Lattice Signature can be as Simple as Lattice Encryption

Dingfeng Ye, Jun Xu, Guifang Huang, Lei Hu

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

Abstract. Existing lattice signature schemes are much less efficient than encryption schemes due to the rejection sampling paradigm. We give a new construction which avoids rejection sampling by using temporary public keys and structured secrets in a Bliss type scheme. Structured secrets also improve existing lattice encryption schemes to nearly the same extreme efficiency. Our signing algorithm is comparative with this optimized encryption efficiency. Our signature scheme allows the same key pair work as an encryption scheme. 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 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 lattice suite algorithms. There are two main approaches to obtain lattice signatures: one is trapdoor sampling a short integer solution (SIS) to a hard SIS problem, such as [7, 14]; the other is a Fiat-Shamir heuristic in proving knowledge of secrets associated with the public key, such as [12]. 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 (for example Tesla [2]) or a smaller Gaussian noise together with a computation-intensive rejection sampling (for example, Bliss [4] and Dilithium [5]) to get a distribution independent of the secret, which results in a large modular size and much less efficient signing algorithm compared with the lattice encryption schemes. Our first objective is a lattice signature scheme without the requirement of large noise and/or complex sampling.

We follow the Fiat-Shamir approach, though our technique trivially applies to the trapdoor sampling approach where the improved scheme is much less efficient than the one given below. To explain our method, we use notations in a RLWE context: all objects are elements of a ring $R = \mathbb{Z}[T]/(T^n + 1)$ or $R_q = \mathbb{Z}_q[T]/(T^n+1)$, n is a power of 2 or a prime, q is the modulus. We can view a Fiat-Shamir signature as a proof of knowing the small secrets x_1, x_2 satisfying a public linear relation. 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. 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 under various statistical attacks. An attack is called linear (resp. quadratic) if it exploits a statistical variable which is linear (resp. quadratic) on z [9, 8, 17, 13]. In the previous version [18] of this work, we use temporary public key to stop linear attacks, and structured y to prevent quadratic ones. But the later fails (Thanks to AsiaCrypto 2021 referees). We observe that Bliss type signatures have been immune to linear attacks, and with structured secrets they are safe within a bound SN on the number of allowed signatures. SN grows rapidly with the effective noise size D. For suitable parameters the temporary public key trick can increase such D for each certification step. Now in theory we can use a certification path and/or key evolving technique to allow unbounded signing times. In practice, we can use only one certification step to achieve a good SN for practical application even for extremely light weight parameters.

Structured secrets are the key ingredients of our scheme. At security respect, they act as statistical noises of various statistical attacks which are the only known threats to plain lattice signatures. So we regard this a more natural security patch of the plain schemes than the rejection sampling approach. At efficiency respect, this is an efficient way to produce approximate Gaussian distributions. Known attacks on lattice cryptography can not distinguish exact Gaussian distributions and approximate ones, except in the case of rejection sampling. In particular, existing lattice encryption schemes would greatly be improved on efficiency at no price.

Next, we consider integrating both encryption and signature in a simple computing module. The key pair of our signature scheme contains the key pair of a standard LWE encryption scheme, so the encryption function can be integrated into the signature module almost for free. The certification step and verification of the signature involves multiplication (or division) 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. As a result, we can integrate the whole public key functionalities in an extremely light weighted hardware (just simple circuit) which mainly adds up signed integers.

In summary: our signing algorithm is even faster than the fastest previous lattice encryptions, and is comparable in speed with lattice encryptions optimized with structured secrets. It can also be integrated with lattice encryption in extremely light-weight hardware implementation.

2 Preliminary Notations

2.1 Algebra

For any vectors a, b of length l, $a \cdot b$ denotes the inner product of a, b, and $|a| = (a \cdot a)^{1/2}$ denotes the Euclidean norm.

Let $R = \mathbb{Z}[T]/(T^n + 1)$, and $R_q = R/(qR)$, where n is a power of 2 or a prime, $q = 2q'$ is a positive integer called the modulus. Elements of R or R_q are also viewed as vectors of length n: $a = \sum_i a_i T^i = (a_0, a_1, \cdots a_{n-1})$. 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 random variable a (considered in this work) in the ring, where '$\approx$' means the difference of the two sides has zero mean value.

2.2 Distributions

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 the set of vectors of length n with exactly d non-zero entries of $\{-1, 1\}$.

A Gaussian distribution χ_σ with parameter σ over $\mathbb{Z}$ is defined as:

$$\Pr[x] = e^{x^2/2\sigma^2} / \sum_{i \in \mathbb{Z}} e^{i^2/2\sigma^2} \approx \chi_\sigma(x),$$

where $\chi_\sigma(x) = e^{-x^2/2\sigma^2}/(\sqrt{2\pi}\sigma)$ is the density function of continuous Gaussian. It is well-known that

$$E(|x|^2 : x \leftarrow \chi_\sigma) = \sigma^2,\ V(|x|^2 : x \leftarrow \chi_\sigma) = 2^{1/2}\sigma^2,$$

where $E(\cdot)$ means mean value, and $V(\cdot)$ means variation.

