

Kriptografi Atasi Zarah Digital Signature (KAZ-SIGN)

Algorithm Specifications and Supporting Documentation

(Version 1.6.4)

Muhammad Rezal Kamel Ariffin¹ Nur Azman Abu² Kai Chieh Chang (Jay)³
Huang Po Yuan (Daniel)⁴ Zahari Mahad⁵ Liaw Man Cheon⁶
Amir Hamzah Abd Ghafar¹

¹Faculty of Science, Universiti Putra Malaysia

²Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka

³Architecture Design Department, Phison Electronics Corp., Taiwan

⁴Division of Solution Research and Development, Zyxel Solutions Corp., Taiwan

⁵Institute for Mathematical Research, Universiti Putra Malaysia

⁶Antrapolation Technology Sdn. Bhd., Selangor, Malaysia

Table of Contents

1	INTRODUCTION	1
2	THE DESIGN IDEALISME	1
3	MODULAR REDUCTION PROBLEM (MRP)	2
4	COMPLEXITY OF SOLVING THE MRP	2
5	THE HIDDEN NUMBER PROBLEM (HNP) (Boneh and Venkatesan, 2001)	2
6	THE HERMANN MAY REMARKS (Herrmann and May, 2008)	2
7	THE KAZ-SIGN DIGITAL SIGNATURE ALGORITHM	3
7.1	Background	3
7.2	Utilized Functions	3
7.3	System Parameters	3
7.3.1	A methodology for generating system parameters (G_0, G_1)	4
7.3.2	An example of methodology sub-subsection 7.3.1	4
7.4	KAZ-SIGN Algorithms	5
8	THE DESIGN RATIONALE	8
8.1	Proof of Correctness (Verification steps 29, 30, 31, 32, 33, and 34)	8
8.2	Proof of Correctness (Verification steps 35, 36, 37, 38, 39, and 40)	8
8.3	Proof of Correctness (Verification steps 4, 5, 6, 7, and 8: KAZ-SIGN digital signature forgery detection procedure type – 1)	9
8.4	Proof of Correctness (Verification steps 9, 10, 11, 12, and 13: KAZ-SIGN digital signature forgery detection procedure type – 2)	9
8.5	Proof of Correctness (Verification steps 14, 15, 16, 17, and 18: KAZ-SIGN digital signature forgery detection procedure type – 3)	9
8.6	Proof of Correctness (Verification steps 19, 20, 21, 22, and 23: KAZ-SIGN digital signature forgery detection procedure type – 4)	9
8.7	Proof of Correctness (Verification steps 24, 25, 26, 27, and 28: KAZ-SIGN digital signature forgery detection procedure type – 5)	10
8.8	Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 1.	10
8.9	Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 2	10
8.10	Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 3	11
8.11	Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 4	11
8.12	Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type - 5	11
8.13	Extracting α	12
8.13.1	Producing $y_{\alpha 1} \equiv \alpha \pmod{G_1 q Q \rho}$	12

8.13.2	Producing $y_{\alpha 2} \equiv \alpha^{\phi(Q)} \pmod{G_1 q Q \rho}$	12
8.14	Modular Linear Equation of S_1	12
8.15	Implementation of the Hidden Number Problem (HNP)	13
8.16	Analysis on V_2	13
9	DISCUSSION ON EXCLUSION OF ITERATIVE CRT PROCEDURE	16
10	UNDERSTANDING WHY $SK \not\equiv 0 \pmod{G_1 Q \rho}$ IS NECESSARY	18
11	DERIVING THE SECURITY LEVEL OF KAZ-SIGN	18
12	IMPLEMENTATION AND PERFORMANCE	18
12.1	Key Generation, Signing and Verification Time Complexity	18
12.2	Parameter sizes	19
12.3	Key Generation, Signing and Verification Ease of Implementation	19
12.4	Key Generation, Signing and Verification Empirical Performance Data	19
13	ADVANTAGES AND LIMITATIONS	20
13.1	Key Length	20
13.2	Speed	20
13.3	No Verification Failure	20
13.4	Limitation	20
13.4.1	Based on unknown problem, the Modular Reduction Problem (MRP)	20
14	CLOSING REMARKS	21
15	ILLUSTRATIVE FULL SIZE TEST VECTORS – 1	22
16	ILLUSTRATIVE FULL SIZE TEST VECTORS – 2	27
17	ILLUSTRATIVE FULL SIZE TEST VECTORS – 3	28
18	ILLUSTRATIVE FULL SIZE TEST VECTORS – 4	29
19	ILLUSTRATIVE FULL SIZE TEST VECTORS – 5	30
20	ILLUSTRATIVE FULL SIZE TEST VECTORS – 6	31
21	ILLUSTRATIVE FULL SIZE TEST VECTORS – 7	32
22	ILLUSTRATIVE FULL SIZE TEST VECTORS – 8	33
23	ILLUSTRATIVE FULL SIZE TEST VECTORS – 9	34

Name of the proposed cryptosystem: KAZ-SIGN 1.6.4

Principal submitter: Muhammad Rezal Kamel Ariffin
Faculty of Science
Universiti Putra Malaysia
43400 UPM Serdang, Selangor
Malaysia
Email: rezal@upm.edu.my
Phone: +60123766494

Auxilliary submitters: Nor Azman Abu
Kai Chieh Chang (Jay)
Huang Po Yuan (Daniel)
Zahari Mahad
Liaw Man Cheon
Amir Hamzah Abd Ghafar

Inventor of the cryptosystem: Muhammad Rezal Kamel Ariffin

Owner of the cryptosystem: Muhammad Rezal Kamel Ariffin

Alternative point of contact: Amir Hamzah Abd Ghafar
Faculty of Science
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 reduced 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 ran-

domized 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 $f(x_1, x_2, \dots, x_k) = a_1x_1 + a_2x_2 + \dots + a_kx_k$ be a linear polynomial. One can hope to solve the modular linear equation $f(x_1, x_2, \dots, x_k) \equiv 0 \pmod{N}$, that is to be able to find the set of solutions $(y_1, y_2, \dots, y_k) \in \mathbb{Z}_N^k$, 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 $1, \dots, k$. 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, \dots, 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 $f(x_1, x_2, \dots, x_k) = \sum_{i=1}^k a_i x_i \equiv 0 \pmod{N}$ 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 , compute $Q = \prod_{i=1}^{k_0} p_i$ where p_i is chosen consecutively from a list of the first j -primes larger than 2, $P = \{p_i\}_{i=1}^j$ and $k_0 < j$. Let $L_Q = \ell(Q)$. Next, generate a prime q where $L_q = \ell(q) \approx L_Q$. Next choose a prime ρ_0 where $\rho = 2\rho_0 + 1$ is a prime. Ensure that the length of ρ is either 256, 384 or 512 (depending on the security

