Post-Quantum Signcryption From Lattice-Based Signatures

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 signcryption scheme based on the GLP signature inspired from a construction of Malone-Lee. It comes in two flavors, one integrating the usual lattice-based key exchange into GLP and the other merging the signature scheme with a RLWE encryption, which is more efficient, but outputs a larger signcryptext. Using the same set of operations as in existing constructions, our scheme can be implemented efficiently on various platforms, reusing optimized pieces of software or hardware presented in previous works.

Keywords: Post-quantum Signcryption Lattice GLP Signature Key exchange Key encapsulation

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 its 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, signature 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 [26]. 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 [9].
Unfortunately, the techniques used were based on the Diffie-Hellman (or RSA) assumption and their security would be invalided 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 a development, both on theoretical and practical side, of quantum-resistant cryptography.

Our contribution. In this paper, we introduce a construction of a signcryption scheme based on the GLP signature [12]. It is inspired from a Schnorr-like variant of the original work of Zheng [26] proposed by Malone-Lee [20]. 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 schemes provide a tradeoff between efficiency and storage. The key exchange version is slower but uses less memory/bandwidth. Even though the idea of the construction could be translated to other Fiat-Shamir lattice signatures, the interesting aspect of using GLP is that the transformation works out of the box, without adjusting the parameters.

Previous work. Signcryption has not been extensively studied in the post-quantum world yet. Some works on lattice-based schemes exist [15,24,25,17], however, they are all based on trapdoors and do not enable practical instantiations.

Organization of the paper. Section 2 formalizes signcryption and recalls basic tools needed in the construction. Section 3 presents the two versions of our signcryption scheme 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 χ, we use the notation $v \overset{r}{\leftarrow} \chi$. This notation is extended in a natural way to polynomials (of a given degree), $\mathbf{v} \overset{r}{\leftarrow} \chi$ means that the coefficients of $\mathbf{v}$ are all sampled independently from χ. The uniform distribution over a set S is written $\mathcal{U}(S)$. We use $v \overset{r}{\leftarrow} S$ as a shorthand for $v \overset{r}{\leftarrow} \mathcal{U}(S)$. For an odd q, we use the representative in $\left[\frac{-(q-1)}{2}, \frac{q-1}{2}\right]$ to identify cosets of $\mathbb{Z}_q$.

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 $\Gamma_{\mathcal{M},\mathcal{C}} = (\texttt{ParamGen}, \texttt{KeyGenSender}, \texttt{KeyGenReceiver}, \texttt{Signcrypt}, \texttt{Unsigncrypt})$ composed of the five following algorithms:

- $\texttt{ParamGen}(\lambda)$: a randomized algorithm taking as input the security parameter λ 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.
- $\texttt{KeyGenReceiver}()$: 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.
- $\texttt{Signcrypt}(sk_a, pk_b, m)$: 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.
The natural correctness property is that, for every valid m,

$$\texttt{Unsigncrypt}(pk_a, sk_b, \texttt{Signcrypt}(sk_a, pk_b, m)) = m$$

with an overwhelming probability.

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 Security model for signcryption

We naturally extent the security games of encryption and signature to the case of signcryption. As an example, here follow the usual IND-CPA and sUF-CMA games:

Definition 2. The signcryption scheme is said to be IND-CPA secure if the probability of an adversary (having a set of two PPT algorithms $\mathcal{A}, \mathcal{A}'$) winning the following game is negligibly close to $\frac{1}{2}$

1. The challenger first runs the `ParamGen` algorithm and outputs public parameters `params`. After that, the challenger generates Alice's key pair $(pk_a, sk_a) \leftarrow$ `KeyGenReceiver`() and Bob's key pair $(pk_b, sk_b) \leftarrow$ `KeyGenReceiver`().
2. The adversary runs $\mathcal{A}$ on input (pk_a, pk_b). It has access to a `Signcrypt`(sk_a, pk_b, m) and an `Unsigncrypt`(pk_a, sk_b, C) oracle. The algorithm finishes by outputting two messages m_0 and m_1.
3. The challenger chooses a bit $b \overset{r}{\leftarrow} \{0, 1\}$ and outputs $\hat{C} \leftarrow$ `Signcrypt`(sk_a, pk_b, m_b)
4. The adversary then runs $\mathcal{A}'$ on input $(\hat{C}, pk_a, sk_a)$ and finishes by outputting a bit b'.

The adversary wins the IND-CPA game if $b' = b$.

Definition 3. The signcryption scheme is said to be sUF-CMA secure if the probability of the adversary (having a PPT algorithm $\mathcal{A}$) winning the following game is negligible.

