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

Algorithm Specifications and Supporting Documentation

(Version 1.6.3)

Muhammad Rezal Kamel Ariffin[1] Nur Azman Abu[2] Kai Chieh Chang (Jay)[3]
Terry Lau Shue Chien[4] Zahari Mahad[1] Liaw Man Cheon[5]
Amir Hamzah Abd Ghafar[1] Nurul Amiera Sakinah Abdul Jamal[1] Vaishnavi Nagaraja[1]

[1]Institute for Mathematical Research, Universiti Putra Malaysia
[2]Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka
[3]Architecture Design Department, Phison Electronics Corporation, Taiwan
[4]Faculty of Computing & Informatics, Multimedia University Malaysia
[5]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 Kai Chieh Chang (Jay) Terry Lau Shue Chien Zahari Mahad Liaw Man Cheon Amir Hamzah Abd Ghafar Nurul Amiera Sakinah Abdul Jamal Vaishnavi Nagaraja
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, 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 $L_q = \ell(q) \approx L_Q$. Then, compute a random composite integer $G_0 = \left(2^{10}\right)\prod_{i=1}^{k_1} p_i^{e_i}$ 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 $L_{G_1} = \ell(G_1)$. Ensure that $L_{G_1qQ} = \ell(G_1qQ)$ is either 256, 384 or 512 (depending on the security level needed). Otherwise, tweak the generation of the parameters (G_0, q, Q) accordingly. The system parameters are $(k, q, Q, R, G_0, G_1, L_{G_1}, L_{G_1qQ})$.

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

As mentioned in preceeding section, determine 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,

- $N = 360741258397132321959000378383494428498852022631945374390696980091$ $989517655143007152881651302340013944918339154953408659224878103612931$7 $137092037483563399346623614557751044797268991006479248759233160158836$4 $511761215345407301312219847791817434306554847172231930077054686259201$9 $5527456360287632608176655$

- $g = 6007$ and $R = 6151$

- $G_0 = G_{gN} = 2310215128354247255535103303185740711054948921498445110378630455815067460611708800$0 $= (2^8)(3^4)(5^3)(7^2)(11^2)(13^2)(17)(19)(23)(29)(31)(37)(41)(43)(47)(53)(59)(61)(67)(71)(73)(79)(83)(89)(97)(101)(103)(107)(113)(127)(131)(139)(163)(173)(179)(191)(233)(239)(251)(281)(293)(359)$.

- $G_1 = G_{Rg} = 799241301392249419704000 = (2^6)(3^4)(5^3)(7)(11)(13)(17)(23)(29)(41)(43)(53)(73)(89)(179)$

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 $(k, q, Q, R, G_0, G_1, L_{G_1})$
Output: Public verification key pair, $(V_1, V_2)$ and private signing key, $SK$.
1: Choose even random $\alpha \approx 2^{k+L_{G_1}}$
2: Compute public verification key-1, $V_1 \equiv \alpha \pmod{G_1}$.
3: Choose random prime $a$ and random $\omega_1$ both $\approx 2^{32}$.
4: Compute secret parameter $b \equiv a^{\phi(\phi(G_1))} \pmod{\omega_1 \phi(G_1)}$.
5: Compute public verification key-2, $V_2 \equiv Q(\alpha^{\phi(Q)b}) \pmod{qQ}$.
6: Compute secret signing key, $SK \equiv \alpha^{\phi(Q)b} \pmod{G_1 qQ}$
7: Output public verification keys, $(V_1, V_2)$, keep signing key $SK$ secret and destroy parameters $(\alpha, a, b, \omega_1)$.
```

Algorithm 2 KAZ-SIGN Signing Algorithm

```
Input: System parameters $(k, q, Q, R, G_0, G_1, L_{G_1 qQ})$, private signing key, $SK$ and message to be signed, $m$.
Output: Signature, $S$
1: Let $m$ be the message to be signed and let $h = (H(m))$.
2: Choose random prime $r$, and random $\omega_2$ both $\approx 2^{32}$.
3: Compute secret parameter $\beta \equiv r^{\phi(\phi(G_1))} \pmod{\omega_2 \phi(G_1)}$.
4: Compute $S \equiv (SK)(h^{(\phi(qQ)\beta)}) \pmod{G_1 qQ}$.
5: if $\ell(S) \neq L_{G_1 qQ}$ then
6:     Repeat from Step 2
7: else Continue Step 9
8: end if
9: Output signature, $S$, and destroy $(\beta, r, \omega_2)$.
```

Algorithm 3 KAZ-SIGN Verification Algorithm

