***K**riptografi **A**tasi **Z**arah* Digital Signature (KAZ-SIGN)

Algorithm Specifications and Supporting Documentation

(Version 1.5)

Muhammad Rezal Kamel Ariffin[1] Nur Azman Abu[2] Terry Lau Shue Chien[3]
Zahari Mahad[1] Liaw Man Cheon[4] Amir Hamzah Abd Ghafar[1]
Nurul Amiera Sakinah Abdul Jamal[1]

[1]Institute for Mathematical Research, Universiti Putra Malaysia
[2]Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka
[3]Faculty of Computing & Informatics, Multimedia University Malaysia
[4]Antrapolation Technology Sdn. Bhd., Selangor, Malaysia

Table of Contents

Name of the proposed cryptosystem:	KAZ-SIGN
Principal submitter:	Muhammad Rezal Kamel Ariffin Institute for Mathematical Research Universiti Putra Malaysia 43400 UPM Serdang, Selangor Malaysia Email: rezal@upm.edu.my Phone: +60123766494
Auxilliary submitters:	Nor Azman Abu Terry Lau Shue Chien Zahari Mahad Liaw Man Cheon Amir Hamzah Abd Ghafar Nurul Amiera Sakinah Abdul Jamal
Inventor of the cryptosystem:	Muhammad Rezal Kamel Ariffin
Owner of the cryptosystem:	Muhammad Rezal Kamel Ariffin
Alternative point of contact:	Amir Hamzah Abd Ghafar Institute for Mathematical Research Universiti Putra Malaysia 43400 UPM Serdang, Selangor Malaysia Email: amir_hamzah@upm.edu.my Phone: +60132723347

1. INTRODUCTION

The proposed KAZ Digital Signature scheme, KAZ-SIGN (in Malay *Kriptografi Atasi Zarah* - translated literally “cryptographic techniques overcoming particles”; particles here referring to the photons) is built upon the hard mathematical problem coined as the Modular Reduction Problem (MRP). The idea revolves around the difficulty of reconstructing an unknown parameter from a given modular reducted value of that parameter. The target of the KAZ-SIGN design is to be a quantum resistant digital signature candidate with short verification keys and signatures, verifying correctly approximately 100% of the time, based on simple mathematics, having fast execution time and a potential candidate for seamless drop-in replacement in current cryptographic software and hardware ecosystems.

2. THE DESIGN IDEALISME

(i) To be based upon a problem that could be proven analytically to require exponential time to be solved;

(ii) To be able to prove analytically that the cryptosystem is indeed resistant towards quantum computers;

(iii) To utilize problems mentioned in point (i) above in its full spectrum without having to induce “weaknesses” in order for a trapdoor to be constructed;

(iv) To use “simple” mathematics in order to achieve maximum simplicity in design, such that even practitioners with limited mathematical background will be able to understand the arithmetic;

(v) Achieve 128 and 256-bit security with key length roughly equivalent to the non-quantum secure Elliptic Curve Cryptosystem (ECC);

(vi) To achieve maximum speed upon having simplicity in design and short key length;

(vii) To have a sufficiently large signature space;

(viii) The computation overhead for both signing and verification increases slightly even if the key size increases in the future;

(ix) To be able to be mounted on hardware with ease;

(x) The plaintext to signature expansion ratio is kept to a minimum.

One of our key strategy to obtain items (i) - (v) was by utilizing our defined Modular Reduction Problem (MRP). It is defined in the following section.

3. MODULAR REDUCTION PROBLEM (MRP)

Let $N = \prod_{i=1}^{j} p_i$ be a composite number and $n = \ell(N)$. Let p_k be a factor of N. Choose $\alpha \in (2^{n-1}, N)$. Compute $A \equiv \alpha \pmod{p_k}$.

The MRP is, upon given the values (A, N, p_k), one is tasked to determine $\alpha \in (2^{n-1}, N)$.

4. COMPLEXITY OF SOLVING THE MRP

Let $n_{p_k} = \ell(p_k)$ be the bit length of p_k. The complexity to obtain α is $O(2^{n-n_{p_k}})$. When deploying Grover's algorithm on a quantum computer, the complexity to obtain α is $O(2^{\frac{n-n_{p_k}}{2}})$. In other words, if $p_k \approx N^{\delta}$, for some $\delta \in (0,1)$, the complexity to obtain α is $O(N^{1-\delta})$. When deploying Grover's algorithm on a quantum computer, the complexity to obtain α is $O(N^{\frac{1-\delta}{2}})$.

5. THE HIDDEN NUMBER PROBLEM (HNP) (Boneh and Venkatesan, 2001)

Fix p and u. Let $O_{\alpha,g}(x)$ be an oracle that upon input x computes the most u significant bits of $\alpha g^{x} \pmod{p}$. The task is to compute the hidden number $\alpha \pmod{p}$ in expected polynomial time when one is given access to the oracle $O_{\alpha,g}(x)$. Clearly, one wishes to solve the problem with as small u as possible. Boneh and Venkatesan (2001) demonstrated that a bounded number of most significant bits of a shared secret are as hard to compute as the entire secret itself.

The initial idea of introducing the HNP is to show that finding the u most significant bits of the shared key in the Diffie-Hellman key exchange using users public key is equivalent to computing the entire shared secret key itself.

6. THE HERMANN MAY REMARKS (Herrmann and May, 2008)

We will now observe two remarks by Herrmann and May. It discusses the ability and inability to retrieve variables from a given modular multivariate linear equation. But before that we will put forward a famous theorem of Minkowski that relates the length of the shortest vector in a lattice to the determinant (see Hoffstein et al. (2008)).

Theorem 1. *In an ω-dimensional lattice, there exists a non-zero vector v with*

$$\|v\| \leq \sqrt{\omega}\, det(L)^{\frac{1}{\omega}}$$

In lattices with fixed dimension we can efficiently find a shortest vector, but for arbitrary dimensions, the problem of computing a shortest vector is known to be NP-hard under randomized reductions (see Ajtai (1998)). The LLL algorithm, however, computes in polynomial time an approximation of the shortest vector, which is sufficient for many applications.

Remark 1. *Let $f(x_1,x_2,\ldots,x_k) = a_1x_1 + a_2x_2 + \ldots + a_kx_k$ be a linear polynomial. One can hope to solve the modular linear equation $f(x_1,x_2,\ldots,x_k) \equiv 0 \pmod{N}$, that is to be able to find the set of solutions $(y_1,y_2,\ldots,y_k) \in \mathbb{Z}_N^k$, when the product of the unknowns are smaller than the modulus. More precisely, let X_i be upper bounds such that $|y_i| \leq X_i$ for $1,\ldots,k$. Then one can roughly expect a unique solution whenever the condition $\prod_i X_i \leq N$ holds (see Herrmann and May (2008)). It is common knowledge that under the same condition $\prod_i X_i \leq N$ the unique solution $(y_1,y_2,\ldots,y_k)$ can heuristically be recovered by computing the shortest vector in an k-dimensional lattice by the LLL algorithm. In fact, this approach lies at the heart of many cryptographic results (see Bleichenbacher and May (2006); Girault et al. (1990) and Nguyen (2004)).*

We would like to provide the reader with the conjecture and remark given in Herrmann and May (2008).

Conjecture 1. *If in turn we have $\prod_i X_i \geq N^{1+\varepsilon}$ then the linear equation $f(x_1,x_2,\ldots,x_k) = \sum_{i=1}^k a_ix_i \equiv 0 \pmod{N}$ usually has N^ε many solutions, which is exponential in the bit-size of N.*

Remark 2. *From Conjecture 1, there is hardly a chance to find efficient algorithms that in general improve on this bound, since one cannot even output all roots in polynomial time.*

7. THE KAZ-SIGN DIGITAL SIGNATURE ALGORITHM

7.1 Background

This section discusses the construction of the KAZ-SIGN scheme. We provide information regarding the key generation, signing and verification procedures. But first, we will put forward functions that we will utilize and the system parameters for all users.

7.2 Utilized Functions

Let $H(\cdot)$ be a hash function. Let $\phi(\cdot)$ be the usual Euler-totient function. Let $\ell(\cdot)$ be the function that outputs the bit length of a given input.

7.3 System Parameters

From the given security parameter k, determine parameter j. Next generate a list of the first j-primes larger than 2, $P = \{p_i\}_{i=1}^j$. Let $N = \prod_{i=1}^j p_i$. As an example, if $j = 43$, N is 256-bits. Let $n = \ell(N)$ be the bit length of N. Choose a random prime in $g \in \mathbb{Z}_N$ of order G_g where at most $G_g \approx N^\delta$ for a chosen value of $\delta \in (0,1)$ and $\delta \to 0$. That is, $g^{G_g} \equiv 1 \pmod{N}$. Choose a random prime $R \in \mathbb{Z}_{\phi(N)}$ of order G_R, where $G_R \approx \phi(N)^\varepsilon$

for $\varepsilon \to 1$. That is, choose R with a large order in $\mathbb{Z}_{\phi(N)}$. Let $n_{G_R} = \ell(G_R)$ be the bit length of G_R. Such R, has its own natural order in Z_{Gg}. Let that order be denoted as G_{Rg}. We can observe the natural relation given by $R^{G_{Rg}} \equiv 1 \pmod{G_g}$ where $\phi(N) \equiv 0 \pmod{G_g}$ and $\phi(G_g) \equiv 0 \pmod{G_{Rg}}$. Let $n_{\phi(G_g)} = \ell(\phi(G_g))$ be the bit length of $\phi(G_g)$ and $n_{\phi(G_{Rg})} = \ell(\phi(G_{Rg}))$ be the bit length of $\phi(G_{Rg})$. Let q be a random k-bit prime. Let $Q = \prod_{i=1}^{25} p_i = 116431182179248680450031658440253681535$. Ensure that $\phi(\phi(G_{Rg})) < \phi(\phi(Q))$. The system parameters are $(g,k,q,Q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$.