1. The challenger first runs the `ParamGen` algorithm and outputs public parameters `params`. After that, the challenger generates Alice's key pair $(pk_a, sk_a) \leftarrow$ `KeyGenReceiver`$()$ and Bob's key pair $(pk_b, sk_b) \leftarrow$ `KeyGenReceiver`$()$.
2. The adversary runs $\mathcal{A}$ on input (pk_a, pk_b). It has access to a `Signcrypt`(sk_a, pk_b, m) and an `Unsigncrypt`(pk_a, sk_b, C) oracle. The algorithm finishes by outputting a signcryptext $\hat{C}$.

The adversary wins the sUF-CMA game if $\hat{C}$ has not been output by the `Signcrypt` oracle and `Unsigncrypt`$(pk_a, sk_b, \hat{C}) \neq \perp$.

Finding the right security model for signcryption seems to be a bit more complex than for other basic primitives. Indeed, since for every communication each participant plays a dual role using at the same time is own secret and the public value of the other one, it is unclear what power we should give to the adversary. To clarify the situation, signcryption schemes security is defined according to two notions, insider security and outsider security.

Outsider security. In the outsider model, the sender and the receiver are both honest and try to prevent an external adversary from retrieving information about the message or modifying it without being detected. It is the model used in the symmetric case of authenticated encryption where all the valid users are "the same" in the sense that they all have the same power and are not associated to a public identity. Here, the adversary only has access to the public-key of the different users of the system and tries to break the IND-CPA or sUF-CMA property of the scheme. A signcryption scheme insecure in the outsider model would be of no use.

Insider security. The insider security model is more complete but its usefulness in practice is more debatable. Here, the receiver and the sender can be the adversary (a more realistic view is that an adversary is given the private key of one of them). Concerning the dishonest receiver, his goal is to forge a signcryptext (signcrypted with his own public key) on a new message without knowing the sender's secret key. For the dishonest sender, her goal is to unsigncrypt a signcryptext created with her private key. The pertinence of this model is quite context dependent. If Bob is the only entity capable of verifying the authenticity of a message, his power of forging a message for himself is irrelevant. Likewise, if Alice can decrypt signcryptext she created with her signing key, we can consider she knows the message. The argument against this claim is to try to protect past (or future) communication if Alice's private key is somehow compromised. In our case, we will not allow Bob to forge a message because we want to enable non-repudiation, meaning that he is not the only one capable verifying the signature anymore. On the other side, we allow Alice to be able to recover a message from past signcryption and argue that it is acceptable since the loss of a secret key often means a total break of the system.

2.4 Ring Learning with Errors and Decisional Compact Knapsack

The ring learning with errors (RLWE) [19] 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 χ 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 $[-k, k]$ we write $\mathcal{R}_{q,[k]}$. We define the following problems:

Definition 4. (*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 χ, find $\mathbf{s}$.

Definition 5. (*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 $\mathbf{t}_i$'s from $\mathcal{U}(\mathcal{R}_q)$

Definition 6. (*Decisional Compact Knapsack*) In [12], 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 $(\mathbf{a}, \mathbf{a} \cdot \mathbf{s} + \mathbf{s}')$ with $\mathbf{a} \overset{r}{\leftarrow} \mathcal{R}_q$ and $(\mathbf{s}, \mathbf{s}') \overset{r}{\leftarrow} (\mathcal{R}_{q,[1]} \times \mathcal{R}_{q,[1]})$ 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.

Those problems are all believed to be hard, even for an adversary in possession of a large-scale quantum computer.

2.5 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 [19,8,23,16] and can be found in Figure 1. This scheme is really similar 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 χ is a narrow distribution of zero mean, this should happen with overwhelming probability.

2.6 Reconciliation mechanism

A common issue in learning with errors key exchanges [14,21,3,7,6] is that both parties end up with two values that are close to each other but not exactly the same. It

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

RLWE Encrypt($\mathbf{pk}, m$):

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

RLWE Decrypt($\mathbf{c}_1, \mathbf{c}_2, \mathbf{s}$):

```
1: $\mathbf{m} = \mathbf{c}_2 - \mathbf{c}_1 \cdot \mathbf{s} \approx \texttt{Encode}(m)$
2: $m = \texttt{Decode}(\mathbf{m})$
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

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 [10]). For example let 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 [21], 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 appears 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 [3]. 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.

3 Lattice-based Signcryption Schemes

Hereunder, we describe both versions of our scheme. The discussion in section 5 will only be made for the second version for the sake of brevity but the analysis is basically the same.

3.1 KEX-Signcryption

First we describe how to integrate encryption into GLP, 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 [20] 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 [18,4,11,1], this is where post-quantum can meet signcryption. We use the GLP signature scheme as a base because, aside from its own advantages (mainly efficient implementations and easy sampling), it fits the most our construction with its original parameters.

Algorithm 1 Key generation

