Construction of Aigis-sig algorithm

I am not really an expert in anything specially not post-quantum cryptography or just algorithms in general. So, I will try to gather all the available information first and then work on it.

The paper titled "*Experimental authentication of quantum key distribution with PQC*" **[1]** contains very generic definitions of the signature scheme for `KeyGen, Sign, Verify`.

Let's see what they share in the paper.

Algorithm 1: Key Generation Algorithm

Function KeyGen

$A \leftarrow R_q^{k \times l}$;
$s_1, s_2 \leftarrow S_\eta^l \times S_\eta^l$;
$t = As_1 + s_2$;
$pk = (A, t), sk = (s_1, s_2, pk)$;
return (pk, sk)

Algorithm 2: Signature Algorithm

Function Sign $sk{=}(s_1, s_2, pk), \mu$

repeat
$y \leftarrow S_{\gamma_1 - 1}^{l+k}$;
$w = Ay$;
$c = \mathrm{Hash}(\mathrm{HighBits}(w, 2\gamma_2) || \mu)$;
$z = y + cs_1$;
until $||z||_\infty < \gamma_1 - \beta$ *and*
$\mathrm{LowBits}(Ay - cs_2, 2\gamma_2) < \gamma_2 - \beta$;
return $\sigma = (z, c)$

Algorithm 3: Verification Algorithm

Function Verify $pk{=}(A,t), \sigma = (z, c), \mu$

if $||z||_\infty < \gamma_1 - \beta$ *and*
$c = \mathrm{Hash}(\mathrm{HighBits}(Az - ct, 2\gamma_2) || \mu)$ **then**
return *true* ;
return *false* ;

Figure 1 – Aigis-sig Scheme algorithms in Paper

If I had exact definitions of each line here, I could write some code, but no extra information is provided in the paper.

However, the paper does refer to other paper titled '*Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes.* **[2]**' It talks about the construction of the algorithm in more detail. The next step is to check the same and gather more information.

Paper [**2**] uses paper [**3**] for some simple algorithms. The understanding of these is important because they are being used in the signature scheme.

Algorithm 1: $\mathsf{Power2Round}_q(r, d)$

$r := r \bmod^{+} q$;
$r_0 := r \bmod^{\pm} 2^d$;
$r_1 := (r - r_0)/2^d$;
return (r_1, r_0);

Algorithm 2: $\mathsf{Decompose}_q(r, \alpha)$

$r := r \bmod^{+} q$;
$r_0 := r \bmod^{\pm} \alpha$;
if $r - r_0 = q - 1$ **then**
    $r_1 := 0$;
    $r_0 := r_0 - 1$;
else
    $r_1 := (r - r_0)/\alpha$;
end
return (r_1, r_0);

Algorithm 3: $\mathsf{HighBits}_q(r, \alpha)$

$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$;
return r_1;

Algorithm 4: $\mathsf{LowBits}_q(r, \alpha)$

$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$;
return r_0;

Algorithm 5: $\mathsf{MakeHint}_q(z, r, \alpha)$

$r_1 := \mathsf{HighBits}_q(r, \alpha)$;
$v_1 := \mathsf{HighBits}_q(r + z, \alpha)$;
if $r_1 \neq v_1$ **then**
    $h := 1$;
else
    $h := 0$;
end
return h;

Algorithm 6: $\mathsf{UseHint}_q(h, r, \alpha)$

$k := (q - 1)/\alpha$;
$(r_1, r_0) := \mathsf{Decompose}_q(r, \alpha)$;
if $h = 1$ *and* $r_0 > 0$ **then**
    $r_1 := (r_1 + 1) \bmod^{+} k$;
end
if $h = 1$ *and* $r_0 \leq 0$ **then**
    $r_1 := (r_1 - 1) \bmod^{+} k$;
end
return r_1;

Now I will look at them one-by-one and try to understand each line.