For any integer c, denote by $\chi_{\sigma,c}$ the distribution

$$r + (-1)^b c : r \leftarrow \chi_\sigma, b \leftarrow \{0, 1\}.$$

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:

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

We also model complex algebraic combinations of semi-Gaussian distributions over R as Gaussian.

Algebraic combinations of S_d's will be abbreviated 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 $((a \times b + c \times d + \cdots)/n)^{1/2}$.

2.3 Statistics

For any two distributions X, Y on a finite set, denote by $\rho_{Y/X}(\theta)$ the density of x such that

$$Pr[Y = x]/Pr[X = x] = \theta.$$

(X, Y) is called a (k, k_0)-pair where $|k_0| << 1$ if

$$\rho_{Y/X}(\exp(t + k_0)) \approx \chi_k(t)$$

A (X, Y, ρ, θ)-distinguisher is any algorithm $\mathcal{A}$ satisfying:

$$Pr[\mathcal{A}(z) = 1 : z \leftarrow X] = \rho,\ Pr[\mathcal{A}(z) = 1 : z \leftarrow Y] \leq \rho\theta.$$

(X, Y) is said to be (c, θ)-close if there is no (Y, X, ρ, θ)-distinguisher for $\rho < c$.

We use the following function over real c for probability:

$$ef(c) = \int_{t \geq c} (2\pi)^{-1/2} \exp(-t^2/2)dt$$

For a constant positive integer c, $\chi_{\sigma,\underline{u}}$ denotes the random variable which is half time $\underline{u} + \chi_\sigma^c$, half time $-\underline{u} + \chi_\sigma^c$ for any vector $\underline{u}$ of length c; and $\underline{1_c}$ denotes the all 1 vector.

Lemma 1. *For $\sigma >> 1$*

- *$((\chi_\sigma)^t, \rho_{(1+\chi_\sigma)^t})$ is approximately a $(t^{1/2}/\sigma, -t/(2\sigma^2))$-pair, for positive integer $t < \sigma^2$.*
- *$((\chi_\sigma^s)^m, (\chi_{\sigma,\underline{1_s}})^m)$ is approximately a $(m^{1/2}(s+1)/\sigma^2, 0)$-pair, for positive integers s, m, and $m >> \sigma$.*

Lemma 2. *If (X, Y) is a (k, k_0)-pair, (X', Y') is a (k', k_0')-pair; then $(X \times X', Y \times Y')$ is a $((k^2 + k'^2)^{1/2}, k_0 + k_0')$-pair.*

Lemma 3. *A (k, k_0)-pair is $(ef(c), \exp(-k_0 + k^2)ef(c+k)/ef(c))$-close, for any $c > 0$.*

2.4 Hard Problems

RLWE: given $a, b \leftarrow R_q$, $b = as + e$, where $s, e \leftarrow \mathcal{K}$ for some distribution $\mathcal{K}$, to find s, e.
NTRU: given $a = f/g$ in R_q, where $f, g \leftarrow \mathcal{K}$ for some distribution $\mathcal{K}$, to find f, g.
RSIS: given $a, b, c \in R_q$, to find $u, v \in R$, such that $au + bv = c$ and $|u|^2 < (\mathcal{D})^2 n$, $|v|^2 < (\mathcal{D})^2 n$ for some bound $\mathcal{D}$.
RSIS(l): given $\underline{a} = (a_1, \cdots, a_l) \in R_q^l$, and $a_0 \in R_q$, to find $\underline{u} = (u_1, \cdots, u_l) \in R_q^l$, such that $\underline{a} \cdot \underline{u} = \sum_i a_i u_i = a_0$ and $|u_i|^2 < (\mathcal{D}')^2 n$, for $1 \leq i \leq l$ and some bound $\mathcal{D}'$. For parameter sets of this work, we always let $\mathcal{D}' = \mathcal{D}$.

We are not concerning the hardness of RLWE (NTRU, RSIS) asymptotically, instead, we would like to evaluate the concrete security level (in bits) of the problem at specific parameters according to known attacks. We regard the security level is equally good as long as it exceeds the claimed number of bits, and it is only measured with known algorithms; that is, large security margin is not favoured, low resource consumption is the top concern of this work.

