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

Algorithm Specifications and Supporting Documentation

(Version 1.3)

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

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

Table of Contents

Name of the proposed cryptosystem: KAZ-SIGN

Principal submitter: Muhammad Rezal Kamel Ariffin
Institute for Mathematical Research
Universiti Putra Malaysia
43400 UPM Serdang
Malaysia
Email: rezal@upm.edu.my
Phone: +60123766494

Auxilliary submitters: Nor Azman Abu
Terry Lau Shue Chien
Zahari Mahad
Liaw Man Cheon
Amir Hamzah Abd Ghafar
Nurul Amiera Sakinah Abdul Jamal

Inventor of the cryptosystem: Muhammad Rezal Kamel Ariffin

Owner of the cryptosystem: Muhammad Rezal Kamel Ariffin

Alternative point of contact: Amir Hamzah Abd Ghafar
Institute for Mathematical Research
Universiti Putra Malaysia
43400 UPM Serdang
Malaysia
Email: amir_hamzah@upm.edu.my
Phone: +60132723347

1. INTRODUCTION

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

2. THE DESIGN IDEALISME

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

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

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

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

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

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

(vii) To have a sufficiently large signature space;

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

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

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

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

3. MODULAR REDUCTION PROBLEM (MRP)

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

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

4. COMPLEXITY OF SOLVING THE MRP

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

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

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

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

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

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

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

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

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

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

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

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

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

7. THE KAZ-SIGN DIGITAL SIGNATURE ALGORITHM

7.1 Background

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

7.2 Utilized Functions

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

7.3 System Parameters

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

That is, choose R with a large order in $\mathbb{Z}_{\phi(N)}$. Let $n_{G_R} = \ell(G_R)$ be the bit length of G_R. Such R, has its own natural order in $Z_{\phi(G_g)}$. Let that order be denoted as G_{Rg}. We can observe the natural relation given by $R^{G_{Rg}} \equiv 1 \pmod{G_g}$ where $\phi(N) \equiv 0 \pmod{G_g}$ and $\phi(G_g) \equiv 0 \pmod{G_{Rg}}$. Let $n_{\phi(G_g)} = \ell(\phi(G_g))$ be the bit length of $\phi(G_g)$ and $n_{\phi(G_{Rg})} = \ell(\phi(G_{Rg}))$ be the bit length of $\phi(G_{Rg})$. Let q be a random k-bit prime where $(q-1)2^{-1}$ is a prime. The system parameters are $(g,k,q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$.

7.4 KAZ-SIGN Algorithms

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

Algorithm 1 KAZ-SIGN Key Generation Algorithm

Input: System parameters $(g,k,q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$.
Output: Public verification key pair, $V = (V_1,V_2)$, and private signing key, α

1: Choose random $\alpha \in (2^{n_{\phi(G_g)}-2}, 2^{n_{\phi(G_g)}-1})$.
2: Compute public verification key, $V_1 \equiv \alpha \pmod{G_{Rg}q}$.
3: Compute public verification key, $V_2 = H(\alpha^4 \pmod{q^2})$.
4: Output public verification key pair, $V = (V_1,V_2)$ and private signing key α.

Algorithm 2 KAZ-SIGN Signing Algorithm

Input: System parameters $(g,k,q,N,R,G_g,G_{Rg},n,n_{\phi(G_g)})$, private signing key, α, and message to be signed, $m \in \mathbb{Z}_N$
Output: Signature pair, $S = (S_1,S_2)$.

1: Compute the hash value of the message, $h = H(m)$.
2: Choose random $r \in (2^{n_{\phi(G_g)}-2}, 2^{n_{\phi(G_g)}-1})$.
3: Compute $S_1 \equiv r \pmod{G_{Rg}q^2}$.
4: Compute $GS_1 = \gcd(S_1, G_{Rg})$.
5: Compute $GS_{12} = \gcd(r, \phi(G_{Rg}q^2(G_{S1}^{-1})))$.
6: **if** $GS_{12} < 100$ **then**
7:     Repeat from Step 2.
8: **end if**
9: Compute $S_2 \equiv GS_1(\alpha^{S_1} + h)r^{-1} \pmod{G_{Rg}q^2}$.
10: **if** S_2 does not exist **then**
11:     Repeat from Step 2.
12: **end if**
13: Output signature pair, $S = (S_1,S_2)$, and destroy r.

Algorithm 3 KAZ-SIGN Verification Algorithm

Input: System parameters $(g, k, q, N, R, G_g, G_{Rg}, n, n_{\phi(G_g)})$, public verification key pair, $V = (V_1, V_2)$, message, m, and, signature pair, $S = (S_1, S_2)$.