Algorithm 1

Algorithm 1: $\mathsf{Power2Round}_q(r, d)$

1. $r := r \bmod^+ q$;
2. $r_0 := r \bmod^{\pm} 2^d$;
3. $r_1 := (r - r_0)/2^d$;
4. **return** (r_1, r_0);

https://www.math.ucdavis.edu/~anne/WQ2007/mat67-Common_Math_Symbols.pdf

$:=$ (the **equal by definition sign**) means "is equal by definition to". This is a common alternate form of the symbol "$=_{\text{Def}}$", which appears in the 1894 book *Logica Matematica* by the logician Cesare Burali-Forti (1861–1931). Other common alternate forms of the symbol "$=_{\text{Def}}$" include "$\stackrel{\text{def}}{=}$" and "$\equiv$", the latter being especially common in applied mathematics.

2.2 Definitions

Modular Reductions. For an even positive integer α, we define $r' = r \bmod^{\pm} \alpha$ as the unique element in the range $(-\frac{\alpha}{2}, \frac{\alpha}{2}]$ such that $r' = r \bmod \alpha$. For an odd positive integer α, we define $r' = r \bmod^{\pm} \alpha$ as the unique element in the range $[-\frac{\alpha-1}{2}, \frac{\alpha-1}{2}]$ such that $r' = r \bmod \alpha$. For any positive integer α, we define $r' = r \bmod^+ \alpha$ as the unique element in the range $[0, \alpha)$ such that $r' = r \bmod \alpha$. When the exact representation is not important, we simply write $r \bmod \alpha$.

Algorithm 2

Algorithm 2: $\mathsf{Decompose}_q(r, \alpha)$

1. $r := r \bmod^+ q$;
2. $r_0 := r \bmod^{\pm} \alpha$;
3. **if** $r - r_0 = q - 1$ **then**
4. $\quad r_1 := 0$;
5. $\quad r_0 := r_0 - 1$;
6. **else**
7. $\quad r_1 := (r - r_0)/\alpha$;
8. **end**
9. **return** (r_1, r_0);

The definitions here are explained in the 'Modular Reduction' text above. By this I mean there are no new expressions that were not mentioned before.

Algorithm 3 and 4

Algorithm 3: $\mathsf{HighBits}_q(r, \alpha)$

```
(r_1, r_0) := Decompose_q(r, α);
return r_1;
```

Algorithm 4: $\mathsf{LowBits}_q(r, \alpha)$

```
(r_1, r_0) := Decompose_q(r, α);
return r_0;
```

These functions make use of `Decompose`$_q$`(r,a)` function to get one part each from the returned value.

Algorithm 5

Algorithm 5: $\mathsf{MakeHint}_q(z, r, \alpha)$

```
r_1 := HighBits_q(r, α);
v_1 := HighBits_q(r + z, α);
if r_1 ≠ v_1 then
    h := 1;
else
    h := 0;
end
return h;
```

Algorithm 6

Algorithm 6: $\mathsf{UseHint}_q(h, r, \alpha)$

```
k := (q − 1)/α;
(r_1, r_0) := Decompose_q(r, α);
if h = 1 and r_0 > 0 then
    r_1 := (r_1 + 1) mod⁺ k;
end
if h = 1 and r_0 ≤ 0 then
    r_1 := (r_1 − 1) mod⁺ k;
end
return r_1;
```

Algorithms 5 and 6 are not being called anywhere (in the algorithms shared before) so they are being used in the functions `KeyGen, Sign, Verify` most probably.

The Aigis-sig algorithm for key generation, signing and verification were taken from paper [2]. The signature scheme is given like below in the same paper.

- **Key generation**: randomly choose $\mathbf{A} \xleftarrow{\$} R_q^{k\times\ell}$, and $\mathbf{s}_1 \xleftarrow{\$} S_\eta^\ell, \mathbf{s}_2 \xleftarrow{\$} S_\eta^k$, compute $\mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2$. Return the public key $pk = (\mathbf{A}, \mathbf{t})$ and secret key $sk = (\mathbf{s}_1, \mathbf{s}_2, pk)$.
- **Signing**: given the secret key $sk = (\mathbf{s}_1, \mathbf{s}_2, pk)$ and a message $\mu \in \{0,1\}^*$,
  1. randomly choose $\mathbf{y} \xleftarrow{\$} S_{\gamma_1-1}^\ell$;
  2. compute $\mathbf{w} = \mathbf{A}\mathbf{y}$ and $c = \mathsf{H}(\mathsf{HighBits}(\mathbf{w}, 2\gamma_2)\|\mu)$;
  3. compute $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$;
  4. If $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ or $\mathsf{LowBits}(\mathbf{A}\mathbf{y} - c\mathbf{s}_2, 2\gamma_2) \geq \gamma_2 - \beta$, restart the computation from step 1), where β is a bound such that $\|c\mathbf{s}_1\|_\infty, \|c\mathbf{s}_2\|_\infty \leq \beta$ hold for all possible $c, \mathbf{s}_1, \mathbf{s}_2$. Otherwise, output the signature $\sigma = (\mathbf{z}, c)$.
- **Verification**: given the public key $pk = (\mathbf{A}, \mathbf{t})$, a message $\mu \in \{0,1\}^*$ and a signature $\sigma = (\mathbf{z}, c)$, return 1 if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ and $c = \mathsf{H}(\mathsf{HighBits}(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)\|\mu)$, otherwise return 0.

