GLYPH: A New Instantiation of the GLP Digital Signature Scheme

Arjun Chopra*

Abstract

In 2012 Güneysu, *et al.* proposed GLP, a practical and efficient post-quantum digital signature scheme based on the computational hardness of the Ring Learning With Errors problem. It has some advantages over more recent efficient post-quantum digital signature proposals such as BLISS and Ring-TESLA, but Ring Learning With Errors hardness is more fully understood now than when GLP was published a half decade ago. Although not broken, GLP as originally proposed is no longer considered to offer strong levels of security.

We propose GLYPH, a new instantiation of GLP, parametrised for 128 bits of security under the very conservative assumptions proposed in [2], which gives a strong assurance that it will be secure against forgery even if there are further developments in lattice cryptanalysis. Parameters to obtain this strong security level in an efficient manner were not possible within the original formulation of GLP, as they are not compatible with a signature compression algorithm, and to address this we also propose a new form of the compression algorithm which works efficiently with wider ranges of parameters.

We have produced a software implementation of GLYPH, and we place it in the public domain at `github.com/quantumsafelattices/glyph`.

Keywords: Cryptography, Post-Quantum Cryptography, Lattice, Ring-LWE, Ring Learning With Errors, Digital Signature, GLP

1 Introduction

Lattice-based cryptographic primitives are emerging as promising post-quantum alternatives for classical asymmetric public key cryptography. There are now mature proposals for efficient and secure Diffie–Hellman-like key agreements such as [5, 2, 12, 13] which have strong security guarantees and have been shown to work within real-world protocols. However practical post-quantum alternatives to classical *digital signature* schemes are only now beginning to emerge. Three main contenders for post-quantum alternatives to current digital signature schemes are the 'BLISS' proposal of Ducas *et al.* in [8], the 'Ring-TESLA' proposal of Akleylek *et al.* in

*`arjun.chopra.vsc@outlook.com`. `Associate Consultant, VS Communications`

[3], latterly improved by Barreto *et al.* in [4] and by the present author in [7], and the 'GLP' proposal of Güneysu, *et al.* in [10].

BLISS is especially efficient and compact, so is a natural contender for a post-quantum digital signature scheme, however this efficiency and compactness is obtained by use of Discrete Gaussian sampling, which is especially hard to secure against timing and other side-channel attacks as demonstrated, for example, in [9] and [11]. Therefore it may be advantageous to consider signature schemes that do not employ Discrete Gaussian sampling.

Ring-TESLA is not as efficient as BLISS, as it was designed with the especial assurance of a 'tight security reduction' to the Ring Learning With Errors (Ring-LWE) problem in mind. However, recently a flaw has been found in the reduction, although this has not been shown to lead to a practical attack. Recent improvements have been proposed to Ring-TESLA such as in [4] and [7], although they do not address the security proof's flaws.

GLP is a strong contender, as it is more efficient than Ring-TESLA and, unlike both BLISS and Ring-TESLA, does not use Discrete Gaussian sampling. It has a security reduction to the Decisional Compact Knapsack (DCK) problem which is a specific case of the Ring-LWE problem. However, Ring-LWE security is better understood now than it was five years ago when [10] was first published, and the instantiation proposed there is no longer considered to offer the desired security against modern cryptanalysis. GLP had originally proposed two parameters sets intended to offer either 100, or more than 256 bits of security, however these estimates have been substantially reduced over time, as noted in [8].

In this paper we shall propose a new instantiation of GLP which offers over 128 bits of security under the conservative model proposed in [2] by Alkim *et al*, which makes optimistic assumptions about the capabilities of an adversary, so should offer long term assurance. We call our instantiation GLYPH. A key innovation in GLYPH, which keeps the scheme as efficient as possible, is an improved 'signature compression' routine. The compression routine originally proposed in [10] works only for certain paramter sets, and is not compatible with the ones we propose here.

2 Preliminaries

In this section we shall recap preliminaries necessary to understand the GLP signature scheme.