For $n = 512$, $q = 256$, and $\lambda = 80$-bit security, we set $\mathcal{D} = 32$ in the RSIS problem which is associated with the problem to forge a lattice signature or a certificate. According to known attacks: lattice reductions [10, 15, 3, 1] (BKZ block size $340, 200$ is regarded ok for security level $128, 80$ bits respectively), brute force attacks at scarce secrets [16], statistical tests in this work, we assume $\mathcal{K} = S_k$ with $k = 45$ is ok for the NTRU (RLWE) problem (which is associated with the basic security assumption of both encryption and signature) even with additional information on secrets obtained from statistical attacks. Similarly for $q = 1024$, $n = 512$, $\lambda = 128$, we set $\mathcal{D} = 100$, and $k = 127$; for $q = 2^{16}$, $n = 1024$, $\lambda = 128$, we set $\mathcal{D} = 2^{12}$, and $k = 65$.

2.5 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 $u = \mathrm{COM}(x, w, r)$; the verifier gives a random challenge c; the prover then computes a respond $a = \mathrm{RES}(x, w, r, c)$; and the verifier computes $\mathrm{Verify}(u, c, a, x)$, such that there is an algorithm to extract a valid w' for x given several (u, c_i, a_i, x) with different c_i that verifies. A Σ protocol is called zero knowledge (ZK) if (u, c, a) is statistically independent of w; witness hiding with bound B if no useful (with respect to any attacks) information of w can be derived from B 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 protocol: the bound B is preserved as the number of signatures allowed.

When w, r is in a ring, and R is a linear relation over the ring, then the basic form of $\mathrm{RES}(x, w, r, c)$ is $r - wc$. We use the following variation of the basic form in this work. Given a vector $\underline{a} = (a_1, \cdots, a_l) \in R_q^l$, $a_0 \in R_q$, a witness $\underline{u} = (u_1, \cdots, u_l) \in R_q^l$, and a sampling algorithm $\mathcal{A}$, we denote the FS protocol

to prove knowledge of the solution to RSIS(l) by

$$\mathrm{FS}(\underline{a}, \underline{u}, \mathcal{A}) := \left\{ \begin{array}{l} (\underline{r1}, \underline{r2}) \leftarrow \mathcal{A}; Y1 = \underline{a} \cdot \underline{r1}; Y2 = \underline{a} \cdot \underline{r2}; \\ (c1, c2) = h(Y1, Y2, \underline{a}, \cdots); \underline{z} = \underline{u} + c1\underline{r1} + c2\underline{r2}; \\ \text{return } (Y1, Y2, \underline{z}) \end{array} \right\},$$

where h is a cryptographic hash function to the distribution $S^2_{d_c}$, where $d_c = 19$ (13) for 128 (80) bits security respectively.

2.6 Cryptography

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

$$\mathrm{KeyGen}(para) \rightarrow (sk, pk)$$

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:

$$\mathrm{Enc}(pk, m) \rightarrow c$$

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

The problem to retrieve sk from pk is called the trapdoor problem on the key pair. The security requirement is that the trapdoor problem is hard, and without the secret key, one can not 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 [6].

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, where the certificate part is common for all signatures by the same temporary key. Our scheme consists of the following algorithms:

–

$$\mathrm{Keygen}(a) := \left\{ \begin{array}{l} x_1, x_2 \leftarrow \mathcal{K}; \\ rs \leftarrow Rgen(); \\ sk = (x_1, x_2, rs); \\ pk = X = ax_1 + x_2; \end{array} \right\},$$

where $a \in R_q$ is the system parameter, $\mathcal{K}$ is a distribution over R_q defined by scheme, $Rgen()$ is the algorithm which generates the secret rs for sampling structured random numbers.

–

$$\mathrm{Cert}(sk, pk) := \left\{ \begin{array}{l} w_1, w_2 \leftarrow \mathcal{K}; \\ b = (q' - Xw_1)^{-1} w_2; X' = Xb; b' = q'b; \\ \underline{a} = (aX', a, X', 1); \\ \underline{u} = (x_1 w_1, x_1 w_2, x_2 w_1, x_2 w_2); \\ (Y1, Y2, \underline{z}) = \mathrm{FS}(\underline{a}, \underline{u}, rSample(rs)); \\ tpk = (X', b, b'); tsk = (w_1, w_2, rs); \\ cert = (pk, tpk, \underline{a}, Y1, Y2, \underline{z}); \end{array} \right\},$$

where the omitted argument of the hash function (implicit in FS()) should include b'; (tpk, tsk) is the temporary key pair; the requirements on the sampler $rSample()$ will be explained later.