```
Input: System parameters $(k,q,Q,R,G_0,G_1,L_{G_1qQ})$, public verification key pair, $(V_1,V_2)$,
    message, $m$, and signature, $S$.
Output: Accept or reject signature
1: Compute $h = H(m)$.
2: Compute $Y_1 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1Q}$ and $S_{F1} = CRT([\frac{V_2}{Q},Y_1],[q,G_1Q])$.
3: Compute $Y_2 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1}$ and $S_{F2} = CRT([\frac{V_2}{Q},Y_2],[\frac{qQ}{e},G_1])$ where $e = gcd(Q,G_1)$.
4: Compute $w_0 \equiv (S \pmod{G_1qQ}) - S$.
5: if $w_0 \neq 0$ then
6:     Reject signature $\perp$
7: else Continue Step 9
8: end if
9: Compute $w_1 = \ell(S) - L_{G_1qQ}$.
10: if $w_1 \neq 0$ then
11:     Reject signature $\perp$
12: else Continue Step 14
13: end if
14: Compute $w_2 \equiv (S \pmod{G_1qQ}) - S_{F1}$.
15: if $w_2 = 0$ then
16:     Reject signature $\perp$
17: else Continue Step 19
18: end if
19: Compute $w_3 \equiv (S \pmod{\frac{G_1qQ}{e}}) - S_{F2}$ 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 $w_4 \equiv QS \pmod{qQ}$. 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 $y_1 \equiv R^S \pmod{G_0}$.
30: Compute $y_2 \equiv R^{((V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1})} \pmod{G_0}$.
31: if $y_1 = y_2$ then
32:     accept signature
33: else reject signature $\perp$
34: end if
```

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,

$$\begin{aligned} R^S &\equiv R^{(\alpha^{(\phi(Q)b)})(h^{(\phi(qQ)\beta)})\ (\mathrm{mod}\ G_1)} \\ &\equiv R^{(\alpha^{(\phi(Q))})(h^{(\phi(qQ))})\ (\mathrm{mod}\ G_1)} \\ &\equiv R^{(V_1^{\phi(Q)}(h^{\phi(qQ)})\ (\mathrm{mod}\ G_1)} \pmod{G_0} \end{aligned}$$

because $\alpha \equiv V_1 \pmod{G_1}$, $b \equiv 1 \pmod{\phi(G_1)}$ and $\beta \equiv 1 \pmod{\phi(G_1)}$. 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 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,

$$w_0 \equiv (S \pmod{G_1 qQ}) - S = 0$$

because $S < G_1 qQ$.

8.3 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,

$$w_1 = \ell(S) - L_{G_1 qQ} = 0$$

because of steps 5, 6, 7 and 8 during signing.

8.4 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 $w_2 \equiv (S \pmod{G_1 qQ}) - S_{F1}$, we will have $w_2 \neq 0$.

8.5 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 $w_3 \equiv (S \pmod{\frac{G_1 qQ}{e}}) - S_{F2}$, where $e = gcd(Q, G_1)$ we will have $w_3 \neq 0$.

8.6 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,

$$h^{\phi(qQ)\beta} \equiv 1 \pmod{q}$$

Thus,

$$Qh^{\phi(qQ)\beta} \equiv Q \pmod{qQ}$$

Finally,

$$w_4 \equiv QS \equiv Q(\alpha^{\phi(Q)b}) \pmod{qQ}$$

Hence, $w_5 = w_4 - V_2 = 0$.

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

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

$$S_F \equiv S + G_1 qQx \pmod{\theta G_1 qQ}$$

for some random value of $x \in \mathbb{Z}$ and small value of $\theta \in \mathbb{Z}$, such that $\ell(S_F) \approx \ell(S)$. 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. However, since

$$w_0 \equiv (S_F \pmod{G_1 qQ}) - S_{F0} \neq 0 \in \mathbb{Z}$$

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

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

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

$$S_F \equiv S \pmod{\frac{G_1 qQ}{u}}$$

where u is small composite of Q such that $\ell(S_F) \approx \ell(S)$. That is, $\ell(S_F)$ is not suspicious to the verifier. As an example $u = 15$. It is easy to observe that S_F will pass steps 29, 30, 31, 32, 33, and 34. However, this would result in $\ell(S_F) \neq L_{G_1 qQ}$ and the signature will fail KAZ-SIGN digital signature forgery detection procedure type – 2. That is, $w_1 \neq 0$.

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

An adversary that constructs a forged signature S as follows; compute $Y_1 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1 Q}$ and $S = CRT([\frac{V_2}{Q}, Y_1], [q, G_1 Q])$, and then transmits it as a signature S would result in

$$w_2 \equiv (S \pmod{G_1 qQ}) - S_{F1} = 0.$$

