Withdrawable signatures in Fiat-Shamir with aborts constructions

Ramses Fernandez
ramses.fernandez@fairgate.io

Fairgate Labs

Abstract

This article presents an extension of the work performed by Liu, Baek and Susilo [6] on withdrawable signatures to the Fiat-Shamir with aborts paradigm. We introduce an abstract construction, and provide security proofs for this proposal. As an instantiation, we provide a concrete construction for a withdrawable signature scheme based on Dilithium [3].

1 Introduction

Digital signatures serve as a fundamental cryptographic mechanism that enables entities to bind their identities to pieces of information. The essential purpose of a digital signature is to allow a signer, who has established a public key pk, to sign messages using their private key sk in a way that enables anyone knowing pk to verify both the message's origin and its integrity during transit.

An important paradigm for the creation of digital signatures is the Fiat-Shamir transform, which converts interactive identification protocols into non-interactive digital signature schemes. Starting with a three-move identification protocol, where a prover demonstrates knowledge to a verifier through commitment, challenge, and response steps, the transform replaces the verifier's random challenge with a hash function applied to both the commitment and the message. This creates a digital signature scheme where the signing algorithm computes a commitment, generates a challenge by hashing the commitment with the message, and produces a response using the secret key.

The Schnorr signature scheme is perhaps the most well-known application of the Fiat-Shamir transform which has gained particular attention due to its security characteristics and its valuable properties, such as signature aggregation. These advantages make Schnorr signatures especially attractive for blockchain applications where transaction size reduction and privacy enhancement are crucial considerations.

The impact of digital signatures is particularly important in blockchain technology, where this primitive extends beyond basic transaction authentication, enabling sophisticated smart contract interactions, multi-signature schemes for enhanced security, and threshold signature systems for distributed key management. Furthermore, innovations in signature aggregation and batch verification techniques have contributed significantly to blockchain scalability solutions.

Current public-key cryptographic algorithms serve as the foundation for protecting sensitive electronic information from unauthorized access. These algorithms have successfully withstood attacks from conventional computing systems for decades, due to the hardness of their underlying mathematical problems, prime factorization and the computation of discrete logarithms. However, the emergence of quantum computing presents a significant challenge to this security paradigm as Shor's algorithm demonstrates the potential to solve both the prime factorization and the computation of discrete logarithms efficiently. This means that quantum computers possess computational capabilities that could potentially compromise current cryptographic methods, exposing vulnerable data and information. To address this impending challenge, new cryptographic approaches are being designed to withstand attacks from both traditional computers and future quantum systems. These methods rely on problems, such as lattices, error-corrector codes or isogenies of elliptic curves, which have enhanced mathematical structure, leading to computational problems assumed to be hard both for classical and quantum computers. This framework is known as post-quantum cryptography, which represents a critical advancement in information security, ensuring that digital assets remain protected against evolving technological threats.

The Fiat-Shamir transform can be extended to lattices, leading to the Fiat-Shamir with aborts paradigm due to Lyubashevsky [8]. Fiat-Shamir with aborts provides a framework for constructing digital signature schemes with provable security. This methodology addresses the challenge of generating signatures from lattice-based one-way functions by introducing a controlled rejection sampling technique, aborting. When a potential signature might reveal information about the secret key, the signing algorithm simply aborts and restarts the process. This paradigm converts an interactive identification protocol with a a non-negligible probability of aborting into a signature scheme through iterative execution until a successful completion occurs, the aborting procedure. This transformation eliminates the need for interaction by substituting the verifier's challenge with a hash function evaluation, which security analysis treat as a random oracle.

Amongst the main construction based in the Fiat-Shamir with aborts mechanism, we find Dilithium [3] and HAETAE [2]. Dilithium derives its security from the difficulty of solving certain lattice problems, specifically the Module Learning with Errors and Module Short Integer Solution problems. The importance of the scheme comes from the effective balance between security, signature size, and computational efficiency, making it practical for real-world implementations. HAETAE has been specifically designed to produce more compact and efficiently maskable signatures. While built upon the Fiat-Shamir with aborts paradigm underpinning Dilithium, HAETAE introduces design choices that optimize the complexity-to-compactness ratio, which isparticularly important in space-constrained implementation scenarios.

Digital signature schemes are designed to provide authenticity, integrity, and non-repudiation for signed messages. Once a signature is created, it remains valid indefinitely, and the signer cannot rescind it. This permanence, while typically considered a feature, 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 address this challenge by introducing the concept of withdrawable signatures [6], offering a practical and secure mechanism for signature revocation in situations where this capability is desirable.

The practical applications of withdrawable signatures span multiple domains where signature revocation capability is essential without compromising the signer's private key. In blockchain-based smart contracts, these signatures enable participants to commit to contract conditions while maintaining the ability to revoke their commitment, particularly valuable when contract fulfillment depends on multiple parties or external conditions. Within decentralized e-voting systems, withdrawable signatures provide voters the security to cast their votes while retaining the flexibility to modify their choices before final vote tabulation, allowing voters to respond to new information or developments during the voting period. Additionally, in decentralized escrow services, these signatures facilitate multi-party transactions by allowing participants to revoke their signatures if circumstances change or disputes arise, without compromising the security of other parties' signatures.

1.1 Contributions

This paper builds upon the research conducted by Liu, Baek, and Susilo on withdrawable signatures [6] to present a general methodology for constructing post-quantum withdrawable signatures based on Lyubashevsky's Fiat-Shamir with aborts paradigm [8].

The work first introduces a comprehensive abstract construction that takes a mechanism based in the Fiat-Shamir with aborts paradigm as its starting point and extends it to incorporate withdrawability features. To demonstrate the practical applications of this abstract framework, the paper presents a concrete implementation of withdrawable versions for a prominent post-quantum signature scheme: Dilithium [3].

The integration of withdrawability features with post-quantum cryptography addresses two crucial challenges in modern cryptographic systems. First, it provides the flexibility to revoke signatures when needed, a capability increasingly important in dynamic digital environments. Second, it ensures this functionality remains secure against quantum computing threats.