2.1 Cyclotomic Rings

Let $R = \mathbb{Z}/\langle \Phi_m(x) \rangle = \mathbb{Z}[\zeta_m]$ be the m-th cyclotomic ring, where $\Phi_m(x)$ is the m-th cyclotomic polynomial and ζ_m is a primitive m-th root of unity. The degree n of R is the degree of Φ_m, which is given by the Euler totient function $\phi(m)$. In this paper m shall be a power of two or

shall be prime.

In the case where m is a power of 2, the situation is $n = \phi(m) = m/2$ and $\Phi_m(x) = 1 + x^n$. This case has the advantage of practical efficiencies and simplifications, but has the disadvantage of restricting to a narrow range of parameter sizes. A wider choice of parameter sizes is provided when instead m is prime, from which $n = \phi(m) = m - 1$, and $\Phi_m(x) = 1 + x + x^2 + \ldots + x^n$.

For any integer q we shall let R_q denote the quotient ring R/qR. Multiple bases for the rings R and R_q are considered in the literature and used for efficient implementations, however for ease of exposition we shall here consider only the natural basis for R and R_q given by $\left\{1, \zeta, \zeta^2, \ldots, \zeta^{n-1}\right\}$, also known as the power basis.

Elements of R_q are represented as $\sum_i x_i \cdot \zeta^i$ where the coefficients x_i are integers in $[0, q)$. We shall also refer to coefficients that are negative mod q for consistency with the language in [10]. These shall be interpreted as the appropriate mod q representation in $[0, q)$, for example -3 as 4 mod q in the simple case where $q = 7$. We shall also refer to the absolute value of a coefficient, which shall mean the absolute value of its representation in $[(1 - q)/2, (q - 1)/2]$ mod q.

For any integer K we define $R_{q,K}$ the set of K-*bounded* elements to be those elements of R_q whose coefficients all have absolute value less than K.

For any integer ω, we define S_ω the set of ω-*sparse* elements of R_q to be the $2^\omega \binom{n}{\omega}$ elements for which ω of the x_i are ± 1 mod q, and the remainder are zero.

2.2 The Ring-LWE and DCK problems

We here recall the Ring-LWE distribution, and the associated decision and search problems:

Definition 1 (Ring-LWE Distribution)**.** For an $s \in R_q$ and a distribution χ over R_q, a sample from the Ring-LWE Distribution $A_{s,\chi}$ over $R_q \times R_q$ is generated by sampling a uniformly at random in R_q, sampling e from χ, and outputting $(a, as + e)$.

Definition 2 (Decision Ring-LWE)**.** The Decision Ring-LWE Problem is to distinguish with non-negligible advantage between independent samples from $A_{s,\chi}$ where s is chosen once and for all, and the same number of *uniformly random* and independent samples from $R_q \times R_q$.

Definition 3 (Search Ring-LWE)**.** The Search Ring-LWE Problem is to recover s with non-negligible advantage from samples from $A_{s,\chi}$ where s is chosen once and for all.

We shall consider only the case where χ is the co-ordinate-wise uniform distribution on $\{-1, 0, 1\}$ mod q, which is called the *Decisional Compact Knapsack* (DCK) problem in [10].

The Ring-LWE problem is a special ideal-lattice case of the general Learning With Errors (LWE) problem defined over lattices:

Definition 4 (LWE Distribution)**.** For an $\mathbf{s} \in \mathbb{Z}_q^n$ and a distribution χ over $\mathbb{Z}$, a sample from the LWE Distribution $B_{\mathbf{s},\chi}$ over $\mathbb{Z}_q^n \times \mathbb{Z}_q$ is generated by sampling $\mathbf{b}$ uniformly at random in $\mathbb{Z}_q^n$, sampling e from χ, and outputing $(\mathbf{b}, \mathbf{b} \cdot \mathbf{s} + e \bmod q)$.

Definition 5 (Decision LWE)**.** The Decision LWE Problem is to distinguish with non-negligible advantage between independent samples from $B_{\mathbf{s},\chi}$ where $\mathbf{s}$ is chosen once and for all, and the same number of *uniformly random* and independent samples from $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Definition 6 (Search LWE)**.** The Search LWE Problem is to recover $\mathbf{s}$ with non-negligible advantage from samples from $B_{\mathbf{s},\chi}$ where $\mathbf{s}$ is chosen once and for all.