```
Input: Public parameter $\mathbf{a}$
Output: Key pair $\mathbf{pk}, \mathbf{sk}$
1: $\mathbf{s}, \mathbf{s}' \leftarrow \mathcal{R}_{q,[1]}$
2: return $\mathbf{pk} = \mathbf{a} \cdot \mathbf{s} + \mathbf{s}', \mathbf{sk} = (\mathbf{s}, \mathbf{s}')$
```

Key generation. (Algorithm 1) As in GLP, the key generation is simple and straightforward. It uses a public parameter $\mathbf{a}$ shared among all users and output a

RLWE sample $\mathbf{pk}$ together with its two secrets $\mathbf{s}, \mathbf{s}'$. The error and secret distributions are the same and output a polynomial with uniform coefficients in $\{-1, 0, 1\}$. 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. We use subscripts (e.g. $\mathbf{pk}_a = \mathbf{a} \cdot \mathbf{s}_a + \mathbf{s}'_a$) to differentiate them.

Algorithm 2 KEX-Signcrypt

Input: Public parameter $\mathbf{a}$, Bob's public key $\mathbf{pk}_b$, Alice's key $(\mathbf{s}_a, \mathbf{s}'_a, \mathbf{pk}_a)$, a message m, hash function $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$, symmetric encryption algorithm E
Output: a signcryptext of m: $C = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$

```
 1: $\mathbf{y}_3 \overset{r}{\leftarrow} \mathcal{R}_{q,[k]}$
 2: do
 3:     $\mathbf{y}_1, \mathbf{y}_2 \overset{r}{\leftarrow} \mathcal{R}_{q,[k]}$
 4:     $\mathbf{v} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3$
 5:     $\mathbf{r} \leftarrow \mathtt{HelpRec}(\mathbf{v})$
 6:     $K \leftarrow \mathtt{Rec}(\mathbf{v}, \mathbf{r})$
 7:     $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 8:     $\mathbf{z}_1 \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}_1$, $\mathbf{z}_2 \leftarrow \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{y}_2$
 9: while not $\mathbf{z_1}$ and $\mathbf{z_2}$ in $\mathcal{R}_{q,[k-32]}$
10: $\mathcal{E} \leftarrow E_K(m)$
11: return $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r}$
```

KEX-Signcrypt. (Algorithm 2) The signcrypt procedure contains three interleaved parts: signature, key exchange and encryption. The signature follows from [12] as a Fiat-Shamir signature from a Σ protocol. First, a commitment consisting of a RLWE sample using two masking polynomials $\mathbf{y}_1, \mathbf{y}_2 \in \mathcal{R}_{q,[k]}$ is computed. Then, an unpredictable challenge $\mathbf{c}$ is retrieved by simulating a verifier with a hash function H taking inputs depending on the commitment. Finally, the response consists of two polynomials of the form $\mathbf{z} = \mathbf{s} \cdot \mathbf{c} + \mathbf{y}$. Note that for a signcryption scheme, the hash function should take as input a symmetric key K and both public identities. If the key was not included in the input, the adversary could easily 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 models [9].
The key exchange part is performed by deriving a secret value K from a noisy version of $\mathbf{a} \cdot \mathbf{s}_b \cdot \mathbf{y}_1$. 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 secret key by computing $\mathbf{pk}_b \cdot \mathbf{y}_1 = \mathbf{a} \cdot \mathbf{s}_b \cdot \mathbf{y}_1 + \mathbf{s}'_b \cdot \mathbf{y}_1 \approx \mathbf{a} \cdot \mathbf{s}_b \cdot \mathbf{y}_1$. 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 on be able to retrieve an approximation version of $\mathbf{a} \cdot \mathbf{y}_1$ *without* sending him any other ring element than the polynomials computed in the signature $(\mathbf{c}, \mathbf{z}_1, \mathbf{z}_2)$. As in [7,3], 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 $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$, 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 KEX-Unsigncrypt

Input: Public parameter $\mathbf{a}$, Bob's key $(\mathbf{s}_b, \mathbf{s}'_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$, hash function $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$, symmetric encryption algorithm E

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