–

$$\mathrm{CertVerify}(cert) := \left\{ \begin{array}{l} \text{parse } cert = (pk, tpk, \underline{a}, Y1, Y2, \underline{z}), \\ \text{where } pk = (a, X), tpk = (X', b, b'); \\ \underline{a} = (a_1, a_2, a_3, a_4); \underline{z} = (z_1, z_2, z_3, z_4); \\ \text{if } \{ \\ |z_i|^2 \leq n * \mathcal{D}^2 \text{ for } 1 \leq i \leq 4; \\ X' = Xb; b' = q'b; \\ a_1 = aX'; a_2 = a; a_3 = X'; a_4 = 1; \\ \underline{a} \cdot \underline{z} = Xb' + c1Y1 + c2Y2; \\ \text{where } (c1, c2) = h(Y1, Y2, \underline{a})\}; \\ \text{Return 1} \\ \text{else Return 0} \end{array} \right\}.$$

The algorithms Cert() and CertVerify() can be modified to certificate and verify a new long-term key pair generated by Keygen(), which means the scheme support key-evolving almost for free.

–

$$\mathrm{Sign}(tsk, tpk, m) := \left\{ \begin{array}{l} \text{parse } tsk = (w_1, w_2, rs); tpk = (X', b, b'); \\ y \leftarrow \{1, -1\}; \underline{a} = (X', 1); \underline{u} = (yw_1, yw_2); \\ (Y1, Y2, \underline{z}) = \mathrm{FS}(\underline{a}, \underline{u}, sSample(rs)); \\ \tau = (Y1, Y2, \underline{z}); \end{array} \right\},$$

where the argument of the implicit hash function should include the message m, τ is the signature on m; and $sSample()$ will be explained later.

–

$$\text{Verify}(\tau, m, tpk) := \left\{ \begin{array}{l} \text{parse } \tau = (Y1, Y2, \underline{z}); tpk = (X', b, b'); \\ \text{where } \underline{z} = (z_1, z_2); \\ \text{if } \{ \\ |z_i|^2 \leq n * \mathcal{D}^2 \text{ for } i = 1, 2; \\ X'z_1 + z_2 = b' + c1Y1 + c2Y2; \\ \text{where } (c1, c2) = h(Y1, Y2, (X', 1), m)\}; \\ \text{Return 1} \\ \text{else Return 0} \end{array} \right\}.$$

Note that the signing algorithm is of Bliss type, so the trapdoor problem is NTRU. This makes linear attacks harder than the LWE case. We will consider general statistic attacks and decide the requirements on the samplers next.

4 Security Analysis and Design of The Samplers

The signature part of the scheme is a plain Bliss type signature, and the certificate part is a Σ-protocol on proving knowledge of short integer solutions of an equation which indicates the knowledge of the private keys. So we suppose that the only weakness of the scheme is that the z part of both the certificate and signature leaks information about secrets which might be derived by statistical methods. Our design aims to restrict such leaked information in an unexploitable form: either highly noised or outcome of a one-way function over the large set of secret variables. Known statistical attacks include: linear, quadratic, and guess-and-determine. Our secret samplers prevent the attacks by introducing corresponding noises and (or) making the statistic problem more demanding on the number of samples. We next explain these attacks and the counter-measures of the samplers. It will be verified that our scheme is secure against general statistics attacks assuming the Gaussian of algebraic expression of large numbers of semi-Gaussian variables.

4.1 Linear and guess-and-determine attack

Consider the random variable $z = u + r$ over R where r is Gaussian with parameter σ, $u = xw$ for the certificate and $u = (-1)^i w$ for the signature, the constant secrets are x, w in these two cases respectively. The linear attack can be modeled as learning the constant secrets by investigating the components of z individually given t samples of z. The guess-and-determine attack is to do so by guessing correlations of the samples and the components, where the sample

statistics are used to verify the guess. Guessing the correlations of a subset of samples is to make components of z have more exploitable distributions over these samples; Guessing correlations of the components is to hit a subset of the components which has exploitable joint distribution. The security level restricts the space of guessing. An important issue is to estimate the bound of numbers of samples for a successful attack. One kind of bound is to achieve a high enough distinguishing advantage between the correct guess and wrong ones, which might over-estimate the actual bound because of other attacks.

The following structural guessing space only needs small distinguishing advantage to success, so gives a bound hopefully for all attacks. A correct guess of correlations of the components is a section w_0 (consecutive components) of the secret up to a shift (multiplication by $(-1)^b T^j$ for some j, b). The attack works as follows: first searches the whole set of candidates w_0 (polynomials with a fixed degree bound, fixed constant term 1, $s-1$ other non-zero terms), excludes some according to statistical analysis on $w_0^* z$; then searches connections among remaining candidates. A connection means two candidates can be shift-jointed to a larger section with large overlap. As long as sufficiently large portion of wrong guesses can be excluded, then w may be recovered by a connecting-process among the remaining candidates: if enough sections of w are included in the candidate sets to allow connecting while wrong candidates are much harder to connect, then the attack works. It is not hard to understand that the connecting-process tolerates exponentially many candidates. To resist the attack, we need that the portion of remaining wrong guesses is comparative with the portion of remaining correct guesses.