The above is given in more details later in the paper as shown below.

4.2 The Construction

Let $n, k, \ell, q, \eta_1, \eta_2, \beta_1, \beta_2, \gamma_1, \gamma_2, \omega \in \mathbb{Z}$ be positive integers. Let $R = \mathbb{Z}[x]/(x^n + 1)$ and $R_q = \mathbb{Z}_q[x]/(x^n + 1)$. Denote B_{60} as the set of elements of R that have 60 coefficients are either -1 or 1 and the rest are 0, and $|B_{60}| = 2^{60} \cdot \binom{n}{60}$. When $n = 256$, $|B_{60}| > 2^{256}$. Let $\mathsf{H}_1 : \{0,1\}^{256} \to R_q^{k\times\ell}, \mathsf{H}_2 : \{0,1\}^* \to \{0,1\}^{384}$, $\mathsf{H}_3 : \{0,1\}^* \to S_{\gamma_1-1}^\ell$ and $\mathsf{H}_4 : \{0,1\}^* \to B_{60}$ be four hash functions. We now present the description of our scheme $\Pi_{\mathrm{SIG}} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$:

- $\Pi_{\mathrm{SIG}}.\mathsf{KeyGen}(\kappa)$: first randomly choose $\rho, K \xleftarrow{\$} \{0,1\}^{256}$, $\mathbf{s}_1 \xleftarrow{\$} S_{\eta_1}^\ell, \mathbf{s}_2 \xleftarrow{\$} S_{\eta_2}^k$. Then, compute $\mathbf{A} = \mathsf{H}_1(\rho) \in R_q^{k\times\ell}, \mathbf{t} = \mathbf{A}\mathbf{s}_1 + \mathbf{s}_2 \in R_q^k, (\mathbf{t}_1, \mathbf{t}_0) = \mathsf{Power2Round}_q(\mathbf{t}, d)$ and $tr = \mathsf{H}_2(\rho\|\mathbf{t}_1) \in \{0,1\}^{384}$. Finally, return the public key $pk = (\rho, \mathbf{t}_1)$ and secret key $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$.