It is easy to observe that S will pass steps 29, 30, 31, 32, 33, and 34. However, the signature will fail KAZ-SIGN digital signature forgery detection procedure type - 3.

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

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

$$w_3 \equiv (S \pmod{\frac{G_1 qQ}{e}}) - S_{F2} = 0.$$

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

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

An adversary that constructs a forged signature S without the private random α and at the same time aspires to pass steps 29, 30, 31, 32, 33, and 34 would result in the need to produce a forged signature S that would eventually produce the relation,

$$S \equiv (\lambda^{\phi(Q)b'}) \pmod{qQ}$$

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

8.12 Extracting α

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

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

8.12.1 Producing $y_{\alpha 1} \equiv \alpha \pmod{G_1 qQ}$

From the public parameter $V_1 \equiv \alpha \pmod{G_1}$, the adversary needs to obtain the parameter $\alpha \pmod{qQ}$ to execute the Chinese Remainder Theorem (CRT) to obtain $\alpha \pmod{G_1 qQ}$. To obtain $\alpha \pmod{qQ}$, the adversary will utilize equation S. Observe that

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

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

8.12.2 Producing $y_{\alpha 2} \equiv \alpha^{\phi(Q)} \pmod{G_1 qQ}$

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

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

Then, one needs to produce the parameter $\alpha^{\phi(Q)} \pmod{qQ}$. However,

$$z_2 \equiv S \equiv (\alpha^{(\phi(Q)b)}) \not\equiv \alpha^{\phi(Q)} \pmod{qQ}.$$

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

8.13 Modular Linear Equation of S

In this direction we analyze

$$S \equiv (\alpha^{(\phi(Q)b)})(h^{(\phi(qQ)\beta)}) \pmod{G_1 qQ}$$

Let

1. $X_1 \equiv \alpha^{\phi(Q)b} \pmod{G_1 qQ}$
2. $X_2 \equiv h^{\phi(qQ)\beta} \pmod{G_1 qQ}$

Moving forward we have,

$$X_1 X_2 - S \equiv 0 \pmod{G_1 qQ} \tag{1}$$

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 $\hat{X}_1 \hat{X}_2 \gg G_1 qQ$, then there is no efficient algorithm to output all the roots of (1). That is, (1) usually has $G_1 qQ$ many solutions, which is exponential in the bit-size of $G_1 qQ$.
To this end, since both $\alpha^{\phi(Q)b}$ and $h^{\phi(qQ)\beta}$ are exponentially large, it is clear to conclude that $\hat{X}_1 \hat{X}_2 \gg G_1 qQ$. This implies, there is no efficient algorithm to output all the roots of (1).

8.14 Implementation of the Hidden Number Problem (HNP)

From S, let us denote as follows:

1. $x_1 \equiv \alpha^{(\phi(Q)b)} \pmod{G_1 qQ}$
2. $x_2 \equiv \phi(qQ)\beta$

Thus, S can be re-written as

$$S \equiv (x_1)(h^{x_2}) \pmod{G_1 qQ} \tag{2}$$

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

8.15 Analysis on V_2

Assume we have $V_1 \equiv \alpha \pmod{q}$. Let,

$$W_1 \equiv V_1^{\phi(Q)} \equiv \alpha^{\phi(Q)} \pmod{q}$$

$$W_2 \equiv V_2 Q^{-1} \equiv \alpha^{\phi(Q)b} \equiv V_1^{\phi(Q)b} \pmod{q}$$

The aim is to obtain the system of equations

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

$$b \equiv a^{\phi(\phi(G_1))} \pmod{\phi(G_1)} \tag{4}$$

for some arbitarily chosen prime a.

We obtain ζ by solving the DLP upon $W_2 \equiv \alpha^{\phi(Q)b} \equiv V_1^{\phi(Q)b} \pmod{q}$, where the DL solver works on the base given by $W_1 \equiv V_1^{\phi(Q)} \pmod{q}$. That is, $z_1 \equiv b \equiv \zeta \pmod{\phi(q)}$ and $W_1^{z_1} \equiv V_1^{\phi(Q)b} \equiv W_2 \pmod{q}$. Then, for some prime a, let $z_2 \equiv a^{\phi(\phi(G_1))} \pmod{\phi(G_1)}$.

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

$$z_4 \equiv \zeta \pmod{\frac{\phi(q)}{z_3}} \Rightarrow \phi(Q)z_4 \equiv \phi(Q)\zeta \pmod{\phi(q)}$$
$$z_4 \equiv a^{\phi(\phi(G_1))} \pmod{\phi(G_1)}$$

Hence, we have
$$Q(V_1^{\phi(Q)z_4}) \equiv Q(V_1^{\phi(Q)\zeta}) \equiv V_2 \pmod{qQ}.$$

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

$$S^* \equiv (V_1^{\phi(Q)z_4})(h^{\phi(qQ)}) \pmod{G_1 qQ}$$

