Can Lattice Signature be as Efficient as Lattice Encryption?

Dingfeng Ye

State Key Laboratory of Information Security, Institute of Information Engineering,
Chinese Academy of Sciences, Beijing 100093, China
yedingfeng@iie.ac.cn

Abstract. Existing lattice signature schemes are much less efficient than encryption schemes due to the rejection sampling paradigm. We give a construction of comparable efficiency with lattice encryption that avoids sampling using structured secrets together with temporary keys. Structured secrets (and randoms) also improve existing lattice encryption schemes to nearly the same extreme efficiency. Our signature scheme allows the same parameters of any encryption schemes (a variation of the basic form is needed when the modulus is as small as 1-byte) and has comparable efficiency with our extreme encryption efficiency. For lightweight implementation, our techniques allow integrating of public-key encryption and signature in a simple circuit which only needs to do small integer additions as the main part of the computation.

Keywords: Lattice Signature · Lattice Encryption.

1 Introduction

Lattice encryption and signature are two kinds of public key cryptography supposed to stand with or even replace RSA and ECC, in light of its resistance to quantum computing attacks, and also for its low computational cost and speed advantage over the later two. Some problems with this technology trend may arise, here are two: (1) Quantum computing remains a pure mathematical notion whose physical feasibility might be excluded in future, the phrase "post-quantum" may lose its power in supporting prevailing of lattice cryptography. (2) At present, lattice signature is not compatible with lattice encryption in efficiency: the former is much slower (even slower than ECC) and bigger in size than the latter; this is another disadvantage besides the size issue compared with RSA and ECC. To make lattice cryptography more widely applicable, it is desirable to have a more efficient and light-weighted lattice signature scheme,

and to make extreme use of the speed and lightweight advantages of lattices in parameter choices of both encryption and signature.

There are two main approaches to obtain lattice signatures: one is trapdoor sampling a short integer solution (SIS) to a hard SIS problem; the other is a Fiar-Shamir heuristic in proving knowledge of secrets associated with the public key. The main efficiency obstacle with both approaches is that the plain treatment is insecure: signing leaks secret information and enough signatures may recover the secret key statistically. To remedy this, the existing approach introduces a large uniform noise or a smaller Gaussian noise together with a computation-intensive rejection sampling to get a distribution independent of the secret, which makes the signature schemes much less efficient compared with the lattice encryption schemes at the same security level. Our first objective is a lattice signature scheme without the requirement of large noise and/or complex sampling.

Since the trapdoor sampling approach can not be improved at speed to the extent of our objective, we follow the second approach, and we worked in a RLWE context: all objects are elements of a ring $R = Z[T]/(T^n + 1)$ or $R_q = Z_q[T]/(T^n+1)$, n is a power of 2 or a prime, q is the modulus. Let $X = x_1a+x_2$ be a public key with private key x_1, x_2, we can view a signature as a proof of knowing x_1, x_2 given X. Typically such a proof has the form $(Y = y_1a+y_2, z_1, z_2)$, where $z_1 = y_1 + cx_1$, $z_2 = y_2 + cx_2$, c is the challenge determined by Y, X. For ease of presentation we omit the index $i = 1, 2$, and simply put it for one copy of variables x, y, z. We call such a scheme plain if y is drawn from uniform or Gaussian distribution. It is well known that plain scheme is not secure, and can be attacked as follows: let * denote the automorphism of $R : T \mapsto T^{-1}$, then $c^*c \approx |c|^2$, and $c^*z \approx |c|^2x + c^*y$, where the secret x becomes a constant translation of a known balanced distribution, which can be easily recovered given a few samples. To stop such attack, known methods make distribution of z independent of x by sampling y depending on cx (so-called "reject sampling"), which needs y have a much larger size (for example Tesla [2]) than cx and/or needs intensive floating-point operations (Bliss [4]).

We observe that using a temporary public key may make things different: each temporary key only signs a restricted number (r) of times to invalidate the above linear statistical attack, and then a new temporary key is certificated; this certification is easier to secure. Let $W = w_1X+w_2$ be a temporary public key, the certificate of W by X is of the form $(Y = y_1a+y_2, z_1, z_2)$, where $z_i = y_i+cw_1x_i$, $i = 1, 2$ (In the following, i will be omitted). Now the above linear statistical attack does not work, since the translation wx seems wild because of w. The only known attack is the quadratic statistical attack introduced in [10, 7]. The

basic observation of this attack is that any quadratic function of z has a mean value (called quadratic statistical deviation) which equals a quadratic function of x, if y is balanced sampled independently of w, x, and knowing the quadratic functions of x can recover x. There are two obvious ways to invalidate this attack: the first is balancing the quadratic functions of z by sampling techniques (there are ways more efficient than reject sampling); the second is our approach in this paper: structured y. We make

$$y = \sum_{1 \le j \le t} y_j s_j,$$

where s_j are secret but constant, the number t is a small integer, say $t = 10$. Now the quadratic statistical deviation of z is a function of x and s_js, we can expect this large number ($(t+1)n$) of unknowns can hardly be recovered from far few ($n/2$) equations. In this basic form of our construction: the requirement of large y in conventional lattice signatures vanishes which allows the signature and encryption to share the same parameters.