```
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c}$
2: $K \leftarrow \texttt{Rec}(\mathbf{v} \cdot \mathbf{s}_b, \mathbf{r})$
3: $m \leftarrow E_K^{-1}(\mathcal{E})$
4: return $m$ if $\mathbf{c} = H(\mathbf{v}, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ and $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{R}_{q,[k-32]}$ else $\perp$
```

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.
Exactly as in GLP, Bob will recover the commitment part of the signature $\mathbf{v} = \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ by computing $\mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c}$. 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 $\mathbf{a} \cdot \mathbf{s}_b \cdot \mathbf{y}_1$. Such a value can be found by computing the product $\mathbf{v} \cdot \mathbf{s}_b = \mathbf{a} \cdot \mathbf{s}_b \cdot \mathbf{y}_1 + \mathbf{y}_2 \cdot \mathbf{s}_b$. Once the message is decrypted, Bob verifies the signature by checking the size of $\mathbf{z}_1, \mathbf{z}_2$ and the hash value. He outputs the message if everything is correct and a failure symbol otherwise.

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 KEM-Signcryption

We describe now 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 [2] or Kyber [5]. The high level perspective is now to perform a noisy ElGamal 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 and can enable parallelism. We also apply a small modification in the way the message is hashed. This modification is irrelevant in the random oracle model but could potentially make a difference in practice.

Algorithm 4 KEM-Signcrypt

Input: Public parameter $\mathbf{a}$, Bob's public key $\mathbf{pk}_b$, Alice's key $(\mathbf{s}_a, \mathbf{s}'_a, \mathbf{pk}_a)$, a message m, hash function $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$, symmetric encryption algorithm E
Output: a signcryptext of m: $C = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E})$

```
 1: $K \xleftarrow{r} \{0,1\}^{256}$
 2: $f \leftarrow H(m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 3: do
 4:     $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 5:     $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, f)$
 6:     $\mathbf{z}_1 \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}_1$, $\mathbf{z}_2 \leftarrow \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{y}_2$
 7: while not $\mathbf{z_1}$ and $\mathbf{z_2}$ in $\mathcal{R}_{q,[k-32]}$
 8: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 9: $\mathbf{x} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3 + \mathtt{Encode}(K)$
10: $\mathcal{E} \leftarrow E_K(m)$
11: return $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}$
```

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 almost exactly GLP, the small difference is that the hash function (as in the KEX version) takes as input the hash of 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 $\mathbf{sk}_b$ 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 $\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ acting at the same time as the commitment of the signature and the $\mathbf{c}_1$ part of the encryption scheme.
We see two advantages in using encryption instead of key exchange:
First, we gain efficiency by only having one polynomial multiplication instead of two in the rejection sampling loop and we can now parallelize the symmetric encryption algorithm. Indeed, in the KEX version, the key depends on $\mathbf{y}_1$ and was not known until the end of the rejection sampling procedure, hence, everything had to be sequential and a multiplication with $\mathbf{pk}_b$ 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. That is why we decided to pre-hash the message before signing. 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, reconciliation can be a real issue and hav-

ing the key encoded as a polynomial with coefficients in $\{0, \frac{q-1}{2}\}$ is optimal for the reconciliation 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.

Algorithm 5 KEM-Unsigncrypt

Input: Public parameter $\mathbf{a}$, Bob's key $(\mathbf{s}_b, \mathbf{s}'_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$, hash function $H : * \to \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$, symmetric encryption algorithm E
Output: A message m or failure symbol $\perp$