Output: Accept or reject

1: Compute the hash value of the message to be verified, $h = H(m)$.
2: Compute $GS_{1r} = \gcd(S_1, G_{Rg})$.
3: Compute $\alpha_F \equiv V_1 \pmod{G_{Rg}}$.
4: Compute $w_0 \equiv GS_{1r}(V_1^{S_1} + h)S_1^{-1} \pmod{G_{Rg}q^2}$.
5: Compute $w_1 = w_0 - S_2$.
6: **if** $w_1 = 0$ **then**
7: Reject signature $\perp$
8: **else** Continue step 10
9: **end if**
10: Compute $w_2 \equiv GS_{1r}(\alpha_F^{S_1} + h)S_1^{-1} \pmod{G_{Rg}q^2}$.
11: Compute $w_3 = w_2 - S_2$.
12: **if** $w_3 = 0$ **then**
13: Reject signature $\perp$
14: **else** Continue step 16
15: **end if**
16: Compute $w_4 \equiv S_1S_2 - GS_{1r}h \pmod{q}$
17: Compute $w_5 \equiv GS_{1r}V_1^{S_1} \pmod{q}$
18: Compute $w_6 = w_4 - w_5$
19: **if** $w_6 \neq 0$ **then**
20: Reject signature $\perp$
21: **else** Continue step 23
22: **end if**
23: Compute $w_7 = 2S_1^{-1} \pmod{\left(\frac{\phi(q^2)}{2}\right)}$.
24: Compute $w_{80} \equiv \left((S_1S_2 - GS_{1r}h)(GS_{1r})^{-1}\right)^{2w_7} \pmod{q^2}$ and $w_8 = H(w_{80})$.
25: Compute $w_9 = w_8 - V_2$.
26: **if** $w_9 \neq 0$ **then**
27: Reject signature $\perp$
28: **else** Continue step 30
29: **end if**
30: Compute $z_0 \equiv R^{S_1S_2} \pmod{Gg}$.
31: Compute $y_1 \equiv g^{z_0} \pmod{N}$.
32: Compute $z_1 \equiv R^{GS_{1r}(V_1^{S_1}+h) \pmod{G_{Rg}}} \pmod{G_g}$.
33: Compute $y_2 \equiv g^{z_1} \pmod{N}$.
34: **if** $y_1 = y_2$ **then**
35: accept signature
36: **else** reject signature $\perp$
37: **end if**

Steps 4, 5, 6, 7, 8, and 9 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 1**, steps 10, 11, 12, 13, 14 and 15 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 2**, steps 16, 17, 18, 19, 20, 21, and 22 during verification are known as the **KAZ-SIGN digital signature forgery detection procedure type – 3**, and steps 23, 24, 25, 26, 27, 28, and 29 are known as the **KAZ-SIGN digital signature forgery detection procedure type – 4**.

8. THE DESIGN RATIONALE

8.1 Proof of correctness (Verification steps 30, 31, 32, 33, 34, 35, 36 and 37)

We begin by discussing the rationale behind steps 30, 31, 32, 33, 34, 35, 36 and 37 with relation to the verification process. Observe the following,

$$g^{z_0} \equiv g^{R^{S_1 S_2}} \equiv g^{R^{rGS_{1r}(\alpha^{S_1}+h)(r)^{-1}}} \equiv g^{R^{GS_{1r}(V_1^{S_1}+h)}} \equiv g^{z_1} \pmod{N}.$$

because $\alpha \equiv V_1 \pmod{G_{Rg}}$. As such the verification process does indeed provide an indication that the signature is indeed from an authorized sender with the private signing key α.

8.2 Proof of correctness (Verification steps 4, 5, 6, 7, 8, and 9: KAZ-SIGN digital signature forgery detection procedure type – 1)

In order to comprehend the rationale behind steps 4, 5, 6, 7, 8, and 9, one has to observe the following,

$$w_0 \equiv GS_{1r}(V_1^{S_1}+h)S_1^{-1} \not\equiv GS_{1r}(\alpha^{S_1}+h)S_1^{-1} \pmod{G_{Rg}q^2}$$

because $\alpha \not\equiv V_1 \pmod{G_{Rg}q^2}$. Hence, $w_1 \neq 0$.