The security of our scheme mainly relies on the difficulty to recover the secret from the certificates of an arbitrary number, a new problem introduced in this work. The main attack we can figure out is as follows: guess a part x' of the secret x, and using statistical methods to verify the guess. Note that

$$c^* x'^* z = c^* x'^* x' w + c^* x'^* = |c|^2 |x'|^2 w + ot,$$

where ot is the other part of the observable $c^* x'^* z$. The ratio

$$\rho = |c|^2 |x'|^2 / |ot| \approx (|c||x'| n^{1/2}) / |z|$$

is an indicator of the hardness of the new problem. At present we don't know what is the best statistic attack, so we don't know the exact requirement about ρ. A simple attack establishes the upper bound ρ_0 of ρ for the new problem to be hard. To get smaller ρ, we use a variation of the Σ-protocol and let z have the form $xw + c_1 y_1 + c_2 y_2$, which leads to $\rho = (|x'| n^{1/2})/|z|$. In our examples we have $\rho < 0.1\rho_0$, which we think is enough to stop any statistical attack.

Next, we consider integrating both encryption and signature in a simple computing module. We could simply add a signature scheme to the existing lattice encryption schemes with the temporary W working as the encryption public key. We can do much better. In fact, using structured secret (and/or noise) in an encryption scheme will greatly improve its efficiency. Typically the

public $Enc()$ needs to compute rW (and rX), where the random r has the same distribution with the secret key w. If this distribution is Gaussian, the multiplication by r is treated as the multiplication of general elements in the ring R, which is costly. If we replace the distribution with a semi-Gaussian given by an algebraic expression of some very small and simple elements, then the multiplication by r would just cost tens of shift-additions in R, which makes all lattice encryption scheme achieve nearly the same level of extreme efficiency.

We say a distribution semi-Gaussian with parameter σ, if it is balanced (with mean value 0) and has mean square value σ^2, and not too far away from the Gaussian distribution D_σ. A distribution of vector is called semi-Gaussian if its components all are semi-Gaussian with the same parameter, such a distribution will be denoted as $[\sigma]$. Algebraic operation of semi-Gaussian distributions in R results in semi-Gaussian distribution:

$$[\sigma_1] + [\sigma_2] = [(\sigma_1^2 + \sigma_2^2)^{1/2}]$$

$$[\sigma_1] \cdot [\sigma_2] = [(n\sigma_1^2\sigma_2^2)^{1/2}]$$

Let S_d denote ring elements with exactly d non-zero entries of $\{-1, 1\}$. Multiplication by an element in S_d costs d shift-additions(the unit of computational cost in this work) in the ring, we also say element has cost d. We may treat the uniform distribution of S_d as semi-Gaussian $(d/n)^{1/2}$, and we could approximate any Gaussian parameter with some algebraic expression of very small S_ds. such an element costs no more than the sum of these ds. For the extreme example, let $S(2^k)$ denote the product of k independent copy of S_2, then it achieves the Gaussian parameter $(2^k/n)^{1/2}$ and only has cost $k + 1$. Of course, we should choose an algebraic expression such that the resulted RLWE problem can not be attacked simply by birthday type exhausting(No other attack is known to exploit this secret structure). It is not hard to see that the cost of structured secret w and random r can always be reduced to tens.

The long term secret key x is chosen in S_d for a d big enough to ensure the hardness of the RLWE instance. Any existing lattice encryption scheme can be transformed into a PKC module(doing both encryption and signature) as above, and in the signing algorithm, we can make the sum of the cost of the y_js as small as 7. So the signature scheme is at the same efficiency level as the most optimized encryption algorithm. This makes the efficiency of our lattice PKC depend only on the parameter n and q. For lightweight implementation, small q is favorable, it seems that 1-byte $q \approx 256$ is the extreme lower bound of modulus size for security level no less than 128 bits. We give explicit parameters

the examples according to known attacks: lattice reductions [6, 8, 3], brute force attacks at scarce secrets [9], statistical tests in this work. Another issue related to both security and efficiency is how many times a temporary signing key can be used. We also gives computations to decide this according to the best attack we know.

Note that integer multiplication is not mentioned. In fact, the signing algorithm should ensure the Euclidian norm of z lies in the expected scope, computing the Euclidian norm needs multiplication. This task can be assigned to the supporting software if we wish the hardware extremely light weighted. The verification of the signature (and certificate) involves multiplication of general elements in the ring, which would be delegated to the environment. Such delegation needs to be carried out even when the environment is not trusted. We give a simple protocol for this task.

To summarize, our signature scheme allows the same key used both for encryption and signing, and both are extremely efficient. It is best suited to be exploited in a PKI context, where transporting and verification of the temporary key may be delegated to the PKI service provider. The basic form of our signature is also in leading size efficiency, but the variation seems better suited for 1-byte q. We also give a simple protocol for delegating the main workload for signature verification, which means we can integrate the whole public key functions in a simple device that need not do integer multiplication. This would allow extremely lightweight hardware implementation of PKC for the first time.