A single Ring-LWE sample $(a, as + e)$ corresponds to n LWE samples:

$$(\mathbf{b}_0, \mathbf{b}_0 \cdot \mathbf{s} + e_0 \bmod q), \ldots, (\mathbf{b}_{n-1}, \mathbf{b}_{n-1} \cdot \mathbf{s} + e_{n-1} \bmod q).$$

Each $\mathbf{b}_i$ is the vector of coefficients for the polynomial $\zeta^i a$, $\mathbf{s}$ is vector of coefficients from s, and the e_i are the coefficients of e. Therefore the bit-level hardness of Ring-LWE problems can be estimated from the bit-level hardness of LWE problems.

Concrete assessments of LWE security are given by Albrecht *et al.* in [1], and most recently and conservatively by Alkim *et al.* in [2], which analyses lattice sieving in conjunction with the Block Korkine Zolotarev (BKZ) 2.0 algorithm.

3 The GLP signature scheme

In this section we shall describe the GLP signature scheme as presented in [10]. The components of GLP are Key Generation, Sign and Verify. In addition to the parameters m, n, q in Section 2, there are integer parameters ω, κ, B and auxiliary functions H and F.

GLP requires a κ-bit hash function $H : \{0,1\}^* \to \{0,1\}^\kappa$ and an *encoding* function $F : \{0,1\}^\kappa \to S_\omega$ from the output of H to the ω-sparse elements of R_q.

The output size κ of the hash function H must have at least the required security level λ of the signature scheme, for instance κ could be 256 for up to 128 bits of security and H instantiated as SHA256. It must be chosen so that the output space of H is larger than the number of ω-sparse elements of R_q. The encoding function F must be chosen so that the probability of mapping two hash outputs to the same sparse element is less than one in 2^λ. This will be satisfied if the number of ω-sparse elements of R_q is larger than 2^λ.

To make signatures as compact as possible, GLP also uses a compression algorithm, which allows a polynomial in R_q to be efficiently represented by its 'high bits'. For any integer K we shall define the *K-rounding function* $\lceil \cdot \rfloor_K : R_q \to R_q$ as follows. We can uniquely write each $x_i \in [(1-q)/2, (q-1)/2]$ in the form $x_i = x_i^{(1)}(2K+1) + x_i^{(0)}$ where $x_i^{(0)} \in [-K, K]$ and

$x_i^{(1)} = \left(x_i - x_i^{(0)}\right)/(2K+1)$. Then we define

$$\left\lceil \sum_i x_i \cdot \zeta^i \right\rfloor_K = \sum_i x_i^{(1)} \cdot \zeta^i.$$

The following lemma from [10] shows that with high probability, a polynomial $y \in R_q$ can be efficiently represented by its rounded version $\lceil y \rfloor_K$.

Lemma 1. There is an algorithm $\texttt{Compress}(y, z)$ that for any q, n, B, ω where $2n(B-\omega)/q \geq 1$ and q is much larger than $2(B-\omega)/n$ takes as inputs $y \in R_q$, $z \in R_{q,B-\omega}$ and outputs z' in $R_{q,B-\omega}$ such that

1. $\lceil y + z' \rfloor_{B-\omega} = \lceil y + z \rfloor_{B-\omega}$
2. z' can be represented with only $2n + \lceil \log_2 (2(B-\omega)+1) \rceil \cdot \frac{6(B-\omega)n}{q}$ bits

with probability greater than 0.98 over the choices of y.

We shall now describe the operation of the GLP signature scheme.

Key Generation
Input: Uniformly-sampled public parameter $a \in R_q$.
Output: Private/public key pair (r, t).

1. Sample $s, e \in R_{q,1}$
2. $t \leftarrow as + e$
3. $r \leftarrow (s, e)$
4. Return (r, t)

The private key (s, e) can be theoretically represented in $2n \log_2(3)$ bits of memory, and the public key $as + e$ can be represented in $n \log_2(q)$ bits.