The case $u = xw$: Both u and r are modeled as Gaussian, so is z. Since each component of z has identical distribution, linear attack gets nothing. For guess-and-determine, there are no correlations between samples. Consider

$$z' = w_0^* z = (w_0^* u) + (w_0^* r) = u' + r'.$$

Again all z', u', r' are Gaussian, and for correct guess, the mean value $E(|u'|^2)$ is $k|w_0|^2$ greater than usual, if r were independently Gaussian. This would be distinguishing. Our structure of r make $E(|r'|^2)$ has variation $E(|r'|^2)/(4|w_0|n^{1/2})$ with respect to w_0, which is far greater than $k|w_0|^2$, thus there is no way to exclude any false guesses. So guess-and-determine attack does not work.

The case $u = (-1)^i w$: Each component of z has distribution $\chi_{\sigma,1}$ or χ_σ according to w is non-zero or zero at the position. Linear attack is meant to distinguish $\chi_{\sigma,1}$ with χ_σ with $\sigma = \mathcal{D}$. For guess-and-determine, suppose we

guess the i in t samples and a section with s non-zero entries. A correct guess results in a sample in distribution $X = (1 + \chi_{\mathcal{D}})^{st} \times (\chi_{\mathcal{D},\underline{1_s}})^m$ corresponding to some component of $z' = w_0^* z$; and $n-1$ samples for distributions corresponding to the other components, which are no more apart than the distribution $Y = (\chi_{\mathcal{D}})^{st} \times (\chi_{\mathcal{D}}^s)^m$, where m is the number of signatures by w. For a wrong guess, what we get is no more apart than n samples of Y. Thus the ability of any algorithm to exclude wrong guesses is bounded by probability to exclude sets of n samples of Y while trying to let sets consisting of $n-1$ samples of Y and 1-sample of X remain. The following is some computations for formulating the requirements to defeat the attack.

1. Bounding st: to test all candidates of w_0 we need a workload of

$$2^{s-1}\binom{(s-1)n/k}{s-1}$$

   the whole complexity in bits of the attack is

$$t + (s-1) + (s-1)\log(((n/k)-1)\exp(1)) = \lambda + o(1)$$

   So we get $st \leq (\lambda+8)^2/(4(\log((n/k)-1)\exp(1)+1)) << \mathcal{D}^2$, which will indicate that guessing i is not helpful for our parameter sets.
2. Bounding s: set $t = 0$, we get $s \leq \lambda/(\log((n/k)-1)\exp(1)+1) + o(1)$.
3. To allow connecting of remained correct guesses, the portion of them should be above $1/s$.
4. If (Y, X) is $(1/s', \theta(s'))$-close, and $s'n\theta(s') > 5$ for $1 \leq \sigma' \leq s$, then the portion of wrong guesses remained is very close to that of the correct guesses, and the attack fails.

These condition easily give a bound tSN for the number of times that a temporary key can be used.

4.2 Quadratic Attacks

Quadratic statistical attacks are first introduced in [17, 13]. This kind of attacks exploit correlations over the whole component set, and can be formulated as follows: given two random variables $z = yw + r$, $z' = yw' + r'$ over R, where w, w' are constant (but unknown), r, r' are Gaussian with parameter σ, and y is a variable called the intersection of z, z'. Note that

$$z^* z' = |y|^2 w^* w' + r''$$

which means w^*w' becomes the target of linear attacks as long as r'' is balanced. To deal with this attack, we simply let the pair r, r' share correlated secrets so that r'' contains structured bias to flood w^*w'. One way to do this is as follows. Let $(r_i : 1 \le i \le l)$, $(r'_i : 1 \le i \le l)$ be two secret but fixed sequence over R_q,

$$r_0 = \sum_{1 \le j \le l} r_j y_j,\ r'_0 = \sum_{1 \le j \le l} r'_j y_j,$$

where the shared y_js are variables from some distributions. Let $r = r_0 + y$, $r' = r'_0 + y'$ where y, y' are independent. Now it is easy to see that the mean value of z^*z' is a linear combinations of w^*w' and the $r^*_j r'_j$s. We let $l = 16$ to make such a function un-exploitable. Note that this technique works for the pair (z, z) too, where we let the two sequence be identical.

Our samplers play the trick to all intersected pairs of zs in Cert() and Sign(): all reflexive pairs (z, z); (z_1, z_2) in Sign() and (z_1, z_3), (z_2, z_4) in Cert(). To reduce secret size, we let all sequences share the same set $\{s_i : 1 \le i \le 16\}$, and the distribution of y_js varies for each pair. In samplers $sSample()$ and $rSample()$, the whole $\underline{r1}$ part treat non-reflexive pairs and whole $\underline{r2}$ part treat reflexive pairs.