2 Preliminary

Let $R = Z[T]/(T^n + 1)$, and $R_q = R/(qR)$, where n is a power of 2 or a prime, q is a positive integer called the modulus. Elements of R or R_q are also viewed as vectors of length n: $a = \sum_i a_i T^i = (a_0, a_1, \cdots a_{n-1})$. $< a, b >$ denotes the inner product of a, b, and $|a| =< a, a >^{1/2}$ denotes the euclidian norm. We also use capital letters $X, Y, A, B \cdots$ to denote elements of R or R_q. Let * denote the automorphism of $R : T \mapsto T^{-1}$, then $a^*a \approx |a|^2$ for any (considered in this work) random variable a in the ring, where '$\approx$' means the difference of the two sides has zero mean value.

For any distribution χ, $x \leftarrow \chi$ means sampling x according to χ. When S is a set, $x \leftarrow S$ refers to the uniform distribution of S. S_d denotes vectors of length n with exactly d non-zero entries of $\{-1, 1\}$.

Gaussian distributions χ_σ over $\mathbb{Z}$:

$$\Pr[x] = e^{-x^2/2\sigma^2} / \sum_{i\in\mathbb{Z}} e^{-i^2/2\sigma^2}$$

It is well known that ($E(\cdot)$ means mean value)

$$E(|x|^2 : x \leftarrow \chi_\sigma) = \sigma^2$$

$$erf(k) := Pr[|x| > k\sigma : x \leftarrow \chi_\sigma] = (2/\pi)^{1/2} \int_k^\infty e^{-x^2}/2dx$$

We say a distribution semi-gaussian with parameter σ, if it is balanced(with mean value 0) and has mean square value σ^2, and not too far away from χ_σ. A distribution of vector is called semi-Gaussian if its components all are semi-Gaussian with the same parameter σ, such a distribution will be denoted as $[\sigma]$. Algebraic operation of semi-Gaussian distributions in R results in semi-Gaussian distribution:

$$[\sigma_1] + [\sigma_2] = [(\sigma_1^2 + \sigma_2^2)^{1/2}]$$

$$[\sigma_1] \cdot [\sigma_2] = [(n\sigma_1^2\sigma_2^2)^{1/2}]$$

Algebraic combinations of S_ds will be abreviated as follows: $S(a \times b + c \times d + \cdots)$ will denote $S_a \times S_b + S_c \times S_d + \cdots$, and it has the Gaussian parameter $((a \times b + c \times d + \cdots)/n)^{1/2}$.

RLWE: given $a \leftarrow R_q$, $b = as + e$, where $s, e \leftarrow \mathcal{K}$ for some distribution $\mathcal{K}$, to find s, e.

RSIS: given $a, b, c \leftarrow R_q$, to find $u, v \in R$, such that $au + bv = c$ and $|u|^2 < \mathcal{D}$, $|v|^2 < \mathcal{D}$ for some bound $\mathcal{D}$.

We are not concerning the hardness of RLWE (RSIS) asymptotically, instead, we would like to evaluate the concrete security level (in bits) of the problem at specific parameters according to known attacks, e.g. using the model of [1]. We regard the security level is equally good as long as it exceeds the claimed number of bits. For $n > 500$, 1-byte q, and 128-bit security, we set $\mathcal{D} = n^* 40^2$ in the SIS problem which is associated with the problem to forge a lattice signature; we think any semi-gaussian distribution $\mathcal{K}$ with $k \geq 64$ non-zero entries is ok for the RLWE problem which is associated with the basic security assumption of both encryption and signature. For $q = 1024$, we set $\mathcal{D} = n^* 80^2$, $k \geq 80$; For $q = 3 \times 2^{12} + 1$, we set $\mathcal{D} = n^* 220^2$, $k \geq 256$.

Σ protocols: a Σ protocol (may not be secure in this work) means the process to prove knowledge of a witness w satisfying a relation $R(x, w)$ which works

as follows: the prover first computes a commitment $u = COM(x, w, r)$; the verifier gives a random challenge c; the prover then computes a respond $a = RES(x, w, r, c)$; and the verifier computes $Verify(u, c, a, x)$. A Σ protocol is called zero knowledge (ZK) if (u, c, a) is statistically independent of w; witness indistinguishable (WI) if (u, c, a) is computationally independent of w; witness hiding if it is hard to recover w given unbounded samples of (u, c, a). Any ZK Σ protocol can be transferred to a signature scheme by the FS (Fiat-Shamir) heuristic: $c = h(u, m)$, whose security can be proved if the cryptographic hash function h is modeled as a random oracle. We assume FS heuristic also works for any witness hiding Σ protocol.

When w, r is in a ring, and R is a linear relation over the ring, then the basic form of $RES(x, w, r, c)$ is $r - wc$. Variations of the basic form might be $w - cr$,$c_1w + c_2r$, $w + c_1r_1 + c_2r_2 \cdots$ etc.