```
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c}$
2: $K \leftarrow \texttt{Decode}(\mathbf{x} - \mathbf{v} \cdot \mathbf{s}_b)$
3: $m \leftarrow E^{-1}(\mathcal{E})$
4: return $m$ if $\mathbf{c} = H(\mathbf{v}, H(m, K, \mathbf{pk}_a, \mathbf{pk}_b))$ and $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{R}_{q,[k-32]}$ else $\perp$
```

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.

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 using the forking lemma [22] and a standard hybrid argument.

4.1 Unforgeability

The underlying signature of the signcryption scheme is the GLP signature which is itself a derivative of the original proposal of Lyubashevsky [18]. The security argument is somewhat split over both papers but the idea is to use the forking lemma to get two different signatures that would allow to solve the **DCK** problem. We use the adversary to get two forgeries $\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}$ and $\mathbf{z}'_1, \mathbf{z}'_2, \mathbf{c}'$ for different random oracles but the same random tape (hence the same $\mathbf{y}_1, \mathbf{y}_2$). We have $\mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c} = \mathbf{a} \cdot \mathbf{z}'_1 + \mathbf{z}'_2 - \mathbf{pk}_a \cdot \mathbf{c}' = \mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2$ and thus, with $\mathbf{pk}_a = \mathbf{a} \cdot \mathbf{s}_a + \mathbf{s}'_a$, $\mathbf{a} \cdot (\mathbf{z}_1 - \mathbf{z}'_1 - \mathbf{s}_a \cdot \mathbf{c} + \mathbf{s}_a \cdot \mathbf{c}') + (\mathbf{z}_2 - \mathbf{z}'_2 - \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{s}'_a \cdot \mathbf{c}') = 0$. As pointed in [12], (if $\mathbf{z}_1 - \mathbf{z}'_1 - \mathbf{s}_a \cdot \mathbf{c} + \mathbf{s}_a \cdot \mathbf{c}'$ and $\mathbf{z}_2 - \mathbf{z}'_2 - \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{s}'_a \cdot \mathbf{c}'$ are non-zero) we have found a solution to the **DCK** problem through the reduction in [18]. 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** problem. 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. Both the hash function and the encryption scheme are seen as ideal primitives. The sequence of games for the KEM version can be found in Figure 2. Games for the KEX 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: $b \leftarrow \{0,1\}$
 3: $K \leftarrow \{0,1\}^{256}$
 4: $f \leftarrow H(m_b)$
 5: do
 6:     $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 7:     $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, f, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 8:     $\mathbf{z}_1 \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}_1$, $\mathbf{z}_2 \leftarrow \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{y}_2$
 9: while not $\mathbf{z_1}$ and $\mathbf{z_2}$ in $\mathcal{R}_{q,[k-32]}$
10: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
11: $\mathbf{x} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3 + \mathtt{Encode}(K)$
12: $\mathcal{E} \leftarrow E_K(m_b)$
13: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$
14: 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: $f \leftarrow H(m_b)$
 5: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 6: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, f, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 7: $\mathbf{z}_1, \mathbf{z}_2 \leftarrow \mathcal{R}_{q,[k-32]}$
 8: with probability $P$ goto step 5
 9: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
10: $\mathbf{x} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3 + \mathtt{Encode}(K)$
11: $\mathcal{E} \leftarrow E_K(m)$
12: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$
13: 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: $f \leftarrow H(m_b)$
 5: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 6: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, f, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 7: $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{r} \mathcal{R}_{q,[k-32]}$
 8: with probability $P$ goto step 5
 9: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
10: $\mathbf{a}' \xleftarrow{r} \mathcal{R}_q$
11: $\mathbf{x} \leftarrow \mathbf{a}' \cdot \mathbf{y}_1 + \mathbf{y}_3 + \mathtt{Encode}(K)$
12: $\mathcal{E} \leftarrow E_K(m)$
13: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$
14: 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: $f \leftarrow H(m_b)$
 5: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
 6: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, f, K, \mathbf{pk}_a, \mathbf{pk}_b)$
 7: $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{r} \mathcal{R}_{q,[k-32]}$
 8: with probability $P$ goto step 5
 9: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
10: $\mathbf{x} \xleftarrow{r} \mathcal{R}_{q,[k]}$
11: $\mathcal{E} \leftarrow E_K(m)$
12: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathbf{x}, \mathcal{E}, \mathbf{r})$
13: return $\hat{b}$
```

Fig. 2. Sequence of games. $P = 1 - \left(1 - \frac{64}{2k+1}\right)^{2n}$ is the probability of $\mathbf{z}_1$ or $\mathbf{z}_2$ not belonging to $\mathcal{R}_{q,[k-32]}$ (see [12], footnote 5)

Game 0: Game 0 is the usual CPA game against the signcryption scheme, 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 the $\mathbf{z}_i$ should be exactly the same as a uniform over $\mathcal{R}_{q,[k-32]}$. Hence, we can replace $\mathbf{z}_1$ and $\mathbf{z}_2$ 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 the ciphertext of the KEM by a random value.

In conclusion, using the fact that both $H(.)$ and $E(.)$ are modeled as ideal primitives and that H takes one random unknown value 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 message was signcrypted.

