# Neutrosophic Boolean Function and Rejection Sampling in Post Quantum Cryptography

Shashi Kant Pandey[0000−0002−0818−6984*]

February 2023

**Abstract**

The use of random seeds to a deterministic random bit generator to generate uniform random sampling has been applied multiple times in post-quantum algorithms. The finalists Dilithium and Kyber use SHAKE and AES to generate the random sequence at multiple stages of the algorithm. Here we characterize one of the sampleing techniques available in Dilithium for a random sequence of length 256 with the help of the neutrosophic Boolean function.

The idea of the neutrosophic Boolean function came from the theory of neutrosophy and it is useful to study any ternary distributions. We present the non-existence of neutrobalanced bent functions specifically with respect to the sampling named *SampleInBall* in Dilithium.

**Keywords**: Post-quantum cryptography, Bent function, Neutrosophy, Linear structure
**Mathematics Subject Classification**: 16Z05, 94D10

## 1 Introduction

Any equation or a function having class symbols $x, y, ...$ is termed a "logical equation" or a "logical function" by Georg Boole(1815-1864). Every logical function $f(x)$ can be written as $f(x) = ax + b(1 - x)$, under the convention for the law of symbols by George Boole [1]. If $a$ and $b$ are equal then this function is a balanced function. The extension of the idea of Boole's representation of a Boolean function for two variables $x$ and $y$ is $f(x, y) = f(1, 1)xy + f(1, 0)x(1 - y) + f(0, 1)(1 - x)y + f(0, 0)(1 - x)(1 - y)$. With the abuse of notation, a Boolean function in cryptography is termed as a function $f$ from an $n$ dimensional vector space $\mathbb{F}_2^n$ to the base field $\mathbb{F}_2$ of order 2. Later on, the development of cryptography produces many such mathematical properties of Boolean functions [3–6]. To a certain extent, the enumeration of cryptographic properties of a Boolean function starts with an important transformation of a Boolean function named Walsh transformation($\mathcal{WT}$). This transformation depends on the first-order correlation of a bit stream of a Boolean function. Another important cryptographic requirement of a Boolean function is its nonlinearity($\mathcal{NL}$) and it is based on the maximum value of the Walsh transformation. Those Boolean functions having maximum nonlinearity are called bent Boolean functions [7]. Other than the correlation nature of the input-output of a Boolean function, there is a well-known feature of algebraic resistance and it is named Algebraic immunity($\mathcal{AI}$) [8, 9].

*shashi@setsindia.net, Society For Electronic Transaction and Security, Chennai, TN, Bharat.

The generalization of the Boolean function and its other cryptographic features is an interesting area of work. These functions are named as Generalized Boolean function(GBF) in the literature [10–13]. The generalization of the bent Boolean function proposed by O.S. Rothus in [7]. A generalization of Boolean function based on neutrosophy and logic is presented in [14, 21] and termed as a neutrosophic Boolean function.

The idea of neutrosophy came from the classification of a set. An imagination of three disjoint classifications refers that any set $S$ can be classified as $P_1, P_2$ and $P_3$ such that the given condition for a set is satisfied by elements in $P_1$, not satisfied by elements in $P_2$ and undefined or not decidable for elements in $P_3$. In this way, any set with any condition uniformly obeys the idea of neutrality or neutrosophy. This partition leads us to define a neutrosophic function $\psi$ from any set $S$ to any arbitrary set $K$, where $\psi$ is defined on the partition $P_1$, not defined on $P_2$ and indeterminate on $P_3$. The same analogy is useful for studying every rejection sampling where some condition is fixed for the selection of the sample. In particular, any arbitrary evaluation of a function $f(x)$ can be used to reject or accept in the sampling procedure. One simple example for the partition of the sample can be seen as three sets $P_0 = \{x : 0 = f(x) \mod 2\}$, $P_1 = \{x : 0 \neq f(x) \mod 2\}$ and $P_3 = \{x : rejection\}$. We focus here on the sampling function *SampleInBall* of Dilithium for one of the main random vectors $c$ during the signing process in the algorithm [20]. In section 2, we introduce the LWE problem and its background for lattice-based problems. The sampling in Dilithium with the analogy of neutrosophic Boolean function is presented in section 2.1. Further in section 3 we present the analysis of the extendable output function(XOF) and *SampleInBall* function of Dilithium with the characterization of the neutrosophic bent Boolean function.