- $\Pi_{\mathrm{SIG}}.\mathsf{Sign}(sk, M)$: given $sk = (\rho, K, tr, \mathbf{s}_1, \mathbf{s}_2, \mathbf{t}_0)$ and a message $M \in \{0,1\}^*$, first compute $\mathbf{A} = \mathsf{H}_1(\rho) \in R_q^{k\times\ell}$, $\mu = \mathsf{H}_2(tr\|M) \in \{0,1\}^{384}$, and set $ctr = 0$. Then, perform the following computations:
  1. $\mathbf{y} = \mathsf{H}_3(K\|\mu\|ctr) \in S_{\gamma_1-1}^\ell$ and $\mathbf{w} = \mathbf{A}\mathbf{y}$;
  2. $\mathbf{w}_1 = \mathsf{HighBits}_q(\mathbf{w}, 2\gamma_2)$ and $c = \mathsf{H}_4(\mu\|\mathbf{w}_1) \in B_{60}$;
  3. $\mathbf{z} = \mathbf{y} + c\mathbf{s}_1$ and $\mathbf{u} = \mathbf{w} - c\mathbf{s}_2$;
  4. $(\mathbf{r}_1, \mathbf{r}_0) = \mathsf{Decompose}_q(\mathbf{u}, 2\gamma_2)$;
  5. if $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta_1$ or $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta_2$ or $\mathbf{r}_1 \neq \mathbf{w}_1$, then set $ctr = ctr + 1$ and restart the computation from step (1);
  6. compute $\mathbf{v} = c\mathbf{t}_0$ and $\mathbf{h} = \mathsf{MakeHint}_q(-\mathbf{v}, \mathbf{u} + \mathbf{v}, 2\gamma_2)$;
  7. if $\|\mathbf{v}\|_\infty \geq \gamma_2$ or the number of 1's in $\mathbf{h}$ is greater than ω, then set $ctr = ctr + 1$ and restart the computation from step 1);
  8. return the signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$.
- $\Pi_{\mathrm{SIG}}.\mathsf{Verify}(pk, M, \sigma)$: given the public key $pk = (\rho, \mathbf{t}_1)$, a message $M \in \{0,1\}^*$ and a signature $\sigma = (\mathbf{z}, \mathbf{h}, c)$, first compute $\mathbf{A} = \mathsf{H}_1(\rho) \in R_q^{k\times\ell}, \mu = \mathsf{H}_2(\mathsf{H}_2(pk)\|M) \in \{0,1\}^{384}$. Let $\mathbf{u} = \mathbf{A}\mathbf{z} - c\mathbf{t}_1 \cdot 2^d, \mathbf{w}_1' = \mathsf{UseHints}_q(\mathbf{h}, \mathbf{u}, 2\gamma_2)$ and $c' = \mathsf{H}_4(\mu\|\mathbf{w}_1')$. Finally, return 1 if $\|\mathbf{z}\|_\infty < \gamma_1 - \beta_1, c = c'$ and the number of 1's in $\mathbf{h}$ is $\leq \omega$, otherwise return 0.

I will try to go over the algorithms one-by-one and line-by-line.
I will use the available literature for notations, definitions and results.

Parameters for the Aigis-Sig scheme

Throughout the algorithm we see various parameters being used.

Here is a table mentioning the recommended parameters set.

Table 6. Parameters for Π_{SIG} (The column "Reps." indicates the excepted number of repetitions that the signing algorithm takes to output a valid signature)

Parameters	(k, ℓ, q, d, ω)	(η_1, η_2)	(β_1, β_2)	(γ_1, γ_2)	Reps.	Quant. Sec.
Π_{SIG}-1024	$(4, 3, 2021377, 13, 80)$	$(2, 3)$	$(120, 175)$	$(131072, 168448)$	5.86	90
Π_{SIG}-1280	$(5, 4, 3870721, 14, 96)$	$(2, 5)$	$(120, 275)$	$(131072, 322560)$	7.61	128
Π_{SIG}-1536	$(6, 5, 3870721, 14, 120)$	$(1, 5)$	$(60, 275)$	$(131072, 322560)$	6.67	163

Bibliography

[1] Experimental authentication of quantum key distribution with PQC

[2] Tweaking the asymmetry of asymmetric-key cryptography on lattices: KEMs and signatures of smaller sizes

[3] Crystals-dilithium: a lattice-based digital signature scheme. IACR Trans. Cryptogr. Hardw. Embed. Syst

Π_{SIG} KeyGen (K)

randomly choose ρ

$K \xleftarrow{\$} \{0,1\}^{256}$

$x \xleftarrow{\$} D$

means

sampling x acc. to distribution D

$S_1 \xleftarrow{\$} S_{\eta_1}^{l}$

$S_2 \xleftarrow{\$} S_{\eta_2}^{l}$

S_η → set of ring elements of R that each coeff is taken from $\{-\eta, -\eta+1, \dots, \eta\}$

column vector of length l

$\underbrace{A}_{\text{matrix}} = H_1(\rho) \in R_q^{k \times l}$

H_1 is a hash function

$H_1 : \{0,1\}^{256} \to R_q^{k \times l}$

R is a ring { Ring is a set together with 2 operations (+) & (·) satisfying some properties }

