# SETLA: Signature and Encryption from Lattices

François Gérard and Keno Merckx

Université libre de Bruxelles
{fragerar,kmerckx}@ulb.ac.be

**Abstract.** In data security, the main objectives one tries to achieve are *privacy*, *data integrity* and *authentication*. In a public-key setting, *privacy* is reached through asymmetric encryption and both *data integrity* and *authentication* through signature. Meeting all the security objectives for data exchange requires to use a concatenation of those primitives in an encrypt-then-sign or sign-then-encrypt fashion. Signcryption aims at providing all the security requirements in one single primitive at a lower cost than using encryption and signature together. Most existing signcryption schemes are using ElGamal-based or pairing-based techniques and thus rely on the decisional Diffie-Hellman assumption. With the current growth of a quantum threat, we seek for post-quantum counterparts to a vast majority of public-key primitives. In this work, we propose a lattice-based signcryption scheme in the random oracle model inspired from a construction of Malone-Lee. It comes in two flavors, one integrating the usual lattice-based key exchange into the signature and the other merging the scheme with a RLWE encryption. Our instantiation is based on a ring version of the scheme of Bai and Galbraith as was done in ring-TESLA and TESLA♯. It targets 128 bits of classical security and offers a save in bandwidth over a naive concatenation of state-of-the-art key exchanges and signatures from the literature. Another lightweight instantiation derived from GLP is feasible but raises long-term security concerns since the base scheme is somewhat outdated.

## 1 Introduction

Enabling secure communication between two parties over a public channel is the most natural task one can ask from cryptography. Nevertheless, it is not necessarily obvious what is meant by secure. Since the channel is public, the first difficulty to overcome is to prevent unauthorized people from accessing the transiting data, that is to say, ensuring *privacy* of data. Privacy is enabled by using an encryption scheme. This scheme transforms a plaintext $m$ in an encrypted message $c$ called ciphertext. Using a secret key, the authorized receiver will be able to reverse this transformation but no polynomial time adversary should be able to retrieve any meaningful information on $m$ given only $c$. Having data secretly transmitted is clearly a milestone in securing communication but cannot be seen as the final answer. While answering all the real-life security threats of a communication system seems unfortunately not possible relying solely on mathematics, two common issues in practice are impersonation and data corruption. Hence, *data integrity* (data were not modified) and *authentication* (sender is actually the one they claim to be) should be guaranteed. In a public-key setting, these properties are both ensured by digital signatures which allow a signer to create a signature $\sigma(m)$ on a message $m$, verifiable by anyone knowing their public key.

Those cryptographic primitives have been developed somewhat independently and can be used separately, depending on the context. If the adversary is passive, i.e they can only read the channel but not write on it, encryption can be enough. If the secrecy of the message is not important, signing can be enough. Yet, in a situation in which an active adversary is present during a sensitive communication, *privacy*, *data integrity* and *authentication* must all be guaranteed at the same time. It is clearly possible to use encryption and signature together but it implies accepting the overhead of using two building blocks and forces a careful security analysis since concatenating two cryptographic primitives in a naive way can be dangerous.
In private-key cryptography, a lot of effort has been put toward the development of *authenticated encryption* schemes. The idea is to merge a symmetric encryption scheme with a message authentication code in a single block providing all the security properties listed above. This work gave rise to a dedicated workshop (DIAC) and a (currently ongoing) competition to establish a portfolio called CAESAR.
On the public-key side, the equivalent primitive is called *signcryption*. The goal of a signcryption scheme is to provide the security properties of both encryption and signature at a lower cost than concatenating them. The (academic) story started at CRYPTO in 1997 with the original paper of Zheng [34]. In this work, the author used a clever combination of ElGamal encryption and signature to create an efficient scheme leading a line of research aiming at formalizing, studying security and enhancing signcryption [15].
Unfortunately, the techniques used were based on the Diffie-Hellman (or RSA) assumption and their security would be compromised in case of the emergence of a large quantum computing power. Now, even though it is not clear when or even if a large enough quantum computer will be built, the importance of ensuring the security of communication in today's world is so critical that no risks can be taken and cryptography should be able to answer at the right moment. Designing and analyzing new cryptosystems takes time and trust can only be developed in the long run when the research community has put a huge amount of effort over the years to break it. Furthermore, the quantum threat could also be already present now if an adversary is currently recording long-term confidential encrypted data in order to decrypt it in the future. For those reasons, the post-quantum community is trying to push, as soon as possible, for development, both on theoretical and practical side, of quantum-resistant cryptography.

**Our contribution.** In this paper, we introduce a construction of a signcryption scheme in the Fiat-Shamir with aborts framework of Lyubashevsky based on the signature of Bai and Galbraith [7]. It is inspired from a Schnorr-like variant of the original work of Zheng [34] proposed by Malone-Lee [28]. We provide two versions of the scheme, both relying on the concept of sharing a key while signing and forwarding a symmetric encryption of the message under this key. The first one uses a usual lattice-based key exchange while the second one encrypts the key in a KEM fashion. Those two flavors of the scheme provide a tradeoff between efficiency and storage. The key exchange version is slower but uses less memory/bandwidth. We also provide a concrete instantiation with parameters chosen according to the methodology of [7] enabling correctness of the scheme and compares the gains of using this specific scheme instead of a naive concatenation of signature and key exchange.

**Previous work.** Signcryption has not been extensively studied in the post-quantum world yet. Some works on lattice-based schemes exist [22,33,24,32], however, they are all based on trapdoors and do not provide concrete practical instantiations. Our work is, to our knowledge, the first one studying signcryption using the Fiat-Shamir with aborts technique on lattices. We call the scheme SETLA (Signature and EncrypTion from LAttices) as a tribute to the TESLA family of signatures and to facilitate references to it in the text.

**Organization of the paper.** Section 2 formalizes signcryption and recalls basic tools needed in the construction. Section 3 presents the two versions of SETLA and points out their differences. Section 4 and 5 argue the security, correctness and efficiency of the schemes and section 6 concludes.

## 2 Preliminaries

### 2.1 Notations

Let us first explain which notations will be used through the paper. For the sake of simplicity and readability, they are similar to what is commonly used in the recent literature on the topic. We write polynomials with bold lower cases, e.g. $\mathbf{a} \in \mathbb{Z}[X]$. When a value $v$ is sampled from a distribution $\chi$, we use the notation $v \xleftarrow{r} \chi$. This notation is extended in a natural way to polynomials (of a given degree), $\mathbf{v} \xleftarrow{r} \chi$ means that the coefficients of $\mathbf{v}$ are all sampled independently from $\chi$. The uniform distribution over a set $S$ is written $\mathcal{U}(S)$. We use $v \xleftarrow{r} S$ as a shorthand for $v \xleftarrow{r} \mathcal{U}(S)$. For an odd $q$, we use the representative in $\left[\frac{-(q-1)}{2}, \frac{q-1}{2}\right]$ to identify cosets of $\mathbb{Z}_q$. We use the notation $\lfloor x \rceil_d = (x - [x]_{2^d})/2^d$ with $[x]_{2^d}$ the integer in $(-2^{d-1}, 2^{d-1}]$ congruent to $x$ modulo $2^d$ to denote the $d$-bit modular rounding of $x$ and also extend it to vectors.

### 2.2 Signcryption

A signcryption scheme is a cryptographic primitive aiming to act at the same time as encryption and signature on some data. The usual situation is that of a sender (a.k.a Alice) willing to send a message $m$ to a receiver (a.k.a Bob) while ensuring at the same time privacy, integrity and authentication. It is the public-key analog of authenticated encryption.

**Definition 1.** Formally, a signcryption scheme with message space $\mathcal{M}$ and signcryptext space $\mathcal{C}$ is a tuple $\mathit{\Gamma}_{\mathcal{M},\mathcal{C}} = (\texttt{ParamGen}, \texttt{KeyGenSender}, \texttt{KeyGenReceiver}, \texttt{Signcrypt}, \texttt{Unsigncrypt})$ composed of the five following algorithms:

- $\texttt{ParamGen}(\lambda)$: a randomized algorithm taking as input the security parameter $\lambda$ and outputting the parameters `params` of the system. We consider `params` as an implicit input of all the algorithms.
- $\texttt{KeyGenSender}()$: a randomized algorithm generating a key pair $(sk_a, pk_a)$ for the sender (Alice). We will call $sk_a$ the secret signing key and $pk_a$ the public verification key.