1.2 Related work

The research in this paper revolves around three gravity centres, namely: the work done by Lyubashevsky, the construction of Liu, Baek and Susilo, and the digital scheme Dilithium.

The Fiat-Shamir heuristic [5] provides a method for transforming public-coin interactive proof systems into digital signatures. This transformation works by substituting the verifier's public coin tosses with hash function evaluations. In the random oracle model (ROM), these hash functions are treated as uniform functions that adversaries can access through classical computing methods.

Schnorr's signature scheme stands as a prominent implementation of the Fiat-Shamir heuristic, with its security fundamentally based on the discrete logarithm problem. However, the emergence of quantum computing necessitates two critical adaptations to this framework. First, the discrete logarithm hardness assumption must be replaced with quantum-resistant alternatives. Second, the security model must account for quantum access to the random oracle (QROM), as quantum adversaries can query the hash function in superposition.

Lyubashevsky's work [8] introduced an innovative lattice-based signature scheme that builds upon Schnorr's design while incorporating abortion as a crucial modification. The abort mechanism ensures that the signature distribution remains independent of the signing key, preventing potential attacks against the signature scheme. The protocol manages these aborts through a loop structure, continuing iterations until a successful execution occurs without an abort.

This modified approach, known as Fiat-Shamir with aborts, maintains the fundamental concept of replacing non-final verifier steps with hash function evaluations while adapting to the requirements of lattice-based cryptography. The integration of the abort mechanism represents a significant advancement in developing quantum-resistant signature schemes, providing a robust framework for cryptographic security in the post-quantum era.

On another hand, Liu, Baek and Susilo propose a designated-verifier signature scheme that introduces withdrawability to digital signatures. Their construction generates a withdrawable signature σ for a message μ, distinct from conventional signature schemes.

The scheme's construction centers on a transformation mechanism. When a signer generates a withdrawable signature, it remains verifiable only by the designated verifier. The signature can then follow two paths: it either remains in its withdrawable state through signer inaction, effectively withdrawing the signature, or undergoes transformation through a "Confirm" algorithm. This algorithm converts the withdrawable signature σ into a confirmed signature σ', which becomes verifiable through both parties' public keys while maintaining a deterministic relationship to σ.

The formal construction involves two entities, signer and verifier, with their public keys comprising a set $\pi = \{pk_s, pk_v\}$, respectively. The scheme utilizes the underlying signature structure to construct a withdrawable signature σ specifically designated for the verifier. Subsequently, using the secret key sk_s and σ, the signer can generate a verifiable signature for μ through the public key set π. This confirmed signature σ' maintains a cryptographic link to the original withdrawable signature σ via the public key set π, ensuring the signature transformation's verifiability and traceability.

Finally, we find Dilithium [3], which builds upon the Fiat-Shamir with aborts paradigm. The scheme operates over module lattices, specifically using the module learning with errors (MLWE) and the Module short integer solution (MSIS) problems as its primary security foundations. Its construction employs a ring $R_q = \mathbb{Z}_q[x]/(x^n + 1)$ with carefully chosen parameters n and q. The signature scheme utilizes matrices and vectors over this ring, with dimensions selected to balance security and efficiency.

The key generation in Dilithium produces a public key containing a matrix $\mathbf{A} \in R_q^{k\times l}$ and a vector $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$, where $\mathbf{s}_1$ and $\mathbf{s}_2$ are secret vectors with small coefficients. The signing process involves generating a masking vector $\mathbf{y}$, computing $\mathbf{w} = \mathbf{A}\mathbf{y}$, and using a challenge value c derived from the message and $\mathbf{w}$ to produce the signature. The scheme incorporates rejection sampling to ensure signature security, leading to probabilistic signature generation.

2 Preliminaries

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, and the hash function $H(\cdot,\cdot)$ will operate on the concatenation of its arguments.

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 $w \in \mathbb{Z}_q$, we denote $\|w\|_\infty$ to mean $|w \bmod \pm q|$. 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 $\|w\| = \sqrt{\|w_0\|_\infty^2 + \ldots + \|w_{n-1}\|_\infty^2}$

Similarly, 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 $\mathcal{S}_\alpha$ the set of all elements $w \in R$ such that $\|w\|_\infty \leq \alpha$ for $\alpha \in \mathbb{R}$.

2.2 Basic definitions

A withdrawable signature scheme involves two participating parties: signers and verifiers. The scheme operates in two primary stages: first, the generation of a withdrawable signature, and second, its transformation into a confirmed signature. Both stages are executed by the signer. Concerning the security of a withdrawable signature scheme WS, it is established through three properties, namely: correctness, unforgeability under insider corruption, and withdrawability.

1. Correctness establishes this strong relation between the verification algorithms: if a withdrawable signature σ is successfully verified through the WSVerify algorithm, then its corresponding confirmed signature σ must also be verifiable through the CVerify algorithm.

2. Unforgeability under insider corruption ensures that only the original signer possesses the capability to transform a verifiable withdrawable signature σ (generated using sk_s for verifier pk_v) into its corresponding confirmed signature σ. This requirement holds even when an adversary has obtained the verifier's secret key sk_v, maintaining the exclusive control of the signer over the confirmation process.

3. Finally, withdrawability establishes the indistinguishability of signature origin. Specifically, given a verifiable withdrawable signature σ, no PPT adversary $\mathcal{A}$ should be able to determine whether the signature was generated by the signer or the verifier, provided that the Confirm algorithm has not been executed on σ. This property effectively ensures that both the signer and the designated verifier possess equivalent capabilities in generating withdrawable signatures.