8.3 Proof of correctness (Verification steps 10, 11, 12, 13, 14 and 15: KAZ-SIGN digital signature forgery detection procedure type – 2)

In order to comprehend the rationale behind steps 10, 11, 12, 13, 14 and 15, one has to observe the following;

$$w_2 \equiv GS_{1r}(\alpha_F^{S_1}+h)S_1^{-1} \not\equiv GS_{1r}(\alpha^{S_1}+h)S_1^{-1} \pmod{G_{Rg}q^2}.$$

because $\alpha \not\equiv \alpha_F \pmod{G_{Rg}q^2}$ where $\alpha_F \equiv V_1 \pmod{G_{Rg}}$. Hence, $w_3 \neq 0$.

8.4 Proof of correctness (Verification steps 16, 17, 18, 19, 20, 21, and 22: KAZ-SIGN digital signature forgery detection procedure type – 3)

In order to comprehend the rationale behind steps 16, 17, 18, 19, 20, 21, and 22, one has to observe

$$S_1S_2 - GS_{1r}h \equiv GS_{1r}V_1^{S_1} \pmod{q}$$

because $\alpha \equiv V_1 \pmod{q}$. Hence, $w_6 = 0$.

8.5 Proof of correctness (Verification steps 23, 24, 25, 26, 27, 28, and 29 : KAZ-SIGN digital signature forgery detection procedure type – 4)

In order to comprehend the rationale behind steps 23, 24, 25, 26, 27, 28, and 29, one has to observe

$$w_{80} \equiv \left((S_1S_2 - GS_{1r}h)(GS_{1r})^{-1}\right)^{2w_7} \equiv (\alpha^{S_1})^{2w_7} \equiv \alpha^4 \pmod{q^2}.$$

By computing $w_8 = H(w_{80})$, we finally have $w_9 = 0$.

8.6 Deriving forged signature identifiable by KAZ-SIGN digital signature forgery detection procedure type – 1 and KAZ-SIGN digital signature forgery detection procedure type – 2.

An adversary utilizing a random r_0 computes the corresponding parameter pair given by $(S_1 \pmod{G_{Rg}q^2}, GS_{1r})$. Next, the adversary could compute either one of the following:

1. $S_2 \equiv GS_{1r}(V_1^{S_1} + h)S_1^{-1} \pmod{G_{Rg}q^2}$; or
2. $S_2 \equiv GS_{1r}(\alpha_F^{S_1} + h)S_1^{-1} \pmod{G_{Rg}q^2}$

Since $\alpha \equiv V_1 \equiv \alpha_F \pmod{G_{Rg}}$, the forged signature pair will pass steps 30, 31, 32, 33, 34, 35, 36 and 37. However, the signature pair will fail KAZ-SIGN digital signature forgery detection procedure type – 1 or KAZ-SIGN digital signature forgery detection procedure type – 2.

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

An adversary utilizing a random r_0 computes the corresponding parameter pair given by $(S_1 \pmod{G_{Rg}q^2}, GS_{1r})$. Next, with a random $x \in \mathbb{Z}_{G_{Rg}q^2}$ and random unknown prime $\rho \approx q$, the adversary could compute either one of the following:

i) $S_2 \equiv GS_{1r}(V_1^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}q^2}$; or

ii) $S_2 \equiv GS_{1r}(V_1^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}\rho q}$; or

iii) $S_2 \equiv GS_{1r}(\alpha_F^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}q^2}$; or

iv) $S_2 \equiv GS_{1r}(\alpha_F^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}\rho q}$.

The forged signature pair will not be able to be detected by either the KAZ-SIGN digital signature forgery detection procedure type – 1 or KAZ-SIGN digital signature forgery detection procedure type – 2. It will also pass steps 30, 31, 32, 33, 34, 35, 36 and 37. However, the signature pair will fail KAZ-SIGN digital signature forgery detection procedure type – 3. This is because, one would obtain either:

i) $S_1S_2 - GS_{1r}h \equiv GS_{1r}(V_1^{S_1} + G_{Rg}x) \not\equiv GS_{1r}V_1^{S_1} \pmod{q}$; or

ii) $S_1S_2 - GS_{1r}h \equiv GS_{1r}(\alpha_F^{S_1} + G_{Rg}x) \not\equiv GS_{1r}V_1^{S_1} \pmod{q}$.

As a note, the corresponding parameter S_1 could also be modulo $G_{Rg}\rho q$. Nevertheless, the above output will remain.

An alternative for the adversary would be to derive the corresponding S_1 modulo $G_{Rg}q^2$ by solving the following relation:

$$S_1S_2 - GS_{1r}h \equiv GS_{1r}V_1^{S_1} \pmod{G_{Rg}q^2} \tag{1}$$

However, to solve equation (1), the complexity is is $O(q)$. When deploying Grover's algorithm on a quantum computer, the complexity will be $O(q^{0.5})$. Furthermore q is a k-bit prime number (where k is either 128 or 192 or 256 bits). The adversary will not be able to execute the Chinese Remainder Theorem to reduce this complexity.

