Lattice-based extended withdrawable signatures

Ramses Fernandez
ramses.fernandez@fairgate.io

Fairgate Labs

Abstract

This article presents an extension of the work performed by Liu, Baek and Susilo [16] on extended withdrawable signatures to lattice-based constructions. We introduce a general construction, and provide security proofs for this proposal. As instantiations, we provide concrete construction for extended withdrawable signature schemes based on Dilithium [5] and HAETAE [4].

1 Introduction

Digital signatures enable entities to securely bind their identities to the information they transmit. When a signer with an established public key signs messages using their private key, anyone with knowledge of the public key can verify both the origin and integrity of the message. This fundamental mechanism lies behind the trust in digital communications across countless applications, such as blockchain technology.

Digital signatures are designed to provide authenticity, integrity, and non-repudiation. Once created, a signature remains valid indefinitely, and the signer cannot rescind it. While this permanence is generally considered a feature, in some situation can be considered a drawback and it raises an important question: Is it possible for signers to efficiently revoke their signatures without compromising their private keys or affecting the validity of their other signatures?

Liu, Baek, and Susilo addressed this challenge by introducing the concept of withdrawable signatures [15], offering a practical and secure mechanism for signature revocation in contexts where this capability is desirable. Their work established the foundational definitions and properties of withdrawable signatures, demonstrating their potential utility in various applications.

However, the initial construction for withdrawable signatures had significant limitations, primarily because it was based on designated verifier signatures. This restriction meant that a withdrawable signature could only be verified by a specific, designated user, limiting its broader applicability.

Building on this foundation, Liu, Baek, and Susilo proposed two specific constructions for withdrawable signatures: one based on BLS signatures [1] and another based on Schnorr

signatures. These constructions expanded the practical applicability of the withdrawable signature concept.

Further advancements came when the author extended these concepts to lattice-based cryptography, proposing a general construction for signatures based on Lyubashevsky's Fiat-Shamir with aborts paradigm [17]. As a concrete implementation, the author demonstrated a withdrawable signature scheme based on Dilithium [5] noting that the same construction could potentially be applied to the HAETAE scheme [4].

More recently, Liu, Baek and Susilo, have introduced the concept of *extended* withdrawable signatures [16]. Their work aims to bridge existing gaps by building upon the revisited concept of withdrawable signatures, removing the restriction of employing designated verifier signatures. The key insight was that any 1-out-of-n signature scheme with a specific property ensuring signer ambiguity can achieve what the authors term *extended withdrawability*. Based on this extended definition and accompanying security notions, they developed a comprehensive generic construction for extended withdrawable signatures through any 1-out-of-n signature scheme derived from discrete-log-based primitives, providing a construction for Schnorr signatures as an instantiation.

1.1 Related work

Jakobsson et al. [12] and Chaum [3] independently introduced the designated-verifier signature (DVS) concept in 1995. This area has undergone extensive research for decades, with various assumptions yielding different implementations as demonstrated in [13] and [23] to name a few.

Universal designated-verifier signature schemes (UDVS) represent a significant variation. The first UDVS scheme was developed by Steinfeld et al. [21] using bilinear groups, followed by two additional schemes [22] that expanded conventional Schnorr and RSA signature methods. This pioneering work prompted several subsequent UDVS proposals, being [14] the first lattice-based UDVS proposal.

Ring signatures were originally formulated by Rivest, Shamir, and Tauman [19]. This scheme enables a signer to select a public key set (including their own) and generate a signature representing that set. Implementations span several approaches: RSA-based [19], discrete logarithm-based [11], pairing-based [2], and lattice-based [7].

Ring signatures can be constructed generically through zero-knowledge proofs on a signer index, particularly using one-out-of-many proofs, as demonstrated in [10].

Park et al. [18] introduced claimable ring signatures (CRS), which allow anonymous message signing while preserving the option for later identity disclosure. This parallels withdrawable signatures, which are considered revoked by default—after generation, they remain simulatable and verifiable using the signer's public key only after signer confirmation. The signer's ambiguity property controls this withdrawability. From this angle, claimable ring signatures represent a specific implementation of withdrawable signatures, achieving withdrawability through signer ambiguity.

Building on this concept, Yamashita et al. [24] proposed claimable designated verifier signatures (CDVS), enabling signers to assert their creation of a designated verifier signature when needed. This represents another implementation of withdrawable signatures maintaining signer-ambiguity.

The author has explored withdrawable signatures within the lattice-based framework, extending techniques established by Liu, Baek and Susilo [15]. This research presents a general construction for withdrawable signatures utilizing schemes based on the Fiat-Shamir with aborts paradigm, and offers two concrete implementations: one incorporating Dilithium [5] and another leveraging HAETAE [4].

1.2 Contributions

In this research, we build upon the ideas in [16] to provide a general construction for extended withdrawable signatures in the setting of lattice-based constructions, extending the work done by the author [8] in a natural direction. As an instantiation, we provide a Dilithium-based extended withdrawable signature scheme; noting that the same construction can be applied to HAETAE.

2 Main concepts

2.1 Notation

We write R and R_q to denote the rings $\mathbb{Z}[x]/(x^n+1)$ and $\mathbb{Z}_q[x]/(x^n+1)$ respectively, where q is an integer. We will denote vectors by bold letters, and regular font letters denote elements in R or R_q. Unless otherwise specified, all vectors are assumed to be column vectors. Bold upper-case letters denote matrices. If S is a set, then $a \xleftarrow{\$} S$ means that a is chosen uniformly at random from S. All logarithms are assumed to be base 2.

For any even positive integer α, we define $r' = r \bmod \pm\alpha$ to be the unique element r' in the range $-\frac{\alpha}{2} < r' \leq \frac{\alpha}{2}$ such that $r' \equiv r \bmod \alpha$. For any odd positive integer α, we define $r' = r \bmod \pm\alpha$ to be the unique element r' in the range $-\frac{\alpha-1}{2} \leq r' \leq \frac{\alpha-1}{2}$ such that $r' \equiv r \bmod \alpha$. We refer to these operations as centered reductions modulo α.

For any positive integer α, we define $r' = r \bmod {}^+\alpha$ to be the unique element r' in the range $0 \leq r' < \alpha$ such that $r' \equiv r \bmod \alpha$. When the specific representation is not significant, we simply write $r \bmod \alpha$.

