

Kriptografi Atasi Zarah
Digital Signature v2.0 (KAZ-SIGN v2.0)

Algorithm Specifications and Supporting Documentation

(KAZ-SIGN 2.0)

Muhammad Rezal Kamel Ariffin¹ Abderrahmane Nitaj² Nor Azman Abu³ Zahari Mahad¹
Muhammad Asyraf Asbullah¹ Amir Hamzah Abd Ghafar¹

¹Universiti Putra Malaysia

²Université de Caen Normandie, France

³Universiti Teknikal Malaysia Melaka

Table of Contents

1 INTRODUCTION	1
2 THE DESIGN IDEALISME	1
3 THE DOUBLE DISCRETE LOGARITHM PROBLEM (DDLDP)	2
4 UNIQUENESS	2
5 COMPLEXITY OF SOLVING THE DDLDP	2
6 THE HIDDEN NUMBER PROBLEM (HNP) (Boneh and Venkatesan, 2001)	2
7 COMPLEXITY OF SOLVING THE HNP	3
8 THE KAZ-SIGN v2.0 ALGORITHM	3
8.1 Background	3
8.2 Utilized Functions	3
8.3 System Parameters	3
8.4 KAZ-SIGN v2.0 Algorithms	3
9 PROOF OF CORRECTNESS	4
10 IMPLEMENTATION OF THE DOUBLE DISCRETE LOGARITHM PROBLEM (DDLDP)	5
11 IMPLEMENTATION OF THE HIDDEN NUMBER PROBLEM (HNP)	5
12 DISCRETE LOGARITHM PROBLEM ANALYSIS	5
13 DERIVING THE SECURITY LEVEL OF KAZ-SIGN v2.0	5
14 IMPLEMENTATION AND PERFORMANCE	6
14.1 Partial and Full Key Generation Time Complexity	6
14.2 Parameter sizes	6
14.3 KAZ-SIGN v2.0 Ease of Implementation	6
14.4 KAZ-SIGN v2.0 Empirical Performance Data	6
15 ADVANTAGES AND LIMITATIONS	7
15.1 Key Length	7
15.2 Speed	7
15.3 No Full Key Generation Failure	7
15.4 Limitation	7
15.4.1 Based on not widely used problem, Double Discrete Logarithm Problem (DDLDP) and Hidden Number Problem (HNP)	7
16 CLOSING REMARKS	8
17 ILLUSTRATIVE FULL SIZE TEST VECTORS	9

Name of the proposed cryptosystem: KAZ-SIGN v2.0

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: Abderrahmane Nitaj
Nor Azman Abu
Zahari Mahad
Muhammad Asyraf Asbullah
Amir Hamzah Abd Ghafar

Inventor of the cryptosystem: Muhammad Rezal Kamel Ariffin

Owner of the cryptosystem: Muhammad Rezal Kamel Ariffin

Alternative point of contact: Zahari Mahad
Institute for Mathematical Research
Universiti Putra Malaysia
43400 UPM Serdang, Selangor
Malaysia
Email: zaharimahad@upm.edu.my
Phone: +60122437663

1. INTRODUCTION

The proposed KAZ Digital Signature v2.0 scheme, KAZ-SIGN v2.0 (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 Double Discrete Logarithm Problem (DDLp) and Hidden Number Problem (HNP). The target of the KAZ-SIGN v2.0 design is to be a quantum resistant digital signature candidate with short parameters, executing correctly 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 plaintext-ciphertext space;
- (viii) The computation overhead for both encapsulation and decapsulation 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 ciphertext expansion ratio is kept to a minimum.

One of our key strategy to obtain items (i) - (v) was by utilizing the defined Double Discrete Logarithm Problem (DDLp) and Hidden Number Problem (HNP). It is defined in the following section.

3. THE DOUBLE DISCRETE LOGARITHM PROBLEM (DDLDP)

Let $N = \prod_{i=1}^j p_i$ be a public modulus. Choose random primes (g_1, g_2) where $\text{DLog}_{g_1}(g_2)$ and $\text{DLog}_{g_2}(g_1)$ modulo N does not exist. Let O_{g_1N} be the order of g_1 in \mathbb{Z}_N . Let O_{g_2N} be the order of g_2 in \mathbb{Z}_N . Compute $A \equiv g_1^s g_2^t \pmod{N}$ for some $s \in \mathbb{Z}_{O_{g_1N}}$ and $t \in \mathbb{Z}_{O_{g_2N}}$. The DDLDP is, upon given the parameters (A, g_1, g_2, N) , one is tasked to identify the pair (s, t) .