The pairs across the Cert() and Sign() are not treated, they do not harm security as long as the number of total signatures for the long term key is restricted below a bound SN. For example, the two z_1s in Cert() and Sign() may result in

$$|w_1|^2(-1)^i x_1 + r$$

where we may model r as Gaussian with parameter $\sigma = \mathcal{D}^2 n^{1/2}$. Divided by $k = |w_1|^2$, it becomes

$$(-1)^i x_1 + r'$$

where r' can be modeled as Gaussian with parameter $\mathcal{D}' = \mathcal{D}^2 n^{1/2}/k$. This is the previous guess-and-determine problem with the effective noise size $\mathcal{D}$ multiplied by a factor $\mathcal{D}n^{1/2}/k$; so the bound SN is much larger than the bound for the temporary key, and is easily computable. Note that there are two copies of the same problem, the s should be doubled in computation of SN. In our examples, we don't let tSN, SN assume their maximal values according to above conditions, one can verify that the values we recommend satisfy the above conditions for the attack to fail. The actual $\mathcal{D}$ we used in these computations is about 10% lower than specified, this makes implementation of the sampling easier.

This completes the specification of our scheme for the security part, the remaining details are only relevant to parameter choice or efficiency.

5 Structured Multiplications

Next, we consider the techniques to optimize implementation of lattice public key cryptography, since we aim to give a light-weight integration of all functions. Note that the most computation-intensive operation in lattice PKC is polynomial multiplication if our signature scheme is used. A simple fact is that, 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 $a+T^ib$, 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_d. 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 ($< 2d_c$). 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 exhausting attack. We can lower the algebraic degree and use the sum of products and using other small S_ds to adjust the 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 $\mathcal{K}$, and we denote $S(N)$ as the structured elements of R with Euclidian norm square N, for example, $S(2 \times 12 + 3 \times 10 + 3 \times 9) \in S(81)$.

This observation will make encryption schemes with different Gaussian parameters all achieve the same level of extreme efficiency.

6 Implementation Techniques

To optimize our scheme towards extreme speed and light-weight, we recommend additional techniques in our structured secrets and samplers.

6.1 *Rgen*()

The objective of this algorithm is to generate the secrets $\{s_i : 1 \leq i \leq 16\}$. We need multiplication by s_i is simple. One way to do this is to let it randomly chosen from $S(N_s)$, where N_s is the desired euclidian norm square. We guess that this is not secure. So we use the following algorithm: Suppose $N_s \approx 2^{l_1}3^{l_2}5^{l_3}$. $\{s_i : 1 \leq i \leq 16\}$ is initialize as random elements of weight 1. Then it goes rounds of updates according to the norm factorization $N_s \approx 2^{l_1}3^{l_2}5^{l_3}$: each round each element gets the norm multiplied by the corresponding factor d by a replacement with signed-shifts sum of d previous round elements, until the desired norm N_s is obtained. Meanwhile, the whole process is recorded as the structure of the final elements in a list rsl, which makes multiplication by these elements structured. In the description of the following algorithms, all variables are global and static.

$$RGen() := \left\{ \begin{array}{l} rsl = \{\}; \\ \underline{y} = (y_1, y_2, \cdots y_{16}) = \underline{0}; \\ \text{for } (1 \leq j \leq 16) \\ \qquad s_j \leftarrow S_1; \text{append } s_j \text{ to } rsl; B_s = 1; \\ \text{end-for;} \\ \text{repeat } l_1 \text{ times } Round(2); \\ \text{repeat } l_2 \text{ times } Round(3); \\ \text{repeat } l_3 \text{ times } Round(5); \end{array} \right\},$$

where $Round(\cdot), round(\cdot)$ is defined as

$$Round(i) := \left\{ \begin{array}{l} \text{for } (1 \leq j \leq 16) \\ \qquad \text{repeat } s'_j = round(i) \text{ until } 0.95iB_s \leq |s'_j|^2 \leq 1.05iB_s; \\ \qquad \text{append } \underline{y} \text{ to } rsl; \\ \text{end-for;} \\ \text{for } (1 \leq j \leq 16) \\ \qquad s_j = s'_j; \\ \text{end-for;} \\ B_s = i \times B_s; \end{array} \right\},$$

$$round(i) := \left\{ \begin{array}{l} \text{sample random integers } d_j \geq 0 \text{ for } 1 \leq j \leq 16, \\ \qquad \text{such that } \sum_j d_j = i; \\ \text{sample } y_j \in S_{d_j} \text{ for } 1 \leq j \leq 16; \\ \text{return } (\sum_j y_j s_j) \end{array} \right\}.$$

Note that the list rsl can be compressed by encoding.