For an element $w \in \mathbb{Z}_q$, let $\|w\|_\infty := |w \bmod \pm q|$. For an element $w = \sum_{i=0}^{n-1} w_i x^i \in R$, we define $\|w\|_\infty = \max_i \|w_i\|_\infty$ and $\|w\| = \sqrt{\|w_0\|_\infty^2 + \ldots + \|w_{n-1}\|_\infty^2}$.

For $\mathbf{w} = (w_1, \ldots, w_k) \in R^k$: $\|\mathbf{w}\|_\infty = \max_i \|w_i\|_\infty$ and $\|\mathbf{w}\| = \sqrt{\|w_1\|^2 + \ldots + \|w_k\|^2}$. We denote by $\mathcal{S}_\beta$ the set of all elements $w \in R$ such that $\|w\|_\infty \leq \beta$ for $\alpha \in \mathbb{R}$. We note an abuse of notation: we write $\mathcal{S}$, with no subindex, to indicate that bounds in this case are given by the underlying scheme.

Let B_h denote the subset of elements in R that have exactly h coefficients equal to either -1 or 1, with all remaining coefficients being 0. The cardinality of this set is given by $|B_h| = 2^h \cdot \binom{n}{h}$. We consider a hash function $H : \{0,1\}^* \rightarrow B_h$ which operates on the concatenation of its arguments.

2.2 Extended withdrawability

Extended withdrawable signature schemes enhance the original definition in [15], which involved only a signer and designated verifier, by taking the designated verifier into a group of potential signers. To generate a withdrawable signature, the signer selects a set of public keys π containing all potential signers' public keys, including their own public key (pk). The signer then broadcasts auxiliary information aux through the *shuffling algorithm* Shuffle, where $aux = \pi - \{pk\}$ (the set of all other potential signers' public keys).

The original withdrawable signature scheme operates in two stages, namely: generating a withdrawable signature and transforming it into a confirmed signature. The authors in [16] extend the WSVerify algorithm to enable public verification, allowing any entity with the withdrawable signature σ to verify it using the signer's public key pk.

For the transformation stage, the authors redefine CVerify as $\text{CVerify}(\mu, pk, \sigma, \delta)$, indicating that σ is a valid signature by pk on message μ. Additionally, they introduce $\text{Check}(\mu, pk, \sigma, \delta)$, a new algorithm ensuring δ can be traced to its corresponding withdrawable signature σ.

An extended withdrawable signature scheme consists of seven polynomial-time algorithms, detailed as follows:

$$\text{EWS} = (\text{KeyGen}, \text{Shuffle}, \text{WSign}, \text{WSVerify}, \text{Confirm}, \text{CVerify}, \text{Check})$$

- $(pk, sk) \leftarrow \text{KeyGen}(1^\kappa)$: the key generation algorithm takes the security parameter 1^κ as input to return a key pair (pk, sk).
- $aux \leftarrow \text{Shuffle}(pk, \pi)$: the shuffling algorithm takes the signer's public key pk and the public key set of all possible signers π as input to return the auxiliary information $aux = \pi - \{pk\}$.
- $\sigma \leftarrow \text{WSign}(m, sk, aux)$: the withdrawable signing algorithm takes as input a message μ, signer's secret key sk and auxiliary information aux, to return a new extended withdrawable signature σ on μ of pk.
- $1/0 \leftarrow \text{WSVerify}(\mu, pk, \sigma)$: the extended withdrawable signature verification algorithm takes as input an extended withdrawable signature σ on μ of pk with auxiliary information aux, to return either 1 or 0.
- $\delta \leftarrow \text{Confirm}(\mu, sk, pk, aux, \sigma)$: the confirm algorithm takes as input an extended withdrawable signature σ on μ of pk where $\text{WSVerify}(\mu, pk, \sigma) = 1$ and σ is the withdrawable signature generated by the signer, to return a confirmed signature δ on μ of pk.
- $1/0 \leftarrow \text{CVerify}(\mu, pk, \sigma, \delta)$: the confirmed signature verification algorithm takes as input a confirmed signature δ on μ of pk, to return either 1 or 0.

- $1/0 \leftarrow \textsc{Check}(\mu, pk, \sigma, \delta)$: the checking algorithm takes as input a confirmed signature δ on μ of pk where $\text{CVerify}(\mu, pk, \sigma, \delta) = 1$ and an extended withdrawable signature σ on μ of pk where $\text{WSVerify}(\mu, pk, \sigma) = 1$, to returns 1 if σ is the corresponding withdrawable signature of δ. Otherwise, it returns 0.

2.3 Security concepts

The security notions of an extended withdrawable signature scheme EWS cover three properties:

Definition 2.1 (Correctness)**.** *An extended withdrawable signature scheme EWS is correct if for any security parameter κ, any public key set π, and any message $\mu \in \{0,1\}^*$, when executing:*

$$\begin{aligned}(pk, sk) &\leftarrow \textsc{KeyGen}(1^\kappa), \pi = \{pk_1, \ldots, pk_n\}\\ aux &\leftarrow \textsc{Shuffle}(pk, \pi), aux = \{pk_1, \ldots, pk_{i-1}, pk_{i+1}, \ldots, pk_n\}\\ \sigma &\leftarrow \textsc{WSign}(\mu, sk, aux)\\ \delta &\leftarrow \textsc{Confirm}(\mu, sk, pk, aux, \sigma)\end{aligned}$$

it holds with overwhelming probability in the parameter κ: $1 \leftarrow \textsc{WSVerify}(\mu, pk, \sigma)$, $1 \leftarrow \textsc{CVerify}(\mu, pk, \sigma, \delta)$, and $1 \leftarrow \textsc{Check}(\mu, pk, \sigma, \delta)$.

Concerning correctness, the intuition is that when an extended withdrawable signature σ passes WSVerify, its corresponding confirmed signature δ will pass CVerify, and both signatures together will pass the Check algorithm.