7.4 KAZ-SIGN Algorithms

The full algorithms of KAZ-SIGN are shown in Algorithms 1, 2, and 3.

Algorithm 1 KAZ-SIGN Key Generation Algorithm

```
Input: System parameters $(g,k,q,Q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$
Output: Public verification key pair, $(V,W_A,W_B)$, and private signing key, $\alpha$
 1: Choose random $\alpha \in (2^{n_{\phi(G_g)}-2}, 2^{n_{\phi(G_g)}-1})$.
 2: Compute public verification key, $V \equiv \alpha \pmod{G_{Rg}q}$. Let $\alpha_F \equiv V \pmod{G_{Rg}}$.
 3: if $\gcd(V, G_{Rg}Q) \neq 1$ or $\gcd(\alpha_F, G_{Rg}Q) \neq 1$ then
 4:     Repeat from Step 1.
 5: end if
 6: Determine the order of $\alpha$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{\alpha G_{Rg}Q}$. That is, $\alpha^{G_{\alpha G_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$. Ensure that $G_{\alpha G_{Rg}Q}$ has minimum length of 76-bits (for 128-bit security level), minimum length of 110-bits (for 192-bit security level) and minimum length of 135-bits (for 256-bit security level).
 7: Determine the order of $V$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{VG_{Rg}Q}$. That is, $V^{G_{VG_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$. Ensure that $G_{VG_{Rg}Q}$ has minimum length of 76-bits (for 128-bit security level), minimum length of 110-bits (for 192-bit security level) and minimum length of 135-bits (for 256-bit security level).
 8: Determine the order of $\alpha_F$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{\alpha_F G_{Rg}Q}$. That is, $\alpha_F^{G_{\alpha_F G_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$
 9: Compute $W_0 \equiv \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$ and $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$.
10: Compute $Z_1 \equiv \alpha V^{-1} \pmod{G_{Rg}Q}$. Determine the order of $Z_1$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{Z_1 G_{Rg}Q}$. That is, $Z_1^{G_{Z_1 G_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$.
11: Compute $Z_2 \equiv \alpha\alpha_F^{-1} \pmod{G_{Rg}Q}$. Determine the order of $Z_2$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{Z_2 G_{Rg}Q}$. That is, $Z_2^{G_{Z_2 G_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$.
```

```
12: if $G_{VG_{Rg}Q} \not\equiv 0 \pmod{G_{\alpha_F G_{Rg}Q}}$ or $\phi(Q)W_A \equiv 0 \pmod{G_{Z_1 G_{Rg}Q}}$ or $\phi(Q)W_A \equiv 0$
    $\pmod{G_{Z_2 G_{Rg}Q}}$ then
13:     Repeat from Step 1.
14: end if
15: Compute $W_B \equiv \text{numer}(\frac{G_{\alpha G_{Rg}Q}}{\phi(Q)})$. That is, $\phi(Q)W_B \equiv 0 \pmod{G_{\alpha G_{Rg}Q}}$.
16: Output public verification key pair, $(V, W_A, W_B)$ and keep signing key $\alpha$ secret.
```

Algorithm 2 KAZ-SIGN Signing Algorithm

```
Input: System parameters $(g,k,q,Q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$, private signing key, $\alpha$, message to be signed, $m \in \mathbb{Z}_N$, and $V$.
Output: Signature and salt, $(S,\sigma)$
1: Let $m \in \mathbb{Z}_N$ be the message to be signed.
2: Choose salt $\sigma \in \{0,1\}^{32}$ and let $h = H(m||\sigma) \in \mathbb{Z}$.
3: if $h = H(m||\sigma) \in \mathbb{Z}$ is not a prime then
4:     Repeat from Step 2
5: end if
6: Determine the order of $h$ in $\mathbb{Z}_Q$. Denote the order as $G_{hQ}$. That is, $h^{G_{hQ}} \equiv 1 \pmod{Q}$.
7: Determine the order of $h$ in $\mathbb{Z}_{G_{Rg}Q}$. Denote the order as $G_{hG_{Rg}Q}$. That is, $h^{G_{hG_{Rg}Q}} \equiv 1$
   $\pmod{G_{Rg}Q}$.
8: if $G_{hG_{Rg}Q} \geq G_{VG_{Rg}Q}$ or $G_{hG_{Rg}Q} \geq G_{\alpha G_{Rg}Q}$, or $G_{\alpha G_{Rg}Q} \not\equiv 0 \pmod{G_{hG_{Rg}Q}}$, or $G_{VG_{Rg}Q} \not\equiv$
   $0 \pmod{G_{hG_{Rg}Q}}$, or $G_{hG_{Rg}Q} \not\equiv 0 \pmod{W_B}$, or $\phi(Q) \not\equiv 0 \pmod{G_{hQ}}$ then
9:     Repeat from Step 2 by choosing new salt $\sigma \in \{0,1\}^{32}$.
10: end if
11: Choose random ephemeral prime $\beta \in (2^{n_{\phi(G_g)}-2}, 2^{n_{\phi(G_g)}-1})$. Determine the order of $\beta$
    in $\mathbb{Z}_{G_{hQ}}$. Denote the order as $G_{\beta G_{hQ}}$. That is, $\beta^{G_{\beta G_{hQ}}} \equiv 1 \pmod{G_{hQ}}$.
12: Determine the order of $\beta$ in $\frac{\mathbb{Z}_{G_{hG_{Rg}Q}}}{W_B}$. Denote the order as $G_{\beta G_{hG_{Rg}Q}\text{inv}W_B}$. That is,
    $\beta^{G_{\beta G_{hG_{Rg}Q}\text{inv}W_B}} \equiv 1 \pmod{\frac{G_{hG_{Rg}Q}}{W_B}}$.
13: if $\phi(\phi(G_{Rg})) \not\equiv 0 \pmod{G_{\beta G_{hQ}}}$, or $\phi(\phi(G_{Rg})) \not\equiv 0 \pmod{G_{\beta G_{hG_{Rg}Q}\text{inv}W_B}}$ then
14:     Repeat from Step 2, by choosing new salt $\sigma \in \{0,1\}^{32}$.
15: end if
16: Choose random ephemeral $r_0, r_1 \in \mathbb{Z}_{\phi(G_{Rg}qQ)}$
17: Compute

    $$S \equiv (\alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \pmod{G_{Rg}qQ}$$

18: Output signature and salt, $(S,\sigma)$, and destroy $(\beta, r_0, r_1)$.
```