- `KeyGenReceiver()`: a randomized algorithm generating a key pair $(sk_b, pk_b)$ for the receiver (Bob). We will call $sk_b$ the secret decryption key and $pk_b$ the public encryption key.
- `Signcrypt`$(sk_a, pk_b, m)$: a randomized algorithm taking as input Alice's secret signing key $sk_a$, Bob's public encryption key $pk_b$, a message $m \in \mathcal{M}$ and outputting a signcryptext $C \in \mathcal{C}$.
- `Unsigncrypt`$(pk_a, sk_b, C)$: a deterministic algorithm taking as input Alice's public verification key $pk_a$, Bob's secret decryption key $sk_b$, a signcryptext $C \in \mathcal{C}$ and outputting a either a message $m \in \mathcal{M}$ if the signcryptext is valid or a failure symbol $\perp$.

It should be noted that, for efficiency and simplicity reasons, the two key generation algorithms can be merged in a single `KeyGen` algorithm outputting a key pair $(sk, pk)$ in which $sk$ act simultaneously as decryption and signing key and $pk$ as verification and encryption key.

**Non-repudiation.** There is no settled answer to the question of non-repudiation for a signcryption scheme. Indeed, since we want privacy of the message, it is not clear if a public verification mechanism is required. But if Alice can later repudiate the message in front of a judge, can we really call it a signature? The consensus is to set up a mechanism allowing Bob to generate a signature from the signcryptext at the price of revealing the message. Hence, if at some point Alice tries to be dishonest, he can create a publicly verifiable signature and present it to the judge. Hence, we extend the signcryption scheme with two optional algorithms:

- `SignExtract`$(pk_a, sk_b, C)$: a deterministic algorithm taking the same inputs as `Unsigncrypt` and outputting a publicly verifiable signature $\sigma(m)$.
- `PublicVerif`$(pk_a, \sigma(m))$: a deterministic algorithm taking as input the parameters of the system, the public key of Alice and a signature on $m$ and outputting 1 if $\sigma(m)$ is a valid signature on $m$, 0 otherwise.

In practice, the `SignExtract` algorithm can be merged with `Unsigncrypt` to output at the same time $m$ together with its signature $\sigma(m)$.

### 2.3 Ring Learning with Errors and Decisional Compact Knapsack

The ring learning with errors (RLWE) [27] problem is a variant of the learning with errors problem offering higher efficiency, both in memory and computing power at the price of a globally less understood security. It is parametrized by a positive integer $q$, an irreducible polynomial $p(X)$ of degree $n$ defining the polynomial ring $\mathcal{R} = \mathbb{Z}[X]/\langle p(X)\rangle$ together with its "mod $q$ version" $\mathcal{R}_q = \mathcal{R}/q\mathcal{R}$ and a narrow error distribution $\chi$ of zero mean over $\mathbb{Z}$. To enable efficient computation, we take the usual well-known ring $\mathcal{R}_q = \mathbb{Z}_q[X]/\langle X^n + 1\rangle$ with $q \equiv 1 \mod 2n$. Through the paper, elements in $\mathcal{R}_q$ will be seen alternatively as polynomials or vectors in $\mathbb{Z}_q$ together with negacyclic convolution as multiplication. It should be clear from context which view we use. To denote the set of elements with coefficients in the range $[-B, B]$ we write $\mathcal{R}_{q,[B]}$. We define the following problems, all believed to be hard, even for an adversary in possession of a large-scale quantum computer:

**Definition 2.** (*Search-RLWE*) for a secret $\mathcal{R}_q$ and a (polynomially bounded) number of samples $\mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i \in \mathcal{R}_q$ with $\mathbf{a}_i \overset{r}{\leftarrow} \mathcal{R}_q$ and $\mathbf{e}_i \in \mathcal{R}$ with coefficients sampled from $\chi$, find $\mathbf{s}$.

**Definition 3.** (*Decisional-RLWE*) for a secret $\mathbf{s} \in \mathcal{R}_q$ and a (polynomially bounded) number of samples $\mathbf{t}_i = \mathbf{a}_i \cdot \mathbf{s} + \mathbf{e}_i \in \mathcal{R}_q$ with $\mathbf{a}_i$ and $\mathbf{e}_i$ sampled as above, distinguish, with non-negligible probability, the distribution of the $\mathbf{t}_i$ from $\mathcal{U}(\mathcal{R}_q)$

**Definition 4.** (*Decisional Compact Knapsack*) In [19], the authors use a small parameters version of decisional RLWE called the decisional Compact Knapsack problem (DCK). In that version, the secret and error distributions are $\mathcal{U}(\{-1, 0, 1\})$ which means that the adversary receives tuples of the form $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{e})$ with $\mathbf{a} \overset{r}{\leftarrow} \mathcal{R}_q$ and $(\mathbf{s}, \mathbf{e}) \overset{r}{\leftarrow} (\mathcal{R}_{q,[1]} \times \mathcal{R}_{q,[1]})$, and must distinguish them from samples from $\mathcal{U}(\mathcal{R}_q \times \mathcal{R}_q)$. One can also naturally define the corresponding search problem.

### 2.4 RLWE Encryption

It is possible to construct an ElGamal-like CPA-secure encryption from RLWE. This ideal lattices version has been studied in the literature under the name RLWE encryption [27,14,31,23] and can be found in Figure 1. This scheme is really similar

Decryption key: $\mathbf{s} \overset{r}{\leftarrow} \chi$
Encryption key: $\mathbf{pk} = \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$ with $\mathbf{e} \overset{r}{\leftarrow} \chi$

```
RLWE Encrypt($\mathbf{pk}, m$):
1: $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3 \overset{r}{\leftarrow} \chi$
2: $\mathbf{c}_1 \leftarrow \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$
3: $\mathbf{c}_2 \leftarrow \mathbf{pk} \cdot \mathbf{y}_1 + \mathbf{y}_3 + \mathtt{Encode}(m)$
4: return $\mathbf{c}_1, \mathbf{c}_2$
```

```
RLWE Decrypt($\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}$):
1: $\mathbf{m} = \mathbf{c}_2 - \mathbf{c}_1 \cdot \mathbf{s} \approx \mathtt{Encode}(m)$
2: $m = \mathtt{Decode}(\mathbf{m})$
3: return $m$
```

```
Encode($m$):
1: for $i$ in $1...n$
2:     $\mathbf{m}[i] = m[i] \cdot \lfloor \frac{q-1}{2} \rfloor$
3: return $\mathbf{m}$
```

```
Decode($\mathbf{m}$):
1: for $i$ in $1...n$
2:     if $\mathbf{m}[i]$ in $\left[-\lceil \frac{q}{4} \rceil, \lceil \frac{q}{4} \rceil - 1\right]$
3:         $m[i] = 1$
4:     else
5:         $m[i] = 0$
6: return $m$
```

**Fig. 1.** RLWE Encryption

to ElGamal encryption, the difference lies in the fact that Bob will recover a noisy version of the ring element representing the message. This is why an encoding and a decoding algorithms are used. Basically the encoding function maps a bitstring to a polynomial by encoding one bit per coefficient. The bit $b$ in position $i$ is encoded as $\frac{q-1}{2} \cdot b$. A threshold decoder is applied to recover the message. If the coefficient at position $i$ is closer to $\lfloor \frac{q}{2} \rfloor$ than 0, we set the bit at the same position to 1, else, we set it to 0. Hence, has long as no coefficients are modified by more than $\frac{q}{4}$, the decoding algorithm will recover the correct message. Since $\chi$ is a narrow distribution of zero mean, this should happen with overwhelming probability.

### 2.5 Reconciliation mechanism

A common issue in learning with errors key exchanges [20,29,5,12,11] is that both parties end up with two values that are close to each other but not exactly the same.