Definition 2.1 ([6], Section 3.2)**.** *A withdrawable signature scheme consists of five polynomial-time algorithms KeyGen, WSign, WSVerify, Confirm, CVerify, defined as follows:*

- $(pk, sk) \leftarrow KeyGen(1^\kappa)$*: On input of a security parameter* κ*, the key generation algorithm outputs a key pair for each party in the system:* (pk_s, sk_s) *for the signer and* (pk_v, sk_v) *for the verifier.*

- $\sigma \leftarrow WSign(m, sk_s, \gamma)$*: Given a message* m*, a signer's secret key* sk_s*, and a tuple* $\pi = \{pk_s, pk_v\}$ *containing both the signer's public key* pk_s *and a designated verifier's public key* pk_v *from the set of all public keys* S*, the withdrawable signing algorithm generates a withdrawable signature* σ*. This signature is specifically bound to message* m *under the signer's identity and can only be verified by the designated verifier* pk_v*.*

- $1/0 \leftarrow WSVerify(m, sk_v, pk_s, \sigma)$*: The withdrawable signature verification algorithm validates a signature* σ *on message* m *that was generated by a signer with public key* pk_s*. Using the designated verifier's secret key* sk_v*, it returns* 1 *if the signature is valid and* 0 *otherwise.*

- $\tilde{\sigma} \leftarrow Confirm(m, sk_s, \gamma, \sigma)$*: The confirmation algorithm transforms a withdrawable signature* σ *into a confirmed signature* σ*. It takes as input the original message* m*, the signer's secret key* sk_s*, the public key set* γ*, and the withdrawable signature* σ*. The resulting confirmed signature* σ *serves as a publicly verifiable signature with respect to the key set* γ*.*

- $1/0 \leftarrow CVerify(m, \gamma, \sigma, \tilde{\sigma})$*: The confirmed signature verification algorithm validates the authenticity of a confirmed signature* σ *on message* m *with respect to the public key set* γ*. It takes as additional input the original withdrawable signature* σ *from which the confirmed signature was derived. The algorithm outputs 1 if the confirmed signature is valid and 0 otherwise.*

Definition 2.2. *A withdrawable signature scheme* WS *is correct if, for any security parameter* κ*, any public key set* γ*, and any message* $m \in \{0,1\}^*$*, when executing the sequence KeyGen, WSign, and Confirm, then the corresponding verification algorithms satisfy:*

$$WSVerify(m, sk_v, pk_s, \sigma) = 1 \text{ and } CVerify(m, \gamma, \sigma, \tilde{\sigma}) = 1$$

with overwhelming probability (in the security parameter κ*).*

Definition 2.3. *For a PPT adversary $\mathcal{A}$ and security parameter κ, we define the unforgeability under insider corruption experiment $Exp_{WS,\mathcal{A}}^{EUF\text{-}CMA}(1^\kappa)$ using the following three oracles:*

Algorithm 1 Corruption Oracle

procedure $\mathcal{O}_i^{\text{CORRUPT}}(\cdot)$
    if $i \neq s$ **then**
        $\mathcal{CO} \leftarrow \mathcal{CO} \cup sk_i$
        return sk_i
    else
        return $\perp$
    end if
end procedure

Algorithm 2 Withdrawable Signing Oracle

procedure $\mathcal{O}_{sk_s,\gamma}^{\text{WSIGN}}(\cdot)$
    if $pk_s \in \pi \wedge s \notin \mathcal{CO}$ **then**
        $\sigma \leftarrow \text{WSign}(\mu, sk_s, \gamma)$
        $\mathcal{W} \leftarrow \mathcal{W} \cup \{\sigma\}$
        return σ
    else
        return $\perp$
    end if
end procedure

Algorithm 3 Confirmation Oracle

procedure $\mathcal{O}_{sk_s,\sigma,\gamma}^{\text{CONFIRM}}(\cdot)$
    if $\sigma \in \mathcal{W}$ **then**
        $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
        $\sigma \leftarrow \text{Confirm}(\mu, sk_s, \gamma, \sigma)$
        return σ
    else
        return $\perp$
    end if
end procedure

Using these three oracles, we define the unforgeability experiment $Exp_{WS,\mathcal{A}}^{EUF\text{-}CMA}(1^\kappa)$ as follows:

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

for $i = 1$ to m **do**
    $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\kappa)$
end for
Select $s, v \in [1, m]$ where $v \neq s$
Initialize empty sets $\mathcal{CO} \leftarrow \emptyset$, $\mathcal{W} \leftarrow \emptyset$, $\mathcal{M} \leftarrow \emptyset$
$(\mu^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_i^{\text{Corrupt}}(\cdot),\mathcal{O}_{sk_s,\gamma}^{\text{WSign}}(\cdot),\mathcal{O}_{sk_s,\sigma,\gamma}^{\text{Confirm}}(\cdot)}(1^\kappa, \gamma^*)$
if $\gamma^* = \{pk_s, pk_v\} \wedge j \in \mathcal{CO} \wedge \mu^* \notin \mathcal{M}$ **then**
    if $\text{WSVerify}(\mu^*, sk_v, pk_s, \sigma^*) = 1 \wedge \text{CVerify}(\mu^*, \gamma^*, \sigma^*, \sigma^*) = 1$ **then**
        return 1
    end if
end if
return 0

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

$$\Pr[Exp_{WS,\mathcal{A}}^{EUF\text{-}CMA}(1^\kappa) = 1] \leq negl(1^\kappa)$$

Definition 2.4. *Let $(pk_0, sk_0), (pk_1, sk_1) \leftarrow KeyGen(1^\kappa)$ be two generated public/secret key pairs, and let $\pi = \{pk_0, pk_1\}$. For a randomly selected bit $b \xleftarrow{\$} \{0, 1\}$, a security parameter κ, and a PPT adversary $\mathcal{A}$, we build the withdrawability experiment $Exp_{WS,\mathcal{A}}^{Withdraw}(1^\kappa)$ with the following oracle:*

Algorithm 5 Withdrawable Signing Oracle for Withdrawability Experiment

procedure $\mathcal{O}_{sk_s,\gamma}^{\text{WSIGN}}(\cdot)$
  if $\pi = \{pk_0, pk_1\}$ **then**
    $b \xleftarrow{\$} \{0, 1\}$
    $\sigma_b \leftarrow \text{WSign}(\mu, sk_b, \gamma)$
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu\}$
    return σ_b
  else
    return $\perp$
  end if
end procedure