Algorithm 3 KAZ-SIGN Verification Algorithm

```
Input: System parameters $(g,k,q,Q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$, public verification key pair,
    $(V,W)$, message, $m$, signature and salt, $(S,\sigma)$.
Output: Accept or reject
 1: Determine $h = nextprime(H(m||\sigma))$.
 2: Determine the order of $V$ in $\mathbb{Z}_{G_{Rg}Q}$, denoted as $G_{VG_{Rg}Q}$. That is, $V^{G_{VG_{Rg}Q}} \equiv 1$
    $\pmod{G_{Rg}Q}$.
 3: Determine the order of $h$ in $\mathbb{Z}_{G_{Rg}Q}$. Denote the order as $G_{hG_{Rg}Q}$. That is, $h^{G_{hG_{Rg}Q}} \equiv 1$
    $\pmod{G_{Rg}Q}$.
 4: if $G_{VG_{Rg}Q} \not\equiv 0 \pmod{G_{hG_{Rg}Q}}$ then
 5:     Reject signature $\perp$
 6: else Continue Step 8
 7: end if
 8: Compute $w_0 \equiv S \pmod{G_{Rg}qQ} - S$.
 9: if $w_0 \neq 0$ then
10:     Reject signature $\perp$
11: else Continue Step 13
12: end if
13: Compute $w_1 \equiv Sh^{-1} \pmod{G_{Rg}Q}$. Compute $w_2 \equiv V^{\phi(Q)} \pmod{G_{Rg}Q}$. Compute
    $w_3 = w_1 - w_2$.
14: if $w_3 = 0$ then
15:     Reject signature $\perp$
16: else Continue Step 18
17: end if
18: Compute $w_4 \equiv Sh^{-1} \pmod{G_{Rg}Q}$. Compute $w_5 \equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}Q}$. Compute
    $w_6 = w_4 - w_5$.
19: if $w_6 = 0$ then
20:     Reject signature $\perp$
21: else Continue Step 23
22: end if
23: Compute $w_7 \equiv Sh^{-1} \pmod{G_{Rg}Q}$. Compute $w_8 \equiv w_7 V^{-\phi(Q)} \pmod{G_{Rg}Q}$. Compute
    $w_9 \equiv w_8^{W_A} \pmod{G_{Rg}Q}$.
24: if $w_9 = 1$ then
25:     Reject signature $\perp$
26: else Continue Step 28
27: end if
28: Compute $w_{10} \equiv Sh^{-1} \pmod{G_{Rg}Q}$. Compute $w_{11} \equiv w_{10}\alpha_F^{-\phi(Q)} \pmod{G_{Rg}Q}$. Com-
    pute $w_{12} \equiv w_{11}^{W_A} \pmod{G_{Rg}Q}$.
```

```
29: if $w_{12} = 1$ then
30:     Reject signature $\perp$
31: else Continue Step 33
32: end if
33: Compute $w_{13} \equiv Sh^{-1} \pmod{Q}$.
34: if $w_{13} \neq 1$ then
35:     Reject signature $\perp$
36: else Continue Step 38
37: end if
38: Compute $w_{14} \equiv S^{W_B} \pmod{G_{Rg}Q}$. Compute $w_{15} \equiv h^{W_B} \pmod{G_{Rg}Q}$. Compute $w_{16} =$
    $w_{14} - w_{15}$.
39: if $w_{16} \neq 0$ then
40:     Reject signature $\perp$
41: else Continue Step 43
42: end if
43: Compute the following procedure:
44:     Set $w_{17} = 0$
45:     Set $modulus = 0$
46:     $V_Q \equiv V^{\phi(Q)} \pmod{G_{Rg}}$
47:     for each factor $r_i^{e_i}$ of $G_{Rg}Q$ do
48:         for $soln = 0, 1, 2, \ldots, r_i^{e_i} - 1$ do
49:             if $soln$ mod $\gcd(Q, r_i^{e_i}) \not\equiv h \pmod{\gcd(Q, r_i^{e_i})}$ then next; end if
50:             if $soln$ mod $\gcd(G_{Rg}, r_i^{e_i}) \not\equiv h \cdot V_Q$ mod $\gcd(G_{Rg}, r_i^{e_i})$ then next; end if
51:             if $soln^{W_B}$ mod $\gcd(G_{Rg}Q, r_i^{e_i}) \not\equiv h^{W_B}$ mod $\gcd(G_{Rg}Q, r_i^{e_i})$then next; end if
52:             break
53:         end for
54:         $w_{17} = CRT([w_{17}, soln], (modulus, r_i^{e_i}))$
55:         $modulus = modulus \cdot r_i^{e_i}$
56:     end for
57: Compute $w_{18} \equiv W_{17} - S \pmod{G_{Rg}Q}$.
58: if $w_{18} = 0$ then
59:     Reject signature $\perp$
60: else Continue Step 62
61: end if
62: Compute the following procedure:
63:     Set $w_{19} = 0$
64:     Set $modulus = 0$
65:     $\alpha_{FQ} \equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}}$
```

```
66:     **for** each factor $r_i^{e_i}$ of $G_{Rg}Q$ **do**
67:         **for** $soln = 0, 1, 2, \ldots, r_i^{e_i} - 1$ **do**
68:             **if** $soln \bmod \gcd(Q, r_i^{e_i}) \not\equiv h \pmod{\gcd(Q, r_i^{e_i})}$ **then next; end if**
69:             **if** $soln \bmod \gcd(G_{Rg}, r_i^{e_i}) \not\equiv h \cdot \alpha_{FQ} \bmod \gcd(G_{Rg}, r_i^{e_i})$ **then next; end if**
70:             **if** $soln^{W_B} \bmod \gcd(G_{Rg}Q, r_i^{e_i}) \not\equiv h^{W_B} \bmod \gcd(G_{Rg}Q, r_i^{e_i})$ **then next; end if**
71:             **break**
72:         **end for**
73:         $w_{19} = CRT([w_{19}, soln], (modulus, r_i^{e_i}))$
74:         $modulus = modulus \cdot r_i^{e_i}$
75:     **end for**
76: Compute $w_{20} \equiv W_{19} - S \pmod{G_{Rg}Q}$.
77: **if** $w_{20} = 0$ **then**
78:     Reject signature $\perp$
79: **else** Continue Step 81
80: **end if**
81: Compute $y_1 \equiv g^{(R^S \pmod{G_g})} \pmod{N}$.
82: Compute $y_2 \equiv g^{(R^{(V^{\phi(Q)}(h) \pmod{G_{Rg}})} \pmod{G_g})} \pmod{N}$.
83: **if** $y_1 = y_2$ **then**
84:     accept signature
85: **else** reject signature $\perp$
86: **end if**
```

Steps 3, 4, 5, 6, and 7 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 1**, steps 8, 9, 10, 11, and 12 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 2**, steps 13, 14, 15, 16, and 17 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 3**, steps 18, 19, 20, 21, and 22 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 4**, steps 23, 24, 25, 26, and 27 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 5**, steps 28, 29, 30, 31, and 32 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 6**, steps 33, 34, 35, 36, and 37 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 7**, steps 38, 39, 40, 41, and 42 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 8**, steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, and 61 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 9**, and steps 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, and 80 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 10**.

8. THE DESIGN RATIONALE

8.1 A conjecture related to the values g=6007, R=6151, Q=116431182179248680450031658440253681535 and $N = \prod_{i=1}^{j} p_i$ for determined value of j.

Please refer to Table 1 for values of j corresponding to 128, 192 and 256-bit security level. The following conjecture is a result of an empirical finding from a 1000 data set.

Conjecture 2. *Let $g = 6007$, $R = 6151$, $Q = 116431182179248680450031658440253681535$, $N = \prod_{i=1}^{j} p_i$ for determined value of j. Suppose $\rho \in \mathbb{Z}_{G_{Rg}Q}$ where $\gcd(\rho, G_{Rg}Q) = 1$, we have the following tabulated information,*

	Minimum length order of ρ in $\mathbb{Z}_{G_{Rg}Q}$	Maximum length order of ρ in $\mathbb{Z}_{G_{Rg}Q}$	Average length order of ρ in $\mathbb{Z}_{G_{Rg}Q}$
128-bit security level ($j = 180$)	66-bits	76-bits	74-bits
192-bit security level ($j = 258$)	99-bits	110-bits	108-bits
258-bit security level ($j = 342$)	124-bits	135-bits	133-bits

Table 1

8.2 The complexity of step 8 during signing

8.2.1 The complexity of finding $G_{hG_{Rg}Q} < G_{VG_{Rg}Q}$ and $G_{hG_{Rg}Q} < G_{\alpha G_{Rg}Q}$

From steps 6 and 7 during key generation and Conjecture 2, the relations $G_{hG_{Rg}Q} < G_{VG_{Rg}Q}$ and $G_{hG_{Rg}Q} < G_{\alpha G_{Rg}Q}$ can be obtained in polynomial time.

8.2.2 The complexity of finding $G_{\alpha G_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$

Since we have the relations:

1. $G_{hG_{Rg}Q} < G_{\alpha G_{Rg}Q}$
2. $\phi(G_{Rg}Q) \equiv 0 \pmod{G_{\alpha G_{Rg}Q}}$
3. $\phi(G_{Rg}Q) \equiv 0 \pmod{G_{hG_{Rg}Q}}$,

the relation $G_{\alpha G_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$ can be obtained in polynomial time.

8.2.3 The complexity of finding $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$

Since we have the relations:

1. $G_{hG_{Rg}Q} < G_{VG_{Rg}Q}$
2. $\phi(G_{Rg}Q) \equiv 0 \pmod{G_{VG_{Rg}Q}}$
3. $\phi(G_{Rg}Q) \equiv 0 \pmod{G_{hG_{Rg}Q}}$,

the relation $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$ can be obtained in polynomial time.

8.2.4 The complexity of finding $G_{hG_{Rg}Q} \equiv 0 \pmod{W_B}$

Since we have:

1. $G_{\alpha G_{Rg}Q} \equiv 0 \pmod{W_B}$
2. $G_{\alpha G_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$ can be obtained in polynomial time

the relation $G_{hG_{Rg}Q} \equiv 0 \pmod{W_B}$ can be obtained in polynomial time.

8.2.5 The complexity of finding $\phi(Q) \equiv 0 \pmod{G_{hQ}}$

From the relations:

1. $h^{\phi(Q)} \equiv 1 \pmod{Q}$
2. $h^{G_{hQ}} \equiv 1 \pmod{Q}$, and

the relation $\phi(Q) \equiv 0 \pmod{G_{hQ}}$ can be obtained in polynomial time.

8.3 The complexity of step 13 during signing

8.3.1 The complexity of finding $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hQ}}}$

From the relations:

1. $R^{\phi(\phi(N))} \equiv 1 \pmod{\phi(N)}$
2. $R^{G_{Rg}} \equiv 1 \pmod{G_g}$
3. $\phi(N) \equiv 0 \pmod{G_g}$
4. $\phi(\phi(N)) \equiv 0 \pmod{G_{Rg}}$, which implies $\phi(\phi(\phi(\phi(N)))) \equiv 0 \pmod{\phi(\phi(G_{Rg}))}$

and since N is a product of first $j-$primes larger than 2 from the set $P = \{p_i\}_{i=1}^{j}$, it is clear that the prime decompositions of $\phi(\phi(G_{Rg}))$ consists of small primes from the set P.

Furthermore, from the relations:

5. $h^{\beta^{(\phi(\phi(Q))} \pmod{G_{hQ}})} \equiv h \pmod{Q}$
6. $h^{\beta^{(G_{\beta G_{hQ}})} \pmod{G_{hQ}})} \equiv h \pmod{Q}$ which implies $\phi(\phi(Q)) \equiv 0 \pmod{G_{\beta G_{hQ}}}$
7. $G_{\beta G_{hQ}} < \phi(\phi(G_{Rg})) < \phi(\phi(Q))$
8. $\phi(\phi(Q)) \equiv 0 \pmod{\phi(\phi(G_{Rg}))}$ with high probability

the relation $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hQ}}}$ can be obtained in polynomial time.