It is due to the fact that, as in the encryption scheme, it is often made of ElGamal-like cryptography but with noisy elements. For example in the RLWE version, Alice eventually computes $\mathbf{ass}' + \mathbf{e}'\mathbf{s}$ while Bob has $\mathbf{ass}' + \mathbf{es}'$ (this can really be seen has them agreeing on a noisy version of $g^{ab}$ in Diffie-Hellman). Obviously, the key exchange cannot be considered successful if each party has a different value. The solution is to use a reconciliation mechanism deriving a common value from noisy data (a.k.a fuzzy extractor [16]). For example let us assume we work in $\mathbb{Z}_q$ with elements represented as values in $[-(q-1)/2, ..., (q-1)/2]$. Alice possesses $a$ and Bob $b = a+e$ for a small $e$. They could map their values to, say, $\{0, 1\}$ by partitioning $\mathbb{Z}_q$ in $S_0 = [-\lceil q/4\rceil, \lceil q/4\rceil - 1]$ and $S_1 = [\lceil q/4\rceil, (q-1)/2] \cup [-(q-1)/2, -\lceil q/4\rceil - 1]$ and outputting in which subset lies their value. This works well if $a$ and $b$ are close to 0 or $q/2$ but can fail if they are close to $q/4$ or $-q/4$. To overcome that possibility, Alice can send a reconciliation value $v \in \{0, 1\}$ indicating if $a$ is in $[0, \lceil q/4\rceil] \cup [-(q-1)/2, \lfloor -q/4\rfloor]$ or $[\lceil q/4\rceil + 1, (q-1)/2] \cup [\lceil -q/4\rceil - 1, -1]$. The value $v$ thus conveys no information about the partition in which $a$ lies but helps Bob to reconcile on the correct value using his knowledge of $b$. This approach has been used by Peikert in [29], by applying the above technique separately on each coefficients of an element of $\mathcal{R}_q$ (with a slight modification dealing with the fact that an odd $q$ cannot be split in two equal parts).

Clearly, any reconciliation technique has an error tolerance threshold over which agreement cannot be reached. To increase the threshold, a possibility is to use *multiple* values to agree on a common bit. The motivation is that polynomials used in RWLE-based are often of size 512 or 1024 to ensure the security of the underlying lattice problem while symmetric secrets of bit size 256 appear to be enough, even in a post-quantum world. Hence we should use mappings from $\mathbb{Z}_q^n$ to $\{0, 1\}^{256}$ with $n \in \{512, 1024\}$. Of course mappings for higher $n$ or larger symmetric keys can be used but in practice, those parameters are good enough. For the key exchange version of our construction, we borrow the notations from NewHope [5]. In their paper, they show how to agree on a $n$ bit key from either a polynomial of degree $2n$ or $4n$. The description of their whole reconciliation mechanism is quite tedious and takes a lot of space. Hence we redirect the interested reader to their paper for a full explanation and analysis. By borrowing their notations, we mean that we will use two algorithms $\texttt{HelpRec}(\mathbf{x})$ and $\texttt{Rec}(\mathbf{x}', \mathbf{hr})$ (as defined below) but that the scheme is unaffected by how those functions work under the hood, they could implement any reconciliation mechanism.

- $\texttt{HelpRec}(\mathbf{x})$ taking as input a ring element and outputting a reconciliation vector $\mathbf{hr}$
- $\texttt{Rec}(\mathbf{x}', \mathbf{hr})$ taking as input a ring element and a reconciliation vector and outputting a symmetric key $K$

If $\mathbf{x}$ and $\mathbf{x}'$ are close to each other (the distance between their coefficients is small), the output of $\texttt{Rec}(\mathbf{x}, \mathbf{hr})$ and $\texttt{Rec}(\mathbf{x}', \mathbf{hr})$ are the same.

### 2.6 Fiat-Shamir lattice-based signatures

In [25], Lyubashevsky presented the Fiat-Shamir with aborts technique to construct digital signatures in the random oracle model. It spawned a long line of research enabling practical instantiations of lattice-based signatures such as BLISS, Dilithium or qTESLA. The construction is following the same pattern as Schnorr signatures. It starts by defining an identification scheme with a $\Sigma$ protocol and then use the generic Fiat-Shamir transformation to create a signature scheme. As Schnorr identification protocol acts as a zero-knowledge proof for a discrete logarithm, Lyubashevsky's protocol acts as a zero-knowledge proof for a LWE/SIS instance. In figure 2 we informally recall such a signature instantiated with RLWE. The public parameter $\mathbf{a}$ is uniform in the ring, $E$ and $Y$ denote two small distributions (with $Y$ significantly larger then $E$ but still small in comparison to the modulus $q$). The crucial difference between this signature and Schnorr's scheme is the rejection sampling step performed after generating the $\mathbf{z}_i$. It is required in order to avoid leakage of the secret over the release of signatures. Indeed, since the $\mathbf{y}_i$ are not sampled uniformly over the whole ring, they do not act as a one-time pad perfectly hiding the value $\mathbf{s} \cdot \mathbf{c}$. The rejection sampling procedure will reject some signatures such that the output distribution of the $\mathbf{z}_i$ is statistically independent of the secret and hence do not reveal anything about it.

Public parameter: $\mathbf{a}$
Secret key: $\mathbf{s}, \mathbf{e} \overset{r}{\leftarrow} E$
Public key: $\mathbf{t} \leftarrow \mathbf{a} \cdot \mathbf{s} + \mathbf{e}$

```
Sign(s, m):
1: do
2:     $\mathbf{y}_1, \mathbf{y}_2 \overset{r}{\leftarrow} Y$
3:     $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m)$
4:     $\mathbf{z}_1 \leftarrow \mathbf{s} \cdot \mathbf{c} + \mathbf{y}_1,\ \mathbf{z}_2 \leftarrow \mathbf{e} \cdot \mathbf{c} + \mathbf{y}_2$
5: while Rejected($\mathbf{z_1 z_2}$)
6: return $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$
```

```
Verify($\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{t}, m$):
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z} - \mathbf{t} \cdot \mathbf{c}$
2: return 1 if $\mathbf{c} = H(\mathbf{v})$ and both $\mathbf{z}_i$ are small else 0
```

**Fig. 2.** Generic RLWE signature scheme in the Fiat-Shamir with aborts framework

## 3 Lattice-based Signcryption Schemes

Hereunder, we describe both versions of our scheme. The discussion in section 5 will only be made for the first version for the sake of brevity but the analysis is basically the same. In the following, when we talk about lattice signatures, we mean lattice-based signatures obtained from the Fiat-Shamir transformation.

### 3.1 SETLA-KEX Signcryption

First we describe how to integrate encryption into a lattice signature, following the steps of the ElGamal modification of Zheng. From a high-level point of view, the idea of the original signcryption scheme is to sign a message with an ElGamal signature and to realize a non-interactive Diffie-Hellman ephemeral key exchange at the same time reusing the"commit" value of the signature. The gain in efficiency comes from the fact that the same operation is used in both primitives. Subsequently, the message is symmetrically encrypted with the key derived from the exchange and forwarded to the receiver. While the first scheme of Zheng was not directly translatable in a lattice version, one of its derivative due to Malone-Lee [28] caught our attention. Indeed, even though its primary advantage over Zheng in pre-quantum cryptography was to enable non-interactive non-repudiation, namely that Bob alone can create a valid signature from a signcryptext, the second difference is that it is based on Schnorr signature. The lattice-based signatures schemes coming from identification schemes through Fiat-Shamir transform being Schnorr-like [26,7,17,3], this is where post-quantum can meet signcryption. We use a ring version of the signature proposed by Bai and Galbraith as a base to construct the scheme but it can be generalized to most signatures derived from the original work of Lyubashevsky as long as the parameters offer at the same time security and correctness for the key reconciliation. We actually also have a construction based on GLP working out of the box with the original parameters which is omitted in this conference version of the paper for the sake of compactness.

**Algorithm 1** SETLA Key generation

```
Input: Public parameter $\mathbf{a_1}, \mathbf{a_2}$
Output: Key pair $\mathbf{pk} = (\mathbf{t_1}, \mathbf{t_2}), \mathbf{sk} = (\mathbf{s}, \mathbf{e_1}, \mathbf{e_2})$
1: $\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2 \leftarrow \mathcal{R}_{q,[1]}$
2: $\mathbf{t}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{s} + \mathbf{e}_1$, $\mathbf{t}_1 \leftarrow \mathbf{a}_2 \cdot \mathbf{s} + \mathbf{e}_2$
3: return $\mathbf{pk} = (\mathbf{t}_1, \mathbf{t}_2), \mathbf{sk} = (\mathbf{s}, \mathbf{e}_1, \mathbf{e}_2)$
```

**Key generation.** (Algorithm 1) The key generation is simple and straightforward for a scheme using ideal lattices cryptography. It uses some public parameters $\mathbf{a}_1, \mathbf{a}_2$ shared among all users and output two RLWE samples $\mathbf{pk} = (\mathbf{t}_1, \mathbf{t}_2)$ together with a secret polynomial $\mathbf{s}$. The error and secret distributions are the same and output a polynomial with uniform coefficients in $\{-1, 0, 1\}$. The choice of such a distribution is suboptimal in terms of security since it has low variance and its special structure may enable specialized attacks but has been made for reasons that will come clear later. Note that in the context of signcryption, both Alice and Bob will run the key generation procedure to retrieve their keys since two key pairs are used in the full signcrypt/unsigncrypt procedure. In the following, we use subscripts (e.g. $\mathbf{pk}_a = (\mathbf{t}_{a,1}, \mathbf{t}_{a,2})$ ) to differentiate them.