That is,
$$QS^* \equiv Q(V_1^{\phi(Q)z_4}) \equiv V_2 \pmod{qQ}$$

And,

$$\begin{aligned} y_1 &\equiv R^{S^*} \\ &\equiv R^{(V_1^{\phi(Q)z_4})(h^{\phi(qQ)}) \pmod{G_1}} \\ &\equiv R^{(V_1^{\phi(Q)(1)})(h^{\phi(qQ)}) \pmod{G_1}} \\ y_2 &\equiv R^{(V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1}} \pmod{G_0} \end{aligned}$$

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

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

$$z_4 \equiv \zeta \pmod{\frac{\phi(q)}{z_3}} \Rightarrow \phi(Q)z_4 \not\equiv \phi(Q)\zeta \pmod{\phi(q)}$$
$$z_4 \equiv a^{\phi(\phi(G_1))} \pmod{\phi(G_1)}$$

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 $QS \equiv V_2 \pmod{qQ}$,

we utilize

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

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

$$S^* \equiv (W_1^{z_1})(h^{\phi(qQ)}) \pmod{G_1 qQ}$$

That is,

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

And under the assumption that $\phi(G_1) < \phi(q)$ which will imply $z_1 \equiv 1 \pmod{\phi(G_1)}$, we have

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

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

However, the value $V_1 \equiv \alpha \pmod{q}$ is not available.

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*

$$\begin{aligned} x &\equiv a_0 \pmod{m_0} \\ x &\equiv a_1 \pmod{m_1} \end{aligned}$$

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

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

Then we have

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

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

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

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

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.

$$soln \equiv 1 \pmod{gcd(Q, \gamma_i^{a_i+b_i})} \tag{6}$$
$$soln \equiv VQ \pmod{gcd(G_1, \gamma_i^{a_i+b_i})} \tag{7}$$
$$soln \cdot Q \equiv V_2 \pmod{gcd(qQ, \gamma_i^{a_i+b_i})} \tag{8}$$

This implies that

$$soln \equiv 1 \pmod{\gamma_i^{b_i}} \tag{9}$$
$$soln \equiv VQ \pmod{\gamma_i^{a_i}} \tag{10}$$
$$soln \cdot Q \equiv V_2 \pmod{\gamma_i^{b_i}} \tag{11}$$

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

$$soln \equiv 1 \pmod{\gamma_i^{b_i}} \tag{12}$$
$$soln \equiv VQ \pmod{\gamma_i^{a_i}} \tag{13}$$

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

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

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

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

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

10. DERIVING THE SECURITY LEVEL OF KAZ-SIGN

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

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

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 $V_1 \equiv \alpha \pmod{G_1}$ 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 twice the difficulty level of solving the MRP. Due to the strategies during key generation, we have the complexity $O(SK) = O(2t) = O(2^{2k})$.

11. IMPLEMENTATION AND PERFORMANCE

11.1 Key Generation, Signing and Verification Time Complexity

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

11.2 Parameter sizes

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) (bits)	ECC key size (bits)
1	128	256	256	256	256
3	192	384	384	384	384
5	256	512	512	512	521

Table 1

In the direction of the research, we also make comparison to ECC key length for the three NIST security levels.

11.3 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.

11.4 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	5	8	7
192 - KAZS384	7	10	9
256 - KAZS512	9	16	12

Table 2

12. ADVANTAGES AND LIMITATIONS

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

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

12.1 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 512-bits. ECC would use 521-bit keys and RSA would use 15360-bit keys.

12.2 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.

12.3 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, 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.