With this signing oracle, we have the following experiment:

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

for $i = 0$ to 1 **do**
  $(pk_i, sk_i) \leftarrow \text{KeyGen}(1^\kappa)$
end for
$\pi \leftarrow \{pk_0, pk_1\}$
$b \xleftarrow{\$} \{0, 1\}$
Initialize empty set $\mathcal{M} \leftarrow \emptyset$
if $\pi = \{pk_0, pk_1\} \wedge \mu^* \notin \mathcal{M}$ **then**
  $\sigma_b \leftarrow \text{WSign}(\mu^*, sk_b, \gamma)$
  $b' \leftarrow \mathcal{A}^{\mathcal{O}_{sk_b,\gamma}^{\text{WSign}}(\cdot)}(1^\kappa, \mu^*, \sigma_b)$
  if $b = b'$ **then**
    return 1
  end if
end if
return 0

A withdrawable signature scheme WS is withdrawable if, for any PPT adversary $\mathcal{A}$, and in the absence of the execution of the Confirm algorithm, there exists a negligible function negl such that:

$$\Pr[Exp_{WS,\mathcal{A}}^{Withdraw}(1^\kappa) = 1] \leq \frac{1}{2} + negl(1^\kappa)$$

2.3 Security definitions and computational assumptions

Definition 2.5. *For a signature scheme $DS = (KeyGen, Sign, Verify)$ and a PPT adversary $\mathcal{A}$, consider the following experiment $Exp_{\mathcal{A}}^{EUF\text{-}CMA}$:*

1. *The challenger $\mathcal{B}$ generates a key pair $(pk_s, sk_s) \leftarrow KeyGen(1^\kappa)$ using the system parameters $\mathcal{SP}$. It provides pk_s to $\mathcal{A}$ while retaining sk_s to handle signature queries.*

2. *$\mathcal{A}$ receives access to the signing oracle $\mathcal{O}^{Sign}_{sk_s}(\cdot)$ that computes $\sigma \leftarrow Sign(\mu, sk_s)$ upon request.*

3. *Eventually, $\mathcal{A}$ outputs a forgery attempt (μ^*, σ^*).*

4. *$\mathcal{A}$ succeeds if $Verify(\mu^*, pk_s, \sigma^*) = 1$ and μ^* was not previously queried to $\mathcal{O}^{Sign}_{sk_s}(\cdot)$.*

We say that DS is (t, q_s, ε)-secure under EUF-CMA if no adversary running in time t and making at most q_s signing queries can succeed with probability greater than ε.

Definition 2.6. *A designated-verifier signature scheme DVS consists of four probabilistic polynomial-time algorithms operating on key pairs (pk_s, sk_s) for signers and (pk_d, sk_d) for designated verifiers:*

$$\begin{aligned} DVS = \Big\{ &(pk, sk) \leftarrow KeyGen(1^\kappa) \\ &\sigma \leftarrow Sign(\mu, pk_d, sk_s) \\ &\sigma \leftarrow Simul(\mu, pk_s, sk_d) \\ &0/1 \leftarrow Verify(\mu, pk_s, sk_d, \sigma) \Big\} \end{aligned}$$

The key security property of DVS schemes is non-transferability, which states that for any message-signature pair (μ, σ) that validates under Verify, it should be computationally infeasible to determine whether σ was produced by the signer using Sign or simulated by the designated verifier using Simul, without access to the signer's secret key sk_s. The formal definition of this property follows:

Definition 2.7 (Non-transferability)**.** *For a designated-verifier signature scheme and a PPT adversary $\mathcal{A}$, consider the non-transferability experiment $Exp^{Sign}_{NonTrans,DV,\mathcal{A}}$:*

Algorithm 7 Non-transferability Experiment $\text{Exp}^{\text{Sign}}_{\text{NonTrans,DV},\mathcal{A}}(1^\kappa)$

1: $(pk_s, sk_s), (pk_d, sk_d) \leftarrow \text{KeyGen}(1^\kappa)$
2: Provide $\mathcal{A}$ access to oracles:
3: $\quad \mathcal{O}^{\text{Sign}}_{sk_s, pk_d}(\cdot) : \sigma_0 \leftarrow \text{Sign}(\mu, sk_s, pk_d)$
4: $\quad \mathcal{O}^{\text{Simul}}_{sk_d, pk_s}(\cdot) : \sigma_1 \leftarrow \text{Simul}(\mu, sk_s, pk_d)$
5: $\mathcal{A}$ outputs message μ^*
6: $b \xleftarrow{\$} \{0, 1\}$
7: Provide $\mathcal{A}$ with signature σ_b^*
8: $\mathcal{A}$ outputs bit b'
9: **return** 1 if $b' = b$, else return 0

A DVS achieves non-transferability if for any PPT adversary $\mathcal{A}$, there exists a negligible function negl such that:

$$\Pr[Exp^{Sign}_{NonTrans,DV,\mathcal{A}}(1^\kappa) = 1] \leq \frac{1}{2} + negl(1^\kappa)$$

The security of our scheme rests upon 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 for achieving strong unforgeability, which prevents even slight modifications of existing signatures.