**SETLA-KEX Signcrypt.** (Algorithm 2) The signcrypt procedure contains three interleaved parts: signature, key exchange and encryption. The signature follows the structure of [1,8] as a Fiat-Shamir signature from a $\Sigma$ protocol. First, a commitment consisting of two rounded polynomials $\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d$, $\lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$ depending on a masking value $\mathbf{y}$ is computed. Then, an unpredictable challenge $\mathbf{c}$ is retrieved by simulating a verifier with a random oracle $H$ taking inputs depending on the commitment. Finally, the response consists of a polynomial of the form $\mathbf{z} = \mathbf{s} \cdot \mathbf{c} + \mathbf{y}$. Note that for reasons related specifically to signcryption schemes, the random oracle should take as input a symmetric key $K$ and both public identities. If the key was not included in the input, the adversary playing a signcryption specific CCA2 game would easily be able to distinguish between two messages $m_0, m_1$ by computing both $H(., m_i, ., .)$ and verify the equality with $\mathbf{c}$. Having the public identities in the hash is a common practice in signcryption schemes to prove security in advanced

**Algorithm 2** SETLA-KEX Signcrypt

**Input**: Public parameters $\mathbf{a_1}, \mathbf{a_2}$, Bob's public key $\mathbf{pk}_b$, Alice's keys $(\mathbf{s}_a, \mathbf{e}_{a,1}, \mathbf{e}_{a,2}, \mathbf{pk}_a)$, a message $m$, random oracle $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$, symmetric encryption algorithm $E$
**Output**: a signcryptext of $m$: $C = (\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$

```
 1: do
 2:     $\mathbf{y}, \mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
 3:     $\mathbf{v} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}' = \mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y} + \mathbf{e}_{b,1} \cdot \mathbf{y} + \mathbf{y}'$
 4:     $\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v})$
 5:     $K \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r})$
 6:     $\mathbf{c} \leftarrow H(\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 7:     $\mathbf{z} \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}$
 8:     $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
 9: while not( $\mathbf{z}$ in $\mathcal{R}_{q,[B-\omega]}$ and $\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d = \lfloor \mathbf{w}_1 \rceil_d$ and $\lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d = \lfloor \mathbf{w}_2 \rceil_d$ )
10: $\mathcal{E} \leftarrow E_K(m)$
11: return $\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r}$
```

models [15].
The key exchange part is performed by deriving a secret value $K$ from a noisy version of $\mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y}$. Alice cannot find the exact value since it means she would know Bob's secret key but she can find an approximate value from Bob's public key by computing $\mathbf{t}_{b,1} \cdot \mathbf{y} = \mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y} + \mathbf{e}'_{b,1} \cdot \mathbf{y} \approx \mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y}$. This is exactly the technique employed in lattice-based key exchanges such as NewHope. The efficiency gain comes from the fact that Bob will later be able to retrieve an approximation version of $\mathbf{a}_1 \cdot \mathbf{y}$ *without* sending him any other ring element than the polynomials computed in the signature $(\mathbf{z}, \mathbf{c})$. As in [12,5], Alice gets a symmetric key by applying a reconciliation procedure on the noisy shared value. The last part is straightforward, now that a key is available, a symmetric cipher $E$ is used to encrypt the data.
Finally, Alice outputs the signature $(\mathbf{z}, \mathbf{c})$, the symmetric ciphertext $\mathcal{E}$ and a small reconciliation vector $\mathbf{r}$. It means that the message was at the same time encrypted and authenticated in an asymmetric manner with only the overhead of sending a symmetric ciphertext (obviously we need to send something at least as long as the message for encryption) and a small reconciliation vector on the top of the signature.

**Algorithm 3** SETLA-KEX Unsigncrypt

**Input**: Public parameters $\mathbf{a_1}, \mathbf{a_2}$, Bob's keys $(\mathbf{s}_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$, random oracle $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$, symmetric encryption algorithm $E$
**Output**: A message $m$ or failure symbol $\perp$

```
1: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{z} - \mathbf{t}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{z} - \mathbf{t}_{a,2} \cdot \mathbf{c}$
2: $K \leftarrow \texttt{Rec}(\mathbf{w}_1 \cdot \mathbf{s}_b, \mathbf{r})$
3: $m \leftarrow E_K^{-1}(\mathcal{E})$
4: return $m$ if $\mathbf{c} = H(\lfloor \mathbf{w}_1 \rceil_d, \lfloor \mathbf{w}_2 \rceil_d, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ and $\mathbf{z} \in \mathcal{R}_{q,[B-\omega]}$ else $\perp$
```

**SETLA-KEX Unsigncrypt.** (Algorithm 3) The goal of this algorithm is to allow Bob to find the secret key to decrypt the symmetric cipher and at the same, to

provide authentication of the message through a signature.
First, Bob will recover the commitment part of the signature by rounding the values $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{z} - \mathbf{t}_{a,1} \cdot \mathbf{c}$ and $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{z} - \mathbf{t}_{a,2} \cdot \mathbf{c}$. Without rounding, $\mathbf{c}$ would have been different since Alice queried the random oracle with $\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d$ and $\lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$. The difference with the original signature scheme is that Bob must now find the key $K$ and the message in order to verify the hash value. To recover it, he shall use the reconciliation vector $\mathbf{r}$ with an approximate version of $\mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y}$. Such a value can be found by computing the product $\mathbf{w}_1 \cdot \mathbf{s}_b = \mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y} + \mathbf{e}_{a,1} \cdot \mathbf{s}_b \cdot \mathbf{c} \approx \mathbf{a}_1 \cdot \mathbf{s}_b \cdot \mathbf{y}$. Once the message is decrypted, Bob verifies the signature by checking the size of $\mathbf{z}$ and the hash value. He outputs the message if everything is correct and a failure symbol otherwise.

**SETLA-KEX SignExtract.** This scheme also inherits the capability to perform a signature extraction from the signcryption of Malone-Lee. It is a simple transformation of the unsigncrypt procedure and can be found in Appendix B.

## 3.2 SETLA-KEM Signcryption

Now, we describe the second version of the scheme based on key encapsulation instead of direct key exchange. The approach is similar to the one used in NewHope-Simple [4] or Kyber [10]. The high-level perspective is now to perform a noisy El-Gamal encryption of a chosen key during signature instead of noisy Diffie-Hellman. While in NewHope-Simple the goal of the new approach is to make the protocol simpler by getting rid of the reconciliation mechanism but not really to enhance the scheme, here, using an encryption based method leads to better performances in terms of speed and can enable parallelism, at the cost of a significantly bigger signcryptext.

**Algorithm 4** SETLA-KEM Signcrypt

**Input**: Public parameters $\mathbf{a}_1, \mathbf{a}_2$, Bob's public key $\mathbf{pk}_b$, Alice's key $(\mathbf{s}_a, \mathbf{e}_{a,1}, \mathbf{e}_{a,2}, \mathbf{pk}_a)$, a message $m$, random oracle $H : * \rightarrow \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$, symmetric encryption algorithm $E$
**Output**: a signcryptext of $m$: $C = (\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E})$

```
 1: $K \xleftarrow{r} \{0,1\}^{256}$
 2: do
 3:     $\mathbf{y} \xleftarrow{r} \mathcal{R}_{q,[B]}$
 4:     $\mathbf{c} \leftarrow H(\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 5:     $\mathbf{z} \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}$
 6:     $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
 7: while not( z in $\mathcal{R}_{q,[B-\omega]}$ and $\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d = \lfloor \mathbf{w}_1 \rceil_d$ and $\lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d = \lfloor \mathbf{w}_2 \rceil_d$ )
 8: $\mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
 9: $\mathbf{x} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}' + \texttt{Encode}(K)$
10: $\mathcal{E} \leftarrow E_K(m)$
11: return $\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E}$
```