5 Analysis

In the analysis of the scheme, we are going to use the set of parameters proposed in the original paper of Güneysu, Lyubashevsky and Pöppelmann. Even though we think that crafting parameters to fit the signcryption scheme is interesting, the point of reusing the previous set is to show that we gain something solely by construction while keeping the underlying signature intact. The efficient/lower security set proposed in [12] is the following:

Set I: $n = 512, q = 8383489, k = 2^{14}$

5.1 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 $\Delta_{ab} = \mathbf{s}'_b \cdot \mathbf{y}_1 - \mathbf{s}_b \cdot \mathbf{y}_2 + \mathbf{y}_3$. While in those schemes the parameters are chosen in order to get correctness with overwhelming probability together with strong security, we face here another strong constraint which is that the parameters should also be compatible with the signature scheme. For example, one strategy to reduce the norm of Δ_{ab} is to reduce k. This would give better results for correctness but at the same time decrease the speed of the scheme since the rejection sampling loop would have a harder time to find small enough $\mathbf{z}_i$. Hence, even though having existing parameters such as **Set I** working reasonably well is a nice feature, tailoring parameters and distributions for an optimal trade-off between performances and correctness would be interesting.

We now provide an analysis of the failure probability for the KEM-version. The KEM-Unsigncrypt algorithm recovers the correct key if $\|\Delta_{ab}\|_\infty < \lfloor \frac{q}{4} \rfloor$. In the following, we write $(\mathbf{p})_i$, to denote the i-th coefficient of a polynomial $\mathbf{p}$.
We start by bounding the magnitude of one coefficient $(\Delta'_{ab})_i = (\mathbf{s}'_b \cdot \mathbf{y}_1 - \mathbf{s}_b \cdot \mathbf{y}_2)_i$. 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 $2n$ products between a coefficient of a polynomial in $\mathcal{R}_{q,[1]}$ and a polynomial in $\mathcal{R}_{q,[k]}$.
Let $S \sim \mathcal{U}(\{-1, 0, 1\})$ and $Y \sim \mathcal{U}([-k, k])$ be random variables, we denote their

product SY. Each coefficient of Δ'_{ab} is the sum of n samples from SY, hence $(\Delta'_{ab})_i \sim \sum_{i=1}^{2n}(SY)_i$. Fortunately, computing the exact distribution SY is easy:

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

and

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

Since the value of k is reasonable, to find the distribution of Δ'_{ab}, one could hope to compute $\log(n)$ time the convolution of the distribution with itself. Unfortunately, this approach failed to give accurate enough results because of numerical stability issues. Instead, as in [3], we use the Chernoff-Cramer inequality to bound the sum of the random variables.

Chernoff-Cramer inequality Let χ be a distribution over $\mathbb{R}$ and let $X_1, \ldots, X_{2n}$ be i.i.d. symmetric random variables of law χ. Then, for any t such that $M_\chi(t) = \mathbb{E}[e^{tX}] < \infty$ it holds that

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

Using the above inequality with

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

and setting $\alpha = \lfloor\frac{q}{4}\rfloor - k$, $t \approx 3.54\cdot 10^{-4}, n = 512$ and $k = 2^{14}$ (**Set I**), we find that $\mathbb{P}\left[(\Delta'_{ab})_i > \lfloor\frac{q}{4}\rfloor\right] \approx 2^{-50}$. By applying union bound on the 512 coefficients, we get that the failure probability is $\approx 2^{-41}$. We should note that in reality only the 256 first coefficients encode the key and that the probability can also be lowered by using a multiple coefficients per bit encoding instead of padding with zeroes.

5.2 Compression

To decrease the size of the signature, the authors of [12] proposed a compression technique highly reducing the entropy of $\mathbf{z}_2$. They send $\mathbf{z}'_2 = \texttt{Compress}(\mathbf{a}\cdot\mathbf{z}_1 - \mathbf{t}\cdot\mathbf{c}, \mathbf{z}_2, p, k-32)$ instead of $\mathbf{z}_2$. The idea is that $\mathbf{z}'_2$ will be a polynomial with (most of its) coefficients in $\{-(k-32), 0, k-32\}$ such that $\mathbf{v}^{(1)} = \lfloor\frac{\mathbf{a}\cdot\mathbf{z}_1+\mathbf{z}_2-\mathbf{t}\cdot\mathbf{c}}{2(k-32)+1}\rceil = \mathbf{v}'^{(1)} = \lfloor\frac{\mathbf{a}\cdot\mathbf{z}_1+\mathbf{z}'_2-\mathbf{t}\cdot\mathbf{c}}{2(k-32)+1}\rceil$ (see the original paper for a full analysis of the algorithm). Alice will use $\mathbf{v}^{(1)}$ and Bob $\mathbf{v}'^{(1)}$ as input to the hash function.
Such a technique would also work with signcryption, when receiving the signature, Bob would compute $\mathbf{a}\cdot\mathbf{z}_1+\mathbf{z}'_2-\mathbf{t}\cdot\mathbf{c}$ which is $\mathbf{a}\cdot\mathbf{y}_1+\mathbf{y}'_2$ with $\mathbf{y}'_2 = \mathbf{y}_2+\epsilon$ and carry on with the algorithm. What happens now is that the difference between the value of Alice and the value of Bob before reconciliation is $\Delta_{ab}^{compress} = \mathbf{s}'_b\cdot\mathbf{y}_1 - \mathbf{s}_b\cdot(\mathbf{y}_2+\epsilon)+\mathbf{y}_3$ with ϵ depending on the compression (but with $\|\epsilon\|_\infty < k$). In conclusion the scheme still works but the correctness is lowered. However a tradeoff can be done by compressing less aggressively the coefficients of $\mathbf{y}_2$.