Definition 2.8 (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:*

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

Definition 2.9 (Module Short Integer Solution (MSIS))**.** *For an algorithm $\mathcal{A}$, we define its advantage function $Adv_{MSIS}^{m,k,\gamma}$ in solving the (Hermite Normal Form) $MSIS_{m,k,\gamma}$ problem over the ring R_q as:*

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

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:

$$
|B_h| = 2^h \cdot \binom{n}{h}
$$

Definition 2.10 (SelfTargetMSIS)**.** *Let $H : \{0,1\}^* \to B_h$ be a cryptographic hash function. For an algorithm $\mathcal{A}$, we define its advantage function as:*

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

The above problems are hard if, for any PPT adversary $\mathcal{A}$, the respective advantages are negligible. Furthermore, it will be a requirement the decisional MLWE problem, which asks $\mathcal{A}$y to distinguish between:

1. $(\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \xleftarrow{\$} R_q^{m\times l}$ and $\mathbf{b} \xleftarrow{\$} R_q^m$ are chosen uniformly at random
2. $(\mathbf{A}, \mathbf{b})$ where $\mathbf{A} \xleftarrow{\$} R_q^{m\times l}$ is chosen uniformly at random, $\mathbf{s} \xleftarrow{\$} R_q^l$ is a secret vector, $\mathbf{e} \xleftarrow{\$} \chi^m$ is an error vector, and $\mathbf{b} = \mathbf{As} + \mathbf{e} \in R_q^m$

We say the decisional MLWE problem is hard if for any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\lambda)$ such that:

$$
|\Pr[\mathcal{A}(\mathbf{A}, \mathbf{b}_0) = 1] - \Pr[\mathcal{A}(\mathbf{A}, \mathbf{b}_1) = 1]| \leq \mathsf{negl}(\lambda) \tag{1}
$$

where $\mathbf{b}_0$ is uniform from R_q^m and $\mathbf{b}_1 = \mathbf{As} + \mathbf{e}$.

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.

3 Withdrawable signature schemes

3.1 The abstract construction

In this section we provide a global construction for withdrawable signatures based in the Fiat-Shamir with aborts paradigm and prove the unforgeability and withdrawability as described in section 2.2. This construction follows the paradigm described in [8] combined with ideas in [6]. Let $H : \{0,1\}^* \to R$ be a random oracle.

Algorithm 8 Fiat-Shamir with aborts signature

procedure KeyGen(1^κ)
  $\mathbf{A} \leftarrow R_q^{k\times l}$
  $\mathbf{s}_1 \leftarrow \mathcal{S}^l, \mathbf{s}_2 \leftarrow \mathcal{S}^k$
  $t = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
  return $pk = (\mathbf{A}, \mathbf{t})$, $sk = (\mathbf{s}_1, \mathbf{s}_2)$
end procedure
procedure Sign($\mu, \rho, sk, \mathbf{t}$)
  repeat
    $\mathbf{y} \leftarrow \mathcal{S}^l$
    $\mathbf{w} = \mathbf{A}\mathbf{y}$
    $c = H(\mu, \mathbf{w})$
    $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$
  until $\mathbf{z} \in \mathcal{S}^l_{\alpha_1}$
  return $\sigma = (\mathbf{z}, c)$
end procedure
procedure Verify(μ, σ, pk)
  $\mathbf{w} = \mathbf{A}\mathbf{z} - c\mathbf{t}$
  if $(\mathbf{z} \in \mathcal{S}^l_{\alpha_1}) \wedge (c = H(\mu, \mathbf{w}))$ **then return** 1
  end if
end procedure

Theorem 3.1 ([8], Theorem 2)**.** *Let n be an integer which is a power of 2, then if the above signature scheme is not strongly unforgeable, then there is a polynomial-time algorithm that can solve $SVP_\varepsilon(\Lambda)$, for $\varepsilon = \tilde{O}(n^2)$ and for every lattice Λ corresponding to an ideal in the ring $\frac{\mathbb{Z}[x]}{x^n+1}$.*

Taking the above scheme as starting point, we define a withdrawable lattice-based scheme as follows:

Algorithm 9 Withdrawable lattice-based signature scheme

procedure KeyGen(1^κ)
  $\mathbf{A} \leftarrow R_q^{k\times l}$
  $\mathbf{s}'_1, \mathbf{s}''_1 \leftarrow \mathcal{S}^l, \mathbf{s}'_2, \mathbf{s}''_2 \leftarrow \mathcal{S}^k$
  $\mathbf{t}_s = \mathbf{A}\mathbf{s}'_1 + \mathbf{s}'_2$, $\mathbf{t}_v = \mathbf{A}\mathbf{s}''_1 + \mathbf{s}''_2$
  return $pk_s = (\mathbf{A}, \mathbf{t}_s)$, $sk_s = (\mathbf{s}'_1, \mathbf{s}'_2)$, $pk_v = (\mathbf{A}, \mathbf{t}_v)$, $sk_v = (\mathbf{s}''_1, \mathbf{s}''_2)$
end procedure
procedure WSign(μ, π, sk_s)
  $\pi = \{pk_s, pk_v\}$
  repeat
    $\mathbf{y} \leftarrow \mathcal{S}^l$
    $\mathbf{w} = \mathbf{A}\mathbf{y}$
    $e = H(\mu, \mathbf{w})$
    $\mathbf{z} = \mathbf{y} + e\mathbf{s}'_1$
  until $\mathbf{z} \in \mathcal{S}^m_{\alpha_1}$
  $r = H(\mu, (\mathbf{s}'_2)^T\mathbf{w})$
  $\mathbf{B} \xleftarrow{\$} R_q^{k\times k}$
  $\sigma_1 = \mathbf{A}\mathbf{z} - e\mathbf{t}_s$, $\sigma_2 = \mathbf{B}\mathbf{t}_v + \mathbf{A}(\mathbf{z} + re\mathbf{s}'_1)$, $\sigma_3 = \mathbf{A}r\mathbf{s}'_1, \sigma_4 = \mathbf{B}\mathbf{A}$
  return $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
end procedure
procedure WSVerify(μ, sk_v, pk_s, σ)
  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
  $e' = H(\mu, \sigma_1)$
  if $\lfloor \sigma_2 \rceil = \lfloor \sigma_1 + e'\sigma_3 + e'\mathbf{t}_s + \sigma_4\mathbf{s}''_1 \rceil$ **then return** 1
  end if
end procedure