level needed). Then, compute a random composite integer $G_0 = (2^{10}) \prod_{i=1}^{k_1} p_i^{e_i}$ where p_i is chosen randomly from the list P , e_i chosen randomly from \mathbb{Z}_{11} and $k_1 < j$. Then, choose a random prime R . Such R , has its own natural order in Z_{G_0} . Let that order be denoted as G_1 . We can observe the natural relation given by $R^{G_1} \equiv 1 \pmod{G_0}$ where $\phi(G_0) \equiv 0 \pmod{G_1}$. Let $L_{G_1\rho} = \ell(G_1\rho)$. Ensure that $L_{G_1qQ\rho} = \ell(G_1qQ\rho)$ is approximately either 386, 578 or 770 (depending on the security level needed). Otherwise, tweak the generation of the parameters (G_0, q, Q) accordingly. Let A be the product of the factors of G_1 (and its corresponding exponent), where the largest prime factor is $\approx 2^4$. The system parameters are $(k, q, \rho, Q, R, G_0, G_1, A, L_{G_1\rho}, L_{G_1qQ\rho})$.

7.3.1 A methodology for generating system parameters (G_0, G_1)

As mentioned in the preceding section, determine the parameter j . Let $N = \prod_{i=1}^j p_i$. Choose a random prime in $g \in \mathbb{Z}_N$. Such g , has its own natural order in Z_N . Let that order be denoted as G_{gN} . We can observe the natural relation given by $g^{G_{gN}} \equiv 1 \pmod{N}$ where $\phi(N) \equiv 0 \pmod{G_{gN}}$.

Choose a random prime $R \in \mathbb{Z}_{G_{gN}}$. Such R , has its own natural order in $Z_{G_{gN}}$. Let that order be denoted as G_{Rg} . We can observe the natural relation given by $R^{G_{Rg}} \equiv 1 \pmod{G_{gN}}$ where $\phi(G_{gN}) \equiv 0 \pmod{G_{Rg}}$.

We will then take $G_0 = G_{gN}$ and $G_1 = G_{Rg}$.

7.3.2 An example of methodology sub-subsection 7.3.1

Let $j = 128$. We have,

- $N = 3607412583971323219590003783834944284988520226319453743906969800919895176551430071528816513023400139449183391549534086592248781036129317137092037483563399346623614557751044797268991006479248759233160158836451176121534540730131221984779181743430655484717223193007705468625920195527456360287632608176655$
- $g = 6007$ and $R = 6151$
- $G_0 = G_{gN} = 23102151283542472555351033031857407110549489214984451103786304558150674606117088000 = (2^8)(3^4)(5^3)(7^2)(11^2)(13^2)(17)(19)(23)(29)(31)(37)(41)(43)(47)(53)(59)(61)(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(113)(127)(131)(139)(163)(173)(179)(191)(233)(239)(251)(281)(293)(359)$.
- $G_1 = G_{Rg} = 799241301392249419704000 = (2^6)(3^4)(5^3)(7)(11)(13)(17)(23)(29)(41)(43)(53)(73)(89)(179)$

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 $(k, q, \rho, Q, R, G_0, G_1, L_{G_1\rho}, A)$

Output: Public verification key pair, (V_1, V_2) and private signing key, SK .

- 1: Choose random prime a and random ω_1 both $\approx 2^{32}$.
 - 2: Compute secret parameter $b \equiv a^{\phi(\phi(G_1))(\rho_0-1)} \pmod{\omega_1 \phi(G_1)(\rho-1)}$.
 - 3: Choose even random $\alpha \approx 2^{k+L_{G_1\rho}}$
 - 4: Compute public verification key-1, $V_1 \equiv \alpha \pmod{G_1\rho}$.
 - 5: **if** $\gcd(V_1, \frac{G_1}{A}) \neq 1$ **then**
 - 6: Repeat from Step 4
 - 7: **else** Continue Step 9
 - 8: **end if**
 - 9: Compute public verification key-2, $V_2 \equiv Q(\alpha^{\phi(Q)b}) \pmod{qQ}$.
 - 10: Compute secret signing key, $SK \equiv \alpha^{\phi(Q)b} \pmod{G_1qQ\rho}$
 - 11: **if** $SK \equiv 0 \pmod{G_1Q\rho}$ **then**
 - 12: Repeat from Step 3
 - 13: **else** Continue Step 15
 - 14: **end if**
 - 15: Output public verification keys, (V_1, V_2) , keep signing key SK secret and destroy parameters (α, a, b, ω_1) .
-

Algorithm 2 KAZ-SIGN Signing Algorithm

Input: System parameters $(k, q, \rho, Q, R, G_0, G_1, L_{G_1qQ\rho})$, private signing key, SK and message to be signed, m .

Output: Signature, (S_1, S_2)

- 1: Let m be the message to be signed and let $h = (H(m))$ and $S_2 = 0$.
 - 2: Choose random prime r_1 , and random ω_2 both $\approx 2^{32}$.
 - 3: Compute secret parameter $\beta_1 \equiv r_1^{\phi(G_1)(\rho_0-1)} \pmod{\omega_2 \phi(G_1)(\rho-1)}$.
 - 4: Choose random prime r_2 , and random ω_3 both $\approx 2^{32}$.
 - 5: Compute secret parameter $\beta_2 \equiv r_2^{\phi(G_1)(\rho_0-1)} \pmod{\omega_3 \phi(G_1)(\rho-1)}$.
 - 6: Compute $S_1 \equiv (SK)(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \pmod{G_1qQ\rho}$.
 - 7: Compute $Y_1 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1Q\rho}$ and $S_{F1} = CRT([\frac{V_2}{Q}, Y_1], [q, G_1Q\rho])$.
 - 8: **if** $\ell(S_1) \neq L_{G_1qQ\rho}$ **or** $(S_1 \pmod{G_1qQ\rho}) - S_{F1} = 0$ **then**
 - 9: Compute $h = h + 1$.
 - 10: Compute $S_2 = S_2 + 1$.
 - 11: Repeat from Step 2
 - 12: **else** Continue Step 14
 - 13: **end if**
 - 14: Output signature, (S_1, S_2) , and destroy $(\beta_1, r_1, \omega_2, \beta_2, r_2, \omega_3)$.
-

Algorithm 3 KAZ-SIGN Verification Algorithm

Input: System parameters $(k, q, \rho, Q, R, G_0, G_1, L_{G_1qQ\rho})$, public verification key pair, (V_1, V_2) , message, m , and signature, (S_1, S_2) .