By public-key cryptography (PKC) we refer to two main functions: encryption and signature. In both cases, there is a key generating algorithm:

$$KeyGen(para) \rightarrow (sk, pk)$$

where $para$ is the system parameter, sk, pk are the private and public key respectively. With public-key, anyone can encrypt a message m or verify a signature τ on m:

$$Enc(pk, m) \rightarrow c$$

$$Verify(pk, m, \tau) = 0 \text{ or } 1$$

The security requirement is that only with the secret key, one can get any information about m from the ciphertext c, or produce a signature τ on m accepted by the verification process. Formal definitions are referred to [5].

In lattice PKC, the main attacks are concerned with the RLWE problem instances appearing in the key pair, ciphertext, and signature; and the SIS problem in forging a signature. We don't give formal assumptions to prove attacking our scheme can be reduced to these problems. We follow the "design and attack" paradigm, we are mainly concerned with known attacks on concrete problems or constructions. The formal proof may go straightforwardly as the suitable definition is given, but it would be irrelevant to our claims of security level λ.

3 The new lattice signature scheme

Our signature consists of two parts: a plain signature with a temporary key and a certificate of the temporary key; both are transformed from Σ-protocols via

standard Fiat-Shamir heuristic, so we just describe the underlying Σ-protocols to specify our signature scheme. For the certificate part, we need the Σ-protocol to be secure (witness hiding) to allow 2^{64} times of use of long term key. For each temporary key, we need to decide the number of times a temporary key can be used. We give the rationale for our design in this section.

We only consider only the following key generation:

$$Keyen(a) := \{x_1, x_2 \leftarrow \mathcal{K}; sk = (x_1, x_2); pk = X = ax_1 + x_2; \}$$

where $a \leftarrow R_q$ is the system parameter, $\mathcal{K}$ is a distribution defined by scheme.

Let $X = x_1a + x_2$ be a public key with private key x_1, x_2. In a typical Σ-protocol for proving knowledge of x_1, x_2 given X, the proof is of the form $(Y = y_1a + y_2, z_1, z_2)$, where $z_1 = y_1 + cx_1$, $z_2 = y_2 + cx_2$, c is the challenge determined by Y, X and message m:

$$c_0 = h(X, Y, m)$$

$$c = f(c_0)$$

where h is a cryptographic hash function modeled as a random oracle, f is a map whose image is the uniform distribution of some S_{d_c} with $|S_{d_c}| > 2^\lambda$. Since the treatment of the two copies z_1, z_2 is the same, we will omit the subscript $1, 2$ and simply say x, y, z etc. We call such a protocol plain if y are drawn from a uniform or (semi-)Gaussian distribution.

It is well known that a plain protocol is not secure, and can be attacked as follows. Let $(\cdot)^*$ be the ring homomorphism defined by $T \mapsto T^{-1}$, then

$$c^*z \approx |c|^2x + c^*y,$$

the right hand is a translation proportional to x from a known distribution, so x can be easily determined given a few samples: assume y has gaussian parameter σ, then the sum of k samples of c^*z is the sum S of $k|c|^2x$ and a semi-gaussian with parameter $k^{1/2}|c|\sigma$, as long as $k|c|^2 >> k^{1/2}|c|\sigma$ or $k >> \sigma^2/|c|^2$, we get $x \approx S/(k|c|^2)$. We call this attack the linear statistic attack.

To stop such attack, known methods make distribution of z independent of x by sampling y depending on cx, which needs y have a much larger size (for example Tesla) than cx and/or needs intensive floating-point operations (Bliss). We circum this attack by introducing temporary keys and structured y.

A temporary key pair (tsk, tpk) is generated taking X as system parameter:

$$Keyen(X) := \{w_1, w_2 \leftarrow \mathcal{K}; tsk = (w_1, w_2); tpk = W = w_1 X + w_2; \}$$

Our signature consists of two parts: the first is the plain signature with respect to the temporary key pair and system parameter X; the second is a certification of W by X using secrete key sk. Now the certification of W is of form $(Y = y_1 a + y_2, z_1, z_2)$, where $z_1 = y_1 + cw_1 x_1$, $z_2 = y_2 + c(w_1 x_2 + w_2)$, and

$$c_0 = h(X, Y, W)$$

$$c = f(c_0).$$

The actual certificate is (W, c_0, z_1, z_2), since Y can be recovered from the later. In the following, the subscripts $1, 2$ will be omitted. The verification of the signature is the "and" result of verification of both the plain signature and the certificate, which in addition to verifying the relations formulated above, also verifying $|z|^2 \leq B_z$ for all the four z elements and some bound B_z.

Now the above linear statistical attack does not work, since the translation wx seems wild because of w. The only known attack is the quadratic statistical attack introduced in [10, 7]. This attack can be formulated as follows: consider the following random variable

$$z^* z = (c^* c x^* x)(w^* w) + y^*(cxw) + y(cxw)^* + y^* y$$

where w, y are variables. The later sounds have zero mean value (or $|y|^2$)) for typical choices of w, y distribution, so the constant $x^* x$ is exposed, which can be exploited to recover x.