5.3 Performances

To assess the efficiency, we implemented the KEM version of the scheme by modifying the software implementation of GLP by Güneysu, Oder, Pöppelmann, and Schwabe [13][1]. We run the software on an Intel Core i7-4600M. Results can be found in Table 1. As expected, the KEX version is slower to signcrypt a message. We also made a comparison with GLP to assess the overhead. Again, the experiments confirm the intuition, the KEM version induces a small penalty due to the extra polynomial multiplication and longer input to the hash function while the KEX version is sensibly heavier. The choice of the symmetric encryption scheme being orthogonal to the rest of the scheme, it was not taken into account (we applied a one-time pad with the key).
Regarding storage and bandwidth usage, the KEM version produces signcryptext of 2656 bytes (21248 bits) to which we should add the symmetric ciphertext size. Compression to reduce the size of $\mathbf{z}_2$ was used. The KEX version using the reconciliation mechanism of NewHope would output signcryptext of around 1308 bytes (10464 bits).

	KEX-Signcryption	KEM-Signcryption	GLP
Signcrypt	850108	624607	564392
Unsigncrypt	92168	120607	52969

Table 1. Cycle counts and comparison with the signature scheme

6 Conclusion

In this work we provided a first look into signcryption in a post-quantum world. Knowing that those schemes were originally based on Elgamal cryptography, the translation toward lattices is natural. We finally chose a scheme of Malone-Lee as a starting point and proposed two construction both using the GLP 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, leading to a more efficient scheme but larger signcryptext. These modifications of GLP work with the original parameters of the signature scheme and hence do not induce any other overhead than the construction. Nevertheless, the transformation is generic and could be integrated in others similar signature schemes. Finally, some of the choices in the parameters and the exact construction depend a lot on the context in which the primitives is used. This is not a surprise since signcryption itself is already really contextual.

[1] `https://cryptojedi.org/crypto/#lattisigns`

References

1. 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`.
2. 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`.
3. 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.
4. 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: The Cryptographer's Track at the RSA Conference 2014, San Francisco, CA, USA, February 25-28, 2014. Proceedings*, pages 28–47, Cham, 2014. Springer International Publishing.
5. 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`.
6. 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.
7. 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.
8. 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.
9. A. W. Dent and Y. Zheng. *Practical Signcryption*. Springer, 2010.
10. 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.
11. 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.
12. 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.
13. T. Güneysu, T. Oder, T. Pöppelmann, and P. Schwabe. *Software Speed Records for Lattice-Based Signatures*, pages 67–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.
14. 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`.
15. 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.
16. 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.
17. 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.