Output: Accept or reject signature

- 1: Compute $h = H(m) + S_2$.
- 2: Compute $Y_1 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1Q\rho}$ and $S_{F1} = CRT([\frac{V_2}{Q}, Y_1], [q, G_1Q\rho])$.
- 3: Compute $Y_2 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1\rho}$ and $S_{F2} = CRT([\frac{V_2}{Q}, Y_2], [\frac{qQ}{e}, G_1\rho])$ where $e = \gcd(Q, G_1)$.
- 4: Compute $w_0 \equiv (S_1 \pmod{G_1qQ\rho}) - S_1$.
- 5: **if** $w_0 \neq 0$ **then**
- 6: Reject signature \perp
- 7: **else** Continue Step 9
- 8: **end if**
- 9: Compute $w_1 = \ell(S_1) - L_{G_1qQ\rho}$.
- 10: **if** $w_1 \neq 0$ **then**
- 11: Reject signature \perp
- 12: **else** Continue Step 14
- 13: **end if**
- 14: Compute $w_2 \equiv (S_1 \pmod{G_1qQ\rho}) - S_{F1}$.
- 15: **if** $w_2 = 0$ **then**
- 16: Reject signature \perp
- 17: **else** Continue Step 19
- 18: **end if**
- 19: Compute $w_3 \equiv (S_1 \pmod{\frac{G_1qQ\rho}{e}}) - S_{F2}$ where $e = \gcd(Q, G_1)$.
- 20: **if** $w_3 = 0$ **then**
- 21: Reject signature \perp
- 22: **else** Continue Step 24
- 23: **end if**
- 24: Compute $w_4 \equiv \frac{QS_1}{2} \pmod{qQ}$. Compute $w_5 = w_4 - V_2$.
- 25: **if** $w_5 \neq 0$ **then**
- 26: Reject signature \perp
- 27: **else** Continue Step 29
- 28: **end if**
- 29: Compute $y_1 \equiv R^{S_1} \pmod{G_0}$.
- 30: Compute $y_2 \equiv R^{((V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1})} \pmod{G_0}$.
- 31: **if** $y_1 = y_2$ **then**
- 32: accept signature
- 33: **else** reject signature \perp
- 34: **end if**
- 35: Compute $y_3 \equiv \frac{S_1}{V_1^{\phi(Q)}} \pmod{\frac{G_1\rho}{A}}$.
- 36: Compute $y_4 \equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}}$.
- 37: **if** $y_3 = y_4$ **then**
- 38: accept signature
- 39: **else** reject signature \perp
- 40: **end if**

Steps 4, 5, 6, 7, and 8 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 1**, steps 9, 10, 11, 12, and 13 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 2**, steps 14, 15, 16, 17, and 18 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 3**, steps 19, 20, 21, 22, and 23 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 4**, and steps 24, 25, 26, 27, and 28 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 5**.

8. THE DESIGN RATIONALE

In this section we will analyse the rationale behind the design vis-à-vis a valid signature parameter S .

8.1 Proof of Correctness (Verification steps 29, 30, 31, 32, 33, and 34)

We begin by discussing the rationale behind steps 29, 30, 31, 32, 33, and 34 with relation to the verification process. Observe the following,

$$\begin{aligned} R^{S_1} &\equiv R^{(\alpha^{(\phi(Q)b)})(h^{(\phi(qQ)\beta_1)} + h^{(\phi(qQ)\beta_2)}) \pmod{G_1}} \\ &\equiv R^{(\alpha^{(\phi(Q))})(h^{(\phi(qQ))} + h^{(\phi(qQ))}) \pmod{G_1}} \\ &\equiv R^{(V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1}} \pmod{G_0} \end{aligned}$$

because $\alpha \equiv V_1 \pmod{G_1}$, $b \equiv 1 \pmod{\phi(G_1)(\rho - 1)}$ and $\beta_1, \beta_2 \equiv 1 \pmod{\phi(G_1)(\rho - 1)}$. As such the verification process does indeed provide an indication that the signature is indeed from an authorized sender with the private signing key SK .

8.2 Proof of Correctness (Verification steps 35, 36, 37, 38, 39, and 40)

We begin by discussing the rationale behind steps 35, 36, 37, 38, 39, and 40 with relation to the verification process. Observe the following,

$$\begin{aligned} S_1 &\equiv \alpha^{(\phi(Q)b)} (h^{(\phi(qQ)\beta_1)} + h^{(\phi(qQ)\beta_2)}) \\ &\equiv \alpha^{(\phi(Q)b)} (h^{(\phi(qQ))} + h^{(\phi(qQ))}) \\ &\equiv V_1^{\phi(Q)} 2h^{\phi(qQ)} \pmod{G_1\rho} \end{aligned}$$

Finally,

$$\frac{S_1}{V_1^{\phi(Q)}} \equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}}$$

because $\gcd(V_1, \frac{G_1}{A}) = 1$, $\alpha \equiv V_1 \pmod{\frac{G_1\rho}{A}}$, $b \equiv 1 \pmod{\phi(G_1)(\rho - 1)}$ and $\beta_1, \beta_2 \equiv 1 \pmod{\phi(G_1)(\rho - 1)}$. As such the verification process does indeed provide an indication that the signature is from an authorized sender with the private signing key SK .

8.3 Proof of Correctness (Verification steps 4, 5, 6, 7, and 8: KAZ-SIGN digital signature forgery detection procedure type – 1)

In order to comprehend the rationale behind steps 4, 5, 6, 7, and 8, one has to observe the following,

$$w_0 \equiv (S_1 \pmod{G_1qQ\rho}) - S_1 = 0$$

because $S_1 < G_1qQ\rho$.

8.4 Proof of Correctness (Verification steps 9, 10, 11, 12, and 13: KAZ-SIGN digital signature forgery detection procedure type – 2)

In order to comprehend the rationale behind steps 9, 10, 11, 12, and 13, one has to observe the following,

$$w_1 = \ell(S_1) - L_{G_1qQ\rho} = 0$$

because of steps 8, 11, 12 and 13 during signing.

8.5 Proof of Correctness (Verification steps 14, 15, 16, 17, and 18: KAZ-SIGN digital signature forgery detection procedure type – 3)

In order to comprehend the rationale behind steps 14, 15, 16, 17, and 18, one has to observe the following; obviously S_{F1} is not constructed with secret parameters SK . As such from $w_2 \equiv (S_1 \pmod{G_1qQ\rho}) - S_{F1}$, we will have $w_2 \neq 0$.