To deal with this attack we simply use structured y, so that the mean value of $z^* z$ depends on secrets implied in y. We make

$$y = \sum_{1 \leq j \leq t} y_j s_j,$$

where s_j are secret but fixed; the number t is a small integer, say $t = 10$. Now the mean value of $z^* z$ (or any the quadratic function of z) is a function of x and s_js, we can expect this large number $((t + 1)n)$ of unknowns can hardly be recovered from far few ($n/2$, by symmetry $z^* z$ only has $n/2$ different components) equations; we can not handle it even for $t = 1$. In our construction, bigger t does not mean more computation (we can choose simpler y_j, s_j to balance), it only

adds up a secret key size. So it is not expensive to have a bigger t when further analysis suggests. This is the basic form of our constructions: the requirement of large y in conventional lattice signatures disappears.

The security of our scheme mainly relies on the difficulty to recover the secret key from certificates of arbitrary number, which is the main new problem introduced in this work. The main attack we can figure out would follow the approach: guess a part x' of the secret x, and using statistical methods to verify the guess. Note that

$$c^*x'^*z = c^*x'^*x'w + c^*x'^* = |c|^2|x'|^2w + ot,$$

where ot denotes the other part of the observable $c^*x'^*z$, if $|c|^2|x'|^2$ is too big relative to the size of ot, this may be statistically identified. A simple such test is just to record the biggest absolute value of components of $c^*x'^*z$ for a big sample pool of size $L \leq 2^{64}$. Assume ot and c^*r^*z is semi-gaussian with parameter σ for all wrong guesses r (with total number K), and let

$$\rho = |c|^2|x'|^2/\sigma = |z|/(|c||x'|n^{1/2})$$

where $\sigma = |z|/n^{1/2}$, then the condition for the test to to distinguish a correct guess from wrong guesses is

$$erf^{-1}((n^*K^*L)^{-1}) < erf^{-1}(k^*L)^{-1}) + |w|_1\rho$$

where L is the expected number of guesses to hit a correct guess, $|.|_1$ denotes the 1-norm,i.e. the maxim of absolute value of the components; k is the rate to hit this maxim. Let

$$\rho_0 = \min_{K^*L \leq 2^\lambda}(erf^{-1}((n^*K^*L)^{-1}) - erf^{-1}(k^*L)^{-1}))/|w|_1$$

then a necessary condition for the test to succeed is that $\rho < \rho_0$.

The problem in the actual case is much harder than the above modeling: the gaussian parameter of c^*r^*z depends on r and is secret; those r close to x' can not be distinguished with x'. So the test could at best return a set of candidates close to the correct guess, even though $\rho < \rho_0$ is satisfied. For very sparse x(which is the case in our examples), such information might be enough to recover x, so we should let $\rho << \rho_0$, for example $\rho < 0.1\rho_0$. For x with gaussian parameter ≥ 1, such information could only reduce the gaussian parameter of the related RLWE problem. So we think the new problem should not be weaker

when a larger gaussian parameter of x is used: though it results in smaller ρ_0, and larger ρ.

At present we don't know what is the best statistic test for such a task, so we don't know recommend the requirement about ρ for sparse x. we just minimize ρ as allowed by other requirements. A simple way to reduce ρ is using the following variation of the basic form: let the z has the form $xw + c1y1 + c2y2$, which leads to

$$\rho \approx (|x'|n^{1/2})/|z|$$

the factor $|c| = {d_c}^{1/2}$ is get rid of. The commitments $Y1, Y2$ need to appear in the certificate in this case. Note that the form $z = xw + cy$ is not secure since $c^*z/|c|^2$ is an approximation of y.

The following are data in our examples: For $q = 3^*2^{12} + 1$, we may guess a part $x' \in S_d$ of x with $d < 20$; and with restriction in sampling we get $|w|_1 = 3$, $k < 1$ in definition of ρ_0; with the bound $L < 2^{64}$, and assume the bound of $|z|/n^{1/2} = \sigma$ can be chosen as 220, the basic form has $\rho < 0.04\rho_0$. For $q = 1024$, with $|w|_1 = 2$, $k = 4.5$,$\sigma = 80$ while the other parameters remain the same, the basic form has $\rho < 0.08\rho_0$ and the variation has $\rho < 0.02\rho_0$. For 1-byte modulus (say $q = 256$), $|w|_1 = 2$, $k = 1.5$, $\sigma = 40$ while the other parameters remain the same, the variation has $\rho < 0.04\rho_0$.

Next we discuss how to decide the number k of times a temporary key can be used. For the basic form, we may regard the sum of k samples of c^*z as $k|c|^2w + [k^{1/2}|c|\sigma]$, where σ is the gaussian parameter of z. The information it leaks abut w is characterized by the ratio

$$\theta = k|c|^2/(k^{1/2}|c|\sigma)$$

The smallest θ that can be exploited to harm security by our known attacks) is $\theta_0 \approx 0.2$, where it reduce the effort to guess a half of zero components of $w \in S_{75}$ to $\approx 2^{64}$ which resulted in a 2^{128} complexity attack. So we should have $\theta < 0.2$, or $k < 0.04\sigma^2/c^2$. For the variation, the $|c|$ factor is gone, and we get $k < 0.04\sigma^2$.