12.4 Limitation

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

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

12.4.1 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.

13. 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 $1.5\beta.2$ of KAZ-SIGN. Today, our participation in this NIST exercise has lead us towards new collaborations.

14. 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. The valid KAZ-SIGN signature will pass all 6 KAZ-SIGN digital signature forgery detection procedure types.

G_0 :
23102151283542472555351033031857407110549489214984451103786304558150674606117088
000

R :
6151

G_1 :
399620650696124709852000

q :
222275055113138355774934753

Q :
2231763320506795843869661560

Key generation

α :
306060707684030040439965825102368816679483393875309167008223246 $\approx 2^{208}$

V_1 :
473163401356546388989636

a :
2421359807

ω_1 :
2968765783

b :
21237795648307995292705436467200 1

V_2 :
1301249128110362973489806772092968589542445565353091 60

SK :
1104913111358661086874375430674524984465312940213968524683455825761071 8373376

MRP complexity upon t

$t = \dfrac{\alpha - V_1}{G_1}$:
765878107527434743563505682475015740379 $\approx 2^{129}$

Signing

h :
92323132445097779935154047106237029502263302334133089688219267996968970913530

r :
3763831051

ω_2 :
4019735110

β :
199404239488927226776340398080001

S :
247669878370593927496847629584161012924631982319547297224935397020268383648000

$\ell(S) = 258$

$\ell(G_1 qQ) = 258$

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} :
18958425784218289585746756961445360103102225309927921163138032559656869440000

w_2 :
228711452586375637911100872622715652821529757009619376061797364460611514208000 $\neq 0$

KAZ-SIGN digital signature forgery detection procedure type – 4

S_{F2} :
64756205993316169108668349448889530879943254143507889624456800

w_3 :
247669878370593862740641636267991904256282533430016417281681253512378759191200 $\neq 0$

KAZ-SIGN digital signature forgery detection procedure type – 5

w_4 :
130124912811036297348980677209296858954244556535309160

w_5 :
0

Final verification

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

15. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1 qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv S_V + G_1 qQ$ where S_V is a valid signature. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 1.**

S_V :
247669878370593927496847629584161012924631982319547297224935397020268383648000

S :
501825190526070466473228185980776996258836406614641571256342952676092936192000

KAZ-SIGN digital signature forgery detection procedure type – 1

w_0 :
$-254155312155476538976380556396615983334204424295094274031407555655824552544000 \neq 0$

Final verification

y_1 and y_2 :
509754818097745677567152344977029985547646377390011116600 75996

16. 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 system parameters, $(k,q,Q,R,G_0, G_1,L_{G_1qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv (S_V \pmod{\frac{G_1qQ}{e}}) + G_1qQ$ where S_V is a valid signature and where $e = \gcd(Q, G_1)$. This signature will fail the <u>**KAZ-SIGN digital signature forgery detection procedure type – 1**</u>.

e :
162376055401560

$S_V \pmod{\dfrac{G_{Rg}qQ}{e}}$:
291394102068115694247355492284767881251583623571138151771015200

S :
254155312155476830370482624512310230689696709062975525615031126793976323559200

<u>KAZ-SIGN digital signature forgery detection procedure type – 1</u>

w_0 :
$-254155312155476538976380556396615983334204424295094274031407555655824552544000 \neq 0$

<u>Final verification</u>

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

17. 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 system parameters, $(k,q,Q,R,G_0, G_1,L_{G_1qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and the forged signature is of the form of $S \equiv S_V \pmod{\frac{G_1qQ}{15}}$ where S_V is a valid signature. This signature will fail the **<u>KAZ-SIGN digital signature forgery detection procedure type – 2.</u>**