Definition 2.2 (Unforgeability under insider corruption)**.** *Let us consider the following three oracles:*

Algorithm 1 Corruption Oracle $\mathcal{O}_j^{\text{Corrupt}}(j)$

if $j \neq i$ **then**
  $\mathcal{CO} \leftarrow \mathcal{CO} \cup \{sk_j\}$
  return sk_j
else return $\perp$
end if

Algorithm 2 Signing Oracle $\mathcal{O}_{sk,aux}^{\text{WSign}}(\mu, sk, aux)$

if $(pk = pk_i) \wedge (i \notin \mathcal{CO}) \wedge (pk_i \notin aux)$ **then**
  $\sigma \leftarrow \text{WSign}(\mu, sk, aux)$
  $\mathcal{W} \leftarrow \mathcal{W} \cup \{\sigma\}$
  return σ
else return $\perp$
end if

Algorithm 3 Confirmation Oracle $\mathcal{O}^{\text{CONFIRM}}_{sk,\sigma}(\mu, sk, pk, aux, \sigma)$

if $\sigma \in \mathcal{W}$ **then**
  $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
  $\delta \leftarrow \text{CONFIRM}(\mu, sk, pk, aux, \sigma)$
  return δ
else return $\perp$
end if

Let us consider the unforgeability under insider corruption experiment $Exp^{EUF\text{-}CMA}_{EWS,\mathcal{A}}(1^\kappa)$ *for a PPT adversary* $\mathcal{A}$ *and security parameter* κ *described below:*

Algorithm 4 Unforgeability under Insider Corruption Experiment $\text{Exp}^{\text{EUF-CMA}}_{\text{EWS},\mathcal{A}}(1^\kappa)$

for $j = 1$ to n **do** $(pk_j, sk_j) \leftarrow \text{KEYGEN}(1^\kappa)$
end for
$\mathcal{P} \leftarrow \{pk_j\}_{j=1}^n$
$(pk, sk) \leftarrow (pk_i, sk_i)$ for some $i \in \{1, \ldots, n\}$
Choose $\pi \subseteq \mathcal{P}$ with $|\pi| = n$
Compute $aux_i \leftarrow \text{SHUFFLE}(pk, \pi_i)$ for $pk \in \pi_i$
Initialize $\mathcal{CO}, \mathcal{W}, \mathcal{M} \leftarrow \emptyset$
$(\mu^*, \sigma^*, (\sigma')^*) \leftarrow \mathcal{A}^{\mathcal{O}_j^{\text{Corrupt}}(\cdot), \mathcal{O}^{\text{WSIGN}}_{sk,aux}(\cdot), \mathcal{O}^{\text{CONFIRM}}_{sk,\sigma}(\cdot)}(1^\kappa, \pi^*)$
if
  $(\pi^* \subseteq \mathcal{P}) \wedge (pk_i \in \pi^*) \wedge (i \notin \mathcal{CO}) \wedge (aux^* = \pi^* - \{pk_i\}) \wedge (\mu^* \notin \mathcal{M})$
  $1 = \text{WSVERIFY}(\mu^*, pk^*, \sigma^*)$
  $1 = \text{CVERIFY}(\mu^*, pk^*, \sigma^*, \delta^*)$
  $1 = \text{CHECK}(\mu^*, pk^*, \sigma^*, \delta^*)$ **then**
  return 1
end if
return 0

An extended withdrawable signature scheme EWS is said to satisfy unforgeability under insider corruption (EUF-CMA) if, for all PPT adversaries $\mathcal{A}$*, there exists a negligible function negl such that:*

$$\Pr\left[Exp^{EUF\text{-}CMA}_{EWS,\mathcal{A}}(1^\kappa) = 1\right] \leq negl(1^\kappa).$$

Unforgeability under insider corruption means that only the original signer can transform a valid extended withdrawable signature σ *generated from* sk *with respect to* pk *into its corresponding confirmed signature* δ*, even if an adversary has access to secret keys* sk_j *of other potential signers* $pk_j \in aux$*.*

Definition 2.3 (Extended withdrawability)**.** *Let us consider* $\pi = \{pk_1, \ldots, pk_n\}$*,* $pk \in \pi$*,* $aux = \pi - \{pk\}$*, and the oracle below:*

Algorithm 5 Signing Oracle $\mathcal{O}^{\text{WSIGN}}_{sk,aux}(\cdot)$

Input: (μ, sk, aux)
if $\mu \notin \mathcal{M}$ **then**
    $\sigma \leftarrow \text{WSIGN}(\mu, sk, aux)$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
    return σ
else return $\perp$
end if

Let us consider the following experiment:

Algorithm 6 Extended Withdrawability Experiment $\text{Exp}^{\text{Withdraw}}_{\text{EWS},\mathcal{A}}(1^\kappa)$

Input: Security parameter κ
for $i = 1$ to n **do** $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\kappa)$
end for
$\pi \leftarrow \{pk_i\}_{i=1}^n$
$\mathcal{M} \leftarrow \emptyset$
$(pk', aux') \leftarrow \mathcal{A}^{\mathcal{O}^{\text{WSIGN}}_{sk,aux}(\cdot)}(1^\kappa, \mu^*, \pi, \sigma^*)$
if $(\mu^* \in \mathcal{M}) \wedge (aux' = \pi - \{pk'\}) \wedge (\text{WSVERIFY}(\mu^*, pk', \sigma^*)) = 1$ **then**
    return 1
else return 0
end if

An extended withdrawable signature scheme EWS achieves extended withdrawability if, for any PPT adversary $\mathcal{A}$, as long as the Confirm algorithm has not been executed, there exists a negligible function negl such that:

$$\Pr\left[Exp^{Withdraw}_{EWS,\mathcal{A}}(1^\kappa) = 1\right] = 1.$$

Extended withdrawability means that a valid extended withdrawable signature σ verifiable through pk can be adapted to be verifiable with another public key $pk' \in aux$ for the same message m. Specifically, it must satisfy signer ambiguity: $WSVERIFY(\mu, pk', \sigma) = 1$, with the corresponding $aux' = aux - \{pk'\}$.

An extended withdrawable signature scheme EWS is secure if it is correct, unforgeable under insider corruption, and extended withdrawable.

3 Security assumptions

The security of our construction is based on three fundamental lattice-based hardness assumptions. The Module Learning With Errors (MLWE) assumption provides protection against key-recovery attacks, ensuring the confidentiality of secret keys. The SelfTargetMSIS assumption establishes the security foundation against new message forgery attempts, preventing adversaries from generating valid signatures for previously unsigned messages.

Finally, the MSIS assumption is essential to achieve strong unforgeability, which prevents even slight modifications to existing signatures.