8.6 Proof of Correctness (Verification steps 19, 20, 21, 22, and 23: KAZ-SIGN digital signature forgery detection procedure type – 4)

In order to comprehend the rationale behind steps 19, 20, 21, 22, and 23, one has to observe the following; obviously S_{F2} is not constructed with secret parameters SK . As such from $w_3 \equiv (S_1 \pmod{\frac{G_1qQ\rho}{e}}) - S_{F2}$, where $e = \gcd(Q, G_1)$, we will have $w_3 \neq 0$.

8.7 Proof of Correctness (Verification steps 24, 25, 26, 27, and 28: KAZ-SIGN digital signature forgery detection procedure type – 5)

In order to comprehend the rationale behind steps 24, 25, 26, 27, and 28, one has to observe the following;

First we have,

$$h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2} \equiv 2 \pmod{q}$$

Thus,

$$Q(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2Q \pmod{qQ}$$

Finally,

$$w_4 \equiv QS_1 \equiv Q(\alpha^{\phi(Q)b})(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2Q(\alpha^{\phi(Q)b}) \pmod{qQ}$$

Hence, $w_5 = \frac{w_4}{2} - V_2 = 0$.

8.8 Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 1.

An adversary utilizing a valid signature, S_1 and its corresponding S_2 and resend it as follows:

$$S_F \equiv S_1 + G_1qQ\rho x \pmod{\theta G_1qQ\rho}$$

for some random value of $x \in \mathbb{Z}$ and small value of $\theta \in \mathbb{Z}$, such that $\ell(S_F) \approx \ell(S_1)$. That is, $\ell(S_F)$ is not suspicious to the verifier. It is easy to observe that S_F will pass steps 29, 30, 31, 32, 33, and 34 and also steps steps 35, 36, 37, 38, 39, and 40. However, since

$$w_0 \equiv (S_F \pmod{G_1qQ\rho}) - S_1 \neq 0 \in \mathbb{Z}$$

the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 1.

8.9 Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 2

An adversary utilizing a valid signature, S_1 and its corresponding S_2 and resend it as follows:

$$S_F \equiv S_1 \pmod{\frac{G_1qQ\rho}{u}}$$

where u is small composite of Q such that $\ell(S_F) \approx \ell(S_1)$. That is, $\ell(S_F)$ is not suspicious to the verifier. As an example $u = 3$. It is easy to observe that S_F will pass steps 29, 30, 31, 32, 33, and 34 and also steps 35, 36, 37, 38, 39, and 40. However, this would result in $\ell(S_F) \neq L_{G_1qQ\rho}$ and the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 2. That is, $w_1 \neq 0$.

8.10 Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 3

An adversary that constructs a forged signature S_1 as follows (and together with corresponding S_2); compute $Y_1 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1Q\rho}$ and $S_1 = CRT([\frac{V_2}{Q}, Y_1], [q, G_1Q\rho])$, and then transmits it as a signature S_1 would result in

$$w_2 \equiv (S_1 \pmod{G_1qQ\rho}) - S_{F1} = 0.$$

It is easy to observe that S_1 will pass steps 29, 30, 31, 32, 33, and 34 and also steps 35, 36, 37, 38, 39, and 40. However, the signature will fail KAZ-SIGN digital signature forgery detection procedure type - 3.

8.11 Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type – 4

An adversary that constructs a forged signature S_1 as follows (and together with corresponding S_2); compute $Y_2 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1\rho}$ and $S_1 = CRT([\frac{V_2}{Q}, Y_2], [\frac{qQ}{e}, G_1\rho])$ where $e = \gcd(Q, G_1)$, and then transmits it as a signature S_1 would result in

$$w_3 \equiv (S_1 \pmod{\frac{G_1qQ\rho}{e}}) - S_{F2} = 0.$$

where $e = \gcd(Q, G_1)$. It is easy to observe that S_1 will pass steps 29, 30, 31, 32, 33, and 34 and also steps 35, 36, 37, 38, 39, and 40. However, the signature will fail KAZ-SIGN digital signature forgery detection procedure type - 4.

8.12 Deriving Forged Signature Identifiable by KAZ-SIGN Digital Signature Forgery Detection Procedure Type - 5

An adversary that constructs a forged signature S_1 (and together with corresponding S_2); without the private random α and at the same time aspires to pass steps 29, 30, 31, 32, 33, and 34 and also steps 35, 36, 37, 38, 39, and 40, would result in the need to produce a forged signature S_1 that would eventually produce the relation,

$$S_1 \equiv (2\lambda^{\phi(Q)b'}) \pmod{qQ\rho}$$

where $\lambda = V_1 + G_1t$ for some $t \in \mathbb{Z}$ and $b' \equiv 1 \pmod{\phi(G_1)(\rho - 1)}$. It is clear that $\alpha \not\equiv V_1 + G_1t \pmod{qQ}$ and $b' \not\equiv b \pmod{\phi(qQ\rho)}$ with high probability. As such, $w_5 = \frac{w_4}{2} - V_2 \neq 0$ with high probability, where $w_4 \equiv 2Q(\lambda^{\phi(Q)b'}) \pmod{qQ}$. Thus, the signature will fail KAZ-SIGN digital signature forgery detection procedure type - 5.

8.13 Extracting α

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

1. $y_{\alpha 1} \equiv \alpha \pmod{G_1 q Q \rho}$ OR
2. $y_{\alpha 2} \equiv \alpha^{\phi(Q)} \pmod{G_1 q Q \rho}$.

8.13.1 Producing $y_{\alpha 1} \equiv \alpha \pmod{G_1 q Q \rho}$

From the public parameter $V_1 \equiv \alpha \pmod{G_1 \rho}$, the adversary needs to obtain the parameter $\alpha \pmod{qQ}$ to execute the Chinese Remainder Theorem (CRT) to obtain $\alpha \pmod{G_1 q Q \rho}$. To obtain $\alpha \pmod{qQ}$, the adversary will utilize equation S_1 . Observe that if h and qQ are co-prime, we can obtain

$$S_1 \equiv (\alpha^{\phi(Q)b})(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2\alpha^{\phi(Q)b} \not\equiv \alpha \pmod{qQ}$$

Thus, with the available parameters (S_1, V_1) , one is unable to produce $y_{\alpha 1}$.

8.13.2 Producing $y_{\alpha 2} \equiv \alpha^{\phi(Q)} \pmod{G_1 q Q \rho}$

To obtain $y_{\alpha 2}$, one begins with,

$$z_1 \equiv V_1^{\phi(Q)} \equiv \alpha^{\phi(Q)} \pmod{G_1 \rho}.$$

Then, one needs to produce the parameter $\alpha^{\phi(Q)} \pmod{qQ}$. Observe that if h and qQ are co-prime, we can obtain

$$z_2 \equiv S_1 \equiv (\alpha^{\phi(Q)b})(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2\alpha^{\phi(Q)b} \not\equiv \alpha^{\phi(Q)} \pmod{qQ}.$$

Thus, with the available parameters (S_1, V_1) , one is unable to produce $y_{\alpha 2}$.