Sign

Input: Message μ, private key (s, e), public parameter a, public key $t = as + e$.
Output: Signature (z_1, z_2, c).

1. Sample $y_1, y_2 \in R_{q,B}$.
2. $c' \leftarrow H\left(\lceil ay_1 + y_2 \rfloor_{B-\omega} \mid \mu\right)$
3. $c \leftarrow F\left(c'\right)$
4. $z_1 \leftarrow sc + y_1$
5. $z_2 \leftarrow ec + y_2$
6. If any coefficient of z_1 or z_2 exceeds $B - \omega$ in absolute value then goto step 1
7. $z_2 \leftarrow \texttt{Compress}((az_1 - tc), z_2)$
8. Return (z_1, z_2, c)

Restarting in step 6 will reduce performance by extending the signing process. The per-coefficient probability of restarting is approximately $1-(2(B-\omega)+1)\,/\,(2B+1)$, so the overall probability of passing through these steps without restarting is $1 - \left(1 - \frac{2(B-\omega)+1}{2B+1}\right)^{2n}$. A signature (z_1, z_2, c) can be theoretically represented by $n \log_2(2(B-\omega)+1)$ bits of memory for z_1, by $2n + \lceil \log_2\left(2(B-\omega)+1\right)\rceil \cdot \frac{6(B-\omega)n}{q}$ bits for z_2, and by $\omega \log_2(2n)$ bits for c.

Verify

Input: Message μ, signature (z_1, z_2, c), public key t, public parameter a.

1. If any coefficient of z_1, z_2 has absolute value greater than $B - \omega$ then return Reject.
2. $d' \leftarrow H\left(\lceil az_1 + z_2 - tc \rfloor_{B-\omega} \mid \mu\right)$.
3. $d \leftarrow F(d')$
4. If $d = c$ then return Accept, otherwise return Reject.

4 GLYPH: A new instantiation of GLP

In this section we shall describe GLYPH, our new proposal to instantiate GLP, which comprises new parameters, a new rounding function and a new instantiation of the algorithm `Compress`.

We have instantiated the public parameter a for the scheme in a per-domain manner, where each certificate root authority chooses public parameters which it shares with its users. This addresses, as far as possible for a signature scheme, the threat identified in [2], where re-using a public parameter may allow a global all-for-the-price-of-one attack if an adversary can perform a massive precomputation, or if the public parameter was chosen in an untrustworthy way. Below the root authority level there is negligible advantage to further varying public parameters, since a single successful attack against the authority is in itself an all-for-the-price-of-one compromise.

Security is always of especial importance when choosing parameters for any signature scheme, as just one cryptanalytic success against a root authority's public key will allow an adversary to sign certificates at all levels of the trust hierarchy. It is for this reason that we have picked parameters for 128 bits of security using the conservative `NewHope` security analysis described in [2], which makes several assumptions about potential future developments in cryptanalysis that are optimistic from the attacker's perspective. We also include the parameter sets 'Set I' and 'Set II' proposed in [10] for comparison.

	[10] Set I	[10] Set II	GLYPH
m	1024	2048	2048
n	512	1024	1024
q	8383489	16760833	59393
B	16383	32767	16383
ω	32	32	16
Signature size (kB)	0.9	1.9	1.8
Secret key size (kB)	0.1	0.3	0.3
Public key size (kB)	1.5	3.1	2.0
Expected number of repetitions	7	7	7
Hamming weight of q	13	11	5
Conservative security level (bits)	< 80	91	137

We stress that we do not claim the 'Set I' or 'Set II' parameters can be broken by any attack known today in 2^{80} or 2^{91} basic operations, as `NewHope` is very optimistic about an attacker's capabilities. However, even if all of those capabilities were realised, GLYPH would still require more than 2^{128} basic operations to defeat.