# 2 Learning with error problem

The reduction of hard maths problems in the classical domain is an ongoing and open-ended problem. Learning with error is one of those hard maths problems about which more discussion is after the NIST PQC competition. However, it is one of the types of uniformity in the finalists of the PQC competition. Oded Regev was the person who coined this in [15]. The problem of LWE is framed as to find a vector $s \in \mathbb{Z}_2^n$ for some integer $n \geq 1$ and a real number $\epsilon$ such that the following set of linear equations with errors,

$$\begin{aligned} < a_1, s >=_\epsilon b_1 (\mod 2) \\ < a_2, s >=_\epsilon b_2 (\mod 2) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{aligned}$$

where each $a_i$ are sampled from the uniform distribution on $\mathbb{Z}_2^n$ and the inner product $< a_i, s >$ is defined as $\sum_{j=1}^{n}(s_j a_i^j)$ on modulo 2. Here, the meaning of an equation with errors is that each equation is correct independently with probability $1 - \epsilon$. The case of zero error can be easily solvable by Gaussian elimination therefore the insertion of hardness in this problem comes from the sampling of errors. The sampling of $a$ is independently and uniformly on $\mathbb{Z}_2^n$ and $b$ is independent with probability $1 - \epsilon$.

Hash function plays a key role in the uniform distribution for the sampleing of vector $a \in \mathbb{Z}_2^n$. In the next section, we elaborate on the sampling techniques in post-quantum algorithms.

## 2.1 Sampling of errors in post-quantum solutions

The future of Quantum computing is a great threat to the available cryptographic infrastructure. Taking this threat into mind NIST has started an open competition for the possible solutions for public key cryptographic(PKC) algorithms and signature schemes. Most of the finalists in the available procedures are based on the LWE problem proposed by Regev in [15]. This problem refers to the idea of introducing some error with some randomized sampling. In every design of PKC which is based on the LWE problem, there is a provision for the randomization of the public key or secret key. This process is based on the various classical rejection sampling techniques [16–18]. The trade-off between the parameters of rejection sampling and security is interesting for every design. KYBER and Dilithium are the two finalists in the list of NIST final rounds of public key encryption decryption algorithm and digital signature algorithm [19, 20]. Both of the algorithms are based on hard problem learning with errors.

## 2.2 Rejection sampling in Dilithium

Dilithium is a finalist algorithm in the NIST competition for post-quantum cryptography. It is an algorithm for digital signature and is based on the hard lattice problem named learning with errors. The core idea of this problem is to solve a linear system of equations having intentionally inserted errors in their coefficients.

**Dilithium Algorithm**

KeyGen:<br>
$\mathbf{A} \leftarrow R_q^{k\times l}$<br>
$(s_1 s_2) \leftarrow S_\eta^l \times S_\eta^k$<br>
$t = \mathbf{A}s_1 + s_2$<br>
Return $(pk = \mathbf{A}, t)$, $sk = (\mathbf{A}, t, s_1, s_2)$

Sign(sk,M):<br>
Initialize $\mathbf{z} = \perp$<br>
while $\mathbf{z} = \perp$<br>
  $y \leftarrow S_{\gamma_1}^l$<br>
  $w_1 = HighBits(\mathbf{A}y, 2\gamma_2)$<br>
  $\hat{c} \in \{0,1\}^{256} = H(M||w_1)$<br>
  $c \in B_\tau = SampleInBall(\hat{c})$<br>
  $z = y + cs_1$<br>
  if $||z||_\infty \leq \gamma_1 - \beta \, or \, ||LowBis(\mathbf{A}y - c\mathbf{s}_2, 2\gamma_2)||_\infty \geq \gamma_2 - \beta$ then $\mathbf{z} = \perp$<br>
return $\sigma = (\mathbf{z}, c)$

Verification$(pk, M, \sigma = (\mathbf{z}, \hat{c}))$:<br>
$\mathbf{w}_1^{'} = HighBits(\mathbf{A}\mathbf{z} - c\mathbf{t}, 2\gamma_2)$<br>
if return $[||z||_\infty < \gamma_2 - \beta]$ and $[\hat{c} = H(M||w_1')]$