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

An adversary utilizing a random r_0 and random unknown prime $\rho \approx q$ computes the corresponding parameter pair $(S_1 \pmod{G_{Rg}\rho q}, GS_{1r})$. Next, the adversary could compute the following:

$$S_2 \equiv GS_{1r}(V_1^{S_1} + h)S_1^{-1} \pmod{G_{Rg}\rho q}$$

The forged signature pair will not be able to be detected by either the KAZ-SIGN digital signature forgery detection procedure type – 1 or KAZ-SIGN digital signature forgery detection procedure type – 2 or KAZ-SIGN digital signature forgery detection procedure type – 3. It will also pass steps 30, 31, 32, 33, 34, 35, 36 and 37. However, the signature pair will fail KAZ-SIGN digital signature forgery detection procedure type – 4. This is because of the different groups $\mathbb{Z}_{G_{Rg}\rho q}$ and $\mathbb{Z}_{G_{Rg}q^2}$.

Note that, by replacing V_1 with α_F for the above forgery strategy in this section, the forged signature will not pass KAZ-SIGN digital signature forgery detection procedure type – 3. This is because $\alpha_F \not\equiv V_1 \pmod{q}$.

8.9 Extracting $\alpha \pmod{G_{Rg}q^2}$ from S_2.

Observe that,

$$z_1 \equiv S_1S_2 - GS_{1r}h \equiv GS_{1r}\alpha^{S_1} \pmod{G_{Rg}q^2}.$$

Since $G_{Rg} \equiv 0 \pmod{GS_{1r}}$, we can have

$$z_2 \equiv z_1 GS_{1r}^{-1} \equiv \alpha^{S_1} \pmod{G_{Rg2}q^2} \tag{2}$$

where $G_{Rg2} = G_{Rg}GS_{1r}^{-1}$. However, $\gcd(S_1, \phi(G_{Rg2}q^2)) \neq 1$. Suppose $z_3 = \gcd(S_1, \phi(G_{Rg2}q^2))$. One can then proceed to compute $z_4 \equiv z_3S_1^{-1} \pmod{\phi(G_{Rg2}q^2)}$. As a result, one can obtain:

$$z_2^{z_4} \equiv \alpha^{z_3} \pmod{G_{Rg2}q^2} \tag{3}$$

Thus, for both cases (2) and (3), the complexity to obtain α modulo $G_{Rg2}q^2$ is $O(G_{Rg2}q^2)$.

8.10 Extracting α via KAZ-SIGN digital signature forgery detection procedure type – 4

Through the KAZ-SIGN digital signature forgery detection procedure type – 4, the adversary can proceed to obtain the value $w_{81} \equiv \alpha \pmod{q^2}$ from the extracted value $w_{80} \equiv \alpha^4 \pmod{q^2}$ from a valid signature.

Then, the challenge faced the adversary is to retrieve α from $w_{81} \equiv \alpha \pmod{q^2}$. This is the MRP. Since $G_{Rg}q < q^2$, the complexity of solving the MRP via V_1 is much higher than the complexity of solving the MRP via $\alpha \pmod{q^2}$.

As such, the complexity of solving the MRP via $\alpha \pmod{q^2}$ will be the determining factor in identifying the suitable key length for each security level.

8.11 Modular linear equation of S_2.

In this direction we obtain $r_F \equiv S_1 \pmod{G_{Rg}}$.

From the above, observe that one can analyze S_2 as follows,

$$S_2 \equiv GS_{1r}(\alpha^{S_1} + h)r^{-1} \equiv GS_{1r}(V_1^{S_1} + h)r_F^{-1} \pmod{G_{Rg}}$$

Since $G_{Rg} \equiv 0 \pmod{GS_{1r}}$, it implies

$$(\alpha^{S_1} + h)r^{-1} \equiv (V_1^{S_1} + h)r_F^{-1} \pmod{G_{Rg2}}$$

where $G_{Rg2} = G_{Rg} GS_{1r}^{-1}$. Moving forward we have,

$$r_F \alpha^{S_1} - (V_1^{S_1} + h)r + hr_F \equiv 0 \pmod{G_{Rg2}} \quad (4)$$

Let $\hat{\alpha}$ be the upper bound for α^{S_1} and $\hat{r}$ be the upper bound for r. From Conjecture 1, if one has the situation where $\hat{\alpha}\hat{r} \gg G_{Rg2}$, then there is no efficient algorithm to output all the roots of (4). That is, (4) usually has G_{Rg2} many solutions, which is exponential in the bit-size of G_{Rg2}.