We have chosen $m = 2048$ to be a power of 2 with $n = 1024$, which Section 5 shall show achieves 137 bits of security. As explained in Section 2, we could have chosen m to be prime, with $n = m - 1$ less than 1024. This would allow more compact signatures and keys, and still achieve the desired 128 bits of security, but the performance would be worse, as the Number

Theoretic Transform for m prime is considerably less efficient than for m a power of 2. However in practice a prime m achieving the required security is not much smaller than 1024, so key and signature sizes would be nearly unchanged, and therefore we do not recommend m prime in this case.

In line with the original GLP paper we have instantiated the hash function H with SHA256 as this provides the required 128 bits of security. We have instantiated the encoding function F using AES in counter mode, seeding with the input, and setting coefficients from the output. This is approach is different to [10], which used a function specifically tailored for $\omega = 32$. It leads to a small reduction in performance, but allows greater flexibility in parameter choice.

With our new parameter choices above, the `Compress` algorithm described in Lemma 1 fails so often as to be unusable, because our prime q is much smaller than in the original GLP paper and so the requirement for q to be much larger than $2(B - \omega)/n$ no longer holds. To address this we shall introduce a new compression function `Compress2` and a new function K-*floor* $\lfloor \cdot \rfloor_K$ which will take the place of the K-rounding function $\lceil \cdot \rfloor_K$ wherever it appears in the signing, compression and verification code.

Compress2

Input: $y \in R_q$, $z \in R_{q,K}$
Output:$z' \in R_{q,K}$.

1. for $i = 0$ to $n - 1$ do
2. $\quad z'_i \leftarrow$ `CompressCoefficient`(y_i, z_i)
3. end for

CompressCoefficient

Input: $u \in [0, q)$, $v \in [-K, K]$
Output: $v' \in \{-K, 0, K\}$

1. if $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K$ then $v' \leftarrow 0$; return
2. if $u \in [0, K)$ then $v' \leftarrow -K$; return
3. if $u \in [q - K, q)$ and $v > 0$ then $v' \leftarrow K$; return
4. if $\lfloor u + v \rfloor_K < \lfloor u \rfloor_K$ then $v' \leftarrow -K$; return
5. $v' \leftarrow K$; return

For any integer K we shall define the *K-floor function* $\lfloor \cdot \rfloor_K$ on the integers and on the quotient ring R_q as follows. For each integer x_i we shall write $x_i \bmod q = r_i(2K+1) + s_i$ where r_i is an integer and $s_i \in [0, 2K]$. We shall denote r as $\lfloor x \rfloor_K$, and then

$$\left\lfloor \sum_i x_i \cdot \zeta^i \right\rfloor_K = \sum_i \lfloor x_i \rfloor_K \cdot \zeta^i.$$

The Lemma below shows that whenever certain modular conditions are satisfied by the parameters, `Compress2` outperforms `Compress`, as the output can be represented with less memory, and succeeds every time rather than with probability 0.98. Most importantly, these conditions can be satisfied by a range of q, such as the choice recommended above that otherwise would be too small to use with `Compress`.

Lemma 2. If $q \geq 2K+1$ is such that $q = r(2K+1) + s$ where r is an integer and $s \in [K, 2K]$ then the output z' of `Compress2`(y, z) is such that

1. z' can be represented with $n \log_2(3)$ bits.
2. $\lfloor y + z' \rfloor_K = \lfloor y + z \rfloor_K$.

Proof. 1. There are only three possibilities for any of the n coefficients of z', as they can be only $-K$, 0, or K. Therefore z' can be represented with $n \log_2(3)$ bits.

2. The output z' is specified by the coefficients u of y and v of z, therefore it is sufficient to show that $\lfloor u + v' \rfloor_K = \lfloor u + v \rfloor_K$.

   If $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K$ then by the definition of `CompressCoefficient` (line 1) we have $v' = 0$ and so $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K = \lfloor u + v' \rfloor_K$.

   Otherwise, if $u \in [0, K)$ then $v' = -K$ (line 2) and so

   $$u + v' \bmod q = q + u - K \in [r(2K+1) + s - K, r(2K+1) + s)$$

   from which we see $\lfloor u + v' \rfloor_K = r$, since $s - K \geq 0$ by the condition on s. Now, it cannot be the case that $u + v \geq 0$ because this would imply that $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K = 0$ contradicting line 1, so we must have $u + v < 0$ and $\lfloor u + v \rfloor_K = r = \lfloor u + v' \rfloor_K$.