Definition 3.1 (Module Learning With Errors (MLWE)). *For integers m, k and a probability distribution $D : R_q \to [0, 1]$, the advantage of an algorithm $\mathcal{A}$ in solving the decisional $MLWE_{m,k,D}$ problem over the ring R_q is defined as:*

$$\begin{aligned} Adv^{m,k,D}_{MLWE} := &\Pr[b = 1 \mid \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{t} \leftarrow R_q^m; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{t})] \\ &- \Pr[b = 1 \mid \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{s_1} \leftarrow D^k; \boldsymbol{s_2} \leftarrow D^m; b \leftarrow \mathcal{A}(\mathbf{A}, \mathbf{As_1} + \boldsymbol{s_2})] \end{aligned}$$

Definition 3.2 (Module Short Integer Solution (MSIS)). *For an algorithm $\mathcal{A}$, we define its advantage function $Adv^{m,k,\gamma}_{MSIS}$ in solving the (Hermite Normal Form) $MSIS_{m,k,\gamma}$ problem over the ring R_q as:*

$$Adv^{m,k,\gamma}_{MSIS}(\mathcal{A}) := \Pr\left[0 < \|\mathbf{y}\|_\infty \leq \pi \wedge [\ \mathbf{I} \mid \mathbf{A}\ ] \cdot \mathbf{y} = 0 \mid \mathbf{A} \leftarrow R_q^{m\times k}; \mathbf{y} \leftarrow \mathcal{A}(\mathbf{A})\right]$$

Definition 3.3 (SelfTargetMSIS). *Let $H : \{0, 1\}^* \to B_h$ be a cryptographic hash function. For an algorithm $\mathcal{A}$, we define its advantage function as:*

$$\begin{aligned} AdvSelfTargetMSIS^{H,m,k,\gamma_1}(\mathcal{A}) := \Pr\big[\ 0 \leq |\mathbf{y}|\infty \leq \gamma_1\ \wedge\ H([\ \mathbf{I} \mid \mathbf{A}\ ] \cdot \mathbf{y}|M) = c \\ \mid \mathbf{A} \leftarrow R_q^{m\times k}; (\mathbf{y} := (\mathbf{r}, c), M) \leftarrow \mathcal{A}^{\langle H(\cdot)\rangle}(\mathbf{A})\big] \end{aligned}$$

The above problems are hard if, for any PPT adversary $\mathcal{A}$, the respective advantages are negligible.

The intuition is that the MLWE assumption protects against key-recovery attacks, the SelfTargetMSIS is the assumption upon which new message forgery is based, and the MSIS assumption is needed for strong unforgeability. The decisional MLWE will be used to prove withdrawability.

4 Extended withdrawable signatures

4.1 Lattice-based signatures

Our construction for lattice-based extended withdrawable signatures is tailored to those schemes built upon the Fiat-Shamir with aborts paradigm [17], which we recall in this section. Let FSA.DS = {KeyGen, FSA.Sign, FSA.Verify} be a digital signature scheme defined as follows:

Algorithm 7 KeyGen(1^κ)

1: $\mathbf{A} \xleftarrow{\$} R_q^{k\times l}$, $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} \mathcal{S}^l \times \mathcal{S}^k$
2: $\mathbf{t} = \mathbf{As}_1 + \mathbf{s}_2$
3: **return** $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{s}_1, \mathbf{s}_2))$

Algorithm 8 FSA.$\textsc{Sign}(\mu, sk)$

1: **repeat**
2: $\quad \mathbf{y} \xleftarrow{\$} \mathcal{S}^l$
3: $\quad c = H(\mu, \mathbf{Ay})$
4: $\quad \mathbf{z} = \mathbf{y} + c\mathbf{s}_1$
5: **until** $(\textsc{RejectionSample}(\mathbf{z}) = 1) \wedge (\textsc{RejectionSample}(\mathbf{Az} - c\mathbf{t}) = 1)$
6: **return** $\sigma = (c, \mathbf{z})$

Algorithm 9 FSA.$\textsc{Verify}(\mu, \sigma, pk)$

1: $\mathbf{w}' = \mathbf{Az} - c\mathbf{t}$
2: **if** $(\textsc{RejectionSample}(\mathbf{z}) = 1) \wedge (c = H(\mu, \mathbf{w}'))$ **then return** 1
3: **end if**

One observes that the RejectionSample function is formulated in general, with its specific instantiation depending on the underlying cryptographic schemes employed in the different constructions. Its precise definition is inherently dependent on the parameter sets associated with each of these underlying schemes. Intuitively, this function serves as a validation mechanism that determines whether input arguments fall within the acceptable boundaries or constraints defined by the underlying scheme using the Fiat-Shamir with aborts paradigm.

4.2 1-out-of-n signatures

The generic construction for lattice-based extended withdrawable signatures requires the introduction of 1-out-of-n signatures.

Let $\pi = \{pk_1, \ldots, pk_n\}$ denote a public key set and let $i \xleftarrow{\$} \{1, \ldots, n\}$, where (pk_i, sk_i) is the signer's key pair. A 1-out-of-n signature scheme Ambi.DS that provides signer ambiguity in π consists of three PPT algorithms:

$$\textsc{Ambi.DS} = \begin{cases} (pk_i, sk_i) \leftarrow \textsc{Ambi.KeyGen}(1^\kappa) \\ \sigma \leftarrow \textsc{Ambi.Sign}(\mu, sk_i, \pi) \\ 1/0 \leftarrow \textsc{Ambi.Verify}(\mu, \pi, \sigma) \end{cases}$$

Definition 4.1 (Signer ambiguity)**.** *Let* $(pk_1, sk_1), \ldots, (pk_n, sk_n) \leftarrow \textit{Ambi.KeyGen}(1^\kappa)$. *Consider* $\mathcal{P} = \{pk_j\}_{j=1}^n$, *and the signer key pair* (pk_i, sk_i) *for* $i \xleftarrow{\$} \{1, \ldots, n\}$. *Let us consider the following oracle:*

Algorithm 10 $\mathcal{O}_{sk_i,\pi}^{\textsc{Ambi.Sign}}(\cdot)$

1: **if** $\pi \subseteq \mathcal{P}, pk_i \in \pi$ **then**
2: $\quad \sigma \leftarrow \textsc{Ambi.Sign}(\mu, sk_i, \pi)$
3: $\quad \mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
4: $\quad$ **return** σ
5: **else return** $\perp$
6: **end if**

This oracle allows the definition of the signer ambiguity experiment:

Algorithm 11 $\text{Exp}^{\text{SigAmbi}}_{\text{Ambi.DS},\mathcal{A}}(1^\kappa)$

for $j = 1$ to n **do**
  $(pk_j, sk_j) \leftarrow \text{KeyGen}(1^\kappa)$, $\mathcal{P} \leftarrow \{pk_j\}_{j=1}^n$
  $i \xleftarrow{\$} \{1, \ldots, n\}$, $b \xleftarrow{\$} \{0, 1\}$
  $\mathcal{M} \leftarrow \emptyset$
end for
if $(\pi^* \subseteq \mathcal{P}) \wedge (pk'_0, pk'_1 \in \pi^*) \wedge (\mu^* \notin \mathcal{M}) \wedge (1 \leftarrow \textsc{Ambi.Verify}(\mu^*, \pi^*, \sigma_i^*))$ **then**
  $\sigma_b^* \leftarrow \textsc{Ambi.Sign}(\mu^*, sk'_b, \pi^*)$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}^{\textsc{Ambi.Sign}}_{sk_i,\pi}(\cdot)}(1^\kappa, \mu^*, pk'_0, pk'_1, \sigma_b^*)$
  if $b = b'$ **then return** 1
  else return 0
  end if
end if

A 1-out-of-n signature scheme achieves signer ambiguity if, for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that:

$$\Pr\left[Exp^{SigAmbi}_{\text{Ambi.DS},\mathcal{A}}(1^\kappa) = 1\right] \leq \frac{1}{n} + negl(1^\kappa).$$

4.3 Lattice-based 1-out-of-n signatures

Let us denote the secret key of a i^{th}-participant as $sk_i = (\mathbf{s}_1^i, \mathbf{s}_2^i)$. We develop a 1-out-of-$n$ signature scheme based on the Fiat-Shamir with aborts paradigm as follows:

Algorithm 12 $\textsc{KeyGen}(1^\kappa)$

$\mathbf{A} \xleftarrow{\$} R_q^{k \times l}$, $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} \mathcal{S}^l \times \mathcal{S}^k$
$\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
return $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$

Algorithm 13 $\textsc{Ambi.Sign}(\mu, sk_i, \pi)$

Parse $\pi = \{pk_j\}_{j=1}^n$
repeat
  $\mathbf{y} \xleftarrow{\$} \mathcal{S}^l$
  for $j \neq i$ **do** $t_j \xleftarrow{\$} R_q$
  end for
  $\mathbf{n} = \mathbf{A}\mathbf{y} - \sum_{j \neq i} t_j(\mathbf{A}\mathbf{s}_1^j + \mathbf{s}_2^j)$
  $t = H(\mu, \pi, \mathbf{n})$
  $t_i = t - \sum_{j \neq i} t_j$
  $\mathbf{z} = \mathbf{y} + t_i \mathbf{s}_1^i$
until $(\textsc{RejectionSample}(\mathbf{z}) = 1) \wedge (\textsc{RejectionSample}(\mathbf{A}\mathbf{z} - t_i\mathbf{t}) = 1)$
$\sigma = (t_1, \ldots, t_n, \mathbf{z})$
return σ

Algorithm 14 AMBI.VERIFY(μ, σ, π)

Parse $\sigma = (t_1, \ldots, t_n, \mathbf{z})$, $\pi = \{pk_j\}_{j=1}^n$
$t = \sum_{j=1}^n t_j$
$\mathbf{n}' = \mathbf{A}\mathbf{z} - \sum_{j=1}^n t_j(\mathbf{A}\mathbf{s}_1^j + \mathbf{s}_2^j)$
$t' = H(\mu, \pi, \mathbf{n}')$
if (REJECTIONSAMPLE$(\mathbf{z}) = 1) \wedge (t = t')$ **then return** 1
else return 0
end if

4.4 The general construction

Let us consider a signature scheme built upon the Fiat-Shamir with aborts paradigm LATTICE.DS = {KEYGEN, LATTICE.SIGN, LATTICE.VERIFY}. Our generic construction for extended withdrawable signatures is detailed below:

Algorithm 15 KEYGEN(1^κ)

$\mathbf{A} \xleftarrow{\$} R_q^{k \times l}$, $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} \mathcal{S}^l \times \mathcal{S}^k$
$\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
return $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{s}_1, \mathbf{s}_2))$

Algorithm 16 SHUFFLE(pk, π)

$\pi = \{pk_j\}_{j=1}^n$
$pk_i \leftarrow pk, i \xleftarrow{\$} \{1, \ldots, n\}, aux = \pi - \{pk_i\}$
return aux

Algorithm 17 WSIGN(μ, sk, aux)

$\pi = aux \cup pk$
for $j \in \{1, \ldots, n\}$ **do** $s_j \xleftarrow{\$} \mathcal{S}$ ▷ $\mathcal{S}$ is the same set as $\mathbf{s}_1$
end for
$r = s_i = H(\pi, \sum_{j \neq i} \mathbf{A}\mathbf{s}_1 s_j + \mathbf{s}_2)$
$\aleph = \{\mathbf{A}\mathbf{s}_1 s_j\}_{j \neq i}$
$sk' = sk + r\mathbf{s}_1 = (\mathbf{s}_1 + r\mathbf{s}_1, \mathbf{s}_2)$
$\pi' = \{pk_j + \mathbf{A}\mathbf{s}_1 s_j\}_{j=1}^n$
$\sigma_1 = \text{AMBI.SIGN}(\mu, \pi', sk')$
$\sigma_2 = \aleph \cup \mathbf{A}\mathbf{s}_1 s_i$
return $\sigma = (\sigma_1, \sigma_2, aux)$

Algorithm 18 WSVERIFY(μ, pk, aux)

$\sigma = (\sigma_1, \sigma_2, aux), \sigma_2 = \{\mathbf{A}\mathbf{s}_1 s_j\}_{j=1}^n$
$\pi = aux \cup pk = \{pk_j\}_{j=1}^n, \pi' = \{pk_j + \mathbf{A}\mathbf{s}_1 s_j\}_{j=1}^n$
if $1 =$ AMBI.VERIFY(μ, π', σ_1) **then return** 1
else return 0
end if

Algorithm 19 CONFIRM$(\mu, sk, pk, aux, \sigma)$

$\sigma = (\sigma_1, \sigma_2, aux)$
$\pi = aux \cup pk = \{pk_j\}_{j=1}^{n}$
$\aleph = \sigma_2 - \mathbf{A}\mathbf{s}_1 s_i$
for $\mathbf{A}\mathbf{s}_1 s_j \in \aleph$ **do** $r' = s_i' = H(\pi, \sum_{j \neq i} \mathbf{A}\mathbf{s}_1 s_j + \mathbf{s}_2)$
end for
$sk'' = sk + r'\mathbf{s}_1 = (\mathbf{s}_1 + r'\mathbf{s}_1, \mathbf{s}_2)$
$\delta_1 =$ LATTICE.SIGN$(\{\mu, \sigma\}, sk'')$
$\delta_2 =$ LATTICE.SIGN(μ, sk)
$\delta_3 = \mathbf{A}(\mathbf{s}_1 + r'\mathbf{s}_1) + \mathbf{s}_2$
return $\delta = (\delta_1, \delta_2, \delta_3)$