18. V. Lyubashevsky. *Lattice Signatures without Trapdoors*, pages 738–755. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
19. 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.
20. J. Malone-Lee. Signcryption with non-interactive non-repudiation. *Designs, Codes and Cryptography*, 37(1):81–109, 2005.
21. C. Peikert. *Lattice Cryptography for the Internet*, pages 197–219. Springer International Publishing, 2014.
22. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, Jun 2000.
23. 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.
24. F. Wang, Y. Hu, and C. Wang. Post-quantum secure hybrid signcryption from lattice assumption. 6, 01 2012.
25. J. Yan, L. Wang, L. Wang, Y. Yang, and W. Yao. Efficient lattice-based signcryption in standard model. 2013:1–18, 11 2013.
26. 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 KEX 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: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
4: do
5:     $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
6:     $\mathbf{v} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3$
7:     $\mathbf{r} \leftarrow \mathtt{HelpRec}(\mathbf{v})$
8:     $K \leftarrow \mathtt{Rec}(\mathbf{v}, \mathbf{r})$
9:     $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
10:    $\mathbf{z}_1 \leftarrow \mathbf{s}_a \cdot \mathbf{c} + \mathbf{y}_1$, $\mathbf{z}_2 \leftarrow \mathbf{s}'_a \cdot \mathbf{c} + \mathbf{y}_2$
11: while not $\mathbf{z_1}$ and $\mathbf{z_2}$ in $\mathcal{R}_{q,[k-32]}$
12: $\mathcal{E} \leftarrow E_K(m)$
13: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$
14: 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: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
4: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
5: $\mathbf{v} \leftarrow \mathbf{pk}_b \cdot \mathbf{y}_1 + \mathbf{y}_3$
6: $\mathbf{r} \leftarrow \mathtt{HelpRec}(\mathbf{v})$
7: $K \leftarrow \mathtt{Rec}(\mathbf{v}, \mathbf{r})$
8: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
9: $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{r} \mathcal{R}_{q,[k-32]}$
10: with probability P, goto 4
11: $\mathcal{E} \leftarrow E_K(m)$
12: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$
13: 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: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
4: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
5: $\mathbf{a}' \leftarrow \mathcal{R}_q$
6: $\mathbf{v} \leftarrow \mathbf{a}' \cdot \mathbf{y}_1 + \mathbf{y}_3$
7: $\mathbf{r} \leftarrow \mathtt{HelpRec}(\mathbf{v})$
8: $K \leftarrow \mathtt{Rec}(\mathbf{v}, \mathbf{r})$
9: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
10: $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{r} \mathcal{R}_{q,[k-32]}$
11: with probability P, goto 4
12: $\mathcal{E} \leftarrow E_K(m)$
13: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$
14: 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: $\mathbf{y}_3 \xleftarrow{r} \mathcal{R}_{q,[k]}$
4: $\mathbf{y}_1, \mathbf{y}_2 \xleftarrow{r} \mathcal{R}_{q,[k]}$
5: $\mathbf{v} \leftarrow \mathcal{R}_q$
6: $\mathbf{r} \leftarrow \mathtt{HelpRec}(\mathbf{v})$
7: $K \leftarrow \mathtt{Rec}(\mathbf{v}, \mathbf{r})$
8: $\mathbf{c} \leftarrow H(\mathbf{a} \cdot \mathbf{y}_1 + \mathbf{y}_2, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$
9: $\mathbf{z}_1, \mathbf{z}_2 \xleftarrow{r} \mathcal{R}_{q,[k-32]}$
10: with probability P, goto 4
11: $\mathcal{E} \leftarrow E_K(m)$
12: $\hat{b} \leftarrow \mathcal{A}_2(\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E}, \mathbf{r})$
13: return $\hat{b}$
```

Fig. 3. Sequence of games for the KEX version

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

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's 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 KEX-SignExtract

Input: Public parameter $\mathbf{a}$, Bob's keys $(\mathbf{s}_b, \mathbf{s}'_b, \mathbf{pk}_b)$, Alice's public key $\mathbf{pk}_a$, a signcryptext $C = (\mathbf{z}_1, \mathbf{z}_2, \mathbf{c}, \mathcal{E})$, hash function $H : * \rightarrow \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$, symmetric encryption algorithm E
Output: A message m together with its signature $\sigma(m)$

```
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c}$
2: $K \leftarrow \mathtt{Rec}(\mathbf{v} \cdot \mathbf{s}_b, \mathbf{r})$
3: $m \leftarrow E_K^{-1}(\mathcal{E})$
4: return $m, \sigma(m) = (K, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$
```

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 equivalent to a GLP signature on m with a nonce depending on $K, \mathbf{pk}_a$ and $\mathbf{pk}_b$ inserted into the hash function. 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 GLP signature and anyone will be able to perform the verification.

Algorithm 7 PublicVerif

Input: Public parameter $\mathbf{a}$, Alice's public key $\mathbf{pk}_a$, Bob's public key $\mathbf{pk}_b$, a message m, a signature $\sigma(m) = (K, \mathbf{z}_1, \mathbf{z}_2, \mathbf{c})$, hash function $H : * \rightarrow \{\mathbf{v} \mid \mathbf{v} \in \mathcal{R}_{q,[1]}, \|\mathbf{v}\|_1 = 32\}$
Output: A message m or failure symbol $\perp$

```
1: $\mathbf{v} \leftarrow \mathbf{a} \cdot \mathbf{z}_1 + \mathbf{z}_2 - \mathbf{pk}_a \cdot \mathbf{c}$
2: return 1 if $\mathbf{z}_1, \mathbf{z}_2 \in \mathcal{R}_{q,[k-32]}$ and $\mathbf{c} = H(\mathbf{v}, m, K, \mathbf{pk}_a, \mathbf{pk}_b)$ else 0
```

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