8.14 Modular Linear Equation of S_1

In this direction we analyze

$$S_1 \equiv (\alpha^{\phi(Q)b})(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \pmod{G_1 q Q \rho}$$

Let

1. $X_1 \equiv \alpha^{\phi(Q)b} \pmod{G_1 q Q \rho}$
2. $X_2 \equiv h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2} \pmod{G_1 q Q \rho}$

Moving forward we have,

$$X_1 X_2 - S_1 \equiv 0 \pmod{G_1 q Q \rho} \quad (1)$$

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 $\hat{X}_1 \hat{X}_2 \gg G_1 q Q \rho$, then there is no efficient algorithm to output all the roots of (1). That is, (1) usually has $G_1 q Q \rho$ many solutions, which is exponential in the bit-size of $G_1 q Q \rho$.

To this end, since both $\alpha^{\phi(Q)b}$ and $h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}$ are exponentially large, it is clear to conclude that $\hat{X}_1 \hat{X}_2 \gg G_1 q Q \rho$. This implies, there is no efficient algorithm to output all the roots of (1).

8.15 Implementation of the Hidden Number Problem (HNP)

From S_1 , let us denote as follows:

1. $x_1 \equiv \alpha^{\phi(Q)b} \left(1 + h^{\phi(qQ)\beta_2 - \phi(qQ)\beta_1}\right) \pmod{G_1 q Q \rho}$
2. $x_2 \equiv \phi(qQ)\beta_1 \pmod{\phi(G_1 q Q \rho)}$

Thus, S_1 can be re-written as

$$S_1 \equiv (x_1)(h^{x_2}) \pmod{G_1 q Q \rho} \quad (2)$$

for unknown pair (x_1, x_2) . It is obvious that (2) is the HNP.

8.16 Analysis on V_2

Assume we have $V_{1,1} \equiv \alpha \pmod{G_1 q \rho}$. Let,

$$\begin{aligned} W_1 &\equiv \alpha^{\phi(Q)} \equiv V_{1,1}^{\phi(Q)} \pmod{q} \\ W_2 &\equiv V_2 Q^{-1} \equiv \alpha^{\phi(Q)b} \equiv V_{1,1}^{\phi(Q)b} \pmod{q} \end{aligned}$$

The aim is to obtain the system of equations

$$b \equiv \zeta \pmod{\phi(q)} \quad (3)$$

$$b \equiv a^{\phi(\phi(G_1))(\rho_0 - 1)} \pmod{\phi(G_1)(\rho - 1)} \quad (4)$$

for some arbitrarily chosen prime a .

We obtain ζ by solving the DLP upon $W_2 \equiv \alpha^{\phi(Q)b} \equiv V_{1,1}^{\phi(Q)b} \pmod{q}$, where the DL solver works on the base given by $W_1 \equiv V_{1,1}^{\phi(Q)} \pmod{q}$. That is, $z_1 \equiv b \equiv \zeta \pmod{\phi(q)}$

and $W_1^{z_1} \equiv V_{1,1}^{\phi(Q)b} \equiv W_2 \pmod{q}$.

Then, for some prime a , let $z_2 \equiv a^{\phi(\phi(G_1))(\rho_0-1)} \pmod{\phi(G_1)(\rho-1)}$.

Let $z_3 = \gcd(\phi(q), \phi(G_1)(\rho-1))$. Solving the CRT upon (3) and (4) modulo $(\frac{\phi(q)\phi(G_1)(\rho-1)}{z_3})$ would result in $z_4 \pmod{\frac{\phi(q)\phi(G_1)(\rho-1)}{z_3}}$, and since $\frac{\phi(q)\phi(G_1)(\rho-1)}{z_3} \equiv 0 \pmod{\phi(q)}$, and if $\phi(Q) \equiv 0 \pmod{z_3}$, we have

$$\begin{aligned} z_4 &\equiv \zeta \pmod{\frac{\phi(q)}{z_3}} \Rightarrow \phi(Q)z_4 \equiv \phi(Q)\zeta \pmod{\phi(q)} \\ z_4 &\equiv a^{\phi(\phi(G_1))(\rho_0-1)} \pmod{\phi(G_1)(\rho-1)} \end{aligned}$$

Hence, we have

$$Q(V_{1,1}^{\phi(Q)z_4}) \equiv Q(V_{1,1}^{\phi(Q)\zeta}) \equiv V_2 \pmod{qQ}.$$

As such, forgery is doable. That is, the forged signature is of the form

$$S_1^* \equiv (V_{1,1}^{\phi(Q)z_4})(2h^{\phi(qQ)}) \pmod{G_1qQ\rho}$$

That is,

$$QS_1^* \equiv 2Q(V_{1,1}^{\phi(Q)z_4}) \equiv 2V_2 \pmod{qQ}$$

And,

$$\begin{aligned} y_1 &\equiv R^{S_1^*} \\ &\equiv R^{(V_{1,1}^{\phi(Q)z_4})(2h^{\phi(qQ)})} \pmod{G_1} \\ &\equiv R^{(V_{1,1}^{\phi(Q)(1)})(2h^{\phi(qQ)})} \pmod{G_1} \\ y_2 &\equiv R^{(V_{1,1}^{\phi(Q)})(2h^{\phi(qQ)})} \pmod{G_1} \pmod{G_0} \end{aligned}$$

We have $y_1 = y_2$. As such, forgery can occur.

Also observe that,

$$\begin{aligned} y_3 &\equiv \frac{S_1^*}{V_1^{\phi(Q)}} \\ &\equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}} \\ y_4 &\equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}} \end{aligned}$$

We have $y_3 = y_4$. As such, forgery can occur.

On the other hand, when $\phi(Q) \not\equiv 0 \pmod{z_3}$, we have

$$\begin{aligned} z_4 &\equiv \zeta \pmod{\frac{\phi(q)}{z_3}} \Rightarrow \phi(Q)z_4 \not\equiv \phi(Q)\zeta \pmod{\phi(q)} \\ z_4 &\equiv a^{\phi(\phi(G_1)(\rho_0-1))} \pmod{\phi(G_1)(\rho-1)} \end{aligned}$$

Hence, we have

$$Q(V_1^{\phi(Q)z_4}) \not\equiv Q(V_1^{\phi(Q)\zeta}) \equiv V_2 \pmod{qQ}$$

Thus, to satisfy the filtering process of $QS_1 \equiv 2V_2 \pmod{qQ}$, we utilize z_1 (which was obtained by solving the DLP upon W_2) as follows,

$$Q(W_1^{z_1}) \equiv Q(V_1^{\phi(Q)b}) \equiv V_2 \pmod{qQ} \quad (5)$$

where z_1 satisfies equations 3 and 4.