$S_V \pmod{\dfrac{G_{Rg}qQ}{15}}$:
10458253692149157785559110280652761812707852977459308128955011741498801273600

S :
10458253692149157785559110280652761812707852977459308128955011741498801273600

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

w_1 :
$-5 \neq 0$

<u>Final verification</u>

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

18. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1 qQ})$ 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 $Y_2 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1 Q}$ to obtain a forge signature. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 3.**

Y_1 :
58305874827930733404562211

Y_2 :
941094002833589598375885743281580526759107008672000

S :
18958425784218289585746756961445360103102225309927921163138032559656869440000

KAZ-SIGN digital signature forgery detection procedure type – 3

S_{F1} :
18958425784218289585746756961445360103102225309927921163138032559656869440000

w_2 :
0

Final verification

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

19. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and conduct the CRT upon the equation pair $Y_1 = V_2Q^{-1} \pmod{\frac{qQ}{e}}$ and $Y_2 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1}$ where $e = \gcd(Q, G_1)$ to obtain a forge signature. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 4**.

e :
162376055401560

Y_1 :
58305874827930733404562211

Y_2 :
39482774808438509747200

S :
64756205993316169108668349448889530879943254143507889624456800

KAZ-SIGN digital signature forgery detection procedure type – 4

S_{F2} :
64756205993316169108668349448889530879943254143507889624456800

w_3 :
0

Final verification

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

20. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S \equiv (V_1^{(\phi(Q))})(h^{(\phi(qQ))}) \pmod{G_1qQ}$. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5**.

S :
141663962941782831195955616476396326040506968566500248978873588438083171744000

KAZ-SIGN digital signature forgery detection procedure type – 5

w_4 :
425112612678717952770924826090260525947896896124209360

w_5 :
294987699867681655421944148880963666993652339588900200 $\neq 0$

Final verification

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

21. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1 qQ})$ 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 $Y_2 \equiv (V_1^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1}$ to obtain a forge signature. This signature will fail the **KAZ-SIGN digital signature forgery detection procedure type – 5**.

e :
162376055401560

Y_1 :
1

Y_2 :
39482774808438509747200

S :
299113849804946159431756293463810371961989709015980582118741600

KAZ-SIGN digital signature forgery detection procedure type – 5

w_4 :
2231763320506795843869661560

w_5 :
$-130124912811036297348980674977533538447448712665647600 \neq 0$

Final verification

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

22. 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 system parameters, $(k, q, Q, R, G_0, G_1, L_{G_1qQ})$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and construct a forged signature with a forged α of the form $A = V_1 + G_1 T$ for some $T \in \mathbb{Z}$ and forged α is a prime. The constructed forged signature is of the form $S \equiv (A^{\phi(Q)})(h^{\phi(qQ)}) \pmod{G_1 qQ}$. This signature will fail the **<u>KAZ-SIGN digital signature forgery detection procedure type – 5</u>**.

T :
266853658273452650560291876861467412044

A :
30389546326497976114495169776157963701677203552926550317450839618995774537
259956497415236

S :
64886012340720975458409046966274712643019556750020636160782523317333205184000

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

w_4 :
425112612678717952770924826090260525947896896124209360

w_5 :
29498769986768165542194414888096366693652339588900200 $\neq 0$

<u>Final verification</u>

y_1 and y_2 :
50975481809774567756715234497702998554764637739001111660075996

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.