8.3.2 The complexity of finding $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hG_{Rg}Q}\mathbf{inv}W_B}}$

From the relations:

1. $h^{\phi(G_{Rg}Q)} \equiv 1 \pmod{G_{Rg}Q}$
2. $h^{G_{hG_{Rg}Q}} \equiv 1 \pmod{G_{Rg}Q}$
3. $\phi(G_{Rg}Q) \equiv 0 \pmod{G_{hG_{Rg}Q}}$
4. $\beta^{\phi(\phi(G_{Rg}Q))} \equiv 1 \pmod{\phi(G_{Rg}Q)}$
5. $\beta^{\phi(\phi(G_{Rg}Q))} \equiv 1 \pmod{G_{hG_{Rg}Q}}$
6. $G_{hG_{Rg}Q} \equiv 0 \pmod{W_B}$
7. $\beta^{\phi(\phi(G_{Rg}Q))} \equiv 1 \pmod{\frac{G_{hG_{Rg}Q}}{W_B}}$
8. $\phi(\phi(G_{Rg}Q)) \equiv 0 \pmod{G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B}}$
9. $G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B} < \phi(\phi(G_{Rg})) < \phi(\phi(G_{Rg}Q))$
10. $\phi(\phi(G_{Rg}Q)) \equiv 0 \pmod{\phi(\phi(G_{Rg}))}$ with high probability

the relation $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B}}$ can be obtained in polynomial time.

8.4 Proof of correctness (Verification steps 81, 82, 83, 84, 85, and 86)

We begin by discussing the rationale behind steps 81, 82, 83, 84, 85, and 86 with relation to the verification process. Observe the following,

$$\begin{aligned} g^{(R^S \pmod{G_g})} &\equiv g^{R^{(\alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)})}) \pmod{G_{Rg}}} \pmod{G_g}} \\ &\equiv g^{R^{((\alpha^{(\phi(Q))})(h^{(1)}) \pmod{G_{Rg}})} \pmod{G_g}} \equiv g^{(R^{((V^{\phi(Q)}(h) \pmod{G_{Rg}})} \pmod{G_g})} \pmod{N} \end{aligned}$$

because $\alpha \equiv V \pmod{G_{Rg}}$. As such the verification process does indeed provide an indication that the signature is indeed from an authorized sender with the private signing key α.

8.5 Proof of correctness (Verification steps 8, 9, 10, 11, and 12: KAZ-SIGN digital signature forgery detection procedure type – 2)

In order to comprehend the rationale behind steps 8, 9, 10, 11, and 12, one has to observe the following,

$$w_0 \equiv S \pmod{G_{Rg}qQ} - S = 0$$

because $S < G_{Rg}qQ$.

8.6 Proof of correctness (Verification steps 13, 14, 15, 16, and 17: KAZ-SIGN digital signature forgery detection procedure type – 3)

In order to comprehend the rationale behind steps 13, 14, 15, 16, and 17, one has to observe the following;

$$w_1 \equiv Sh^{-1} \equiv \alpha^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})})h^{-1} \not\equiv w_2 \equiv V^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_3 \neq 0$.

8.7 Proof of correctness (Verification steps 18, 19, 20, 21, and 22: KAZ-SIGN digital signature forgery detection procedure type – 4)

In order to comprehend the rationale behind steps 18, 19, 20, 21, and 22, one has to observe

$$w_4 \equiv Sh^{-1} \equiv \alpha^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})})h^{-1} \not\equiv w_5 \equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_6 \neq 0$.

8.8 Proof of correctness (Verification steps 23, 24, 25, 26, and 27: KAZ-SIGN digital signature forgery detection procedure type – 5)

In order to comprehend the rationale behind steps 23, 24, 25, 26, and 27, one has to observe

$$w_7 \equiv Sh^{-1} \equiv \alpha^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})})h^{-1} \not\equiv V^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence,

$$w_8 \equiv w_7 V^{-\phi(Q)} \equiv (\alpha V^{-1})^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})-1}) \pmod{G_{Rg}Q}.$$

Since during key generation we have the condition $\phi(Q)W_A \not\equiv 0 \pmod{G_{Z_1 G_{Rg} Q}}$, we can conclude that

$$w_9 \equiv w_8^{W_A} \not\equiv 1 \pmod{G_{Rg}Q}.$$

8.9 Proof of correctness (Verification steps 28, 29, 30, 31, and 32: KAZ-SIGN digital signature forgery detection procedure type – 6)

In order to comprehend the rationale behind steps 28, 29, 30, 31, and 32, one has to observe

$$w_{10} \equiv Sh^{-1} \equiv \alpha^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})})h^{-1} \not\equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence,

$$w_{11} \equiv w_{10}\alpha_F^{-\phi(Q)} \equiv (\alpha\alpha_F^{-1})^{\phi(Q)}(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})-1}) \pmod{G_{Rg}Q}.$$

Since during key generation we have the condition $\phi(Q)W_A \not\equiv 0 \pmod{G_{Z_2 G_{Rg} Q}}$, we can conclude that

$$w_{12} \equiv w_{11}^{W_A} \not\equiv 1 \pmod{G_{Rg}Q}.$$

8.10 Proof of correctness (Verification steps 33, 34, 35, 36, and 37: KAZ-SIGN digital signature forgery detection procedure type – 7)

In order to comprehend the rationale behind steps 33, 34, 35, 36, and 37, one has to observe since we have the relations $\phi(Q) \equiv 0 \pmod{G_{hQ}}$ and $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hQ}}}$ where $\beta^{G_{\beta G_{hQ}}} \equiv 1 \pmod{G_{hQ}}$, we can conclude

$$\begin{aligned} w_{13} \equiv Sh^{-1} &\equiv h^{(\phi(G_{Rg}Q)r_1 + \beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)})}h^{-1} \\ &\equiv h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(Q)})}h^{-1} \equiv hh^{-1} \equiv 1 \pmod{Q}. \end{aligned}$$

8.11 Proof of correctness (Verification steps 38, 39, 40, 41, and 42: KAZ-SIGN digital signature forgery detection procedure type – 8)

In order to comprehend the rationale behind steps 38, 39, 40, 41, and 42, one has to observe

1. $\phi(Q)W_B \equiv 0 \pmod{G_{\alpha G_{Rg}Q}}$
2. $G_{hG_{Rg}Q} \equiv 0 \pmod{W_B}$
3. $\phi(\phi(G_{Rg})) \equiv 0 \pmod{G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B}}$

$$
\begin{aligned}
w_{14} \equiv S^{W_B} &\equiv (\alpha^{W_B(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(W_B(\phi(G_{Rg}Q)r_1+W_B\beta^{(\phi(\phi(G_{Rg})))} \pmod{G_{Rg}qQ}))}) \\
&\equiv (1)(h^{(W_B\beta^{(\phi(\phi(G_{Rg})))} \pmod{G_{Rg}Q}))}) \\
&\equiv h^{(W_B\beta^{(G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B})} \pmod{G_{hG_{Rg}Q}})} \\
&\equiv h^{(W_B(\beta^{(G_{\beta G_{hG_{Rg}Q}\mathrm{inv}W_B})} \pmod{\frac{G_{hG_{Rg}Q}}{W_B}})))} \\
&\equiv h^{W_B^{(1)}} \equiv h^{W_B} \pmod{G_{Rg}Q}.
\end{aligned}
$$

8.12 Proof of correctness (Verification steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, and 80: KAZ-SIGN digital signature forgery detection procedure type – 9 and type – 10)

In order to comprehend the rationale behind steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, and 80, one has to observe the relations given by

$$S \equiv (\alpha^{(\phi(Q))})(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)}))}) \pmod{G_{Rg}Q},$$

where S is a valid KAZ-SIGN digital signature, $V \not\equiv \alpha \pmod{G_{Rg}Q}$ and $\alpha_F \not\equiv \alpha \pmod{G_{Rg}Q}$, the output of KAZ-SIGN digital signature forgery detection procedure type – 9 and type – 10 upon valid KAZ-SIGN digital signature S would result in $w_{18} \neq 0$ and $w_{20} \neq 0$.

8.13 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 2.

An adversary utilizing a valid signature, S and resends it as follows:

$$S_{F1} \equiv S + G_{Rg}qQx \pmod{\theta G_{Rg}qQ}$$

for some random value of $x \in \mathbb{Z}$ and small value of $\theta \in \mathbb{Z}$, such that $\ell(S_{F1}) \approx \ell(S)$. That is, $\ell(S_{F1})$ is not suspicious to the verifier. It is easy to observe that S_{F1} will pass steps 81, 82, 83, 84, 84, 85, and 86. However, since

$$w_0 \equiv S_{F1} \pmod{G_{Rg}qQ} - S_{F1} \neq 0 \in \mathbb{Z}$$

the signature pair will fail KAZ-SIGN digital signature forgery detection procedure type – 2.

8.14 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 3

An adversary utilizing V, random parameters (r_0, r_1, r_2) and sends either one of the following parameters

$$S_{F2} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) \pmod{G_{Rg}qQ}$$
$$S_{F3} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$$

would result in S_{F2}, S_{F3} passing steps 81, 82, 83, 84, 84, 85, and 86. However, since for $i = 2, 3$ we will have

$$w_1 \equiv S_{Fi}h^{-1} \equiv w_2 \equiv V^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_3 = w_1 - w_2 = 0$. Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type - 3.

8.15 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 4

An adversary utilizing $\alpha_F \equiv V \pmod{G_{Rg}}$, random parameters (r_0, r_1, r_2) and sends either one of the following parameters

$$S_{F4} \equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) \pmod{G_{Rg}qQ}$$
$$S_{F5} \equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$$

would result in S_{F4}, S_{F5} passing steps 81, 82, 83, 84, 84, 85, and 86.
However, since for $i = 4, 5$, we will have

$$w_4 \equiv S_{Fi}h^{-1} \equiv w_5 \equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_6 = w_4 - w_5 = 0$. Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 4.