   Otherwise, if $u \in [q - K, q)$ and $v > 0$ then $v' = K$ (line 3) and so

   $$u + v' \bmod q = u + K - q \in [0, K)$$

   from which we see $\lfloor u + v' \rfloor_K = 0$. Now $u + v \bmod q \in [q - K, q) \cup [0, K)$ so $\lfloor u + v \rfloor_K$ is either $\lfloor u \rfloor_K$ or 0, but $\lfloor u + v \rfloor_K \neq \lfloor u \rfloor_K$ by the above. We conclude $\lfloor u + v' \rfloor_K = 0 = \lfloor u + v \rfloor_K$.

   Otherwise, if $v' = -K$, then $\lfloor u + v' \rfloor_K$ is either $\lfloor u \rfloor_K - 1$ or $\lfloor u \rfloor_K$, and $\lfloor u + v \rfloor_K < \lfloor u \rfloor_K$ (line 4) but also $\lfloor u + v' \rfloor_K \leq \lfloor u + v \rfloor_K$ so we conclude $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K - 1 = \lfloor u + v' \rfloor_K$.

   Otherwise, $v' = K$ and $\lfloor u + v' \rfloor_K$ is either $\lfloor u \rfloor_K + 1$ or $\lfloor u \rfloor_K$, and $\lfloor u + v \rfloor_K > \lfloor u \rfloor_K$ (line 5) but also $\lfloor u + v' \rfloor_K \geq \lfloor u + v \rfloor_K$ so we conclude $\lfloor u + v \rfloor_K = \lfloor u \rfloor_K + 1 = \lfloor u + v' \rfloor_K$.

□

5 Security Analysis

We now analyse the security of GLYPH. Although GLP comes with a security proof that a succesful forger will be able to solve the DCK instance of the Ring-LWE problem, we also need to assess the security of the specific parameters we have chosen.

5.1 Primal key recovery attack

In a primal key recovery attack, the adversary attempts to recover the secret key s from the public key $t = as+e$, which can be viewed as a Ring-LWE sample. The strongest known attacks against Ring-LWE consider it as an instance of the Learning With Errors (LWE) problem. We assume an adversary has access to n samples $(\mathbf{a}_i, t_i) \in \mathbb{Z}^{n+1}$ of the form $\mathbf{a_i} \cdot \mathbf{s} + e_i = t_i \bmod q$ for a fixed secret $\mathbf{s} \in \mathbb{Z}^n$ that is to be recovered and secret error terms $e_i \in \mathbb{Z}$. To optimise the attack they may choose to use a number l of samples that is less than n.

Because our analysis of the primal key recovery merges several analyses in the literature, we present the approach in some detail.

5.1.1 BKZ attack

The l samples $(\mathbf{a_i}, t_i)$ allow us to construct a matrix $A \in \mathbb{Z}_q^{l \times n}$ whose rows are the a_i and vector $\mathbf{t} \in \mathbb{Z}_q^l$ whose entries are the t_i. The adversary builds the lattice

$$\Lambda = \left\{ (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbb{Z}^{l+n+1} : (A| - I_l| - \mathbf{t}) \cdot (\mathbf{x}, \mathbf{y}, \mathbf{z}) = 0 \bmod q \right\},$$

which is of dimension $d = l + n + 1$. They then look to solve the unique Short Vector Problem (unique-SVP) problem in Λ to recover $\mathbf{s}$.

We use the conservative geometric series assumption, that is that when BKZ is run with block size b on a lattice with dimension d, it finds a basis $\mathbf{b}_i^*$ whose Gram-Schmidt norms are given by

$$\|\mathbf{b}_i^*\| = \delta^{d-2i-1} \mathrm{Vol}\,(\Lambda)^{1/d}$$

where $\delta = \left((\pi b)^{1/b} \cdot b/2\pi e \right)^{1/2(b-1)}$. The cost of finding the shortest vector within the block is then estimated conservatively to be $2^{0.292b}$ basic operations with a classical computer, or $2^{0.2075b}$ basic operations with a quantum computer.