**SETLA-KEM Signcrypt.** (Algorithm 4) In the same way as before, one can find three phases: signature, key encapsulation and symmetric encryption. The signature is now more isolated and almost exactly the same as in [8], the small difference

is that the random oracle (as in the KEX version) takes as input the message, the symmetric decryption key and the public identities.
The key encapsulation part is a RLWE encryption of a randomly sampled key $K$. Such an encryption consists of two ciphertexts $\mathbf{c}_1 = \mathbf{a}\cdot\mathbf{y}_1+\mathbf{y}_2$ and $\mathbf{c}_2 = \mathbf{pk}_b\cdot\mathbf{y}_1+\mathbf{y}_3$. Basically, $\mathbf{c}_2$ is the message masked with a ring element depending on the public-key looking random under the decisional-RLWE assumption and $\mathbf{c}_1$ is a value allowing the owner of $\mathbf{s}_b$ to remove the mask without conveying any (computable) information on $\mathbf{y}_1$ under the search-RLWE assumption. Here we gain efficiency by having the value $\lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d$ acting at the same time as the commitment of the signature and the $\mathbf{c}_1$ part of the encryption scheme.
Globally, the KEM version is adding a lot of overhead on the size of the signcryptext which is problematic since this is where we are looking for efficiency. Nevertheless, we see two advantages of using encryption instead of key exchange.First, the scheme is faster because it has less computation in the rejection sampling loop (which can run several times depending on the parameters) and we can now parallelize the symmetric encryption algorithm. Indeed, in the KEX version, the key depends on $\mathbf{y}$ and was not known until the end of the rejection sampling procedure, hence, everything had to be sequential and a multiplication with $\mathbf{t}_{b,1}$ had to be done at each iteration. Now, the symmetric encryption can start at the same time as the rejection sampling. It is fair to say that in general symmetric operations are lightweight in comparison to polynomial multiplication. Nevertheless, if a really large message has to be encrypted, say such that $E_K(m)$ takes as long as the **do**...**while** loop, the saving becomes non-negligible. Obviously, this argument only makes sense if the rejection sampling procedure itself is not affected by the size of the message. One solution would be to pre-hash the message before the loop and only inject this hash in the random oracle. Actually this issue is not specific to signcryption, all the signature schemes using rejection sampling would be badly affected by a really long message if the hash function cannot restart from its previous sate. Hence, in this case, hashing the message once before would save some computation. This small modification could be done in the KEX version as well as in existing Fiat-Shamir lattice-based signatures.
Second, depending on the parameters, if the correctness during reconciliation is a real issue, having the key encoded as a polynomial with coefficients in $\{0, \frac{q-1}{2}\}$ is optimal for the since they are at"maximum distance" in $\mathbb{Z}_q$. Also, because the symmetric key needed being often smaller than the encoding polynomial, having control over the value eases the process of embedding an error-correcting code in the extra space. Even though in the current state of affairs and with the parameters proposed in the next section the KEM version would not outperform neither the KEX version nor the naive concatenation of efficient schemes, we think the construction may be of interest in some contexts.

**SETLA-KEM Unsigncrypt.** (Algorithm 5) The unsigncrypt algorithm follows in the obvious manner. Bob retrieves the $\mathbf{c}_1$ part of the RLWE encryption from the signature and run the decryption algorithm to find the key. Then, he decrypts the symmetric ciphertext and verifies the signature.

**Algorithm 5** SETLA-KEM Unsigncrypt

**Input**: Public parameter $\mathbf{a}_1, \mathbf{a}_2$, Bob's key $(\mathbf{s}_b, \mathbf{s}'_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$, random oracle $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$, symmetric encryption algorithm $E$

**Output**: A message $m$ or failure symbol $\perp$

```
1: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{z} - \mathbf{t}_{a,1} \cdot \mathbf{c}$
2: $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{z} - \mathbf{t}_{a,2} \cdot \mathbf{c}$
3: $K \leftarrow \texttt{Decode}(\mathbf{x} - \mathbf{w}_1 \cdot \mathbf{s}_b)$
4: $m \leftarrow E^{-1}(\mathcal{E})$
5: return $m$ if $\mathbf{c} = H(\mathbf{v}, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ and $\mathbf{z} \in \mathcal{R}_{q,[k-\omega]}$ else $\perp$
```

## 4 Security arguments

The security aspects of interest for signcryption are unforgeability and privacy. The construction combining both a signature scheme using the Fiat-Shamir heuristic and a public key encryption scheme, we argue the security by using the forking lemma [30] and a standard hybrid argument. This does not provide a formal argument of security in a signcryption specific security model since it does not consider the primitive as an encryption *and* a signature but rather successively as an encryption *or* a signature. Nevertheless, having both unforgeability and privacy of the two underlying schemes is a good pointer toward the fact the design is sound. Providing a formal argument in advanced signcryption models is a tedious task (see [6]) and we do not attempt to do so here.

### 4.1 Unforgeability

The underlying signature of the signcryption scheme is the ring variant of the Bai-Galbraith signature which is itself a derivative of the original proposal of Lyubashevsky [26]. The full security argument can be found in [7] but the idea is to use the forking lemma to get two different signatures for the same commitment that would allow us to solve a special SIS instance. We use the adversary to get two forgeries $\mathbf{z}, \mathbf{c}$ and $\mathbf{z}', \mathbf{c}'$ for different random oracles but the same random tape (hence the same $\mathbf{y}$). We have (providing the argument for only one RLWE sample instead of two as in the signature for the sake of simplicity) $\lfloor \mathbf{a} \cdot \mathbf{z} - \mathbf{t}_a \cdot \mathbf{c} \rceil_d = \lfloor \mathbf{a} \cdot \mathbf{z}' - \mathbf{t}_a \cdot \mathbf{c}' \rceil_d = \lfloor \mathbf{a} \cdot \mathbf{y} \rceil_d$. This means that for some small $\mathbf{e}$, $\mathbf{a} \cdot \mathbf{z} - \mathbf{t}_a \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{z}' - \mathbf{t}_a \cdot \mathbf{c}'$ and thus, with $\mathbf{t}_a = \mathbf{a} \cdot \mathbf{s}_a + \mathbf{e}_a$, $\mathbf{a} \cdot (\mathbf{z} - \mathbf{z}' - \mathbf{s}_a \cdot \mathbf{c} + \mathbf{s}_a \cdot \mathbf{c}') + (\mathbf{e}_a \cdot (\mathbf{c}' - \mathbf{c}) + \mathbf{e}) = 0$. As pointed in [7] section 4.2, (if $\mathbf{z} - \mathbf{z}' - \mathbf{s}_a \cdot \mathbf{c} + \mathbf{s}_a \cdot \mathbf{c}'$ and $\mathbf{e}_a \cdot (\mathbf{c}' - \mathbf{c}) + \mathbf{e}$ are non-zero) we have found a solution to the SIS instance. This argument still holds for the signcryption scheme.

### 4.2 Confidentiality

We argue the confidentiality of the scheme with a sequence of games showing semantic security under the **DCK** assumption in the random oracle model. We model the adversary as a tuple of two algorithms $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the first choosing messages for the game according to the public keys and the second trying to guess which one was signcrypted. The encryption scheme $E$ is seen as an ideal primitive. The sequence of games for the KEX version can be found in Figure 4. Games for the KEM version are really similar and can be found in Appendix A.

Game 0:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $\mathbf{y}, \mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
4: $\mathbf{v} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}'$
5: $\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v})$
6: $K \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r})$
7: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
8: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
9: $\mathbf{z} \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}$
10: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
11: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
12: if $\mathbf{z}$ not in $\mathcal{R}_{q,[B-\omega]}$, goto 3
13: $\mathcal{E} \leftarrow E_K(m)$
14: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$
15: return $\hat{b}$
```

Game 1:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $\mathbf{y}, \mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
4: $\mathbf{v} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}'$
5: $\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v})$
6: $K \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r})$
7: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
8: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
9: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
10: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
11: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
12: with probability P, goto 3
13: $\mathcal{E} \leftarrow E_K(m)$
14: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$
15: return $\hat{b}$
```

Game 2:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $\mathbf{y}, \mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
4: $\mathbf{a}' \xleftarrow{r} \mathcal{R}_q$
5: $\mathbf{v} \leftarrow \mathbf{a}' \cdot \mathbf{y} + \mathbf{y}'$
6: $\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v})$
7: $K \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r})$
8: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
9: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
10: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
11: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
12: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
13: with probability P, goto 3
14: $\mathcal{E} \leftarrow E_K(m)$
15: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$
16: return $\hat{b}$
```

Game 3:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $\mathbf{y}, \mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
4: $\mathbf{v} \xleftarrow{r} \mathcal{R}_q$
5: $\mathbf{r} \leftarrow \texttt{HelpRec}(\mathbf{v})$
6: $K \leftarrow \texttt{Rec}(\mathbf{v}, \mathbf{r})$
7: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
8: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
9: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
10: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
11: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
12: with probability P, goto 3
13: $\mathcal{E} \leftarrow E_K(m)$
14: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$
15: return $\hat{b}$
```

**Fig. 3.** Sequence of games for the KEX version

*Game 0:* Game 0 is the usual CPA game against SETLA, the adversary chooses two messages $m_0, m_1$ and tries to guess which one was signcrypted.