In order to ensure (5) is executable, the signature is of the form

$$S_1^* \equiv (W_1^{z_1})(2h^{\phi(qQ)}) \pmod{G_1qQ\rho}$$

That is,

$$QS_1^* \equiv Q(W_1^{z_1}) \equiv 2V_2 \pmod{qQ}$$

And since $z_1 \equiv 1 \pmod{\phi(G_1)}$, we have

$$\begin{aligned} y_1 &\equiv R^{S_1^*} \\ &\equiv R^{(W_1^{z_1})(2h^{\phi(qQ)})} \pmod{G_1} \\ &\equiv R^{(V_1^{\phi(Q)b})(2h^{\phi(qQ)})} \pmod{G_1} \\ &\equiv R^{(V_1^{\phi(Q)(1)})(2h^{\phi(qQ)})} \pmod{G_1} \pmod{G_0} \\ y_2 &\equiv R^{(V_1^{\phi(Q)})(2h^{\phi(qQ)})} \pmod{G_1} \pmod{G_0} \end{aligned}$$

We have $y_1 = y_2$. As such forgery can occur.

Also observe that,

$$\begin{aligned} y_3 &\equiv \frac{S_1^*}{V_1^{\phi(Q)}} \\ &\equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}} \\ y_4 &\equiv 2h^{\phi(qQ)} \pmod{\frac{G_1\rho}{A}} \end{aligned}$$

We have $y_3 = y_4$. As such, forgery can occur.

However, the value $V_{1,1} \equiv \alpha \pmod{G_1 q \rho}$ is not available.

An attempt to obtain $V_{1,1}$ would be from $S_1 \equiv (\alpha^{(\phi(Q)b)}) (h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \pmod{G_1 q Q \rho}$.

Observe,

$$S_1 \equiv (\alpha^{(\phi(Q)b)}) (h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \not\equiv \alpha \pmod{G_1 q \rho}$$

Also observe,

$$S_1 \equiv 2\alpha^{(\phi(Q)b)} \not\equiv \alpha \pmod{q}$$

Hence, to obtain $V_{1,1} \equiv \alpha \pmod{G_1 q \rho}$ by conducting the CRT upon $S_1 \pmod{q}$ with $V_1 \equiv \alpha \pmod{G_1 \rho}$ is not feasible.

9. DISCUSSION ON EXCLUSION OF ITERATIVE CRT PROCEDURE

In this section we discuss the exclusion of the iterative CRT procedure that is visible in KAZ-SIGN versions 1.6.0 and 1.6.1.

Lemma 2. *Consider CRT equation*

$$x \equiv a_0 \pmod{m_0}$$

$$x \equiv a_1 \pmod{m_1}$$

where $\gcd(m_0, m_1) \neq 1$. Then this CRT has solution if and only if $\gcd(m_0, m_1) \mid (a_0 - a_1)$

Proof. Omitted. □

Consider

$$G_1 = \left(\prod_{i=1}^l \gamma_i^{a_i} \right) \left(\prod_{i=1}^n \alpha_i^{a_{i+l}} \right)$$

$$Q = \left(\prod_{i=1}^l \gamma_i^{b_i} \right) \left(\prod_{i=1}^m \beta_i^{b_{i+l}} \right)$$

Then we have

$$G_1 q Q = q \left(\prod_{i=1}^l \gamma_i^{a_i + b_i} \right) \left(\prod_{i=1}^n \alpha_i^{a_{i+l}} \right) \left(\prod_{i=1}^m \beta_i^{b_{i+l}} \right)$$

Once attacker can forge signature by solving the following CRT equations.

$$S' \equiv \begin{cases} 1 & \pmod{\gcd(Q, r_i^{e_i})} \\ V_Q & \pmod{\gcd(G_1, r_i^{e_i})} \\ V_2 & \pmod{\gcd(q \cdot Q, r_i^{e_i})} \end{cases}$$

where $r_i^{e_i} = q, \gamma_i^{a_i+b_i}, \alpha^{a_i+l}, \beta^{a_i+l}$ and $V_Q = (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1}$

We consider the case that $r_i^{e_i} = \gamma_i^{a_i+b_i}$, first. Then the forgery will succeed if the following CRT equations are solved.

$$\text{soln} \equiv 1 \pmod{\gcd(Q, \gamma_i^{a_i+b_i})} \quad (6)$$

$$\text{soln} \equiv V_Q \pmod{\gcd(G_1, \gamma_i^{a_i+b_i})} \quad (7)$$

$$\text{soln} \cdot Q \equiv V_2 \pmod{\gcd(qQ, \gamma_i^{a_i+b_i})} \quad (8)$$

This implies that

$$\text{soln} \equiv 1 \pmod{\gamma_i^{b_i}} \quad (9)$$

$$\text{soln} \equiv V_Q \pmod{\gamma_i^{a_i}} \quad (10)$$

$$\text{soln} \cdot Q \equiv V_2 \pmod{\gamma_i^{b_i}} \quad (11)$$

Note that equation (6) always holds, because $\gamma_i^{b_i} \mid Q$ and $Q \mid V_2$. Therefore, it only needs to consider that

$$\text{soln} \equiv 1 \pmod{\gamma_i^{b_i}} \quad (12)$$

$$\text{soln} \equiv V_Q \pmod{\gamma_i^{a_i}} \quad (13)$$

By Lemma 2, we are trying to do some tricks such that

$$V_Q \not\equiv 1 \pmod{\gcd(\gamma_i^{a_i}, \gamma_i^{b_i})} \quad (14)$$

This will cause the CRT forgery cannot be conducted. We do this by choosing specific α in key generation. Suppose 2 is one of the common factors of Q and G_1 . We choose $\alpha = 2\omega$, where ω is a random number. Then

$$\begin{aligned} V_Q &\equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1} \pmod{\gcd(2^{a_i}, 2^{b_i})} \\ &\equiv (\alpha^{\phi(Q)})(h^{\phi(qQ)}) \pmod{\gcd(2^{a_i}, 2^{b_i})} \\ &\equiv ((2\omega)^{\phi(Q)})(h^{\phi(qQ)}) \pmod{\gcd(2^{a_i}, 2^{b_i})} \\ &\equiv 0 \pmod{\gcd(2^{a_i}, 2^{b_i})} \text{ (if } \min\{a_i, b_i\} \leq \phi(Q)) \end{aligned}$$

By properly choosing parameters, we can make sure $\min\{a_i, b_i\} \leq \phi(Q)$. Hence, this will make equation (14) hold.