4. UNIQUENESS

Assume there exists (α, β) such that $g_1^s g_2^t \equiv g_1^\alpha g_2^\beta \pmod{N}$. We will have,

$$g_1^{s-\alpha} \equiv g_2^{\beta-t} \pmod{N}$$

Which implies either,

$$g_1 \equiv g_2^{\zeta_1} \pmod{N}$$

or

$$g_2 \equiv g_1^{\zeta_2} \pmod{N}$$

for some $\zeta_1, \zeta_2 \in \mathbb{Z}_{\phi(N)}$. This is false, since we have chosen random primes (g_1, g_2) where $\text{DLog}_{g_1}(g_2)$ and $\text{DLog}_{g_2}(g_1)$ modulo N does not exist.

5. COMPLEXITY OF SOLVING THE DDLDP

The complexity to obtain s is $O(s)$. Due to the strategies during key generation, we have the complexity $O(s) > O(2^k)$ where k is the chosen security level, either 128, 192 or 256. When deploying Grover's algorithm on a quantum computer, the complexity to obtain s is $O(2^{\frac{k}{2}})$.

6. 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.

7. COMPLEXITY OF SOLVING THE HNP

The complexity to obtain $\alpha \pmod{p}$ is $O(p)$. When deploying Grover's algorithm on a quantum computer, the complexity to obtain $\alpha \pmod{p}$ is $O(p^{\frac{1}{2}})$.

8. THE KAZ-SIGN v2.0 ALGORITHM

8.1 Background

This section discusses the construction of the KAZ-SIGN v2.0 scheme. We provide information regarding the partial key agreement generation and full key agreement generation procedures. But first, we will put forward functions that we will utilize and the system parameters for all users.

8.2 Utilized Functions

Let $\ell(\cdot)$ be the function that outputs the bit length of a given input. Let $\text{DLog}_u(v)$ modulo p be the function that outputs w such that $v \equiv u^w \pmod{p}$.

8.3 System Parameters

From the given security parameter k (either 128, 192 or 256; depending on the security level needed), prepare a list of the first j -primes larger than 2, $P = \{p_i\}_{i=1}^j$ (as of printing it is suggested $j = 65, 96, 122$). Let $N = \prod_{i=1}^j p_i$ be a public modulus. Choose random primes (g_1, g_2) where $\text{DLog}_{g_1}(g_2)$ and $\text{DLog}_{g_2}(g_1)$ modulo N does not exist. Let O_{g_1N} be the order of g_1 in \mathbb{Z}_N . Let O_{g_2N} be the order of g_2 in \mathbb{Z}_N . The system parameters are $(g_1, g_2, O_{g_1N}, O_{g_2N}, N)$.

8.4 KAZ-SIGN v2.0 Algorithms

The full algorithms of KAZ-SIGN v2.0 are shown in Algorithms 1, 2, and 3.

Algorithm 1 KAZ-SIGN v2.0 Key Generation Algorithm

Input: System parameters $(g_1, g_2, O_{g_1N}, O_{g_2N}, N)$.

Output: Verification key, V and signing key pair, (s, t) .

- 1: Choose random $s \in \mathbb{Z}_{O_{g_1N}}$ and $t \in \mathbb{Z}_{O_{g_2N}}$
 - 2: Compute $V \equiv g_1^s g_2^t \pmod{N}$.
 - 3: Verification key V and keep signing key pair (s, t) .
-

Algorithm 2 KAZ-SIGN v2.0 Signing

Input: System parameters $(g_1, g_2, O_{g_1N}, O_{g_2N}, N)$, signing key pair (s, t) , suitable hash function $H(\cdot)$ and message m .

Output: Signature, S .