Algorithm 10 Withdrawable lattice-based signature scheme *(continuation)*

procedure $\textsc{Confirm}(\mu, sk_s, \gamma, \sigma)$
  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4), \pi = \{pk_s, pk_v\}$
  $r' = H(\mu, (\mathbf{s}_2')^T \sigma_1)$
  repeat
    $\mathbf{y}_s \leftarrow \mathcal{S}^l$
    $e_s = H(\mu, \mathbf{A}\mathbf{y}_s)$
    $\mathbf{z}_s = \mathbf{y}_s + e_s \mathbf{s}_1'$
  until $\mathbf{z}_s \in \mathcal{S}^l_{\alpha_1}$
  repeat
    $\mathbf{y}_v \leftarrow \mathcal{S}^l$
    $e_v = H(pk_v, h(\mathbf{y}_v))$
    $\mathbf{z}_v = y_v + r' e_v \mathbf{s}_1'$
  until $\mathbf{z}_v \in \mathcal{S}^l_{\alpha_1}$
  $\delta_1 = e_s, \delta_2 = \mathbf{z}_s + r' e_s \mathbf{s}_1', \delta_3 = e_v, \delta_4 = \mathbf{z}_v$
  return $\tilde{\sigma} = (\delta_1, \delta_2, \delta_3, \delta_4)$
end procedure
procedure $\textsc{CVerify}(\mu, \gamma, \sigma, \tilde{\sigma})$
  $\pi = \{pk_s, pk_v\}, \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4), \tilde{\sigma} = (\delta_1, \delta_2, \delta_3, \delta_4)$
  $\delta_1' = H(\mu, \mathbf{A}\delta_2 - \mathbf{t}_s\delta_1 - \sigma_3\delta_1)$, $\delta_3' = H(\mathbf{t}_v, \mathbf{A}\delta_4 - \sigma_3\delta_3)$
  if $(\delta_1 = \delta_1') \wedge (\delta_3 = \delta_3')$ **then return** 1
  end if
end procedure

The proofs for Theorem 3.2 and Theorem 3.3 below, closely follow the structure of their analogous Theorems 12 and 13 in [6]. The fundamental arguments remain valid when adapting the security assumptions to our context, with the primary distinction being the underlying signature scheme.

Theorem 3.2. *If a signature scheme is unforgeable against chosen-message attacks, then the associated withdrawable scheme defined using algorithm 9 and algorithm 10 is unforgeable under insider corruption in the random oracle model with reduction loss $L = q_{H_1}$ for q_{H_1} the number of hash queries to the random oracle H.*

Proof. Let $\mathcal{B}$ be an adversary breaking the EUF-CMA of the underlying signature scheme Sign. We assume that $\mathcal{B}$ runs another adversary $\mathcal{A}$ which can break the unforgeability under insider corruption in the random oracle model of the withdrawable signature scheme.

Setup: let us assume that $\mathcal{B}$ has access to a simulator $\mathcal{C}$. Suppose $\mathcal{C}$ executes the EUF-CMA game of Sign, denoted as $\mathrm{Exp}_{\mathcal{A}}^{\text{EUF-CMA}}$ which includes an oracle $\mathcal{O}_{sk_s}^{\text{Sign}}(\cdot)$, where $\mathcal{O}_{sk_s}^{\text{Sign}}(\cdot) : \omega \leftarrow \mathrm{Sign}(\mu, sk_s)$. $\mathcal{C}$ first generates $(pk_s, sk_s) \leftarrow \mathrm{KeyGen}(1^\kappa)$, then $\mathcal{B}$ obtains pk_s from $\mathcal{C}$.

Let us assume that $\mathcal{B}$ then generates other public keys in S as $S = \{pk_1, \ldots, pk_{s-1}, pk_{s+1}, \ldots, pk_m\}$ 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 $\pi = \{pk_s, pk_v\}$ where $s \neq v$ 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 $pk_i, i \in [1, m]$ 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: $\mathcal{C}$ 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 μ_i and queries for withdrawable signature with corresponding signer pk_s and specific verifier pk_v. If the signer of withdrawable signature is not pk_s, abort. Otherwise, $\mathcal{B}$ sets μ_i as the input of $\mathcal{C}$. $\mathcal{B}$ then asks the signing output of $\mathcal{C}$ as $\omega_i = \mathrm{Sign}(\mu_i, sk_s)$.

Upon reception of ω_i, $\mathcal{B}$ responds the signature query for the specific verifier pk_v chosen by $\mathcal{A}$ as follows:

- $\mathcal{O}^{\text{WSign}}_{sk_s,\gamma}(\cdot)$: With the output of $\mathcal{C}$, $\mathcal{B}$ can compute the withdrawable signature $\sigma_i \leftarrow \mathcal{O}^{\text{WSign}}_{sk_s,\gamma}(\cdot)$ for $\mathcal{A}$ with $\omega_i = (e_i, \mathbf{z}_i) = (H(\mu, h(\mathbf{y})), \mathbf{z}_i)$ as follows:
  1. Takes $r_i \xleftarrow{\$} \mathcal{S}_\eta$
  2. Computes $\sigma_i = (\sigma_{1,i}, \sigma_{2,i}, \sigma_{3,i}, \sigma_{4,i})$ as described in the algorithm.
- $\mathcal{O}^{\text{Confirm}}_{sk_s,\sigma,\gamma}(\cdot)$: $\mathcal{B}$ then queries for the signature of μ_i again to $\mathcal{C}$ and returns a corresponding $\omega_{s,i} = (e_{s,i}, \mathbf{z}_{s,i})$ instead. With ω_i, $\omega_{s,i}$ and σ_i, $\mathcal{B}$ can compute the confirmed signature $\tilde{\sigma}_i \leftarrow \mathcal{O}^{\text{Confirm}}_{sk_s,\sigma,\gamma}(\cdot)$ for $\mathcal{A}$ as follows:
  1. Takes $\mathbf{y}_{j,i} \xleftarrow{\$} \mathcal{S}_y^m$ and $e_{j,i} \xleftarrow{\$} \mathcal{S}_\eta$.
  2. Computes $\tilde{\sigma}_i = \{\delta_{1,i}, \delta_{2,i}, \delta_{3,i}, \delta_{4,i}\}$

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

Forgery: in this phase $\mathcal{B}$ returns a withdrawable signature σ^* for $\gamma^* = \{pk_s, pk_v\}$ on some μ^* that has not been queried before. Then σ^* could be transformed into σ^* under γ^* for signer pk_s correctly. After $\mathcal{A}$ transforms σ^* into $\tilde{\sigma}^*$, if $\tilde{\sigma}^*$ could not be verified through CVerify$(\mu^*, \gamma^*, \sigma^*, \tilde{\sigma}^*)$, abort. Otherwise, if $\tilde{\sigma}^* = (\delta_1^*, \delta_2^*, \delta_3^*, \delta_4^*)$ is valid, $\mathcal{B}$ then could obtain a forged signature ω^* for pk_s on μ^*.