Algorithm 20 CVERIFY$(\mu, pk, \sigma, \delta)$

$\delta = (\delta_1, \delta_2, \delta_3)$
if $(1 =$ LATTICE.VERIFY$(\{\mu, \sigma\}, \delta_3, \delta_1)) \wedge (1 =$ LATTICE.VERIFY$(\mu, pk, \delta_2))$ **then return** 1
else return 0
end if

Algorithm 21 CHECK$(\mu, pk, \sigma, \delta)$

$\sigma = (\sigma_1, \sigma_2, aux), \sigma_2 = \{\mathbf{A}\mathbf{s}_1 s_j\}_{j=1}^{n}, \mathbf{A}\mathbf{s}_1 r = \mathbf{A}\mathbf{s}_1 s_i, \delta = (\delta_1, \delta_2, \delta_3)$
$\pi = aux \cup pk = \{pk_j\}_{j=1}^{n}, \pi' = \{pk_j + \mathbf{A}\mathbf{s}_1 s_j\}_{j=1}^{n}$
if
  $\mathbf{A}(\mathbf{s}_1 + r\mathbf{s}_1) + \mathbf{s}_2 = \mathbf{A}(\mathbf{s}_1 + r'\mathbf{s}_1) + \mathbf{s}_2$
  $1 =$ WSVERIFY(μ, pk, σ)
  $1 =$ CVERIFY$(\mu, pk, \sigma, \delta)$
  $1 =$ LATTICE.VERIFY$(\{\mu, \sigma\}, \mathbf{A}(\mathbf{s}_1 + r\mathbf{s}_1) + \mathbf{s}_2, \delta_1)$ **then return** 1
else return 0
end if

4.5 Two instantiations: Dilithium and HAETAE

The construction presented in Section 4.4 can be applied to a number of lattice-based signature schemes satisfying the security assumptions set in Section 3. One can construct extended withdrawable signatures built upon Dilithium [5] or HAETAE [4] with the replacement LATTICE.SIGN/VERIFY $\leftrightarrow$ DILITHIUM.SIGN/VERIFY or HAETAE.SIGN/VERIFY.

The function REJECTIONSAMPLE in these situations is given by the parameters defined in [5] and [4], respectively. In the case of Dilithium, we set β as the maximum possible coefficient of $c\mathbf{s}_i$, γ_1 is large enough so the signature does not reveal the secret key and small enough so that the signature is not forged, $\gamma_2 = \gamma_1/2$, and η is a small integer. We set $B_h \coloneqq B_{60}$. Similarly, one can instantiate REJECTIONSAMPLE for the HAETAE-based construction following the parameter set suggested in [4].

5 Security analysis

The security analysis of the constructions in this research follows the arguments and ideas used in [16]. We start proving the unforgeability under insider corruption.

Theorem 5.1. *If the underlying signature scheme is secure against EUF-CMA on the random oracle model, then our generic construction for the extended withdrawable signature from lattice-based primitives is unforgeable under insider corruption in the random oracle model with reduction loss $L = q_{2H-1}$ where q_{2H-1} denotes the number of hash queries to the random oracle H.*

Proof. Let $\mathcal{C}$ be an algorithm that violates the EUF-CMA security of the underlying signature scheme LATTICE.DS. We construct a simulator $\mathcal{B}$ that utilizes adversary $\mathcal{A}$, which breaks unforgeability under insider corruption, to enable $\mathcal{C}$ to compromise the EUF-CMA security of LATTICE.DS in the random oracle model.

Setup: simulator $\mathcal{B}$ interacts with algorithm $\mathcal{C}$. Consider that $\mathcal{C}$ runs the EUF-CMA security game of LATTICE.DS, denoted as $\text{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}$ which includes a signing oracle $\mathcal{O}_{sk_i}^{\text{LATTICE.SIGN}}(\cdot)$, where $\mathcal{O}_{sk_i}^{\text{LATTICE.SIGN}}(\cdot) : \omega_i \leftarrow \text{LATTICE.SIGN}(\mu, sk_i)$. Initially, $\mathcal{C}$ generates $(pk_i, sk_i) \leftarrow \text{KEYGEN}(1^\kappa)$, and simulator $\mathcal{B}$ receives pk_i from $\mathcal{C}$. Simulator $\mathcal{B}$ then establishes $\mathcal{P} = \{pk_j\}_{j=1}^n$ where $pk_i \in \mathcal{P}, i \in \{1, \ldots, n\}$ and supplies $\mathcal{P}$ to adversary $\mathcal{A}$. Subsequently, $\mathcal{A}$ defines the public key set $\pi_i \subseteq \mathcal{P}$ where $pk_i \in \pi_i$ and provides π_i to $\mathcal{B}$.

Oracle simulation: $\mathcal{B}$ responds to oracle queries as follows.

- Corruption query: adversary $\mathcal{A}$ requests secret keys for $pk_j, j \in \{1, \ldots, n\}$ during this phase. If $\mathcal{A}$ requests the secret key for pk_i, $\mathcal{B}$ terminates the simulation. Otherwise, $\mathcal{B}$ provides the corresponding sk_j to $\mathcal{A}$, and adds sk_j to the corrupted secret key list $\mathcal{CO}$.
- H-Query: adversary $\mathcal{A}$ issues hash queries during this phase. Algorithm $\mathcal{C}$ implements H as a random oracle, and $\mathcal{B}$ forwards the responses from $\mathcal{C}$ to answer $\mathcal{A}$'s hash queries.
- Signature query: $\mathcal{A}$ submits a message μ_i and requests an extended withdrawable signature with signer pk_i and $\pi_i \subseteq \mathcal{P}$. If $pk_i \notin \pi_i$, $\mathcal{B}$ terminates. Otherwise, $\mathcal{B}$ inputs μ_i to $\mathcal{C}$. Simulator $\mathcal{B}$ then obtains the signature from $\mathcal{C}$ as $\omega_i \leftarrow \text{LATTICE.SIGN}(\mu_i, sk_i)$. With $\omega_i = (t_i', \mathbf{z}_i')$ received from $\mathcal{C}$, $\mathcal{B}$ addresses the signature query from $\mathcal{A}$ as follows:
  - $\mathcal{O}_{sk_i,aux}^{\text{WSIGN}}(\cdot)$: $\mathcal{B}$ computes $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, aux_i)$.
  - $\mathcal{O}_{sk_i,\sigma}^{\text{CONFIRM}}(\cdot)$: $\mathcal{B}$ generates the confirmed signature response for $\mathcal{A}$ as follows:
    1. $\delta_{1,i} \leftarrow \text{LATTICE.SIGN}(\mu_i, \sigma_i, sk_i')$
    2. $\delta_i = (\delta_{1,i}, \omega_i, \delta_{3,i})$

  Concurrently, $\mathcal{B}$ updates $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu_i\}$ and $\mathcal{W} \leftarrow \mathcal{W} \cup \{\sigma_i\}$.