4 Structured secrets and random numbers

Next, we consider more structures of secrets (x, w, s_j) and random numbers (y_j) to improve efficiency, and for integrating both encryption and signature in a simple computing module.

Note that the most computation-intensive operation in lattice PKC is polynomial multiplication if our signature scheme is used. In fact, using structured

secret (and/or noise) will avoid multiplication of general elements in the ring in the whole encryption scheme and the signing algorithm(only verification of signatures need such operation). multiplication by a structured element will be done by tens of shift-additions in R: a shift-addition is of the form $a + T^i b$, where the shift T^i is cheap in software and free in hardware. We will count the computation cost in number of shift-additions, and say the cost of an element (denoted as $c()$) meaning the cost of multiplication by it.

Note that elements in S_d have cost d, and $c(a+b) = c(ab) = c(a) + c(b)$. The structure we pose on the secrets and random numbers is that they are algebraic expressions of some S_ds. We have the following observation:

In RLWE problem in PKC: $(a, as + e)$ the distribution $\mathcal{K}$ can be replaced with a structure $S(\mathcal{K})$ of cost in tens ($< 2d_c$).

The fact is that no known attacks on RLWE can make uses of this structure except: the birthday exhausting of s (or e).

To see this, note first that the gaussian parameter of $\mathcal{K}$(< 10 in lattice PKC) can be approximated by a structure of extremely low cost(consider the product of (S_2s and S_3s) which is weak for the birthday attack. We can lower the algebraic degree and use the sum of products and using other small S_ds to adjust the birthday attack cost to the security level. Usually, we can get the desired structure of cost $< 2d_c$. Example: the structure $S(2 \times 12 + 3 \times 10 + 3 \times 9$ has the same gaussian parameter as S_{81}, and has cost 35, and birthday complexity $|S_2|S_5||S_3||S_9| > 2^{128}$. For any distribution $\mathcal{K}$, the distribution $S(\mathcal{K})$ means a structure of such low cost but high enough birthday complexity, and with approximate gaussian parameter as $\mathcal{K}$.

This observation will make encryption schemes with different gaussian parameters all achieves the same level of extreme efficiency. Any RLWE-based encryption scheme can be transformed in to a PKC module to realize both encryption and signature as follows: The system parameter is a; each user has a long term key pair $((x_1, x_2), X), X = x_1 a + x_2$, where $x_i \in S_d$ for suitable d; and additional s_js, as_js and Xs_js as needed by the signing algorithm; the temporary secrets w_1, w_2 are drawn from $S(\mathcal{K})$, and the key pair $((w_1, w_2), W = ax_1 + x_2)$ is used as the key pair of both the encryption scheme and the plain signing scheme with respect to the system parameter X. If the original encryption algorithm needs to compute $rX + r_1, rW + r_2$, where rs are drawn from $\mathcal{K}'$, we also replace $\mathcal{K}'$ with $S(k')$. It is clear that each encryption costs $< 4d_c$, and decryption costs $< 2d_c$. The condition for $\mathcal{K}$ (of gaussian parameter σ) is that the expected norm square of cxw is considerably smaller(say $<$0.05 times) than that of z, i.e.

$^2 d_c n < 0.05 B_z$ for the basic form; and $d\sigma^2 n < 0.05 B_z$ for the variation; and the later holds for all existing lattice encryption schemes.

To see the efficiency of the signing algorithm, we need to get into more details of the signing algorithm defined by the following sub-algorithms : the algorithm $sGen()$ to generate s_j,$A_j = s_j a$, and $X_j = s_j a$; $yGen()$ to generate ys; $wGen()$ to generate the temporary key pair return the certificate; $Sign(m)$ to produce a signature given message m. In the description of these algorithms, all variables are global and static.

In basic form of the scheme, these algorithms are defined as follows:

$$sGen() := \left\{ \begin{array}{l} \text{for } (1 \le j \le t) \\ \qquad s_j \leftarrow S_1; A_j = s_j a; X_j = s_j X; B_s = 1; \\ \text{repeat } k \text{ times } R(2); \\ \text{repeat } l \text{ times } R(3); \end{array} \right\},$$

where k, l makes $2^k 3^l$ approximate the expect value, and $R(), r()$ is defined as

$$R(i) := \left\{ \begin{array}{l} \text{for } (1 \le j \le t) \\ \qquad \text{repeat } s'_j = r(i) \text{ until } 0.95 i B_s \le |s'_j|^2 \le 1.05 i B_s; \\ \qquad A'_j = \sum_{1 \le m \le t} y_m A_m; \\ \qquad X'_j = \sum_{1 \le m \le t} y_m X_m; \\ \text{for } (1 \le j \le t) \\ \qquad s_j = s'_j; A_j = A'_j; X_j = X'_j; B_s = i^* B_s; \end{array} \right\},$$

$$r(i) := \left\{ \begin{array}{l} \text{sample random integers } d_j \ge 0 \text{ for } 1 \le j \le t, \\ \qquad \text{such that } \sum_j d_j = i; \\ \text{sample } y_j \in S_{d_j} \text{ for } 1 \le j \le t; \\ \text{return } (y = \sum_j y_j s_j) \end{array} \right\}.$$