The standard deviation of coefficients drawn from $\{-1, 0, 1\}$ is 0.81, and the block size b is selected to be as small as possible subject to it being possible to find the unique solution, which is detected when $\mathbf{s}$ has projected norm $0.81\sqrt{b}$ smaller than $\left\|\mathbf{b}_{d-b}^*\right\|$.

5.1.2 Exhaustive and Meet-in-the-middle attacks

It is possible to exhaust over all possible coefficients of s with asymptotically[1] 3^n basic operations. Following the approach described in Section 5.1 of [1] this can be reduced to $3^{0.5n}$ basic operations with a meet-in-the-middle attack, but with an increased memory requirement. An adversary computes and stores a vector J_0 of the inner products of the first half of each $\mathbf{a_i}$ with the first half of each candidate s, and likewise a vector J_1 of the inner products of the second halves. They then look for instances where $J_0 + J_1 - \mathbf{t} \bmod q$ is small.

5.1.3 Hybrid attack

It is possible to combine a BKZ attack and a meet-in-the-middle attack. A meet-in-the-middle attack is run to recover the final r co-ordinates of $\mathbf{s}$. This is combined with a precomputed basis of the first $n - r$ co-ordinates to recover the full secret. We can view the BKZ and meet-in-the-middle attacks above as special cases of this hybrid, when $r = 0$ or $r = n$, so in this sense it is the 'only' primal key recovery attack that we need to analyse.

In more detail, let A' be the first $n - r$ columns of A, and let A'' be the remaining r columns. Let $\mathbf{s}'$ be the first $n - r$ co-ordinates of $\mathbf{s}$ and $\mathbf{s}''$ be the final r co-ordinates. Let $\mathbf{t}'$ be the first $n - r$ co-ordinates of $\mathbf{t}$ and $\mathbf{t}''$ be the final r co-ordinates. An adversary constructs the lattice

$$\Lambda = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{Z}^{l+n-r} : \left(A' | - I_l\right) \cdot (\mathbf{x}, \mathbf{y}) = 0 \bmod q \right\},$$

which is of dimension $d = l + n - r$. An adversary runs BKZ on this lattice with pre-chosen block-size b to obtain a reduced basis of vectors $\mathbf{b}_i^*$.

The adversary then runs a meet-in-the-middle attack on s''. They compute and store a vector of inner products J_0 of the first half of each row from A'' with the first half of each candidate s'', and likewise the inner products J_1 of the second halves. They then use their precomputed basis to perform Bounded Distance Decoding (BDD) and find instances where $J_0 + J_1 - \mathbf{t}'' \bmod q$ is very close to a point in Λ. For such instances, they find a close vector s' in Λ. For BDD to succeed, it is necessary that the projected norm $0.81\sqrt{b}$ of $\mathbf{s}'$ be smaller than $\|\mathbf{b}_{d-b}^*\|$.

5.2 Dual key recovery attack

In a dual key recovery attack, the adversary attempts to distinguish the public key $t = as + e$ from a uniformly sampled element of R_q via a BKZ lattice attack. Unlike our primal analysis, the approach here mirrors exactly that in [2] so we omit details.

[1] In reality there will be an additional factor of $2n$ in the cost so this is optimistic from the attacker's perspective.

5.3 Forgery attack

In a forgery attack, an adversary attempts to produce a forged signature for a message μ of their choice, without knowledge of the secret key. We assume the following approach:

Forgery attack

Input: Message μ, public key t, public parameter a.

1. Sample uniform $y \in R_q$.
2. $c' \leftarrow H(y, \mu)$.
3. $c \leftarrow F(c')$.
4. $h \leftarrow tc + y$
5. Use BKZ to find $z_1, z_2 \in R_{q,B-\omega}$ with $az_1 + z_2 = h$.
6. Return (z_1, z_2, c).