10. UNDERSTANDING WHY $SK \not\equiv 0 \pmod{G_1 Q \rho}$ IS NECESSARY

Observe,

$$\begin{aligned} h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2} &\equiv 2 \pmod{q} \\ Q\rho(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) &\equiv 2Q\rho \pmod{qQ\rho} \\ G_1 Q\rho(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) &\equiv 2G_1 Q\rho \pmod{G_1 qQ\rho} \end{aligned}$$

Now, suppose $SK \equiv 0 \pmod{G_1 Q \rho}$ (i.e. $\frac{SK}{G_1 Q \rho} = \kappa \in \mathbb{Z}$)

$$\kappa G_1 Q\rho(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2\kappa G_1 Q\rho \pmod{\kappa G_1 qQ\rho}$$

Which implies,

$$S_1 \equiv SK(h^{\phi(qQ)\beta_1} + h^{\phi(qQ)\beta_2}) \equiv 2SK \pmod{G_1 qQ\rho}$$

11. DERIVING THE SECURITY LEVEL OF KAZ-SIGN

The challenge faced by the adversary is to retrieve α from $V_1 \equiv \alpha \pmod{G_1 \rho}$. It is protected by the MRP. The MRP representation is given as follows:

$$t = \frac{\alpha - V_1}{G_1 \rho}$$

Due to the strategies during key generation, we have the complexity $O(t) = O(2^k)$ where k is the chosen security level, either 128, 192 or 256.

As such, the complexity of solving the MRP via $V_1 \equiv \alpha \pmod{G_1 \rho}$ will be the determining factor in identifying the suitable key length for each security level.

Another alternative challenge faced by the adversary is to produce the secret signing key, SK . The difficulty is much larger than the difficulty level of solving the MRP. Due to the strategies during key generation, we have the complexity $O(SK) = O(G_1 qQ\rho) \gg O(t)$.

12. IMPLEMENTATION AND PERFORMANCE

12.1 Key Generation, Signing and Verification Time Complexity

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

12.2 Parameter sizes

In the direction of the research, we also make comparison to ECC key length for the three NIST security levels. We provide here information on size of the key and signature based on NIST security level.

NIST Security Level	Security level, k	Public key size, (V_1, V_2) (bits)	Private Signing key Size, SK	Signature Size, (S_1, S_2) (bits)	ECC key size (bits)
1	128	386	386	$386 + \eta$	256
3	192	578	578	$578 + \eta$	384
5	256	770	770	$770 + \eta$	521

Table 1

where η is negligible size of S_2 . A practical implementation value of η is 8 bytes.

12.3 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.

12.4 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 - KAZS256	8	21	9
192 - KAZS384	13	54	13
256 - KAZS512	33	100	22

Table 2

13. ADVANTAGES AND LIMITATIONS

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

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

13.1 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 564-bits. ECC would use 521-bit keys and RSA would use 15,360-bit keys.

13.2 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.

13.3 No Verification Failure

It is apparent that the execution of **KAZ-SIGN digital signature forgery detection procedures type – 1, type – 2, type – 3, type – 4, and type – 5** 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.

13.4 Limitation

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

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

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

The MRP is not a well explored 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.

14. 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. Next, 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 β .2 of KAZ-SIGN.

15. ILLUSTRATIVE FULL SIZE TEST VECTORS – 1

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a valid KAZ-SIGN signature $S = (S_1, S_2)$. The valid KAZ-SIGN signature will pass all 6 KAZ-SIGN digital signature forgery detection procedure types.

G_0 :

2310215128354247255535103303185740711054948921498445110378630455815067460611708
8000

R :

6151

G_1 :

399620650696124709852000

A :

324324000

q :

246208917987764371328101733

Q :

1115881660253397921934830780

ρ_0 :

291449771828691044491032061778151323489

ρ :

582899543657382088982064123556302646979

Key generation

α :

245860455403105736328864953886518636840306133926479369884264760451054022807
418877169739525086485584158 $\approx 2^{337}$

V_1 :

104475401134535854444692508164333952495896689277511154871820158

a :

2649922049

ω_1 :

2818055362

b :

95284489401884030339116838849891272873325348041840341027073189478400001

V_2 :

158650205776564614614775089219867068769915329992733320

SK :

3129909774553300436270580991165077263098190233036453844772432142485286979565
4571407023020239066597648402573138469376

MRP complexity upon t

$$t = \frac{\alpha - V_1}{G_1 \rho} :$$

1055472795021570370086684868302383399933 $\approx 2^{130}$

Signing

$h :$

340282366920938463463374607431768211456

$r_1 :$

3980649377

$\omega_2 :$

3890247524

$\beta_1 :$

118825564102322620141878274292012672230439748618410038843558456197120001

$r_2 :$

2967761627

$\omega_3 :$

4030685362

$\beta_2 :$

50048064471881845957278446228044628993952582979095986149275237089280001

$S_1 :$

51201551095750608630188424692183141007892567105560535563658753969909561955468
283258833313702039754714490864858698752

$S_2 :$

3

$\ell(S_1) = 385$

$\ell(G_1qQ\rho) = 385$

Verification

KAZ-SIGN digital signature forgery detection procedure type – 1

w_0 :

0

KAZ-SIGN digital signature forgery detection procedure type – 2

w_1 :

0

KAZ-SIGN digital signature forgery detection procedure type – 3

S_{F1} :

292867450871490208807528317638814371432025578383042597062281981178022997424453
60774780730620918938489816977342378752

w_2 :

2191480600860158774943559292830170386469000926725627585743055585210726221302292
2484052583081120816224673887516320000 $\neq 0$

KAZ-SIGN digital signature forgery detection procedure type – 4

S_{F2} :

9427222537319803962145226410712638505368024753364389564420904473688915892039175
01924427282009187258752

w_3 :

– 23746370115655390356651029591331079069646177673665554496463941616099137021463
4796296100574740760900000 $\neq 0$

KAZ-SIGN digital signature forgery detection procedure type – 5

w_4 :

158650205776564614614775089219867068769915329992733320

w_5 :

0

Final verification

y_1 and y_2 :

1950515261305462389431673403111858145543588512927979533108520798400199169865
2620801

y_3 and y_4 :

568020163865290175115023241625324091490907380828160587

16. ILLUSTRATIVE FULL SIZE TEST VECTORS – 2

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1qQ\rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv S_V + G_1qQ\rho$ where S_V is a valid signature of S_1 . This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 1.**