*Game 1:* By virtue of the rejection sampling performed during signcryption, the output distribution of $\mathbf{z}$ should be exactly the same as a uniform over $\mathcal{R}_{q,[k-\omega]}$. Hence, we can replace $\mathbf{z}$ by random elements over this range without modifying the view of the adversary.

*Game 2:* Using the **DCK** assumption, we can replace the public key of Bob by a random element in $\mathcal{R}_q$ without being detected by the polynomial time adversary.

*Game 3:* In game 3, we use the same argument again to replace $\mathbf{v}$ by a uniformly random value (and hence $K$ is uniform as well by design of `Rec`).

In conclusion, using the fact that both $H(.)$ and $E(.)$ are modeled as ideal primitives and that $H$ takes one random unknown to the adversary value ($K$) uncorrelated to the message, they do not reveal anything about their inputs. Hence, the values given to $\mathcal{A}_2$ looks all random and independent from the messages. Thus, the adversary cannot guess which one was signcrypted.

### 4.3 ROM vs QROM

It is known that the forking lemma cannot be used if the adversary has quantum access to the random oracle. This issue has been recently discussed a lot in the literature on lattice-based signatures and some schemes took it into consideration ([3]) while others ignored it to focus on performances ([18]). Since our goal is to improve practicability, we decided to stick to the classical ROM. Having a classical reduction is essential to claim provable security but the implications of the QROM issue in practice are not clear enough to require QROM security for all the schemes. We redirect the interested reader to [21,9] for more details.

## 5 Analysis and parameters

### 5.1 Parameter selection

To select the parameters, we followed the methodology described in [7]. It allows the scheme to reduce to worst-case problems on ideal lattices. Be careful that it does not mean that the parameters are chosen such that the problem we reduce to is hard (since the proofs are non-tight) but merely that the reduction works. This is a common practice and as pointed in [21], it is reasonable to assume that it does not create any security issue. Our parameters can be found in Table 1. The dimension $n$ has been set to 1024 because it seems to be the minimal lattice dimension such that RLWE is hard with such a small error distribution. The value $m$ represents the number of rows of the LWE instance written in matrix form. Here it means that we work with two polynomials (which are explicit in the construction) since $m = 2n$. The entropy of the output of the random oracle is given by $\kappa = \log_2\left(2^\omega\binom{n}{\omega}\right)$, that is to say the logarithm of the cardinality of the set $\{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$. The modulus $q = 2^{25} - 2^{12} + 1$ is a prime such that $q \equiv 1 \mod 2n$ .The parameters $d$ and $B$ are chosen such that the rejection probability of the signature is not too high in order to keep the runtime reasonable and $q^{m-n} \geq \frac{2^{(d+1)m+\kappa}}{(2B)^n}$.
To assess the security of the scheme, we used the *LWE-Estimator* tool of Albrecht and al. [2]. We ran the estimator with the following command:

```
n = 1024; q = 33550337;
stddev = sqrt(2/3); alpha = alphaf(sigmaf(stddev), q)
_ = estimate_lwe(n, alpha, q,
          secret_distribution=(-1,1),reduction_cost_model=BKZ.sieve)
```

It estimates a bit security of 131 against the most efficient attack. The estimation of the hardness of directly forging the signature without recovering the private key has been made in the same way as in [13]. It gave overwhelming results, which is not a surprise since the parameters are a harder version of the most secure parameters set of [19].

| $n$ | $m$ | $\omega$ | $d$ | $B$ | $q$ | $\kappa$ |
|---|---|---|---|---|---|---|
| 1024 | 2048 | 16 | 15 | $2^{15}$ | $33550337 \approx 2^{25}$ | 131 |

**Table 1.** Parameters targeting 128 bits of classical security.

## 5.2 Failure probability

The main bottleneck of the signcryption scheme is the correctness regarding decryption. Indeed, as in a lot of RLWE-based protocols, the two parties end up with two ring elements close to each other but not exactly the same. In our case, the difference between the value of Alice and the value of Bob is $\Delta_{ab} = \mathbf{e}_{b,1} \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c} \cdot \mathbf{s}_b + \mathbf{y}'$. While in those schemes the parameters are chosen in order to get correctness with overwhelming probability, we face here a strong constraint which is that the parameters should also be compatible with the signature scheme. In their case, the $\mathbf{y}$ is coming from the error distribution and hence is very small. In our case, it is the masking polynomial for the signature $\mathbf{s} \cdot \mathbf{c} + \mathbf{y}$ which should be significantly larger. Obviously, one strategy to reduce the norm of $\Delta_{ab}$ is to reduce $B$. This would give better results for correctness but unfortunately decrease the speed of the scheme since the rejection sampling loop would have to run longer to find a small enough $\mathbf{z}$. This is the reason why we decided to use such a small distribution for the secret and the errors. Of course, it is possible to work with slightly larger distributions in a more specific context in which correctness matters less.

We now provide an analysis of the failure probability for the KEX-version. Using the reconciliation method of [5], the KEX-Unsigncrypt algorithm recovers the correct key if $\|\Delta_{ab}\|_\infty < \lfloor \frac{3q}{8} \rfloor$ (actually the requirement is that the $\ell_1$ norm of packs of 4 coefficients should be smaller than $\lfloor \frac{3q}{4} \rfloor - 2$). In the following, we write $(\mathbf{p})_i$, to denote the $i$-th coefficient of a polynomial $\mathbf{p}$.
We shall bound the magnitude of one coefficient $(\Delta'_{ab})_i = (\mathbf{e}'_{b,1} \cdot \mathbf{y})_i$. Since the polynomial product is computed modulo $\langle X^n + 1 \rangle$ and all distributions are symmetric, one such coefficient is the result of a sum of $n$ products between a coefficient of a polynomial in $\mathcal{R}_{q,[1]}$ and a polynomial in $\mathcal{R}_{q,[B]}$.
Let $S \sim \mathcal{U}(\{-1, 0, 1\})$ and $Y \sim \mathcal{U}([-B, B])$ be random variables, we denote their product $SY$. Each coefficient of $\Delta'_{ab}$ is the sum of $n$ samples from $SY$, hence $(\Delta'_{ab})_i \sim \sum_{i=1}^{n} (SY)_i$. Fortunately, computing the exact distribution $SY$ is easy:

$$\mathbb{P}[SY = 0] = \frac{2B+3}{6B+3}$$

$$\mathbb{P}[SY = z \mid z \in [-B, B] \backslash \{0\}] = \frac{2}{6B+3}$$

Since the value of $B$ is reasonable, to find the distribution of $\Delta'_{ab}$, one could hope to compute $\log(n)$ time the convolution of the distribution with itself. Unfortunately, this approach failed to give accurate results because of numerical stability issues. Instead, as in [5], we use the Chernoff-Cramer inequality to bound the sum of the random variables.

**Chernoff-Cramer inequality** Let $\chi$ be a distribution over $\mathbb{R}$ and let $X_1, \ldots, X_n$ be i.i.d. symmetric random variables of law $\chi$. Then, for any $t$ such that $M_\chi(t) = \mathbb{E}[e^{tX}] < \infty$ it holds that

$$\mathbb{P}\left[\left|\sum_{i=1}^{n} x_i\right| > \alpha\right] < 2e^{-\alpha t + n\log(M_\chi(t))}.$$

Using the above inequality with

$$M_{SY}(t) = \frac{2B+3}{6B+3} + \frac{2}{6B+3} \cdot \left(\frac{e^{t(B+1)}-1}{e^t-1} + \frac{e^{-t(B+1)}-1}{e^{-t}-1} - 2\right)$$

and setting $\alpha = \lfloor \frac{q}{4} \rfloor - B - n\omega$, $t \approx 2.5 \cdot 10^{-5}$, $n = 1024$ and $k = 2^{15}$ (our parameters from the previous section), we find that $\mathbb{P}\left[(\Delta'_{ab})_i > \lfloor \frac{3q}{16} \rfloor\right] \approx 2^{-115}$. By virtue of the union bound on the 1024 coefficients, we get that the failure probability is $\approx 2^{-105}$.