- 1: Compute the hash of the message, m to be signed $h = H(m)$.
 - 2: Choose random ephemeral $e_1 \in \mathbb{Z}_{O_{g_1N}}$ and $e_2 \in \mathbb{Z}_{O_{g_2N}}$.
 - 3: Compute $S_1 \equiv g_1^{e_1} g_2^{e_2} \pmod{N}$.
 - 4: Compute $S_2 \equiv (h - sS_1)e_1^{-1} \pmod{\phi(N)}$.
 - 5: Compute $S_3 \equiv h - tS_1 - e_2S_2 \pmod{\phi(N)}$.
 - 6: Output signature $S = (S_1, S_2, S_3)$ and destroy (e_1, e_2) .
-

Algorithm 3 KAZ-SIGN v2.0 Verification

Input: Signature $S = (S_1, S_2, S_3)$, verification key V . suitable hash function $H(\cdot)$ and message m .

Output: Accept or reject signature.

- 1: Compute the hash of the message, m to be verified $h = H(m)$.
 - 2: Compute $Y_1 \equiv V^{S_1} S_1^{S_2} g_2^{S_3} \pmod{N}$
 - 3: Compute $Y_2 \equiv (g_1 g_2)^h \pmod{N}$.
 - 4: **if** $Y_1 = Y_2$ **then**
 - 5: accept signature
 - 6: **else** reject signature \perp
 - 7: **end if**
-

9. PROOF OF CORRECTNESS

Observe the following,

$$\begin{aligned} Y_1 &\equiv V^{S_1} S_1^{S_2} g_2^{S_3} \\ &\equiv (g_1^s g_2^t)^{S_1} (g_1^{e_1} g_2^{e_2})^{S_2} g_2^{S_3} \\ &\equiv g_1^{sS_1 + e_1S_2} g_2^{tS_1 + e_2S_2 + S_3} \\ &\equiv (g_1 g_2)^h \\ &\equiv Y_2 \pmod{N} \end{aligned}$$

10. IMPLEMENTATION OF THE DOUBLE DISCRETE LOGARITHM PROBLEM (DDLDP)

It is clear that the following parameters are implementing the DDLDP:

1. $V \equiv g_1^s g_2^t \pmod{N}$
2. $S_1 \equiv g_1^{e_1} g_2^{e_2} \pmod{N}$

11. IMPLEMENTATION OF THE HIDDEN NUMBER PROBLEM (HNP)

It is clear that the following parameters are implementing the HNP:

1. $V \equiv x g_2^t \pmod{N}$ where $x \equiv g_1^s \pmod{N}$
2. $S_1 \equiv y g_2^{e_2} \pmod{N}$ where $y \equiv g_1^{e_1} \pmod{N}$

12. DISCRETE LOGARITHM PROBLEM ANALYSIS

Since $DLog_{g_1}(g_2)$ and $DLog_{g_2}(g_1)$ modulo N does not exist, an immediate DLP scenario does not occur with the equations (V, S_1) . However, assume that h is a primitive root in \mathbb{Z}_N such that we have $h^{z_1} \equiv g_1 \pmod{N}$ and $h^{z_2} \equiv g_2 \pmod{N}$ where N is either 1, 2, 4, p^k , or $2p^k$ where p is an odd prime number and k is a positive integer. One would now have the following equations:

$$V \equiv h^{z_1 s + z_2 t} \pmod{N} \quad (1)$$

$$S_1 \equiv h^{z_1 e_1 + z_2 e_2} \pmod{N} \quad (2)$$

Upon obtaining the DLP solution for (1) and (2), one could proceed to obtain two equations with four variables. Furthermore, parameter of N in KAZ-SIGN v2.0 does not subscribe to either 1, 2, 4, p^k , or $2p^k$ where p is an odd prime number and k is a positive integer.

13. DERIVING THE SECURITY LEVEL OF KAZ-SIGN v2.0

The challenge faced by the adversary is to retrieve (s, t) from V and (e_1, e_2) from S_1 . It is protected by the HNP.

Due to the strategies during key generation, we have the complexity $O(s) > O(2^k)$ where k is the chosen security level, either 128, 192 or 256. When deploying Grover's algorithm on a quantum computer, the complexity to obtain s is greater than $O(2^{\frac{k}{2}})$.