6.2 Sampling of $\underline{r1}$

Let d_y be the integer such that $0.8\mathcal{D}^2 n \approx 2d_c d_y N_s$ (in fact we should first choose d_y then decide N_s), and

$$rd1(i) := \left\{ \begin{array}{c} \text{sample random integers } d_j \geq 0 \text{ for } 1 \leq j \leq 16, \\ \text{such that } \sum_j d_j = d_y; \\ \text{sample } y_j \in S_{d_j} \text{ for } 1 \leq j \leq 16; \\ \text{return } (\sum_j y_j s_j, \sum_j y_j s_{j+i \pmod{16}}) \end{array} \right\}.$$

In $rSample()$, suppose $\underline{r1} = (r1_1, r1_2, r1_3, r1_4)$, we let

$$(r1_1, r1_3) = rd1(1); (r1_2, r1_4) = rd1(2).$$

In $sSample()$, we let

$$(r1_1, r1_2) = rd1(3).$$

6.3 Sampling of $\underline{r2}$

Let $p_i = 1/16 + (2i-1)/256$ for $1 \leq i \leq 8$, and $p_{i+8} = 1/16 - (2i-1)/256$ for $1 \leq i \leq 8$.

$$rd2(i) := \left\{ \begin{array}{l} \text{sample random integers } d_j \geq 0 \text{ for } 1 \leq j \leq 16, \\ \text{according to the distribution } E(d_j) = d_y p_{j+i} \text{ and such that } \sum_j d_j = d_y; \\ \text{sample } y_j \in S_{d_j} \text{ for } 1 \leq j \leq 16; \\ \text{return } (\sum_j y_j s_j) \end{array} \right\}.$$

We let the 6 components of $\underline{r2}$ in $rSample()$ and $sSample()$ be generated by $rd2(i) : 1 \leq i \leq 6$ respectively. This makes the counter-measures for the reflexive pairs in quadratic attacks do not cancel each other.

6.4 Efficiency

What is required about d_y? First consider its impact on security of commitments Ys, which are RLWE instances with quite large gaussian parameter. A safe way is to let the components of $\underline{r1}$s and $\underline{r2}$s never repeat (too small d_y may results in such repeats), i.e. each RLWE instance is fresh. But it seems that such repeat is not so harmful: firstly repeats are hard to detect; and secondly even few repeats

at problem instances with the same coefficients are known, the large gaussian parameter makes them remain hard. Now look at how d_y affects zs. Each z is an instance of the assumed-hard problem (nonlinear equations of scarce variables with number of variables far greater than equations). These instances share some common variables (the secrets x, s, w, and at present we don't know how to exploit this. We select d_y to keep these instances at least λ-bit information-theoretic apart from each other. This implies the entropy of each component of $\underline{r2}$ is greater than $\lambda/2$. For $\lambda = 128, 80$, we set $d_y = 7, 4$ respectively.

Another implementation issue is how to ensure the norm of the zs within the desired bound. If the signing device has the resources to compute norm squares (which needs integer multiplications), we may choose to append (before return) a loop to the $FS()$ function to ensure this by just refreshing the hash values:

$$ensure() := \left\{ \begin{array}{l} i = 0; \\ \text{while there's } z \text{ violating norm bound, do} \\ \qquad i++; \\ \qquad (c1, c2) = h_i(Y1, Y2, \cdots); \\ \qquad \underline{z} = \underline{u} + c1\underline{r1} + c2\underline{r2}; \\ \text{end-while} \\ i_h = i; \end{array} \right\},$$

where $h_i(Y1, Y2, \cdots)$ can be implemented as consecutive output of a stream cipher initialized using $(Y1, Y2, \cdots)$ ($h_0 = h$), and the integer i_h is included as a part of the signature (or certificate). When the signing device does not support integer multiplication, we may choose to let the caller do the check and let the device redo when this fails. We may choose the above parameters to make the average repeating times < 0.1.

Now it is easy to see that the cost of the algorithm Sign() is under $1.1(4d_c + 6d_y)$. The cost of Cert() is much higher since it does too much: key generation and pre-computations for the signing algorithm; yet when shared by each signature, it is much lower. Note that the computations in Cert() are mainly integer additions together with few general multiplications and 1 inversion in the ring.

6.5 Examples

We let $n = 512$, $q = 256$, $k = 45$, $\mathcal{D} = 32$, $d_c = 13$, $d_y = 4$ for 80 bits security. Let $N_s = 2^{12}$, i.e. $l_1 = 12, l_2 = l_3 = 0$. In sampling of rs, we ensure their norm squares do not exceed $1.1d_yN_s$. The average cost of each signature is about 80; while each encryption of LAC [11] cost 256 plus complex coding overhead. If we

choose the random secrets in encryption drawn from $S(2 \times 7 + 2 \times 8 + 2 \times 8)$, then the integrated encryption costs 52, and without any encoding the decryption virtually never fails. We recommend the bound of number of signatures for each temporary key is $tSN = 2^{12}$, and for the long term key $SN = 2^{25}$.