Forgery: $\mathcal{B}$ obtains an extended withdrawable signature σ_{μ^*} for pk_i with $aux^* \subseteq \mathcal{P}$ on some previously unqueried message μ^*. Subsequently, adversary $\mathcal{A}$ converts σ_{μ^*} into δ_{μ^*}. If $0 \leftarrow \text{CVERIFY}(\mu, pk, \sigma_{\mu^*}, \delta_{\mu^*})$ or $0 \leftarrow \text{CHECK}(\mu, pk, \sigma_{\mu^*}, \delta_{\mu^*})$, the simulation terminates.

Otherwise, if δ_{μ^*} is valid, $\mathcal{B}$ extracts a forged signature ω^* for pk_i on μ^*, σ_{μ^*} directly from $\mathcal{A}$'s forged confirmed signature $\delta_{\mu^*} = (\delta_1^*, \omega^*)$.

Consequently, we can leverage $\mathcal{A}$ to violate the unforgeability in the EUF-CMA model of our underlying signature scheme LATTICE.DS, which contradicts the security properties of our underlying signature scheme.

Probability of successful simulation: all requested extended withdrawable signatures σ_i and their corresponding confirmed signatures δ_i can be properly simulated, and the forged signature is reducible since message μ^* cannot be selected for a signature query as it will be utilized for signature forgery. Therefore, the probability of successful simulation equals $\frac{1}{q_{2H-1}}$ for q_{2H-1} queries. □

Before moving ahead with the proof of the extended withdrawability of our generic construction, we need to provide the proof of the signer ambiguity in the random oracle for our underlying 1-out-of-n signature scheme, which follows from the theorem below:

Theorem 5.2. *The 1-out-of-n signature scheme from lattice-based primitives satisfies the signer-ambiguity in the random oracle model.*

Proof. Simulator $\mathcal{B}$ establishes the challenge public key set $\mathcal{P} = \{pk_j\}_{j=1}^n$ and corresponding secret key set $\mathcal{R} = \{sk_j\}_{j=1}^n$. We denote the signer as pk_i where $i \in \{1, \ldots, n\}$.

Oracle simulation: $\mathcal{B}$ processes oracle queries as follows.

- H-Query: during this phase, adversary $\mathcal{A}$ submits hash queries. Simulator $\mathcal{B}$ implements H as a random oracle.
- Signature query: during this phase, adversary $\mathcal{A}$ requests signature generation. $\mathcal{A}$ provides a message μ_i and requests a 1-out-of-n signature with public key set $\pi_i \subseteq \mathcal{P}$, where signer $pk_i \in \pi_i$. $\mathcal{B}$ addresses the signature request from $\mathcal{A}$ via $\mathcal{O}_{sk_i,\pi}^{\text{AMBI.SIGN}}(\cdot)$: $\mathcal{B}$ generates and returns $\sigma_i \leftarrow \text{AMBI.SIGN}(\mu_i, sk_i, \pi_i)$.

Challenge: adversary $\mathcal{A}$ presents simulator $\mathcal{B}$ with a challenge message μ^* together with public key set $\pi^* \subseteq \mathcal{S}$, where $\pi^* = \{pk_j\}_{j=1}^n$, and two public keys $pk_0', pk_1' \subseteq \pi^*$. The secret keys corresponding to pk_0', pk_1' are denoted as sk_0', sk_1'.

Simulator $\mathcal{B}$ selects $\{t_1, \ldots, t_n\} \xleftarrow{\$} B_h$ and $\mathbf{e} \xleftarrow{\$} \mathcal{S}^l$. Subsequently, simulator $\mathcal{B}$ randomly selects $b \xleftarrow{\$} \{0, 1\}$ and computes $\sigma_b \leftarrow \text{AMBI.SIGN}(\mu^*, sk_b', \pi^*)$ as: $\sigma_b = (t_1, \ldots, t_n, \mathbf{z})$ where $\mathbf{z} = \mathbf{e}$.

Output: adversary $\mathcal{A}$ produces a bit b'. Given that neither b nor sk_b is utilized in the generation phase of σ_b, adversary $\mathcal{A}$ can only succeed with probability $\frac{1}{2}$.

Probability of successful simulation: no termination occurs in our simulation, therefore the probability of successful simulation equals 1. □

Theorem 5.3. *The construction for lattice-based extended withdrawable signature satisfies the extended withdrawability in the random oracle model.*

Proof. Given that the underlying 1-out-of-n signature scheme AMBI.DS possesses signer ambiguity, simulator $\mathcal{B}$ constructs the challenge public key set for $\mathcal{A}$ as $\pi = \{pk_j\}_{j=1}^{n}$ where the signer's public key is represented as $pk_i \in \pi$. The corresponding auxiliary information is defined as $aux = \pi - \{pk_i\}$.

Oracle simulation: Simulator $\mathcal{B}$ responds to oracle queries through the following mechanisms.

- H-Query: during this phase, adversary $\mathcal{A}$ initiates hash queries. Simulator $\mathcal{B}$ implements H as a random oracle.
- Signature query: Adversary $\mathcal{A}$ provides m_i and requests an extended withdrawable signature with public key pk_i and auxiliary information aux. Simulator $\mathcal{B}$ addresses this signature request by generating $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, aux)$ through the $\mathcal{O}^{\text{WSIGN}}_{sk,aux}(\cdot)$ oracle. Concurrently, simulator $\mathcal{B}$ updates the message set $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu_i\}$.

Challenge: in the challenge phase, adversary $\mathcal{A}$ presents simulator $\mathcal{B}$ with a message $\mu^* \notin \mathcal{M}$, ensuring μ^* has not been previously queried. Simulator $\mathcal{B}$ then selects $s_j^* \xleftarrow{\$} \mathcal{S}$ for $j \in \{1, \ldots, n\}$ and constructs the challenge withdrawable signature σ^*. Subsequently, simulator $\mathcal{B}$ formulates $\sigma^* = (\sigma_1^*, \sigma_2^*, aux)$.