The critical stage of the forgery attack is step 5, which we approach by constructing a random-SVP instance and solving using BKZ. Unlike the unique-SVP instance considered for key recovery, the adversary does not have flexibility in the number l of samples that they use. The required (z_1, z_2) will be a vector of length $(B-\omega)\sqrt{2n}$, whose projection onto blocksize b is $(B-\omega)\sqrt{b}$. We therefore pick b to be as small as possible subject to this projection being smaller than $\|\mathbf{b}_0^*\| = \delta^{d-1} q^{n/d}$.

5.4 Security results for GLYPH parameters

We record the results of the above security analysis for the GLYPH parameters recommended in Section 4.

	primal key recovery	dual key recovery	forgery
b	477	601	469
l	746	856	1024
r	176	-	-
Attack cost ($\log_2$ basic classical computer operations)	139.5	175.8	137.17

6 Conclusion

Lattice-based cryptography is a promising post-quantum alternative to classical public key cryptography. The GLP algorithm is a competitive digitial signature scheme, for which we have recommended GLYPH, a new instantiation to address developments in lattice cryptanalysis since the original proposal in [10].

References

[1] M.R. Albrecht, R. Player, and S. Scott. On the concrete hardness of Learning with Errors. *Journal of Mathematical Cryptology*, 9(3):169–203, 2015.

[2] E. Alkim, L. Ducas, T Pöppelmann and P. Schwabe. Post-quantum key exchange — a new hope. In *USENIX Security 2016*.

[3] S. Akleylek, N. Bindel, J. Buchmann, J. Krämer, and G. A. Marson. An Efficient Lattice-Based Signature Scheme with Provably Secure Instantiation. `http://eprint.iacr.org/2016/030`.

[4] P. S. L. M. Barreto, P. Longa, M. Naehring, J. E. Ricardini, and G. Zanon. Sharper Ring-LWE Signatures. `http://eprint.iacr.org/2016/1026`.

[5] 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. `http://eprint.iacr.org/2014/599`.

[6] L. G. Bruinderink, A. Hülsing, T. Lange, and Y. Yarom. Flush, Gauss, and Reload – a Cache Attack on the BLISS Lattice-Based Signature Scheme *Cryptographic Hardware and Embedded Systems CHES 2016*, volume 9813 of Lecture Notes in Computer Science, pages 323–345. Springer, 2016

[7] A. Chopra. Improved Parameters for the Ring-TESLA Digital Signature Scheme. `http://eprint.iacr.org/2016/1099`

[8] L. Ducas, A. Durmas, T. Lepoint, and V. Lyubashevsky. Lattice signatures and Bimodal Gaussians. In R. Canetti and J. A. Garay, editors, *CRYPTO 2013*, Part I, volume 8042 of LNCS, pages 40–56, Santa Barbara, CA, USA, Aug. 18–22, 2013. Springer, Heidelberg, Germany.

[9] T. Espitau, P–A Fouque, B. Gerard, and M. Tibouchi. Side-Channel Attacks on BLISS Lattice-Based Signatures – Exploiting Branch Tracing Against strongSwan and Electromagnatic Emanations in Microcontrollers. `http://eprint.iacr.org/2017/583`

[10] T. Guneysu, V. Lyubashevsky, and T. Pöppelmann. Practical Lattice-Based Cryptography: A Signature Scheme for Embedded Systems. In E. Prouff and P. Schaumont, editors, *Cryptographic Hardware and Embedded Systems CHES 2012*, volume 7428 of Lecture Notes in Computer Science, pages 530–547. Springer, 2012.

[11] P. Pessl and L. G. Bruinderink, and Y. Yarom. To BLISS-B or not to be - Attacking strongSwan's Implementation of Post-Quantum Signatures. http://eprint.iacr.org/2017/490

[12] V. Singh. A Practical Key Exchange for the Internet using Lattice Cryptography. http://eprint.iacr.org/2015/138.

[13] V. Singh and A. Chopra. Even More Practical Key Exchanges for the Internet Using Lattice Cryptography http://eprint.iacr.org/2015/1120.