For 128 bits security, in the light-weight domain, we recommend $n = 512$, $q = 1024$, $k = 127$, $\mathcal{D} = 100$, $d_c = 19$, $d_y = 7$, $N_s = 2^3 \times 3^4 \times 5^2$. The distribution for random secrets in encryption is $S(6 \times 7 + 6 \times 7 + 7 \times 7)$. Each signature cost about 130, each encryption cost 74. We can set $tSN = 2^{15}$, and $SN = 2^{30}$.

If we wish a huge bound SN, we have to make n larger. For $n = 1024$, we can set $q = 2^{16}$, $k = 65$, $\mathcal{D} = 2^{12}$. We can set $tSN = 2^{40}$, $SN = 2^{80}$. In this case, we can use the basic form of the Fiat-Shamir protocol to make the size of signature about 1 ring element. The resulted effective noise size is $\mathcal{D} = 2^{12}/(18)^{1/2}$, it is safe to set $tSN = 2^{32}$ and $SN = 2^{64}$. Each signature costs < 100 in this case.

7 Applications

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 Cert(), CertVerify(), Verify() which involve multiplication/division of general elements in R_q, can also be delegated efficiently.

To compute AB for any public random A, B: Pick $v, v' \leftarrow S(\mathcal{K})$ and send the helper $(A, B, vB + v')$, who returns $(u = AB, u' = A(vB + v'))$, and now $u' = vu + v'A$ means u is the right answer: any success cheating means the helper can recover v, v', contradicting the RLWE hardness.

To compute w/A for sensitive general A and $w \in S_k$: Pick $v, v', v_0, v_1, v_2 \leftarrow S(\mathcal{K})$, let $A' = Av$, delegate-compute $A'' = A'X$, let $B' = v(w + A''v_2) + A'v', B = v_0A' + v_1B'$, and send the helper (A', B, B'), who returns $(u = B/A', u' = B'/A')$. Now verify $u = v_0 + v_1u'$, then accept $u' - v_2X - v'$ as the right answer.

Now that the whole public key functions are realized using only very limited computations, so allow a simple integrated circuit implementation. Together with construction of integrated symmetric key functionalities similar as [19], 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. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - dilithium: Digital signatures from module lattices. IACR Cryptol. ePrint Arch. p. 633 (2017), http://eprint.iacr.org/2017/633
6. Galbraith, S.D.: Mathematics of Public Key Cryptography. Cambridge University Press (2012), https://www.math.auckland.ac.nz/%7Esgal018/crypto-book/crypto-book.html
7. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 197–206. ACM (2008). https://doi.org/10.1145/1374376.1374407, https://doi.org/10.1145/1374376.1374407
8. Hoffstein, J., Howgrave-Graham, N., Pipher, J., Silverman, J.H., Whyte, W.: NTRUSIGN: digital signatures using the NTRU lattice. In: Joye, M. (ed.) Topics in Cryptology - CT-RSA 2003, The Cryptographers' Track at the RSA Conference 2003, San Francisco, CA, USA, April 13-17, 2003, Proceedings. Lecture Notes in Computer Science, vol. 2612, pp. 122–140. Springer (2003). https://doi.org/10.1007/3-540-36563-X_9, https://doi.org/10.1007/3-540-36563-X_9
9. Hoffstein, J., Pipher, J., Silverman, J.H.: NSS: an NTRU lattice-based signature scheme. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding. Lecture Notes in Com-

puter Science, vol. 2045, pp. 211–228. Springer (2001). https://doi.org/10.1007/3-540-44987-6_14, https://doi.org/10.1007/3-540-44987-6_14

10. Lenstra, A.K., Lenstra, H.W., Lovasz, L.: Factoring polynomials with rational coefficients. MATH. ANN **261**, 515–534 (1982)
11. Lu, X., Liu, Y., Zhang, Z., Jia, D., Xue, H., He, J., Li, B.: LAC: practical Ring-LWE based public-key encryption with byte-level modulus. IACR Cryptol. ePrint Arch. p. 1009 (2018), https://eprint.iacr.org/2018/1009
12. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 738–755. Springer (2012). https://doi.org/10.1007/978-3-642-29011-4_43, https://doi.org/10.1007/978-3-642-29011-4_43
13. 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
14. Prest, T., Fouque, P.A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Ricosset, T., Seiler, G., Whyte, W., Zhang, Z.: FALCON. Tech. rep., National Institute of Standards and Technology (2020), available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions (2020)
15. 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
16. 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
17. 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
18. Ye, D.: Can lattice signature be as efficient as lattice encryption? IACR Cryptol. ePrint Arch. p. 2 (2021), https://eprint.iacr.org/2021/002
19. 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