To this end, since α^{S_1} is exponentially large, it is clear to conclude that $\hat{\alpha}\hat{r} \gg G_{Rg2}$. This implies, there is no efficient algorithm to output all the roots of (4).

8.12 Implementation of the Hidden Number Problem

From S_2 to obtain α or r, is the hidden number problem.

8.13 Another "Expensive" Problem Related To KAZ-SIGN: The Second Order Discrete Logarithm Problem (2-DLP)

Let N be a composite number, g a random prime in $\mathbb{Z}_N$ of order G_g where at most $G_g \approx N^\delta$ for $\delta \in (0,1)$ and $\delta \to 0$. That is, $g^{Gg} \equiv 1 \pmod{N}$. Choose a random prime $Q \in \mathbb{Z}_{\phi(N)}$ of order G_Q, where $G_Q \approx \phi(N)^\varepsilon$ for $\varepsilon \to 1$. That is, choose Q with a large order in $Z_{\phi(N)}$. Such Q, has it own natural order in $Z_{\phi(G_g)}$. Let that order be denoted as G_{Qg}. We can observe the natural relation given by $Q^{G_{Qg}} \equiv 1 \pmod{G_g}$ and $\phi(N) \equiv 0 \pmod{G_g}$.

Then choose a random integer $x \in \mathbb{Z}_{\phi(G_g)}$ where $x \approx \phi(G_g)$. Suppose from the relation given by

$$g^{Q^x \pmod{\phi(N)}} \equiv A \pmod{N} \quad (5)$$

one has solved the Discrete Logarithm Problem (DLP) upon equation (5) in polynomial time on a classical computer and obtained the value X where $Q^x \not\equiv X \pmod{\phi(N)}$ and $g^X \equiv A \pmod{N}$, The relation $Q^x \not\equiv X \pmod{\phi(N)}$ would result in the non-existence of the discrete logarithm solution for $Q^x \equiv X \pmod{\phi(N)}$.

The 2-DLP is, upon given the values (A, g, N, Q), one is tasked to determine $x \in \mathbb{Z}_{\phi(G_g)}$ where $x \approx \phi(G_g)$ such that equation (5) holds.

Let $Q^x \equiv T_1$ (mod $\phi(N)$. From the predetermined order of $g \in \mathbb{Z}_N$, during the process of solving the DLP upon equation (5), a collision would occur prior to the full cycle of g. As such, the process of solving the DLP upon equation (5) to obtain $X \approx N^{\delta}$ would occur in polynomial time on a classical computer. And since $T_1 < \phi(N)$ and $T_1 \approx N_1$, the relation $Q^x \not\equiv X \pmod{\phi(N)}$ will hold.

Furthering on the discussion, one has the relation $g^{G_g} \equiv 1 \pmod{N}$. As such, from the value $X < G_g$ obtained from equation (5), one can construct the set of solutions given by $T_0 = X + G_g t$ for $t = 0, 1, 2, 3, \ldots$. Now let $Q^x \equiv T_1 \pmod{\phi(N)}$. Following through,since T_1 is an element from the set of solutions, one can have the relation

$$t_{T_1} = \frac{T_1 - X}{G_g}$$

Since $G_g, X \approx N^{\delta}$, and $\phi(N) \approx N$, the complexity to obtain t_{T_1} is $O(N^{1-\delta})$. When deploying Grover's algorithm on a quantum computer, the complexity to obtain t_{T_1} is $O(N^{\frac{1-\delta}{2}})$.

To this end, note that if one proceeds to solve the DLP upon $Q^x \equiv X \pmod{G_g}$, one can obtain the value $x_0 \equiv x \pmod{G_{Qg}}$. From the preceding sections, this is in fact the MRP. It is easy to see that with correct choice of parameters (x, G_{Qg}), the complexity of 2-DLP and MRP can be made the same. Hence, a more "non-expensive" method in discussing the needs of the KAZ-SIGN is directly via the MRP.

9. KEY GENERATION, SIGNING AND VERIFICATION TIME COMPLEXITY

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

10. SPECIFICATION OF KAZ-SIGN

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

Number of primes in P	$n = \ell(N)$	Total security level, k
195	1662	128
290	2667	192
390	3783	256

Table 1

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 security level information from Table 1 (for $\delta = 0.23$) where $\ell(V_2)$ is the length of an output generated by a 256-bit hash function.