$r(i)$ has cost i.

$$Sign(m) := \left\{ \begin{array}{l} \text{repeat } y1 = r(d_y);\ \text{until } \rho_1 B_y \leq |y1|^2 \leq \rho_2 B_y; \\ Y0 = \sum_j y_j X_j; \\ \text{repeat } y2 = r(d_y);\ \text{until } \rho_1 B_y \leq |y2|^2 \leq \rho_2 B_y; \\ Y = Y0 + y2; \\ c_0 = h(Y, m); \\ c = f(c_0); \\ z_1 = y1 + cw_1, z_2 = y2 + cw_2; \\ \text{return } (z_1, z_2, c_0); \end{array} \right\}.$$

$$wGen() := \left\{ \begin{array}{l} w_1 \leftarrow S(\mathcal{K}); w_2 \leftarrow S(\mathcal{K}); \\ W1 = w_1 x_1; W2 = w_1 x_2 + w_2; \\ W = w_1 X + w_2; \\ \text{repeat } y1 = r(d_y);\ \text{until } \rho_1 B_y \leq |y1|^2 \leq \rho_2 B_y; \\ Y0 = \sum_j y_j A_j; \\ \text{repeat } y2 = r(d_y);\ \text{until } \rho_1 B_y \leq |y2|^2 \leq \rho_2 B_y; \\ Y = Y0 + y2; \\ c_0 = h(Y, m); \\ c = f(c_0); \\ z_1 = y1 + cW1, z_2 = y2 + cW2; \\ \text{return } (W, z_1, z_2, c_0); \end{array} \right\}.$$

The cost of $Sign()$ is $(2k+1)d_y + 2d_c$, and the cost of $wGen()$ is $< (2k+1)d_y + 2d_c + 3 \times 2d_c$, where k is the expected number of repeat to get a desired norm (usually $k < 1.1$).

What is required about d_y? We should guarantee all the commitments Ys in the lifetime of X remains secure. Each single $Y = y_1 X + y_2$ is an RLWE instance with quite large gaussian parameter. A safe way is to let y_1s and y_2s never repeat, which can be achieved if the space of y is large enough, i.e. $|\{y\}| > (2^{64})^2 2^{\lambda} (2^{64}$ is the bound for signing times). But it seems that repeat of y is not so harmful: repeat y_1 k times just reduces the gaussian parameter by a factor of $k^{1/2}$, and no more bad effect is known. If we let $k < 10$, then the resulted RLWE instance is still much harder than the instance for the key pair; which can be satisfied if $|\{y\}| > (2^{70})$ for $t = 10, n > 500$; which means $d_y = 7$ is enough. This make the signing algorithm at the same level of efficiency as the encryption.

The description of the variation is almost the same except: two copies of Y, repeat of two copies of c to get z having the desired norm. We just give the $Sign()$ algorithm.

$$Sign(m) := \left\{ \begin{array}{l} \text{repeat } y11 = r(d_y); \text{ until } \rho_1 B_y \le |y11|^2 \le \rho_2 B_y; \\ Y0 = \sum_j y_j X_j; \\ \text{repeat } y11 = r(d_y); \text{ until } \rho_1 B_y \le |y12|^2 \le \rho_2 B_y; \\ Y1 = Y0 + y12; \\ \text{repeat } y21 = r(d_y); \text{ until } \rho_1 B_y \le |y21|^2 \le \rho_2 B_y; \\ Y0 = \sum_j y_j X_j; \\ \text{repeat } y22 = r(d_y); \text{ until } \rho_1 B_y \le |y22|^2 \le \rho_2 B_y; \\ Y2 = Y0 + y22; \\ i = 0; \\ \text{repeat} \\ \quad (c1_i, c2_i) = h_i(Y1, Y2, m); \\ \quad c1 = f(c1_i); c2 = f(c2_i); \\ \quad z_1 = c1y11 + c2y12 + w_1; z_2 = c1y21 + c2y22 + w_2; \\ \quad i++; \\ \text{until} \\ \quad |z_1|^2 < B_z \& |z_2|^2 < B_z; \\ \text{return } (z_1, z_2, Y1, Y2, i); \end{array} \right\}.$$

where $h_i(Y1, Y2, m)$ can be consecutive output of a stream cipher initialized using $Y1, Y2, m$. The expected number k of repeat times can be made < 1.1. It is clear that this algorithm cost $k(6d_y + 4d_c)$.

5 Examples

We assume $n \approx 512, \lambda = 128$, thus $d_c = 19$; and assume $d_y = 7$. In the examples we set $B_z = n \times D^2$ where D varies with q; for the basic form $B_y = B_z - d^2 d_c$; for the variation $B_y = (B_z - d^2 d_c)/(2d_c)$; the integer k, l is such that $B_s = 2^k 3^l \approx B_y/d_y$.