## 5.3 Performances

Even if the construction of the signcryption scheme is conceptually interesting on its own, its usage only makes sense if we gain something over the trivial solution of concatenating an encryption/key exchange and a signature scheme. In table 2, we compare the performances regarding bandwidth between SETLA and a selection of schemes of the same kind. Since lattice-based schemes are already doing great in terms of speed, especially when they can take advantage of SIMD instructions, reducing bandwidth will a major factor for adoption in the future. We decided to compare SETLA to the pairs given in the table for the following reasons:

- Dilithium + Kyber: They were very recently designed and are part of the same family of algorithms.
- qTESLA + NewHope: They are the two up-to-date RLWE based schemes and are both candidates for future standardization.
- TESLA♯ + Kyber: This seems to be the most efficient pair regarding compactness out of the reasonably secure Fiat-Shamir/key exchange schemes in the literature.

For the record, we also indicates the performances of the GLP version of signcryption that is using the original parameters of the signature [19]. It obviously gives goods results since we get the key exchange for free without modifying the parameters but the security has been reduced so much over the years that it does not seem reasonable to use it without further modifications. The signcryptext size for SETLA was computed without the symmetric cipher (since it depends on the size of the message itself and would be needed in the naive construction as well) and with Peikert's reconciliation which is less efficient but more compact than the one of NewHope but still gives good correctness results in practice. The last column compares the gain in compactness of signcryptext when using SETLA instead of the mentioned scheme. We see that at the price of a larger public key, SETLA outperforms the naive concatenation of popular schemes by a significant margin. This is not a surprise since we only have to output a signature and the key exchange is done implicitly. The large public key comes partially from the lack of flexibility of RLWE which limits fast implementations to power of two cyclotomics and $m$ as a multiple of $n$. We also evaluated the speed of a research oriented implementation made to verify that the design of the scheme is sound. The SETLA-KEX signcrypt procedure took 1 735 234 cycles while the unsigncrypt procedure took 391 944 cycles. Those tests were made on an Intel Core i7-4600M processor using a reasonable but unoptimized implementation and can be greatly improved using AVX2 parallel instructions for the polynomial operations, even if the scheme is already quite fast.

| Scheme | \|sk\| | \|pk\| | \|Signcryptext\| | Gain |
|---|---|---|---|---|
| Dilithium[18]+Kyber[10] | 2863(=463+2400) | 2560(=1472+1088) | 3852(=2700+1152) | 48% |
| qTESLA+NewHope[5] | 3648(=1856+1792) | 4800(=2976+1824) | 4896(=2720+2176) | 60% |
| TESLA♯[8]+Kyber | 4512(=2112+2400) | 4416(=3328+1088) | 2768(=1616+2176) | 29% |
| SETLA-KEX | 608 | 6400 | 1972 | - |
| GLP-Signcrypt | 202 | 1475 | 1247 | - |

**Table 2.** Comparison between similar schemes using the naive construction. Values are in bytes.

## 6 Conclusion

In this work we presented a lattice-based signcryption scheme called SETLA. We chose a scheme of Malone-Lee as a starting point and proposed two construction both using the Bai-Galbraith signature at their cores. The first construction directly embeds a RLWE key exchange in the signature exactly as in the classical signcryption scheme while the second one uses RLWE encrypt as a key encapsulation mechanism. The KEX version seems to globally outperform the KEM version since even if it is heavier in terms of computation, this is not the main issue with lattices. We proposed a set of parameters targeting 128 bits of classical security following the reduction of Bai and Galbraith. We provided an analysis of correctness and a comparison with most recent schemes (using the naive construction) in the literature regarding signcryptext size. We also made a research oriented implementation to verify the soundness of the scheme while providing reasonable benchmarks. We conclude that it is possible to instantiate SETLA with parameters providing security, correctness and efficiency while still outperforming the naive construction of encrypt-then-sign with state-of-the-art schemes.

## References

1. S. Akleylek, N. Bindel, J. Buchmann, J. Krmer, and G. A. Marson. An efficient lattice-based signature scheme with provably secure instantiation. Cryptology ePrint Archive, Report 2016/030, 2016. `https://eprint.iacr.org/2016/030`.
2. M. R. Albrecht, R. Player, and S. Scott. On the concrete hardness of learning with errors. Cryptology ePrint Archive, Report 2015/046, 2015. `https://bitbucket.org/malb/lwe-estimator`.

3. E. Alkim, N. Bindel, J. Buchmann, Ö. Dagdelen, E. Eaton, G. Gutoski, J. Krämer, and F. Pawlega. Revisiting tesla in the quantum random oracle model. Cryptology ePrint Archive, Report 2015/755, 2015. `http://eprint.iacr.org/2015/755`.
4. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Newhope without reconciliation. Cryptology ePrint Archive, Report 2016/1157, 2016. `http://eprint.iacr.org/2016/1157`.
5. E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe. Post-quantum key exchange—a new hope. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 327–343, Austin, TX, 2016. USENIX Association.
6. J. Baek, R. Steinfeld, and Y. Zheng. Formal proofs for the security of signcryption. *Journal of Cryptology*, 20(2):203–235, Apr 2007.
7. S. Bai and S. D. Galbraith. An improved compression technique for signatures based on learning with errors. In J. Benaloh, editor, *Topics in Cryptology – CT-RSA 2014. Proceedings*, pages 28–47, Cham, 2014. Springer International Publishing.
8. P. S. L. M. Barreto, P. Longa, M. Naehrig, J. E. Ricardini, and G. Zanon. Sharper ring-lwe signatures. Cryptology ePrint Archive, Report 2016/1026, 2016. `https://eprint.iacr.org/2016/1026`.
9. D. Boneh, Ö. Dagdelen, M. Fischlin, A. Lehmann, C. Schaffner, and M. Zhandry. Random oracles in a quantum world. Cryptology ePrint Archive, Report 2010/428, 2010. `https://eprint.iacr.org/2010/428`.
10. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, and D. Stehlé. Crystals – kyber: a cca-secure module-lattice-based kem. Cryptology ePrint Archive, Report 2017/634, 2017. `http://eprint.iacr.org/2017/634`.
11. J. W. Bos, C. Costello, L. Ducas, I. Mironov, V. Nikolaenko, A. Raghunathan, and D. Stebila. Frodo : Take off the ring ! practical , quantum-secure key exchange from lwe. 2016.
12. J. W. Bos, C. Costello, M. Naehrig, and D. Stebila. Post-quantum key exchange for the tls protocol from the ring learning with errors problem. In *2015 IEEE Symposium on Security and Privacy*, pages 553–570, May 2015.
13. Ö. Dagdelen, R. El Bansarkhani, F. Göpfert, T. Güneysu, T. Oder, T. Pöppelmann, A. H. Sánchez, and P. Schwabe. High-speed signatures from standard lattices. In D. F. Aranha and A. Menezes, editors, *Progress in Cryptology - LATINCRYPT 2014*, pages 84–103, Cham, 2015. Springer International Publishing.
14. R. de Clercq, S. S. Roy, F. Vercauteren, and I. Verbauwhede. Efficient software implementation of ring-lwe encryption. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 339–344, San Jose, CA, USA, 2015. EDA Consortium.
15. A. W. Dent and Y. Zheng. *Practical Signcryption*. Springer, 2010.
16. Y. Dodis, L. Reyzin, and A. Smith. *Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data*, pages 523–540. Springer Berlin Heidelberg, 2004.
17. L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky. Lattice signatures and bimodal gaussians. In R. Canetti and J. A. Garay, editors, *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 40–56, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
18. L. Ducas, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle. Crystals – dilithium: Digital signatures from module lattices. Cryptology ePrint Archive, Report 2017/633, 2017. `https://eprint.iacr.org/2017/633`.
19. T. Güneysu, V. Lyubashevsky, and T. Pöppelmann. *Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems*, pages 530–547. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
20. X. L. Jintai Ding, Xiang Xie. A simple provably secure key exchange scheme based on the learning with errors problem. Cryptology ePrint Archive, Report 2012/688, 2012. `https://eprint.iacr.org/2012/688`.