Therefore, we can use $\mathcal{A}$ to break the unforgeability in the EUF-CMA model of our underlying signature scheme, 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 μ^* cannot be chosen for a signature query as it will be used for the signature forgery. Therefore, the probability of successful simulation is $q_{H_1}^{-1}$. □

Theorem 3.3. *If a signature scheme relies in the hardness of the decisional MLWE problem, then the associated withdrawable scheme defined using algorithm 9 and algorithm 10 is withdrawable in the random oracle model.*

Proof. Let us consider an adversary $\mathcal{A}$ who can (t, ε)-break the withdrawability of our withdrawable signature scheme. We now construct a simulator $\mathcal{B}$ able to solve the decisional MLWE problem.

Setup: let $\mathcal{B}$ set a challenge public set $\theta = \{pk_0, pk_1\}$ and the associated secret set $\vartheta = \{sk_0, sk_1\}$. We denote the signer key pair as (pk_b, sk_b) and (pk_{1-b}, sk_{1-b}) for the specific verifier.

Oracle simulation: in this phase $\mathcal{B}$ responds the oracle queries from $\mathcal{A}$ as follows:

- H query: the adversary $\mathcal{A}$ makes hash queries to $\mathcal{B}$, who simulates the hash function H as a random oracle.
- Signature query: here $\mathcal{A}$ outputs a message μ_i and queries its withdrawable signature for the corresponding signer pk_b and specific verified pk_{1-b}. Then $\mathcal{B}$ responds this query running $\mathcal{O}^{\text{WSign}}_{sk_s,\gamma}(\cdot) = (\sigma_{b,1}, \sigma_{b,2}, \sigma_{b,3}, \sigma_{b,4})$ and sets $\mathcal{M} \leftarrow \mathcal{M} \cup \{\mu_i\}$.

Challenge: in this phase the adversary $\mathcal{A}$ provides $\mathcal{B}$ with a message $\mu^* \notin \mathcal{M}$ and queries for the withdrawable signature for the corresponding signer and the specific verifier. Upon reception of μ^*, $\mathcal{B}$ computes the signature σ_b^* for $b \leftarrow \{0, 1\}$.

Guess: in this phase $\mathcal{A}$ outputs a guess bit b' for b. The simulator outputs *true* if $b' = b$; *false* otherwise.

Probability of breaking the withdrawability property: we observe that σ_0^* and σ_1^* are indistinguishable. Assuming the hardness of the decisional MLWE problem, the probability of guessing correctly b' is negligible.

Probability of successful simulation: this probability is 1 since there are no abortions in the simulation. □

3.2 An instantiation

From [3] 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 $B_h \coloneqq B_{60}$.

Algorithm 11 Dilithium

procedure KeyGen(1^κ)
  $\mathbf{A} \xleftarrow{\$} R_q^{k\times l}$, $(\mathbf{s}_1, \mathbf{s}_2) \xleftarrow{\$} \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$
  $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$
  return $(pk = (\mathbf{A}, \mathbf{t}), sk = (\mathbf{A}, \mathbf{t}, \mathbf{s}_1, \mathbf{s}_2))$
end procedure
procedure Sign(μ, sk)
  repeat
    $\mathbf{y} \xleftarrow{\$} S_{\gamma_1 - 1}^l$
    $\mathbf{w} = \mathrm{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$
    $c \in B_h = H(\mu, \mathbf{w})$
    $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$
  until $(\|\mathbf{z}\|_\infty < \gamma_1 - \beta) \wedge (\|\mathrm{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta)$
  return $\sigma = (\mathbf{z}, c)$
end procedure
procedure Verify(μ, pk, σ)
  $\mathbf{w}' = \mathrm{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$
  if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ **and** $c = H(\mu, \mathbf{w}')$ **then return** 1
  end if
end procedure

Taking the above algorithm as starting point and following the general construction in 3.1, algorithm 9 and algorithm 10, it is straightforward to set a withdrawable digital signature based in Dilithium.

Since Dilithium relies on the hardness of MSIS, Theorem 3.2 applies to prove that the proposal below is unforgeable under insider corruption. The hardness of the decisional MLWE problem makes Theorem 3.3 apply to prove withdrawability.

Below follows the withdrawable construction:

Algorithm 12 Dilithium-based withdrawable signature scheme

procedure KeyGen(1^κ)
  $\mathbf{A} \xleftarrow{\$} R_q^{k\times l}$
  $(\mathbf{s}_1', \mathbf{s}_2') \xleftarrow{\$} \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$, $(\mathbf{s}_1'', \mathbf{s}_2'') \xleftarrow{\$} \mathcal{S}_\eta^l \times \mathcal{S}_\eta^k$
  $\mathbf{t}_s = \mathbf{A}\mathbf{s}_1' + \mathbf{s}_2'$, $\mathbf{t}_v = \mathbf{A}\mathbf{s}_1'' + \mathbf{s}_2''$
  return $pk_s = (\mathbf{A}, \mathbf{t}_s), sk_s = (\mathbf{A}, \mathbf{t}_s, \mathbf{s}_1', \mathbf{s}_2')$, $pk_v = (\mathbf{A}, \mathbf{t}_v), sk_v = (\mathbf{A}, \mathbf{t}_v, \mathbf{s}_1'', \mathbf{s}_2'')$
end procedure