14. IMPLEMENTATION AND PERFORMANCE

14.1 Partial and Full Key Generation Time Complexity

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

14.2 Parameter sizes

We will utilize $g_1 = 65563$ and $g_2 = 65617$.

NIST Security Level	Security level, k	Number of primes in list P , j	Order (O_{g_1N}, O_{g_2N}) size (bits)	Verification key size V (bits)	Signing Key Size, (s, t) (bits)	Signature Size, (S_1, S_2, S_3) (bits)
1	128	65	$\approx (128, 128)$	432	256	1,292
3	192	96	$\approx (192, 192)$	702	384	2,102
5	256	122	$\approx (256, 256)$	942	512	2,822

Table 1

14.3 KAZ-SIGN v2.0 Ease of Implementation

The algebraic structure of KAZ-SIGN v2.0 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.

14.4 KAZ-SIGN v2.0 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 100 key generations, 100 signings and 100 verifications are conducted:

Security level	Time (ms)		
	Key Generation	Signing	Verification
128	3	10	17
192	9	29	40
256	12	52	97

Table 2

15. ADVANTAGES AND LIMITATIONS

As we have seen, KAZ-SIGN v2.0 can be evaluated through:

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

15.1 Key Length

KAZ-SIGN v2.0 key length is comparable to non-post quantum algorithms such as ECC and RSA. For 256-bit security, the KAZ-SIGN v2.0 key size is 942-bits. ECC would use 521-bit keys and RSA would use 15,360-bit keys.

15.2 Speed

KAZ-SIGN v2.0'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 encapsulation and decapsulation is $O(n^3)$ where parameter n here is the input length.

15.3 No Full Key Generation Failure

From the proof of correctness, the probability of decapsulation failure is 0.

15.4 Limitation

As we have seen, limitation of KAZ-SIGN v2.0 can be evaluated through:

1. Based on not widely used problem, the Double Discrete Logarithm Problem (DDLDP) and Hidden Number Problem (HNP).

15.4.1 Based on not widely used problem, Double Discrete Logarithm Problem (DDLDP) and Hidden Number Problem (HNP)

The DDLDP and HNP are not known hard mathematical problems that are quantum resistant and are subject to future cryptanalysis success in solving the defined challenge either with a classical or quantum computer.

16. CLOSING REMARKS

The KAZ-SIGN v2.0 key generation mechanism exhibits properties that might result in it being a desirable post quantum key agreement mechanism scheme.

To this end, the security is based on DDLP and HNP, which are not widely used problems. We opine that the acceptance of DDLP and HNP as potential quantum-resistant hard mathematical problems will come hand in hand with a secure cryptosystem designed upon it. We welcome all comments on the KAZ-SIGN v2.0 digital signature algorithm, 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.

17. ILLUSTRATIVE FULL SIZE TEST VECTORS

The following are parameters that illustrate KAZ-SIGN v2.0 for 128-bit security. In this illustration, we provide a valid KAZ-SIGN v2.0 signature $S = (S_1, S_2, S_3)$. The valid KAZ-SIGN v2.0 signature will pass the verification procedure.

Key generation

N :

9680693320350411581735712527156160041331448806285781880953481207107506184928318
589548473667621840334803765737814574120142199988285

g_1 :

65537

g_2 :

65539

s :

51391170912764190373354282301938775411

t :

15558732012432950691121044741653438480

V :

31488249359805585458601647868574211332297749366106257980702852385922758479739
53210567590692111325563070084725337037024755504867538

Signing

e_1 :

83028107879996739747335363200985034503

e_2 :

12886137506727726782364037266927725704

S_1 :

27317710298520888288453776518937564257175586492355422166417658594969746414964563
56827801756193276909948214343936972249817190369853

S_2 :

23693739069049209106766094653944973226094684341683387747068892215706687661716640
8734677637941247015176465259611258021331933979779

S_3 :

472991502697595297547856354365330086447267764194686832894743534423421792266414334
009116057326796099029554692233837495171034716564

Verification

Y_1 :

3358422775285580869900850539974105816184301665352835825870401511169844231505434622
751287103703924069326258553123075557143258575836

Y_2 :

33584227752855808699008505399741058161843016653528358258704015111698442315054346227
51287103703924069326258553123075557143258575836

References

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.