Simulate: adversary $\mathcal{A}$ proceeds to simulate the withdrawable signature σ^* of pk on a new public key $pk' \in aux$ as $(\sigma')^* = ((\sigma_1')^*, (\sigma_2')^*) = (\sigma_1^*, \sigma_2^*, aux')$, where $aux' = \pi - \{pk_i\}$.

Probability of breaking the extended withdrawability property: since AMBI.DS achieves signer ambiguity, given the challenge withdrawable signature δ^*, adversary $\mathcal{A}$ has only a $1/n$ probability of correctly identifying the actual signer's identity pk_i. We note that $1 \leftarrow \text{WSVERIFY}(\mu^*, pk', \sigma^*)$ as the verification process remains consistent as follows: the verifier derives the same $\pi = aux' \cup \{pk'\}$ and $(\pi')^*$. In such situation we have $1 \leftarrow \text{AMBI.VERIFY}(\mu^*, (\pi')^*, \sigma_1^*)$.

Probability of successful simulation: no termination occurs in our simulation, thus the probability of successful simulation equals 1. □

6 Conclusion and future research

Withdrawable signatures offer a compelling solution to the challenge of transaction revocation in blockchain systems, a functionality that has traditionally been difficult to implement but is increasingly recognized as valuable for many use cases.

This research builds upon the ideas introduced by Liu et al. [16], extending the work presented by the author in [8] to create a broader framework for withdrawable signatures. Our construction demonstrates significant versatility, as it can be effectively applied to any construction utilizing the Fiat-Shamir with aborts paradigm.

Several promising research directions arise. First, efficiency optimization remains an important consideration for practical implementation. Second, the incorporation of additional cryptographic properties such as blindness or multiparty functionality could enhance the utility of these signatures in complex systems.

References

[1] Boneh, Dan, Ben Lynn, and Hovav Shacham. "Short signatures from the Weil pairing." In International conference on the theory and application of cryptology and information security, pp. 514-532. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[2] Boneh, Dan, Craig Gentry, Ben Lynn, and Hovav Shacham. "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 Berlin Heidelberg, 2003.

[3] Chaum, David, and Eugène Van Heyst. "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 Berlin Heidelberg, 1991.

[4] Cheon, Jung Hee, Hyeongmin Choe, Julien Devevey, Tim Güneysu, Dongyeon Hong, Markus Krausz, Georg Land, Marc Möller, Damien Stehlé, and MinJune Yi. "HAETAE: Shorter lattice-based fiat-shamir signatures." IACR Transactions on Cryptographic Hardware and Embedded Systems 2024, no. 3 (2024): 25-75.

[5] Ducas, Léo , Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. "Dilithium: A lattice-based digital signature scheme." IACR Transactions on Cryptographic Hardware and Embedded Systems (2018): 238-268.

[6] ElGamal, Taher. "A public key cryptosystem and a signature scheme based on discrete logarithms." IEEE transactions on information theory 31, no. 4 (1985): 469-472.

[7] Esgin, Muhammed F., Ron Steinfeld, Amin Sakzad, Joseph K. Liu, and Dongxi Liu. "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 International Publishing, 2019.

[8] Fernandez, Ramses. "Withdrawable signatures in Fiat-Shamir with aborts constructions." Cryptology ePrint Archive (2025).

[9] Fiat, Amos, and Adi Shamir. "How to prove yourself: Practical solutions to identification and signature problems." In Conference on the theory and application of cryptographic techniques, pp. 186-194. Berlin, Heidelberg: Springer Berlin Heidelberg, 1986.

[10] Groth, Jens, and Markulf Kohlweiss. "One-out-of-many proofs: Or how to leak a secret and spend a coin." In Annual International Conference on the Theory and Applications of Cryptographic Techniques, pp. 253-280. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015.

[11] Herranz, Javier, and Germán Sáez. "Forking lemmas for ring signature schemes." In International Conference on Cryptology in India, pp. 266-279. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003.

[12] Jakobsson, Markus, Kazue Sako, and Russell Impagliazzo. "Designated verifier proofs and their applications." In International Conference on the Theory and Applications of Cryptographic Techniques, pp. 143-154. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996.

[13] Li, Yong, Willy Susilo, Yi Mu, and Dingyi Pei. "Designated verifier signature: definition, framework and new constructions." In International Conference on Ubiquitous Intelligence and Computing, pp. 1191-1200. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[14] Li, BaoHong, YanZhi Liu, and Sai Yang. "Lattice-based universal designated verifier signatures." In 2018 IEEE 15th International Conference on e-Business Engineering (ICEBE), pp. 329-334. IEEE, 2018.

[15] Liu, Xin, Joonsang Baek, and Willy Susilo. "Withdrawable signature: how to call off a signature." In International Conference on Information Security, pp. 557-577. Cham: Springer Nature Switzerland, 2023.

[16] Liu, Xin, Willy Susilo, and Joonsang Baek. "Extended Withdrawable Signature." In International Conference on Data Security and Privacy Protection, pp. 119-140. Singapore: Springer Nature Singapore, 2024.

[17] Lyubashevsky, Vadim. "Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures." In International Conference on the Theory and Application of Cryptology and Information Security, pp. 598-616. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[18] Park, Sunoo, and Adam Sealfon. "It wasn't me! Repudiability and claimability of ring signatures." In Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part III 39, pp. 159-190. Springer International Publishing, 2019.

[19] Rivest, Ronald L., Adi Shamir, and Yael Tauman. "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 Berlin Heidelberg, 2001.

[20] Rivest, Ronald L., Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems." Communications of the ACM 21, no. 2 (1978): 120-126.

[21] Steinfeld, Ron, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. "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 Berlin Heidelberg, 2003.

[22] Steinfeld, Ron, Huaxiong Wang, and Josef Pieprzyk. "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 Berlin Heidelberg, 2004.

[23] Thorncharoensri, Pairat, Willy Susilo, and Joonsang Baek. "Aggregatable certificateless designated verifier signature." IEEE Access 8 (2020): 95019-95031.

[24] Yamashita, Kyosuke, Keisuke Hara, Yohei Watanabe, Naoto Yanai, and Junji Shikata. "Designated verifier signature with claimability." In Proceedings of the 10th ACM Asia Public-Key Cryptography Workshop, pp. 21-32. 2023.