NIST Security Level	Number of primes in P	Security level, k	Length of parameter N (bits)	Public key size, (V_1, V_2) (bits)	Private key size, α (bits)	Signature Size (S_1, S_2) (bits)	ECC key size (bits)
1	195	128	1662	$\approx 218 + 256 = 474$	≈ 384	≈ 700	256
3	290	192	2667	$\approx 332 + 256 = 588$	≈ 576	≈ 1046	384
5	390	256	3783	$\approx 450 + 256 = 706$	≈ 768	≈ 1409	521

Table 2

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

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 - KAZ1662	108	384	161
192 - KAZ2667	117	426	375
256 - KAZ3783	123	513	1186

Table 3

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 706-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, and type – 4** 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.

14. ILLUSTRATIVE FULL SIZE TEST VECTORS – 1

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3, 5, 7, \ldots, 1193\}$. In this illustration, we provide a valid KAZ-SIGN signature pair $S = (S_1, S_2)$. The valid KAZ-SIGN signature will pass all 4 KAZ-SIGN digital signature forgery detection procedure types.

N :
11407459538923317956992856472478034719938444515550504873191432724561240
77590743470594911066332509539066597551699013776050581430643550167887346
08191843438244914234171652904624354475977819424112770780591839969259477
16504087172783276497191148945981028252785414086386424445171843610035797
71215355642643202854238405781778429260537407631182034795543825674983533
93399124947777143263677778788796582423578186360342165106147009426252 83
16073897973467968535780096264794534714067794192763112712608847144283240
4185 $\approx 2^{1662}$

g :
6007

G_g :
17720681536509435215163237615189945478576380515988874530058447706012782
9364641006743420972687893421171393095806176000 $\approx 2^{387} \approx 2^{0.232(1662)} \approx N^{0.232}$

R :
6151

G_{Rg} :
3567853131480071022029641200 0 $\approx 2^{95} \approx N^{0.057}$

q :
18627106629136517523513455977536297977 9

Key generation

α :
119648675149800676507745088101116425428374212690189712761950433680013980536288952135553484479144299921754187859632 69 $\approx 2^{383}$

V_1 :
2094185182448701905319928164821044095341949158961290587886894107269

$t_{\alpha V_1} = \dfrac{\alpha - V_1}{G_{Rg} q}$:
1800344120951868743001728342526473219585390642772 $\approx 2^{161}$

$\alpha^4 \pmod{q^2}$:
267216841398013221587445837240581592236612762096013356075082782770603759 23046

$V_2 = H(\alpha^4 \pmod{q^2})$:
`ba391875ff50ea9e738a5cc0159b1339bb8fa0e2f270ef378e5c5240fcbc4732`

Signing

h :
910064287414137313665450137965897620831174852451769694397574145815418786 14238

r :
1185287286461866994330768150085143491865530563662615076112510207886464645378568617070553002615601048748145745865656 0

S_1 :
383280853942879221199791598440756269797113868564031038689232946075485855619571589870248043417102016752560

S_2 :
898376279818222649749655416712652166826778988758637952582049773171351708655925733391607667230757292074179

GS_1 and GS_{1r} :
2160

Verification

KAZ-SIGN digital signature forgery detection procedure type – 1

w_0 :
761679607153335214391753240705006834963459891264482167623026291359549745006546876886604796156804399106779

$w_1 = w_0 - S_2$:
$-$ 136696672664887435357902176007645331863319097494155784959023481811801963649378856505002871073952892967400

KAZ-SIGN digital signature forgery detection procedure type – 2

w_2 :
896521076337686999332735353368153838617423165326855510413694787982572568814970755009584582939499182134779

$w_3 = w_2 - S_2$:
$-$ 18555203480535650416920063344498328209355823431782442168354985188779139840954978382023084291258109939400

KAZ-SIGN digital signature forgery detection procedure type – 3

w_4 :
57336069090460161702150769343223740998

w_5 :
57336069090460161702150769343223740998

$w_6 = w_4 - w_5$:
0

KAZ-SIGN digital signature forgery detection procedure type – 4

w_7 :
1590281275589685874846763137233424549967118327810933629745453682578197517
9315

w_{80} :
2672168413980132215874458372405815922366127620960133560750827827706037592
3046

$H(w_{80})$:
`ba391875ff50ea9e738a5cc0159b1339bb8fa0e2f270ef378e5c5240fcbc4732`

$w_9 = H(w_{80}) - V_2$:
0

MRP complexity upon $w_{81} = \alpha \pmod{q^2}$

$w_{81} = \alpha \pmod{q^2}$:
2569344103747874058684908223729786629800855093160215735786452403680426903
8715