8.16 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 5

An adversary utilizing V, random parameters (r_0, r_1, r_2, β) and sends either one of the following parameters

$$S_{F6} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) \pmod{G_{Rg}qQ}$$
$$S_{F7} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$$
$$S_{F8} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)})}) \pmod{G_{Rg}qQ}$$
$$S_{F9} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) \pmod{G_{Rg}qQ}$$
$$S_{F10} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$$
$$S_{F11} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)})}) \pmod{G_{Rg}qQ}$$

would result in $S_{F6}, S_{F7}, S_{F8}, S_{F9}, S_{F10}, S_{F11}$ passing steps 81, 82, 83, 84, 84, 85, and 86. However, since for $i = 6, 7$, we will have

$$w_7 \equiv S_{Fi}h^{-1} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1)}) \pmod{G_{Rg}qQ}.$$

Hence,

$$w_8 \equiv w_7 \cdot V^{-\phi(Q)} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0)})(h^{(\phi(G_{Rg})\phi(Q)r_1)}) \pmod{G_{Rg}Q}.$$

Following through we have,

$$w_9 = w_8^{W_A} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0)W_A})(h^{(\phi(G_{Rg})\phi(Q)r_1)W_A}) \equiv 1 \pmod{G_{Rg}Q}$$

because $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$ and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$

Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 5.

For S_{F8} we can observe

$$w_7 \equiv S_{F8}h^{-1} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q)})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)})}) \pmod{G_{Rg}Q}.$$

Hence, $w_8 \equiv w_7 \cdot V^{-\phi(Q)} \equiv h^{(\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)})} \pmod{G_{Rg}Q}$. Following through we will have

$$\begin{aligned} w_9 \equiv w_8^{W_A} &\equiv (V^{(\phi(G_{Rg})\phi(Q)r_0)W_A})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q))W_A}}) \\ &\equiv 1 \pmod{G_{Rg}Q} \end{aligned} \tag{1}$$

Because of

1. $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$ and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$.

   And

2. From the relation
$$\beta^{\phi(\phi(G_{Rg}))} \equiv \beta_0 \pmod{\phi(G_{Rg}Q)}$$
   and
$$\beta^{\phi(\phi(G_{Rg}))} \equiv 1 \pmod{\phi(G_{Rg})}.$$

This implies $\beta^{\phi(\phi(G_{Rg}))} = \beta_0 + \phi(G_{Rg}Q)k_0$ for some $k_0 \in \mathbb{Z}$ and $\beta^{\phi(\phi(G_{Rg}))} = 1 + \phi(G_{Rg})k_1$ for some $k_1 \in \mathbb{Z}$. Since $\phi(G_{Rg}Q) \equiv 0 \pmod{\phi(G_{Rg})}$, we have

$$\beta_0 - 1 = \phi(G_{Rg})\kappa$$

for some $\kappa \in \mathbb{Z}$. Then, we can have

$$(\beta_0 - 1)W_A = \phi(G_{Rg})W_A\kappa$$

And because $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$, we have

$$(\beta_0 - 1)W_A \equiv 0 \pmod{G_{VG_{Rg}Q}}$$

which in turn implies
$$(\beta_0 - 1)W_A \equiv 0 \pmod{G_{hG_{Rg}Q}}$$
because $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$. Hence, (1) is true.

Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 5.

For $i = 9, 10$ we will have

$$w_7 \equiv S_{Fi}h^{-1} \equiv V^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_8 \equiv w_7 \cdot V^{-\phi(Q)} \equiv 1 \pmod{G_{Rg}Q}$ and $w_9 \equiv w_8^{W_A} \equiv 1 \pmod{G_{Rg}Q}$. Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 5.

For S_{F11}, we can observe

$$\begin{aligned} w_7 &\equiv S_{F11}h^{-1} \\ &\equiv (V^{\phi(Q)})(h^{(\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)})}) \pmod{G_{Rg}Q}. \end{aligned}$$

Hence,

$$\begin{aligned} w_8 &\equiv w_7 \cdot V^{-\phi(Q)} \\ &\equiv h^{(\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)}))} \pmod{G_{Rg}Q}. \end{aligned}$$

Following through we have,

$$\begin{aligned} w_9 &\equiv w_8^{W_A} \\ &\equiv h^{((\beta^{(\phi(\phi(G_{Rg})))}-1)W_A \pmod{\phi(G_{Rg}Q)})} \\ &\equiv 1 \pmod{G_{Rg}Q} \end{aligned}$$

as explained for (1).

Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 5.

8.17 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 6

An adversary utilizing $\alpha_F \equiv V \pmod{G_{Rg}}$, random parameters (r_0, r_1, r_2, β) and sends either one of the following parameters

$$\begin{aligned} S_{F12} &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) \pmod{G_{Rg}qQ} \\ S_{F13} &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ} \\ S_{F14} &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \pmod{G_{Rg}qQ} \\ S_{F15} &\equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) \pmod{G_{Rg}qQ} \\ S_{F16} &\equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ} \\ S_{F17} &\equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \pmod{G_{Rg}qQ} \end{aligned}$$

would result in $S_{F12}, S_{F13}, S_{F14}, S_{F15}, S_{16}, S_{17}$ passing steps 81, 82, 83, 84, 84, 85, and 86. However, since for $i = 12, 13$, we will have

$$w_{10} \equiv S_{Fi}h^{-1} \equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1)}) \pmod{G_{Rg}qQ}.$$

Hence, $w_{11} \equiv w_{10} \cdot \alpha_F^{-\phi(Q)} \equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0)})(h^{(\phi(G_{Rg})\phi(Q)r_1)}) \pmod{G_{Rg}Q}$. Following through we have,

$$w_{12} \equiv w_{11}^{W_A} \equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0)W_A})(h^{(\phi(G_{Rg})\phi(Q)r_1)W_A}) \equiv 1 \pmod{G_{Rg}Q}$$

because $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$, and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{\alpha_F G_{Rg}Q}}$ and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$.

Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 6.

For S_{F14} we can observe

$$\begin{aligned} w_{10} &\equiv S_{F14}h^{-1} \\ &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0)+\phi(Q)})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)}))}) \pmod{G_{Rg}Q}. \end{aligned}$$

Hence,

$$\begin{aligned} w_{11} &\equiv w_{10} \cdot \alpha_F^{-\phi(Q)} \\ &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0)})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)}))}) \pmod{G_{Rg}Q}. \end{aligned}$$

Following through we have,

$$\begin{aligned} w_{12} &\equiv w_{11}^{W_A} \\ &\equiv (\alpha_F^{(\phi(G_{Rg})\phi(Q)r_0)W_A})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)}))W_A}) \\ &\equiv 1 \pmod{G_{Rg}Q}. \end{aligned} \tag{2}$$

Because of

1. $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$, and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{\alpha_F G_{Rg}Q}}$ and $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$.

   And

2. From the relation
$$\beta^{\phi(\phi(G_{Rg}))} \equiv \beta_0 \pmod{\phi(G_{Rg}Q)}$$
   and
$$\beta^{\phi(\phi(G_{Rg}))} \equiv 1 \pmod{\phi(G_{Rg})}.$$

This implies $\beta^{\phi(\phi(G_{Rg}))} = \beta_0 + \phi(G_{Rg}Q)k_0$ for some $k_0 \in \mathbb{Z}$ and $\beta^{\phi(\phi(G_{Rg}))} = 1 + \phi(G_{Rg})k_1$ for some $k_1 \in \mathbb{Z}$. Since $\phi(G_{Rg}Q) \equiv 0 \pmod{\phi(G_{Rg})}$, we have

$$\beta_0 - 1 = \phi(G_{Rg})\kappa$$

for some $\kappa \in \mathbb{Z}$. Then, we can have

$$(\beta_0 - 1)W_A = \phi(G_{Rg})W_A\kappa.$$

And because $W_A = \frac{G_{VG_{Rg}Q}}{W_0}$, where $W_0 = \gcd(\phi(G_{Rg}), G_{VG_{Rg}Q})$, we have

$$(\beta_0 - 1)W_A \equiv 0 \pmod{G_{VG_{Rg}Q}}$$

which in turn implies

$$(\beta_0 - 1)W_A \equiv 0 \pmod{G_{hG_{Rg}Q}}$$

because $G_{VG_{Rg}Q} \equiv 0 \pmod{G_{hG_{Rg}Q}}$. Hence, (2) is true.

Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 6.

For $i = 15, 16$ we will have

$$w_{10} \equiv S_{Fi}h^{-1} \equiv \alpha_F^{\phi(Q)} \pmod{G_{Rg}Q}.$$

Hence, $w_{11} \equiv w_{10} \cdot \alpha_F^{-\phi(Q)} \equiv 1 \pmod{G_{Rg}Q}$ and $w_{12} \equiv w_{11}^{W_A} \equiv 1 \pmod{G_{Rg}Q}$. Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 6.

For S_{F17}, we can observe

$$\begin{aligned} w_{10} &\equiv S_{F17}h^{-1} \\ &\equiv (\alpha_F^{\phi(Q)})(h^{(\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)})}) \pmod{G_{Rg}Q}. \end{aligned}$$

Hence,

$$\begin{aligned} w_{11} &\equiv w_{10} \cdot \alpha_F^{-\phi(Q)} \\ &\equiv h^{(\beta^{(\phi(\phi(G_{Rg})))}-1 \pmod{\phi(G_{Rg}Q)})} \pmod{G_{Rg}Q}. \end{aligned}$$

Following through we have,

$$\begin{aligned} w_{12} &\equiv w_{11}^{W_A} \\ &\equiv h^{((\beta^{(\phi(\phi(G_{Rg})))}-1)W_A \pmod{\phi(G_{Rg}Q)})} \\ &\equiv 1 \pmod{G_{Rg}Q}. \end{aligned}$$

as explained for (1).

Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 6.

8.18 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 7

An adversary utilizing V or $\alpha_F \equiv V \pmod{G_{Rg}}$, and random parameters (r_0, r_1, r_2, r_3) and sends either the parameter

1. $S_{F18} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q)})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}r_2 \pmod{G_{Rg}qQ}$
2. $S_{F19} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q)(1+G_{Rg}r_2))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}r_3 \pmod{G_{Rg}qQ}$ OR
3. $S_{F20} \equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q)})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}r_2 \pmod{G_{Rg}qQ}$
4. $S_{F21} \equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q)(1+G_{Rg}r_2))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}r_3 \pmod{G_{Rg}qQ}$

would result in S_{F18}, S_{F19}, S_{F20} and S_{F21} passing steps 81, 82, 83, 84, 84, 85, and 86.

However, since for $i = 18, 19, 20, 21$, we will have

$$w_{13} \equiv S_{Fi}h^{-1} \equiv (h + G_{Rg}r_2)h^{-1} \not\equiv 1 \pmod{Q}.$$

Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 7.

8.19 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 8

Let $\gamma = \gcd(Q, G_{Rg})$ where $\gamma > 1$. And adversary could conduct the CRT upon the system of equations given by

$$S_{F20} \equiv h \pmod{\frac{Q}{\gamma}} \quad (3)$$

$$S_{F20} \equiv h(V^{\phi(Q)}) \pmod{G_{Rg}} \quad (4)$$

OR

$$S_{F21} \equiv h \pmod{\frac{Q}{\gamma}} \quad (5)$$

$$S_{F21} \equiv h(\alpha_F^{\phi(Q)}) \pmod{G_{Rg}}. \quad (6)$$

Conducting the CRT upon (3) and (4) would result in

$$S_{F20} \equiv h(V^{\phi(Q)}) \pmod{\frac{QG_{Rg}}{\gamma}}. \quad (7)$$

Conducting the CRT upon (5) and (6) would result in

$$S_{F21} \equiv h(\alpha_F^{\phi(Q)}) \pmod{\frac{QG_{Rg}}{\gamma}}. \tag{8}$$

Since $\frac{QG_{Rg}}{\gamma} \equiv 0 \pmod{Q}$, for $i = 20, 21$ we will have

$$\begin{aligned} S_{Fi} &\equiv h \pmod{Q} \\ S_{Fi} &\equiv h(V^{\phi(Q)}) \pmod{G_{Rg}} \end{aligned}$$

that would result in S_{F20} and S_{F21} passing steps 81, 82, 83, 84, 85, and 86. However, for $i = 20, 21$ we will have

$$w_{16} \equiv S_{Fi}^{W_B} - h^{W_B} \not\equiv 0 \pmod{G_{Rg}Q}$$

due to operation on different groups $\frac{\mathbb{Z}_{QG_{Rg}}}{\gamma}$ and $\mathbb{Z}_{G_{Rg}Q}$.

Thus, the signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 8.

8.20 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 9 and type – 10

It is clear that steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, and 75 during verification would produce forged signature that passes steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, and 61.

However, these signatures will fail KAZ-SIGN digital signature forgery detection procedure type – 9 and type – 10.

8.21 Extracting α

An approach to forge the signature would be to produce either one of the following:

1. $y_{\alpha 1} \equiv \alpha \pmod{G_{Rg}qQ}$ OR
2. $y_{\alpha 2} \equiv \alpha^{(\phi(G_{Rg}Q)r_0 + \phi(Q))} \pmod{G_{Rg}qQ}$ for some random value r_0.

8.21.1 Producing $y_{\alpha 1} \equiv \alpha \pmod{G_{Rg}qQ}$

From the public parameter $V \equiv \alpha \pmod{G_{Rg}q}$ and adversary can produce:

1. $\alpha \pmod{G_{Rg}q}$
2. $\alpha \pmod{G_{Rg}}$
3. $\alpha \pmod{q}$.

Thus, the adversary needs to obtain the corresponding parameters to execute the Chinese Remainder Theorem (CRT) to obtain $\alpha \pmod{G_{Rg}qQ}$.

1. $\alpha \pmod{Q} \rightarrow$ to execute CRT with $\alpha \pmod{G_{Rg}q}$
2. $\alpha \pmod{qQ} \rightarrow$ to execute CRT with $\alpha \pmod{G_{Rg}}$
3. $\alpha \pmod{G_{Rg}Q} \rightarrow$ to execute CRT with $\alpha \pmod{q}$.

8.21.1.1 To obtain $\alpha \pmod{Q}$

To obtain $\alpha \pmod{Q}$, the adversary will utilize equation S. Observe that

$$S \equiv (h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(Q)}))}) \not\equiv \alpha \pmod{Q}$$

Thus, this option is not viable.

8.21.1.2 To obtain $\alpha \pmod{qQ}$

To obtain $\alpha \pmod{qQ}$, the adversary will utilize equation S. Observe that

$$S \equiv (\alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{((\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \not\equiv \alpha \pmod{qQ}$$

Thus, this option is not viable.

8.21.1.3 To obtain $\alpha \pmod{G_{Rg}Q}$

To obtain $\alpha \pmod{G_{Rg}Q}$, the adversary will utilize equation S. Observe that

$$S \equiv (\alpha^{(\phi(Q))})(h^{(\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}Q)}))}) \not\equiv \alpha \pmod{G_{Rg}Q}$$

Thus, this option is not viable.

8.21.2 Producing $y_{\alpha 2} \equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))} \pmod{G_{Rg}qQ}$

An adversary is able to produce $y_{\alpha 2}$ when the value of β is known. Let us assume $\beta = 1$. The adversary would have the following relations:

$$z_1 \equiv Sh^{-1} \equiv \alpha^{(\phi(Q))} \pmod{G_{Rg}Q}.$$

Observe that

$$\gcd(\phi(Q), \phi(G_{Rg}Q) = \omega \neq 1.$$

As such, the adversary can compute

$$d = \frac{1}{\phi(Q)} \quad (\text{mod } \frac{\phi(G_{Rg}Q)}{\omega})$$

Which implies $d\phi(Q) = 1 + \frac{\phi(G_{Rg}Q)}{\omega}t$ for some $t \in \mathbb{Z}$. Hence, the adversary will obtain

$$z_2 \equiv z_1^d \equiv \alpha^{d\phi(Q)} \equiv \alpha^{1+\frac{\phi(G_{Rg}Q)}{\omega}t} \pmod{G_{Rg}Q}.$$

With an arbitrary chosen value r_0 and under the condition that $\phi(Q) \equiv 0 \pmod{\omega}$, the adversary would proceed to compute the following 2 relations:

$$\begin{aligned} z_3 \equiv z_2^{(\phi(G_{Rg}Q)r_0+\phi(Q))} &\equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))+\frac{\phi(G_{Rg}Q)}{\omega}t(\phi(G_{Rg}Q)r_0+\phi(Q))} \\ &\equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))} \pmod{G_{Rg}Q} \\ z_4 \equiv V^{(\phi(G_{Rg}Q)r_0+\phi(Q))} &\equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))} \pmod{q} \end{aligned}$$

Then upon conducting the CRT on the pair (z_3, z_4), will result in

$$z_5 \equiv \alpha^{(\phi(G_{Rg}Q)r_0)+\phi(Q))} \pmod{G_{Rg}qQ}.$$

Hence, obtaining $y_{\alpha 2}$. However, the assumption that secret β is known has to be fulfilled first.

8.22 Modular linear equation of S

In this direction we analyze

$$S \equiv (\alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \pmod{G_{Rg}qQ}$$

Let

1. $X_1 \equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))} \pmod{G_{Rg}qQ}$
2. $X_2 \equiv (h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}))}) \pmod{G_{Rg}qQ}$

Moving forward we have,

$$X_1 X_2 - S \equiv 0 \pmod{G_{Rg} q Q} \tag{9}$$

Let $\hat{X_1}$ be the upper bound for X_1 and $\hat{X_2}$ be the upper bound for X_2. From Conjecture 1, if one has the situation where $\hat{X_1}\hat{X_2} \gg G_{Rg}qQ$, then there is no efficient algorithm to output all the roots of (9). That is, (9) usually has $G_{Rg}qQ$ many solutions, which is exponential in the bit-size of $G_{Rg}qQ$.

To this end, since both $\alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))}$ and $h^{(\phi(G_{Rg}Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)})}$ are exponentially large, it is clear to conclude that $\hat{X_1}\hat{X_2} \gg G_{Rg}qQ$. This implies, there is no efficient algorithm to output all the roots of (9).

8.23 Implementation of the Hidden Number Problem (HNP)

From S, let us denote as follows:

1. $x_1 \equiv \alpha^{(\phi(G_{Rg}Q)r_0+\phi(Q))} \pmod{G_{Rg}qQ}$
2. $x_2 \equiv \phi(G_{Rg}Q)r_1 + \beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)}$

Thus, S can be re-written as

$$S \equiv (x_1)(h^{x_2}) \pmod{G_{Rg}qQ} \tag{10}$$

for unknown pair (x_1, x_2). It is obvious that (10) is the HNP.