For $q = 3 \times 2^{12} + 1, n = 512$, we set $D = 220, d = 256, k = 4, l = 11, \rho_1 = 0.65, \rho_2 = 1$; $S(S_d) = S(4 \times 5 \times 5 + 4 \times 4 \times 5 + 4 \times 4 \times 5)$; experimental costs of $sGen(), wGen(), Sign(), Enc()$ for the basic form are 1280,129,60,68 respectively, and each temporary key can be used 100 times.

For parameters $q = 1024, n = 509$, we set $D = 80, d = 81, \rho_1 = 0.65, \rho_2 = 1$; $S(S_d) = S(5 \times 6 + 5 \times 6 + 5 \times 6)$. For the basic form $k = 17, l = 1$, experimental

costs of $sGen(), wGen(), Sign(), Enc()$ are 1115,121,60,60 respectively, and each temporary key can be used 10 times. For the variation,$k = 7, l = 4$, experimental costs of $sGen(), wGen(), Sign(), Enc()$ are 756,188,128,60 respectively, and each temporary key can be used 200 times.

The 1-byte q case: $q = 256, n = 509$, or $q = 257, n = 512$. We set $D = 40, d = 75, k = 7, l = 4, \rho_1 = 0.65, \rho_2 = 0.93$, $S(S_d) = S(2 \times 12 + 3 \times 10 + 3 \times 9)$, experimental costs of $sGen(), wGen(), Sign(), Enc()$ for the variation are 625,198,128,70 respectively, and each temporary key can be used 50 times.

6 Applications

Our constructions allow the same key pair is used both for encryption and signing, and both are extremely efficient. Note that transporting and verification of the temporary keys may be implemented using a public directory which functions as in common PKI , this makes the signer only need to send signatures to the verifier, and the verifier only need to verify the signature if the public service is trusted(its misbehavior is easy to detect). This greatly reduces the workload both for signer and verifier. If the verifier is resource-limited, the main workload in verification, computing zX, can also be delegated efficiently. Pick $v, v' \leftarrow S(\mathcal{K})$ and send the helper $(z, X, vX + v')$, who returns $(u = zX, u' = z(vX + v'))$, and now $u' = vu + v'z$ means u is the right answer: any success cheating means the helper can recover v, v', contradicting the RLWE hardness.

Now that the whole public key functions are realized using only very limited computations, so can integrate into a simple circuit implementation. Together with construction of integrated symmetric key functionalities similar as [11], we could get a small but full-fledged crypto circuit (co-chip).

References

1. Albrecht, M.R., Curtis, B.R., Deo, A., Davidson, A., Player, R., Postlethwaite, E.W., Virdia, F., Wunderer, T.: Estimate all the {LWE, NTRU} schemes! In: Catalano, D., Prisco, R.D. (eds.) Security and Cryptography for Networks - 11th International Conference, SCN 2018, Amalfi, Italy, September 5-7, 2018, Proceedings. Lecture Notes in Computer Science, vol. 11035, pp. 351–367. Springer (2018). https://doi.org/10.1007/978-3-319-98113-0_19, https://doi.org/10.1007/978-3-319-98113-0_19
2. Alkim, E., Bindel, N., Buchmann, J., Dagdelen, Ö.: TESLA: tightly-secure efficient signatures from standard lattices. IACR Cryptol. ePrint Arch. **2015**, 755 (2015), http://eprint.iacr.org/2015/755

3. Chen, Y., Nguyen, P.Q.: Bkz 2.0: Better lattice security estimates. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. pp. 1–20. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
4. Ducas, L., Durmus, A., Lepoint, T., Lyubashevsky, V.: Lattice signatures and bimodal gaussians. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 40–56. Springer (2013), https://doi.org/10.1007/978-3-642-40041-4_3
5. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012), https://www.math.auckland.ac.nz/%7Esgal018/crypto-book/crypto-book.html
6. Lenstra, A.K., Lenstra, H.W., Lovasz, L.: Factoring polynomials with rational coefficients. MATH. ANN **261**, 515–534 (1982)
7. Nguyen, P.Q., Regev, O.: Learning a parallelepiped: Cryptanalysis of GGH and NTRU signatures. J. Cryptol. **22**(2), 139–160 (2009), https://doi.org/10.1007/s00145-008-9031-0
8. Schnorr, C., Euchner, M.: Lattice basis reduction: Improved practical algorithms and solving subset sum problems. Math. Program. **66**, 181–199 (1994). https://doi.org/10.1007/BF01581144, https://doi.org/10.1007/BF01581144
9. Son, Y., Cheon, J.H.: Revisiting the hybrid attack on sparse and ternary secret lwe. Cryptology ePrint Archive, Report 2019/1019 (2019), https://eprint.iacr.org/2019/1019
10. Szydlo, M.: Hypercubic lattice reduction and analysis of GGH and NTRU signatures. In: Biham, E. (ed.) Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2656, pp. 433–448. Springer (2003), https://doi.org/10.1007/3-540-39200-9_27
11. Ye, D., Shi, D., Wang, P.: Lightweight ae and hash in a single round function. Cryptology ePrint Archive, Report 2018/1126 (2018), https://eprint.iacr.org/2018/1126