$t_{\alpha w_{81}} = \dfrac{\alpha - w_{81}}{q^2}$:
344839568354241152569342511565821233594 $\approx 2^{129}$

Final verification

y_1 and y_2 :
9678733104992034888628894584286731645027347443872038782768171489732529388
0048877490496307118868165050375032594242014959669840046107246260536803183
4888203384559829798330927698615360241642588019411254504903877747159053146
6701871613002290052643640816468457435808334759136405882613474759745455101
1451522048434905036144454872522946456115748221581710227263448217652008147
5714830034648821466616203937051374293791936095743419505390553360154708920
60746760512011109369326662461764018069641813629469515808984342

15. ILLUSTRATIVE FULL SIZE TEST VECTORS – 2

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3,5,7,\ldots,1193\}$. In this illustration, we provide a forged KAZ-SIGN signature pair $S = (S_1,S_2)$ where the system parameters, $(N,g,G_g,R,G_{Rg},\alpha,V_1,V_2,h,r,S_1)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_2 \equiv G_{1r}(V_1^{S_1}+h)S_1^{-1} \pmod{G_{Rg}q^2}$. This signature pair will fail the **KAZ-SIGN digital signature forgery detection procedure type - 1**.

S_2 :
76167960715333521439175324070500683496345989126448216762302629135954974
5006546876886604796156804399106779

GS_1 and GS_{1r} :
2160

KAZ-SIGN digital signature forgery detection procedure type – 1

w_0 :
76167960715333521439175324070500683496345989126448216762302629135954974
5006546876886604796156804399106779

w_1 :
0

Final verification

y_1 and y_2 :
96787331049920348886288945842867316450273474438720387827681714897325293
88004887749049630711886816505037503259424201495966984004610724626053680
31834888203384559829798330927698615360241642588019411254504903877747159
05314667018716130022900526436408164684574358083347591364058826134747597
45455101145152204843490503614445487252294645611574822158171022726344821
76520081475714830034648821466616203937051374293791936095743419505390553
36015470892060746760512011109369326662461764018069641813629469515808984
342

16. ILLUSTRATIVE FULL SIZE TEST VECTORS – 3

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3, 5, 7, \ldots, 1193\}$. In this illustration, we provide a forged KAZ-SIGN signature pair $S = (S_1, S_2)$ where the system parameters, $(N, g, G_g, R, G_{Rg}, \alpha, V_1, V_2, h, r, S_1)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_2 \equiv G_{1r}(\alpha_F^{S_1} + h)S_1^{-1} \pmod{G_{Rg}q^2}$. This signature pair will fail the **KAZ-SIGN digital signature forgery detection procedure type – 2**.

α_F :
25765466249370643715119715269

S_2 :
89652107633768699933273535336815383861742316532685551041369478798257256
8814970755009584582939499182134779

GS_1 and GS_{1r} :
2160

KAZ-SIGN digital signature forgery detection procedure type – 2

w_2 :
89652107633768699933273535336815383861742316532685551041369478798257256
8814970755009584582939499182134779

w_3 :
0

Final verification

y_1 and y_2 :
96787331049920348886288945842867316450273474438720387827681714897325293
88004887749049630711886816505037503259424201495966984004610724626053680
31834888203384559829798330927698615360241642588019411254504903877747159
05314667018716130022900526436408164684574358083347591364058826134747597
45455101145152204843490503614445487252294645611574822158171022726344821
76520081475714830034648821466616203937051374293791936095743419505390553
36015470892060746760512011109369326662461764018069641813629469515808984
342

17. ILLUSTRATIVE FULL SIZE TEST VECTORS – 4

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3, 5, 7, \ldots, 1193\}$. In this illustration, we provide a forged KAZ-SIGN signature pair $S = (S_1, S_2)$ where the system parameters, $(N, g, G_g, R, G_{Rg}, \alpha, V_1, V_2, h, r, S_1)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_2 \equiv G_{1r}(V_1^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}q^2}$. This signature pair will pass the **KAZ-SIGN digital signature forgery detection procedure type – 1 and type – 2**. However, this signature pair will fail the **KAZ-SIGN digital signature forgery detection procedure type – 3**.

x :
10618684067963136096872551407440065459774895754182940862141292049797260
8607722