9. SPECIFICATION OF KAZ-SIGN

The challenge faced by the adversary is to retrieve α from $V \equiv \alpha \pmod{G_{Rg}q}$. It is protected by the MRP. The MRP representation is given as follows:

$$t = \frac{\alpha - V}{G_{Rg}q}$$

Due to the strategies during key generation, we have the complexity $O(t) = O(q)$.

As such, the complexity of solving the MRP via $V \equiv \alpha \pmod{G_{Rg}q}$ will be the determining factor in identifying the suitable key length for each security level.

The following is the security specification for $\delta = 0.23$.

Number of primes in P	$\ell(q)$	$n = \ell(N)$	Total security level, k
180	134	1509	128
258	198	2321	192
342	264	3241	256

Table 2

10. IMPLEMENTATION AND PERFORMANCE

10.1 Key Generation, Signing and Verification Time Complexity

It is obvious that the time complexity for all three procedures is in polynomial time.

10.2 Parameter sizes

We provide here information on size of the key and signature based on security level information from Table 2 (for $\delta = 0.23$).

NIST Security Level	Number of primes in P	Security level, k	Length of parameter N (bits)	Public key size, (V, W_A, W_B) (bits)	Private key size, α (bits)	Signature Size (S, σ) (bits)	ECC key size (bits)
1	180	128	1509	$\approx 220 + 40 + 32$ = 292	≈ 352	$\approx 690 + 32 = 722$	256
3	258	192	2321	$\approx 340 + 60 + 64$ = 464	≈ 530	$\approx 1050 + 32 = 1082$	384
5	342	256	3241	$\approx 444 + 64 + 88$ = 596	≈ 700	$\approx 1390 + 32 = 1422$	521

Table 3

In the direction of the research, we also make comparison to ECC key length for the three NIST security levels. KAZ-SIGN key length did not achieve its immediate target of having approximately the same key length as ECC, but further research might find means and ways.

10.3 Key Generation, Signing and Verification Ease of Implementation

The algebraic structure of KAZ-SIGN has an abundance of programming libraries available to be utilized. Among them are:

1. GNU Multiple Precision Arithmetic Library (GMP); and
2. Standard C libraries.

10.4 Key Generation, Signing and Verification Empirical Performance Data

In order to obtain benchmarks, we evaluate our reference implementation on a machine using GCC Compiler Version 6.3.0 (MinGW.org GCC-6.3.0-1) on Windows 10 Pro, Intel(R) Core(TM) i7-4710HQ CPU @ 2.50GHz and 8.00 GB RAM (64-bit operating system, x64-based processor).

We have the following empirical results when conducting 100 key generations, 100 signings and 100 verifications:

Security level	Time (ms)		
	Key generation	Signing	Verification
128 - KAZ1509	4297	531	281
192 - KAZ2321	10047	875	531
256 - KAZ3241	18125	1375	1094

Table 4

11. ADVANTAGES AND LIMITATIONS

As we have seen, KAZ-SIGN can be evaluated through:

1. Key length
2. Speed
3. No verification failure

11.1 Key Length

KAZ-SIGN key length is comparable to non-post quantum algorithms such as ECC and RSA. For 256-bit security, the KAZ-SIGN key size is approximate 600-bits. ECC would use 521-bit keys and RSA would use 15360-bit keys.

11.2 Speed

KAZ-SIGN's speed analysis results stem from the fact that it has short key length to achieve 256-bit security plus its textbook complexity running time for both signing and verifying is $O(n^3)$ where parameter n here is the input length.

11.3 No verification failure

It is apparent that the execution of **KAZ-SIGN parameter suitability detection procedure** together with **KAZ-SIGN digital signature forgery detection procedure type – 1, type – 2, type – 3, type – 4, type – 5, type – 6, type – 7, type – 8, type – 9, and type – 10** within the verification procedure will enable the verification computational process by the recipient to verify or reject a digital signature that was received by the recipient with probability equal to 1. That is, the probability of verification failure is 0.

11.4 Limitation

As we have seen, limitation of KAZ-SIGN can be evaluated through:

1. Based on unknown problem, the Modular Reduction Problem (MRP)

11.4.1 Based on unknown problem, the Modular Reduction Problem (MRP)

The MRP is not a known hard mathematical problem which is quantum resistant and is subject to future cryptanalysis success in solving the defined challenge either with a classical or quantum computer.

12. CLOSING REMARKS

The KAZ-SIGN digital signature exhibits properties that might result in it being a desirable post quantum signature scheme. In the event that new forgery methodologies are found, as long as the procedure can also be done by the verifier, then one can add the new forgery methodology into the verification procedure. At the same time, the same forgery methodology can be inserted into the signing procedure in order to eliminate any chances the signer will produce a signature that will be rejected.

To this end, the security of the MRP is an unknown fact. We opine that, the acceptance of MRP as a potential quantum resistant hard mathematical problem will come hand in hand with a secure cryptosystem designed upon it. We welcome all comments on the KAZ-SIGN digital signature, either findings that nullify its suitability as a post quantum digital signature scheme or findings that could enhance its deployment and use case in the future.

Finally, we would like to put forward our heartfelt thanks to Prof. Dr. Abderrahmane Nitaj from Laboratoire de Mathématiques Nicolas Oresme, Université de Caen Basse Normandie, France for insights, comments, and friendship throughout the process and special thanks to Prof. Dr. Daniel J. Bernstein from University of Illinois at Chicago, United States of America who has given his thoughts and efforts throughout versions 1.0 until $1.5\beta.2$ of KAZ-SIGN.

13. ILLUSTRATIVE FULL SIZE TEST VECTORS – 1

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). That is, $P = \{3, 5, 7, \dots, 1087\}$. In this illustration, we provide a valid KAZ-SIGN signature S. The valid KAZ-SIGN signature will pass all 7 KAZ-SIGN digital signature forgery detection procedure types.

N :

1665409992402569056088099162882616633362634244067356501888501184798944673390411\6
0490173267662421037651076925218135417482822328634005702894401991339669414651118\4
5637269507076961986313197141424158604886280314066047206653222207353469933659597\5
3415679244320546140681916938894958694783504509315984550444746877596669802184487\7
3122994100821551380848897549374242095332359872258964174269418980707061566230310\9
$86271334632962653419873630528847259413332189960852075555 \approx 2^{1509}$

g :

6007

G_g :

6642524914739203510335957556368291920623114068857378765257238167887987635099098\5
$8902490872774504562957760000 \approx 2^{355} \approx 2^{0.235(1509)} \approx N^{0.235}$

R :

6151

G_{Rg} :

$964284630129748924872876000 \approx 2^{90} \approx N^{0.059}$

q :

2009559865622718903330596030154428804188\1

Q :

11643118217924868045003165844025368153\5

Key generation

α :
3572844728718886657954859714207486534618359213340196222220619112904936164811931338447665974422999017187577 $\approx 2^{351}$

V :
1204111334741742319641142332668642402838188599546545341475703819957

W_A :
1469533186727

W_B :
10568862430976285227607327

α_F :
741496958222320836580123577

MRP complexity upon t

$t = \dfrac{\alpha - V}{G_{Rg}q}$:
184377511733517544363052733386450922873 $\approx 2^{128}$

Signing

h :
79776735220587049571440525922467175573570400741227212455814989620815793975251

r_0 :
102778712139004328118940402307833599435848073890568073307809521911017628943807

r_1 :
107475675068633532113808768378082886868504532169498556487403543813081924622751

β :
108199061612480436491931600532549502099493421033904279930457002775665738610319

S :
13733892285867581103702266824538712585156702752194140498445570685474755926856372
9099476697801601975675251

<u>Verification</u>

<u>KAZ-SIGN digital signature forgery detection procedure type – 2</u>

w_0 :
0

<u>KAZ-SIGN digital signature forgery detection procedure type – 3</u>

w_3 :
67235523807345563523023482326536338744094561597222977613035380000

<u>KAZ-SIGN digital signature forgery detection procedure type – 4</u>

w_6 :
88965241133113520290507016834037353577455320277846236927716540000

<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>

w_9 :
90827433257489982376327405407682284714748779297087471260395680001

<u>KAZ-SIGN digital signature forgery detection procedure type – 6</u>

w_{12} :
41629240243016241922483394145187713827593190511165090994348020001

<u>KAZ-SIGN digital signature forgery detection procedure type – 7</u>

w_{13} :
1

KAZ-SIGN digital signature forgery detection procedure type – 8

w_{16} :
0

KAZ-SIGN digital signature forgery detection procedure type – 9

w_{18} :
82009559393408921618305076841284040656426552550666040355470740000

KAZ-SIGN digital signature forgery detection procedure type – 10

w_{20} :
82009559393408921618305076841284040656426552550666040355470740000

Final verification

y_1 and y_2 :
43668737997361719050693661454056355268870675602026963449199091755758497671352589
94605278332941511870917664149605687492343324742410174612593067384928443847170187
39804485364625835930569319525483983792689626812528808609436769619549493754884455
22621301035004245816351073613862899056045456813429682963080802174934243165880471
23448015341091671486884406782533119364430887903919524345336638541420434708069758
6123133948897448949124458504201263535922785519567414 23

14. ILLUSTRATIVE FULL SIZE TEST VECTORS – 2

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F2} \equiv (V^{(\phi(G_{Rg}Q)r_0 + \phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) \pmod{G_{Rg}qQ}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 3**.

r_0 :
112244679145025821045929212022420382281083829442885064813534066394335392109232

r_1 :
114475449272976211837636387966496782634094368421703212235613376496419952942408