$R = \mathbb{Z}[X]/(X^n+1)$

$R_q = \mathbb{Z}_q[X]/(X^n+1)$

q is the modulo

$k \times l$ is the matrix size

There is also quotient ring of the form R/I where I is an ideal

I maybe X^n+1 and n is power of 2

this makes $x^n + 1$ [cyclotomic polyⁿ]

whose complex roots are primitive roots of unity

$$t = As_1 + s_2 \in R_q^k$$

$$(t_1, t_0) = \text{Power2Round}_q(t, d)$$

$$tr = \underbrace{H_2}_{\text{Hash fun}^n}(p \underbrace{\|}_{\text{concatenation}} t_1) \in \{0,1\}^{384}$$

public key $pk = (p, t_1)$

secret key $sk = (p, K, tr, s_1, s_2, t_0)$

$$\boxed{\Pi_{\text{sig}}\ \text{Sign}(sk, M)}$$

M is message

$M \in \{0,1\}^*$

$$A = H_1(p) \in R_q^{k \times l}$$

$$\mu = H_2(tr \| M) \in \{0,1\}^{384}$$

$ctr = 0$

Then some computations are performed

I will write them in the form of

step number — computation

1 $y = H_3(K \| \mu \| ctr) \in S^{\ell}_{\gamma_1 - 1}$

$w = Ay$ H_3 is hash funⁿ

$H_3 : \{0,1\}^* \to S^{\ell}_{\gamma_1 - 1}$

2 $w_1 = \text{HighBits}_q(w, 2\gamma_2)$

$c = H_4(\mu \| w_1) \in B_{60}$

H_4 is hash funⁿ $H_4 : \{0,1\}^* \to B_{60}$

B_{60} is set of elements of R that have 60 coeff which are either -1 or 1 & rest are 0.

3 $z = y + cs_1$ $u = w - cs_2$

4 $(r_1, r_0) = \text{Decompose}(u, 2\gamma_2)$

5 if $\|z\|_\infty \geq \gamma_1 - \beta_1$

or $\|r_0\|_\infty \geq \gamma_2 - \beta_2$ or $r_1 \neq w_1$

then $ctr = ctr + 1$ goto step 1

6 $v = ct_0$ & $h = \text{MakeHint}_q(-v, u + v, 2\gamma_2)$

7 if $\|v\|_\infty \geq \gamma_2$ or no. of 1's in $h > \omega$

then $ctr = ctr + 1$ goto step 1

8 return $\sigma = (z, h, c)$ signature

in the signing algorithm above, we notice the form $\|w\|_\infty$. For an element $w \in Z_q$ we write $\|w\|_\infty$ to mean $|w \bmod^{\pm} q|$

- $\|w\|_\infty$ is the l_∞ norm.
- norm refers to total length of all vectors in a space
- l_∞ norm gives the longest magnitude among each element of a vector
- $\|w\|_\infty = \max_i \|w_i\|_\infty$

we also see a lot of symbols (parameters) mentioned

$(k, l, q, d, \omega, \eta_1, \eta_2, \beta_1, \beta_2, \gamma_1, \gamma_2)$

The value used for these parameters is shared in the text (a table) previously.

The choice for these affect the security provided by the underlying hard problems SIS, LWE

Π_{SIG} Verify(pk, M, σ)

public key (p, t_1) $\in \{0,1\}^*$ message $\sigma = (z, h, c)$ signature

$$A = H_1(p) \in R_q^{k \times l}$$

$$\mu = H_2(H_2(pk) \,\|\, M) \in \{0,1\}^{384}$$

$$u = Az - ct_1 \cdot 2^d$$

$$w_1' = \mathrm{UseHints}_q(h, u, 2\gamma_2)$$

$$c' = H_4(\mu \,\|\, w_1')$$

if $\|z\|_\infty < \gamma_1 - \beta_1$ and $c = c'$

and number of 1's in $h \leq \omega$

return true

else

return false

There are 3 papers which seem to be most crucial when understanding the algorithms.

Experimental authentication of QKD with PQC

(paper in focus)

↑

Tweaking the Asymmetry of Asymmetric-Key Cryptography on Lattices: KEMs & Signatures of smaller sizes

(like it says, size optimization)

↑

CRYSTALS-Dilithium: A lattice-based Digital Signature scheme

(NIST PQC round-3 finalist)

contains the construction in more detail and with explainations.