21. E. Kiltz, V. Lyubashevsky, and C. Schaffner. A concrete treatment of fiat-shamir signatures in the quantum random-oracle model. In J. B. Nielsen and V. Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018*, pages 552–586, Cham, 2018. Springer International Publishing.
22. F. Li, F. T. Bin Muhaya, M. K. Khan, and T. Takagi. Lattice-based signcryption. *Concurrency and Computation: Practice and Experience*, 25(14):2112–2122, 2013.
23. Z. Liu, H. Seo, S. Sinha Roy, J. Großschädl, H. Kim, and I. Verbauwhede. *Efficient Ring-LWE Encryption on 8-Bit AVR Processors*, pages 663–682. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.
24. X. Lu, Q. Wen, Z. Jin, L. Wang, and C. Yang. A lattice-based signcryption scheme without random oracles. *Frontiers of Computer Science*, 8(4):667–675, Aug 2014.
25. V. Lyubashevsky. Fiat-shamir with aborts: Applications to lattice and factoring-based signatures. In M. Matsui, editor, *Advances in Cryptology – ASIACRYPT 2009*, pages 598–616, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.
26. V. Lyubashevsky. *Lattice Signatures without Trapdoors*, pages 738–755. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
27. V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. *J. ACM*, 60(6):43:1–43:35, Nov. 2013.
28. J. Malone-Lee. Signcryption with non-interactive non-repudiation. *Designs, Codes and Cryptography*, 37(1):81–109, 2005.
29. C. Peikert. *Lattice Cryptography for the Internet*, pages 197–219. Springer International Publishing, 2014.
30. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, Jun 2000.
31. T. Pöppelmann, T. Oder, and T. Güneysu. *High-Performance Ideal Lattice-Based Cryptography on 8-Bit ATxmega Microcontrollers*, pages 346–365. Springer International Publishing, Cham, 2015.
32. S. Sato and J. Shikata. Lattice-based signcryption without random oracles. In T. Lange and R. Steinwandt, editors, *Post-Quantum Cryptography*, pages 331–351, Cham, 2018. Springer International Publishing.
33. J. Yan, L. Wang, L. Wang, Y. Yang, and W. Yao. Efficient lattice-based signcryption in standard model. 2013:1–18, 11 2013.
34. Y. Zheng. *Digital signcryption or how to achieve cost(signature & encryption) cost(signature) + cost(encryption)*, pages 165–179. Springer Berlin Heidelberg, 1997.

## A Security games for the KEM version

Game 0:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $K \xleftarrow{r} \{0,1\}^{256}$
4: $\mathbf{y} \xleftarrow{r} \mathcal{R}_{q,[B]}$
5: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
6: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
7: $\mathbf{z} \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}$
8: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
9: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
10: if z not in $\mathcal{R}_{q,[B-\omega]}$, goto 3
11: $\mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
12: $\mathbf{x} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}' + \mathtt{Encode}(K)$
13: $\mathcal{E} \leftarrow E_K(m)$
14: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E})$
15: return $\hat{b}$
```

Game 1:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $K \xleftarrow{r} \{0,1\}^{256}$
4: $\mathbf{y} \xleftarrow{r} \mathcal{R}_{q,[B]}$
5: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
6: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
7: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
8: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
9: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
10: with probability P, goto 3
11: $\mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
12: $\mathbf{x} \leftarrow \mathbf{t}_{b,1} \cdot \mathbf{y} + \mathbf{y}' + \mathtt{Encode}(K)$
13: $\mathcal{E} \leftarrow E_K(m)$
14: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E})$
15: return $\hat{b}$
```

Game 2:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $K \xleftarrow{r} \{0,1\}^{256}$
4: $\mathbf{y} \xleftarrow{r} \mathcal{R}_{q,[B]}$
5: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
6: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
7: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
8: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
9: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
10: with probability P, goto 3
11: $\mathbf{y}' \xleftarrow{r} \mathcal{R}_{q,[B]}$
12: $\mathbf{a}' \xleftarrow{r} \mathcal{R}_q$
13: $\mathbf{x} \leftarrow \mathbf{a}' \cdot \mathbf{y} + \mathbf{y}' + \mathtt{Encode}(K)$
14: $\mathcal{E} \leftarrow E_K(m)$
15: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E})$
16: return $\hat{b}$
```

Game 3:

```
1: $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathbf{pk}_a, \mathbf{pk}_b)$
2: $b \xleftarrow{r} \{0,1\}$
3: $K \xleftarrow{r} \{0,1\}^{256}$
4: $\mathbf{y} \xleftarrow{r} \mathcal{R}_{q,[B]}$
5: $\mathbf{h}_1 \leftarrow \lfloor \mathbf{a}_1 \cdot \mathbf{y} \rceil_d, \mathbf{h}_2 \leftarrow \lfloor \mathbf{a}_2 \cdot \mathbf{y} \rceil_d$
6: $\mathbf{c} \leftarrow H(\mathbf{h}_1, \mathbf{h}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
7: $\mathbf{z} \xleftarrow{r} \mathcal{R}_{q,[B-\omega]}$
8: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{y} - \mathbf{e}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{y} - \mathbf{e}_{a,2} \cdot \mathbf{c}$
9: if $\mathbf{h}_1 \neq \lfloor \mathbf{w}_1 \rceil_d$ or $\mathbf{h}_1 \neq \lfloor \mathbf{w}_2 \rceil_d$, goto 3
10: with probability P, goto 3
11: $\mathbf{x} \xleftarrow{r} \mathcal{R}_q$
12: $\mathcal{E} \leftarrow E_K(m)$
13: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}, \mathbf{c}, \mathbf{x}, \mathcal{E})$
14: return $\hat{b}$
```

**Fig. 4.** Sequence of games for the KEM version

**Game 0** → **Game 1**: Rejection sampling
**Game 1** → **Game 2**: Decisional Compact Knapsack/RLWE
**Game 2** → **Game 3**: Decisional Compact Knapsack/RLWE

## B Publicly verifiable signature from signcryptext

An interesting feature of Malone-Lee's signcryption scheme is that the receiver Bob can himself create a fully valid publicly verifiable signature under Alice's secret key on the message he unsigncrypted. Even if we chose to start from this scheme for its similarity with Schnorr signature (and thus, lattice-based signature), this really helpful feature carries to our construction. Below are the algorithms for the KEX version but the same technique can trivially be applied to the KEM version.

**Algorithm 6** SETLA-KEX SignExtract

**Input**: Public parameters $\mathbf{a_1}, \mathbf{a_2}$, Bob's keys $(\mathbf{s}_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}, \mathbf{c}, \mathcal{E}, \mathbf{r})$, random oracle $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$, symmetric encryption algorithm $E$
**Output**: A message $m$ together with its signature $\sigma(m)$ or a failure symbol

```
1: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{z} - \mathbf{t}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2 \leftarrow \mathbf{a}_2 \cdot \mathbf{z} - \mathbf{t}_{a,2} \cdot \mathbf{c}$
2: $K \leftarrow \mathtt{Rec}(\mathbf{w}_1 \cdot \mathbf{s}_b, \mathbf{r})$
3: $m \leftarrow E_K^{-1}(\mathcal{E})$
4: $b \leftarrow \mathbf{c} = H(\lfloor \mathbf{w}_1 \rceil_d, \lfloor \mathbf{w}_2 \rceil_d, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ and $\mathbf{z} \in \mathcal{R}_{q,[B-\omega]}$
5: return $m, \sigma(m) = (K, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$ if $b = 1$ else $\perp$
```

**SETLA-KEX SignExtract.** (Algorithm 6) To extract a publicly verifiable signature $\sigma(m)$ from a signcryptext, Bob will use the fact that the output of KEX-Signcrypt is essentially equivalent to a TESLA♯ signature on $m$ with a nonce depending on $K, \mathbf{pk}_a$ and $\mathbf{pk}_b$ queried to the random oracle. Since a verifier should obviously know the message to validate the signature, the confidentiality of the key $K$ is not required anymore. Thus, KEX-SignExctract will output $m$ and $K$ together with the signature and anyone will be able to perform the verification.

**Algorithm 7** PublicVerif

**Input**: Public parameters $\mathbf{a_1}, \mathbf{a_2}$, Alice's public key $\mathbf{pk}_a$, Bob's public key $\mathbf{pk}_b$, a message $m$, a signature $\sigma(m) = (K, \mathbf{z}, \mathbf{c})$, hash function $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = \omega\}$
**Output**: 1 if the signature is valid, 0 otherwise

```
1: $\mathbf{w}_1 \leftarrow \mathbf{a}_1 \cdot \mathbf{z} - \mathbf{t}_{a,1} \cdot \mathbf{c}$, $\mathbf{w}_2$
2: return 1 if $\mathbf{c} = H(\lfloor \mathbf{w}_1 \rceil_d, \lfloor \mathbf{w}_2 \rceil_d, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ and $\mathbf{z} \in \mathcal{R}_{q,[B-\omega]}$ else 0
```

**PublicVerif.** (Algorithm 7) The public verification is the same as in usual lattice-based signatures, except that the hash function also takes as input $K, \mathbf{pk}_a$ and $\mathbf{pk}_b$.