S_{F2} :
16612260593863145080343955269113499168711368585378784946787743949414720217089527
40335596638760409432815251

KAZ-SIGN digital signature forgery detection procedure type – 3

w_3 :
0

Final verification

y_1 and y_2 :
43668737997361719050693661454056355268870675602026963449199091755758497671352589
94605278332941511870917664149605687492343324742410174612593067384928443847170187
39804485364625835930569319525483983792689626812528808609436769619549493754884455
22621301035004245816351073613862899056045456813429682963080802174934243165880471
23448015341091671486884406782533119364430887903919524345336638541420434708069758
612313394889744894912445850420126353592278551956741423

15. ILLUSTRATIVE FULL SIZE TEST VECTORS – 3

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F5} \equiv (\alpha_F^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 4**.

r_0 :
9811175691116266562678596713616311593288564246813250092798462382288386639126 3

r_1 :
8419377250329483522897410086440966350156926198092271523750576282016432332038 4

r_2 :
10819344889609066541020679415433801817980501954934448843598794794697588512940 7

S_{F5} :
104618182933588042384617119341426079980980743375307534435200107295309351224511 85
97032492344502994608215251

KAZ-SIGN digital signature forgery detection procedure type – 4

w_6 :
0

Final verification

y_1 and y_2 :
436687379973617190506936614540563552688706756020269634491990917557584976713525 89
946052783329415118709176641496056874923433247424101746125930673849284438471701 87
398044853646258359305693195254839837926896268125288086094367696195494937548844 55
226213010350042458163510736138628990560454568134296829630808021749342431658804 71
234480153410916714868844067825331193644308879039195243453366385414204347080697 58
61231339488974489491244585042012635359227855195674142 3

16. ILLUSTRATIVE FULL SIZE TEST VECTORS – 4

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F6} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) \pmod{G_{Rg}qQ}$. This signature will fail the **<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>**.

r_0 :
7290485638534807050759849073940884728724392887655395842878956946042122960 7404

r_1 :
9757636096317061002804825383648084679080187495727175926283798426520684791 0326

S_{F6} :
121132796063004800327093601667882193914445093475866533373983735209681119232452 33
42299463543174397057455251

<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>

w_9 :
1

<u>Final verification</u>

y_1 and y_2 :
4366873799736171905069366145405635526887067560202696344919909175575849767135 2589
9460527833294151187091766414960568749234332474241017461259306738492844384717 0187
3980448536462583593056931952548398379268962681252880860943676961954949375488 4455
2262130103500424581635107361386289905604545681342968296308080217493424316588 0471
2344801534109167148688440678253311936443088790391952434533663854142043470806 9758
61231339488974489491244585042012635359227855195674 1423

17. ILLUSTRATIVE FULL SIZE TEST VECTORS – 5

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F7} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5**.

r_0 :
950761042973116402209973987652761415000326456402952599382360027093112046815 78

r_1 :
9696107985084722965470041865928256663385550846946311225128980791362186270784 6

r_2 :
8206993773585268934901356603802379261181510192153056723875024221541075919738 7

S_{F7} :
117788608023938323938001938850907828031695199236199688042321975763897847943036 85
93603611405739664010655251

KAZ-SIGN digital signature forgery detection procedure type – 5

w_9 :
1

Final verification

y_1 and y_2 :
4366873799736171905069366145405635526887067560202696344919909175575849767135258 9
9460527833294151187091766414960568749234332474241017461259306738492844384717018 7
3980448536462583593056931952548398379268962681252880860943676961954949375488445 5
2262130103500424581635107361386289905604545681342968296308080217493424316588047 1
2344801534109167148688440678253311936443088790391952434533663854142043470806975 8
61231339488974489491244585042012635359227855195674142 3

18. ILLUSTRATIVE FULL SIZE TEST VECTORS – 6

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F8} \equiv (V^{(\phi(G_{Rg})\phi(Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg})\phi(Q)r_1+\beta^{(\phi(\phi(G_{Rg})))} \pmod{\phi(G_{Rg}qQ)})}) \pmod{G_{Rg}qQ}$. This signature will fail the **<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>**.

r_0 :
94082560105547131529016766214334884214406960698420181108238361492050730960340

r_1 :
65720427855858752406460327809885015224249315648554701901069574133946112788027

β :
109608379134681605170132951462015515029829900858659860881446129122794343533367

S_{F8} :
11842767854513774815712574785285792398815475310325737173004721336046288476462102
71890213417455872726555251

<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>

w_9 :
1

<u>Final verification</u>

y_1 and y_2 :
43668737997361719050693661454056355268870675602026963449199091755758497671352589
94605278332941511870917664149605687492343324742410174612593067384928443847170187
39804485364625835930569319525483983792689626812528808609436769619549493754884455
22621301035004245816351073613862899056045456813429682963080802174934243165880471
23448015341091671486884406782533119364430887903919524345336638541420434708069758
612313394889744894912445850420126353592278551956741423

19. ILLUSTRATIVE FULL SIZE TEST VECTORS – 7

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F18} \equiv (V^{(\phi(G_{Rg}Q)r_0+\phi(Q))})(h^{(\phi(G_{Rg}Q)r_1+1)}) + G_{Rg}Qr_2 \pmod{G_{Rg}qQ}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 7**.

r_0 :
111861207963148495300154086022918340777206663801766087260508609191580768030084

r_1 :
924957448838812902798074461260462270938335881859092432397837669161374802750 25

r_2 :
109648844228513019529025961365555930824535275404167239075686417660026746915178

S_{F18} :
371613439344618841980632202014160939549159253559964937154573879642353362226226 10
95475560285669530871632 51

KAZ-SIGN digital signature forgery detection procedure type – 7

w_{13} :
188645521446046729217839888006450395 76

Final verification

y_1 and y_2 :
436687379973617190506936614540563552688706756020269634491990917557584976713525 89
946052783329415118709176641496056874923433247424101746125930673849284438471701 87
398044853646258359305693195254839837926896268125288086094367696195494937548844 55
226213010350042458163510736138628990560454568134296829630808021749342431658804 71
234480153410916714868844067825331193644308879039195243453366385414204347080697 58
612313394889744894912445850420126353592278551956741423

20. ILLUSTRATIVE FULL SIZE TEST VECTORS – 8

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_{F20} \equiv h(V^{\phi(Q)}) \pmod{\frac{QG_{Rg}}{\gamma}}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 8**.

γ :
1963788631084825545

S_{F20} :
44230394190414966033747797635057049559520103251

KAZ-SIGN digital signature forgery detection procedure type – 8

w_{16} :
$-$54732936377288187679800732369132839565606419127241890217068732000

Final verification

y_1 and y_2 :
43668737997361719050693661454056355268870675602026963449199091755758497671352589
94605278332941511870917664149605687492343324742410174612593067384928443847170187
39804485364625835930569319525483983792689626812528808609436769619549493754884455
22621301035004245816351073613862899056045456813429682963080802174934243165880471
23448015341091671486884406782533119364430887903919524345336638541420434708069758
612313394889744894912445850420126353592278551956741423

21. ILLUSTRATIVE FULL SIZE TEST VECTORS – 9

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 3). This is an example for $j = 180$. That is, $P = \{3, 5, 7, \ldots, 1087\}$. In this illustration, we provide a forged KAZ-SIGN signature S where the system parameters, $(N, g, q, Q, G_g, R, G_{Rg}, \alpha, V, W, h)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and S_{F22} is derived as per steps 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56 during verification. The signature will fail the KAZ-SIGN digital signature forgery detection procedure type – 9 and type – 10.

S_{F22} :
1107942963024677017389774846860148416213996708411283991594493525 1

KAZ-SIGN digital signature forgery detection procedure type – 9

w_{18} :
0

KAZ-SIGN digital signature forgery detection procedure type – 10

w_{20} :
0

Final verification

y_1 and y_2 :
4366873799736171905069366145405635526887067560202696344919909175575849767135258 9
9460527833294151187091766414960568749234332474241017461259306738492844384717018 7
3980448536462583593056931952548398379268962681252880860943676961954949375488445 5
2262130103500424581635107361386289905604545681342968296308080217493424316588047 1
2344801534109167148688440678253311936443088790391952434533663854142043470806975 8
612313394889744894912445850420126353592278551956741423

References

Ajtai, M. (1998). The shortest vector problem in L2 is NP-hard for randomized reductions. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 10–19.

Bleichenbacher, D. and May, A. (2006). New attacks on RSA with small secret CRT-exponents. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pages 1–13. Springer.

Boneh, D. and Venkatesan, R. (2001). Hardness of computing the most significant bits of secret keys in Diffie-Hellman and related schemes. In *Advances in Cryptology-CRYPTO'96: 16th Annual International Cryptology Conference Santa Barbara, California, USA August 18–22, 1996 Proceedings*, pages 129–142. Springer.

Girault, M., Toffin, P., and Vallée, B. (1990). Computation of approximate L-th roots modulo n and application to cryptography. In *Advances in Cryptology—CRYPTO'88: Proceedings 8*, pages 100–117. Springer.

Herrmann, M. and May, A. (2008). Solving linear equations modulo divisors: On factoring given any bits. In *Advances in Cryptology-ASIACRYPT 2008: 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings 14*, pages 406–424. Springer.

Hoffstein, J., Pipher, J., Silverman, J. H., and Silverman, J. H. (2008). *An introduction to mathematical cryptography*, volume 1. Springer.

Nguyen, P. Q. (2004). Can we trust cryptographic software? Cryptographic flaws in GNU Privacy Guard v1. 2.3. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 555–570. Springer.