Algorithm 13 Dilithium-based withdrawable signature scheme *(continuation)*

procedure WSIGN(μ, sk_s, π)
  $\pi = (pk_s, pk_v)$
  repeat
    $\mathbf{y} \xleftarrow{\$} \mathcal{S}^{l}_{\gamma_1 - 1}$
    $\mathbf{w} = \mathrm{HighBits}(\mathbf{A}\mathbf{y}, 2\gamma_2)$
    $c \in B_h = H(\mu, \mathbf{w})$
    $\mathbf{z} = \mathbf{y} + c\mathbf{s}'_1$
  until $(\|\mathbf{z}\|_\infty < \gamma_1 - \beta) \wedge (\|\mathrm{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}'_{\mathbf{2}}, 2\gamma_2)\|_\infty < \gamma_2 - \beta)$
  $r = H(\mu, \mathrm{HighBits}((\mathbf{s}'_2)^T \mathbf{A}\mathbf{y}, 2\gamma_2))$
  $\mathbf{B} \xleftarrow{\$} R_q^{k \times k}$
  $\sigma_1 = \mathbf{A}\mathbf{z} - c\mathbf{t}_s$, $\sigma_2 = \mathbf{B}\mathbf{t}_v + \mathbf{A}(\mathbf{z} + rc\mathbf{s}'_1)$, $\sigma_3 = \mathbf{A}r\mathbf{s}'_1, \sigma_4 = \mathbf{B}\mathbf{A}$
  return $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
end procedure
procedure WSVERIFY$(\mu, sk_v, pk_s, \sigma)$
  $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$
  $c' = H(\mu, \mathrm{HighBits}(\sigma_1, 2\gamma_2))$
  if $\lfloor \sigma_2 \rceil = \lfloor \sigma_1 + c'\sigma_3 + c'\mathbf{t}_s + \sigma_4 \mathbf{s}''_1 \rceil$ **then return** 1
  end if
end procedure
procedure CONFIRM(μ, sk_s, π, σ)
  $\pi = (pk_s, pk_v), \sigma = (\sigma_1, \sigma_2, \sigma_3)$
  $r' = H(\mu, \mathrm{HighBits}((\mathbf{s}'_2)^T \sigma_1, 2\gamma_2))$
  repeat
    $\mathbf{y}_s \xleftarrow{\$} \mathcal{S}^{l}_{\gamma_1 - 1}$
    $c_s = H(\mu, \mathrm{HighBits}(\mathbf{A}\mathbf{y}_s, 2\gamma_2))$
    $\mathbf{z}_s = \mathbf{y}_s + c_s \mathbf{s}'_1$
  until $(\|\mathbf{z}_s\|_\infty < \gamma_1 - \beta) \wedge (\|\mathrm{LowBits}(\mathbf{A}\mathbf{y}_s - c\mathbf{s}'_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta)$
  repeat
    $\mathbf{y}_v \xleftarrow{\$} \mathcal{S}_{\gamma_1 - 1}$
    $c_v = H(\mathbf{t}_v, \mathrm{HighBits}(\mathbf{A}\mathbf{y}_v, 2\gamma_2))$
    $\mathbf{z}_v = \mathbf{y}_v + r' c_v \mathbf{s}'_1$
  until $(\|\mathbf{z}_v\|_\infty < \gamma_1 - \beta) \wedge (\|\mathrm{LowBits}(\mathbf{A}\mathbf{y}_v - c\mathbf{s}'_2, 2\gamma_2)\|_\infty < \gamma_2 - \beta)$
  $\delta_1 = c_s$, $\delta_2 = \mathbf{z}_s + r' c_s \mathbf{s}'_1$
  $\delta_3 = c_v$, $\delta_4 = \mathbf{z}_v$
  return $(\delta_1, \delta_2, \delta_3, \delta_4)$
end procedure
procedure CVERIFY$(\mu, \pi, \sigma, \tilde{\sigma})$
  $\pi = (pk_s, pk_v), \sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4), \tilde{\sigma} = (\delta_1, \delta_2, \delta_3, \delta_4)$
  $\delta'_1 = H(\mu, \mathrm{HighBits}(\mathbf{A}\delta_2 - \mathbf{t}_s\delta_1 - \sigma_3\delta_1, 2\gamma_2))$
  $\delta'_3 = H(\mathbf{t}_v, \mathrm{HighBits}(\mathbf{A}\delta_4 - \sigma_3\delta_3, 2\gamma_2))$
  if $(\delta_1 = \delta'_1) \wedge (\delta_3 = \delta'_3)$ **then return** 1
  end if
end procedure

Remark 3.1. *The identity $r = r'$ follows from the identity $c = c'$ (Section 3.1, [3]). The same arguments used in [3] apply in this work to justify why the function CVERIFY returns 1 correctly, therefore we refer the reader to Section 1.1 and Section 3.1 in [3] for the technical details.*

4 Conclusion and future research

This work uses the ideas in [6] to extend the Fiat-Shamir with aborts paradigm [8] with withdrawability and defines a general construction for withdrawable lattice-based digital signature schemes.

We demonstrated our approach by creating a withdrawable version of Dilithium, though the same principles could be applied to other signature schemes like HAETAE [2]. Our construction maintains the security properties of the underlying signature scheme while adding the ability to withdraw signatures when needed by the signer.

Several directions remain for future research, including optimizing our construction's efficiency and exploring additional features such as blindness, multiparty capabilities, or enhancing this construction with *extended withdrawability*, where rather than limiting verification to a specific entity, we can ensure the universal verifiability of the withdrawable signature by employing any signature scheme that can maintain signer ambiguity (this is work done in [7]). Another potential line of research is given by practical applications, particularly how these constructions could enhance quantum resistance in blockchain systems. The ability to withdraw signatures could prove valuable in blockchain environments where transaction revocation is desirable but traditionally difficult to implement.

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] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.