S_2 :
36221688592028277861806102659303141259702767567329838129865594939084446
28553699075548950793547793051467799

GS_1 and GS_{1r} :
2160

KAZ-SIGN digital signature forgery detection procedure type – 3

w_4 :
24066027619102262377401085416454550544

w_5 :
57336069090460161702150769343223740998

w_6 :
$-$33270041471357899324749683926769190454

Final verification

y_1 and y_2 :
96787331049920348886288945842867316450273474438720387827681714897325293
88004887749049630711886816505037503259424201495966984004610724626053680
31834888203384559829798330927698615360241642588019411254504903877747159
05314667018716130022900526436408164684574358083347591364058826134747597
45455101145152204843490503614445487252294645611574822158171022726344821
76520081475714830034648821466616203937051374293791936095743419505390553
36015470892060746760512011109369326662461764018069641813629469515808984
342

18. ILLUSTRATIVE FULL SIZE TEST VECTORS – 5

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3,5,7,\ldots,1193\}$. In this illustration, we provide a forged KAZ-SIGN signature pair $S = (S_1, S_2)$ where the system parameters, $(N, g, G_g, R, G_{Rg}, \alpha, V_1, V_2, h, r, S_1)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_2 \equiv G_{1r}(\alpha_F^{S_1} + h + G_{Rg}x)S_1^{-1} \pmod{G_{Rg}q^2}$. This signature pair will pass the **KAZ-SIGN digital signature forgery detection procedure type – 1 and type – 2**. However, this signature pair will fail the **KAZ-SIGN digital signature forgery detection procedure type – 3**.

α_F :
25765466249370643715119715269

x :
10292570878411527362985895757184114510600700155840767275769719533899661
0287616

S_2 :
25926009443463365313589620010673337933679600055940946817131870184278638
4071104964657529549011435148658779

GS_1 and GS_{1r} :
2160

KAZ-SIGN digital signature forgery detection procedure type – 3

w_4 :
165142597253735214432446640655721095637

w_5 :
57336069090460161702150769343223740998

w_6 :
107806528163275052730295871312497354639

Final verification

y_1 and y_2 :
9678733104992034888628894584286731645027347443872038782768171489732529 3
8800488774904963071188681650503750325942420149596698400461072462605368 0
3183488820338455982979833092769861536024164258801941125450490387774715 9
0531466701871613002290052643640816468457435808334759136405882613474759 7
4545510114515220484349050361444548725229464561157482215817102272634482 1
7652008147571483003464882146661620393705137429379193609574341950539055 3
3601547089206074676051201110936932666246176401806964181362946951580898 4
342

19. ILLUSTRATIVE FULL SIZE TEST VECTORS – 6

The following are parameters that illustrate KAZ-SIGN for 128-bit security (refer to Table 2). This is an example for $j = 195$. That is, $P = \{3, 5, 7, \ldots, 1193\}$. In this illustration, we provide a forged KAZ-SIGN signature pair $S = (S_1, S_2)$ where the system parameters, $(N, g, G_g, R, G_{Rg}, \alpha, V_1, V_2, h, r)$ are the same as in **ILLUSTRATIVE FULL SIZE TEST VECTORS – 1** and $S_1 \equiv r \pmod{G_{Rg}\rho q}$ and $S_2 \equiv GS_{1r}(V_1^{S_1} + h)S_1^{-1} \pmod{G_{Rg}\rho q}$. This signature pair will pass the **KAZ-SIGN digital signature forgery detection procedure type – 1, type – 2 and type – 3**. However, this signature pair will fail the **KAZ-SIGN digital signature forgery detection procedure type – 4**.

ρ :
25765466249370643715119715269

S_1 :
38328085394287922119979159004128098235898447712885717733355141843099320
07490812634408195138152771071 20560

S_2 :
29719339281599439145650707365536734023799072194611705382204228269452658
73744814289429851394301295462 59579

GS_1 and GS_{1r} :
2160

KAZ-SIGN digital signature forgery detection procedure type – 4

w_7 :
12055485020885182345868886859695725401374311440640181188606350691281905
997689

w_{80} :
32547142287675725316375416153708109141324660055984491776946220327241251

$H(w_{80})$:
`deccb2c9009a8b043d2f0e6adaa9b1343eb6e87865338d8536a01da9a84939f1`

$w_9 = H(w_{80}) - V_2 \neq 0$

Final verification

y_1 and y_2 :
9678733104992034888628894584286731645027347443872038782768171489732529388004887749049630711886816505037503259424201495966984004610724626053680318348882033845598297983309276986153602416425880194112545049038777471590531466701871613002290052643640816468457435808334759136405882613474759745455101145152204843490503614445487252294645611574822158171022726344821765200814757148300346488214666162039370513742937919360957434195053905533601547089206074676051201110936932666246176401806964181362946951580898434 2

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.