S_V :

51201551095750608630188424692183141007892567105560535563658753969909561955468
283258833313702039754714490864858698752

S :

11519913190735569853165031128039480528543431088673904269156597645050895323491
2024416801721798233806929466303346618752

KAZ-SIGN digital signature forgery detection procedure type – 1

w_0 :

– 63997580811605089901461886588211664277541743781178507127907222480599391279443
741157968408096194052214975438487920000 $\neq 0$

Final verification

y_1 and y_2 :

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and y_4 :

568020163865290175115023241625324091490907380828160587

17. ILLUSTRATIVE FULL SIZE TEST VECTORS – 3

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1qQ\rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv (S_V \pmod{\frac{G_1qQ\rho}{e}}) + G_1qQ\rho$ where S_V is a valid signature of S_1 and where $e = \gcd(Q, G_1)$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 1.**

$e :$

63633859549260

$S_V \pmod{\frac{G_1qQ\rho}{e}} :$

7052585525754264926480123451579530598403406985997834114774510312079002189
89282705628326707268426358752

$S :$

63997580811605795160014462014704312289886901734238347468605822264010868730
474949058187397378899680541682706914278752

KAZ-SIGN digital signature forgery detection procedure type – 1

$w_0 :$

– 6399758081160508990146188658821166427754174378117850712790722248059939127
9443741157968408096194052214975438487920000 $\neq 0$

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

18. ILLUSTRATIVE FULL SIZE TEST VECTORS – 4

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1qQ\rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv S_V \pmod{\frac{G_1qQ\rho}{3}}$ where S_V is a valid signature of S_1 . This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 2.**

$$S_V \pmod{\frac{G_1qQ\rho}{3}} :$$

85364972213472153625471669667086981561980712514415308117206056495099677691
72455820187708304577053237840572533418752

$S :$

85364972213472153625471669667086981561980712514415308117206056495099677691
72455820187708304577053237840572533418752

KAZ-SIGN digital signature forgery detection procedure type – 2

$w_1 :$

$$-3 \neq 0$$

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

19. ILLUSTRATIVE FULL SIZE TEST VECTORS – 5

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1 Q \rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and conduct the CRT upon the equation pair $Y_1 = V_2 Q^{-1} \pmod{q}$ and $Y_2 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1 Q \rho}$ to obtain a forge signature, S . This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 3.**

$Y_1 :$

142174758693083840279588094

$Y_2 :$

1469514990928433888394011671482770545570061821745210295673268381332851894
9610789831498752

$S :$

2928674508714902088075283176388143714320255783830425970622819811780229974
2445360774780730620918938489816977342378752

KAZ-SIGN digital signature forgery detection procedure type – 3

$S_{F1} :$

2928674508714902088075283176388143714320255783830425970622819811780229974
2445360774780730620918938489816977342378752

$w_2 :$

0

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

20. ILLUSTRATIVE FULL SIZE TEST VECTORS – 6

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1 q Q \rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and conduct the CRT upon the equation pair $Y_1 = V_2 Q^{-1} \pmod{\frac{qQ}{e}}$ and $Y_2 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1 \rho}$ where $e = \gcd(Q, G_1)$ to obtain a forge signature, S . This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 4.**

$e :$

63633859549260

$Y_1 :$

142174758693083840279588094

$Y_2 :$

16577725536200450015218524322221329385120222831177464613430752

$S :$

94272225373198039621452264107126385053680247533643895644209044736889

1589203917501924427282009187258752

KAZ-SIGN digital signature forgery detection procedure type – 4

$S_{F2} :$

94272225373198039621452264107126385053680247533643895644209044736889

1589203917501924427282009187258752

$w_3 :$

0

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

21. ILLUSTRATIVE FULL SIZE TEST VECTORS – 7

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1qQ\rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S \equiv (V_1^{(\phi(Q))})(2h(\phi(qQ))) \pmod{G_1qQ\rho}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5.**

S :

55175945118465133398708318499171415329524266820736844173681552640768818040
176854893351443567841865617459358510298752

KAZ-SIGN digital signature forgery detection procedure type – 5

w_4 :

163417673433191886738792472854218101709635179416430240

w_5 :

4767467656627272124017383634351032939719849423696920 $\neq 0$

Final verification

y_1 and y_2 :

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and y_4 :

568020163865290175115023241625324091490907380828160587

22. ILLUSTRATIVE FULL SIZE TEST VECTORS – 8

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1 q Q \rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and conduct the CRT upon the equation pair $Y_1 \equiv 1 \pmod{\frac{qQ}{e}}$ where $e = \gcd(Q, G_1)$ and $Y_2 \equiv (V_1^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1 \rho}$ to obtain a forge signature, S . This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5.**

$e :$

63633859549260

$Y_1 :$

1

$Y_2 :$

16577725536200450015218524322221329385120222831177464613430752

$S :$

281801631769949597859083174531630410158231056013982668253851299644265636
586444344045774288539164574752

KAZ-SIGN digital signature forgery detection procedure type – 5

$w_4 :$

137370008086689597273190618781334612988476996957286260

$w_5 :$

$-21280197689875017341584470438532455781438333035447060 \neq 0$

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

23. ILLUSTRATIVE FULL SIZE TEST VECTORS – 9

The following are parameters that illustrate KAZ-SIGN for 128-bit security. In this illustration, we provide a forged KAZ-SIGN signature S where the parameters, $(S_1, S_2, k, q, Q, \rho, R, G_0, G_1, L_{G_1 q Q \rho})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and construct a forged signature S with a forged α of the form $A_0 = V_1 + G_1 \rho T$ for some $T \in \mathbb{Z}$ and forged α is a prime. The constructed forged signature is of the form $S \equiv (A_0^{\phi(Q)})(2h^{\phi(qQ)}) \pmod{G_1 q Q \rho}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5.**

$T :$

210051794687383489479899642695360182531

$A_0 :$

48929190921519062072513102642288006464338989323226127356604376652168870609
475393418516343796532168158

$S :$

227377408611454196238439763336437901381377037343639792175887242232819623454
32844090145753272298145883620681875258752

KAZ-SIGN digital signature forgery detection procedure type – 5

$w_4 :$

45109344636834428501847976519543455299648972046815640

$w_5 :$

$-113540861139730186112927112700323613470266357945917680 \neq 0$

Final verification

y_1 and $y_2 :$

19505152613054623894316734031118581455435885129279795331085207984001991698652620801

y_3 and $y_4 :$

568020163865290175115023241625324091490907380828160587

References

- Ajtai, M. (1998). The shortest vector problem in L_2 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.