A short explanation of the algorithm of Dilithium is presented in the above block. Here $A$ represents a matrix of size $k \times l$, where each of the elements $a_{ij}$ is a polynomial taken from the ring $R_q$, $\frac{\mathbb{Z}_q[x]}{<x^n-1>}$, where $n = 256$ and $q = 8380417$ for every version of Dilthium algorithm. The

| NIST Security Level | q | d | $\tau$ | $\gamma_1$ | $\gamma_2$ | (k,l) | $\eta$ | $\beta$ | $\omega$ |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 8380417 | 13 | 39 | $2^{17}$ | (q-1)/88 | (4,4) | 2 | 78 | 80 |
| 3 | 8380417 | 13 | 49 | $2^{19}$ | (q-1)/32 | (6,5) | 4 | 196 | 55 |
| 5 | 8380417 | 13 | 60 | $2^{19}$ | (q-1)/32 | (8,7) | 2 | 120 | 75 |

Table 1: Dilithium security parameters

key generation part of the algorithm uses sampling of the public key and secret key from the output of the hash function. The matrix $A$ is sampled from the output of the extendable output function(XOF) and it is part of the public as well as secret key. $(s_1, s_2)$ are part of the secret key and they are sampled from the XOF function denoted as $S_\eta^l$ and $S_\eta^k$. Secret keys $s_1$ and $s_2$ are elements of the same ring $R_q$. Other parameters for security leveling are presented in table 1.

The function ExpandS computes each of the $l + k$ polynomials in $s_1$ and $s_2$ independently. For the $i - th$ polynomial, $0 \leq i \leq l + k$, it absorbs the 64 bytes of $\rho$ concatenated with 2 bytes representing $i$ in little-endian byte order into SHAKE-256. Then the output bytes are used to create a sequence of uniformly random positive numbers in the range $\{0, ..., 2\eta\}$ by performing rejection sampling. Concretely, the lower and upper four bits of each output byte are interpreted as two integers in $\{0, ..., 15\}$. Then, in the case of $\eta = 2$, an integer is accepted when it is less than 15 and then reduced modulo 5. In the case of $\eta = 4$, an integer is accepted when it is less than $2\eta + 1 = 9$. Finally, the polynomial coefficients are obtained by subtracting the positive numbers from $\eta$.

Now we present a discussion on the sampling technique of $s_1$ and $s_2$ for three versions of Dilithium named security levels 2,3 and 5. Here we frame some generalized Boolean functions w.r.t these sampling techniques.

**Dilithium-2:**
This is the first version of Dilithium and here the security parameter $\eta$ is selected as 2, therefore the sampling function for this version is $S_2$. The function ExpandS, used for generating the secret vectors in key generation, maps a seed $\rho'$ to $(s_1, s_2) \in S_\eta^l \times S_\eta^k$. The value of $k$ and $l$ is 4, therefore the sampling of vectors $s_1$ and $s_2$ independently with the help of XOF uniform distribution. In total there is $l + k$ polynomial need to be sampled from the output of XOF therefore for this version, eight polynomials have to be sampled from index $0 \leq i \leq 7$. Since $rho'$ is fixed 64 bytes for all input of XOF function, therefore, we assume this to be known and constant, therefore to generate each polynomial or 256 integer sequence we feed only 2 bytes representing $i$ in little-endian byte order into SHAKE-256. For the sake of convenience in the case of version-2, where $0 \leq i \leq 7$, the input vector of SHAKE-256 can be taken as three bits, and if we take little-endian order then they are of sixteen bits. The mapping without little-endian byte order input can be treated as a Boolean function $f : \mathbb{Z}_2^3 \mapsto \mathbb{Z}_5$. The mapping with little-endian order can be treated as a generalized Boolean function $f : \mathbb{Z}_2^{16} \mapsto \mathbb{Z}_5$.

**Dilithium-3:**
This is the second version of Dilithium and here $\eta$ is selected as 4, therefore the sampling function for this version is $S_4$. The function ExpandS, used for generating the secret vectors in key generation, maps a seed $\rho'$ to $(s_1, s_2) \in S_4^6 \times S_4^5$.

Similar to the explanation in Dilithium-2, the mapping without little-endian byte order input can be treated as a Boolean function $f : \mathbb{Z}_2^4 \mapsto \mathbb{Z}_9$. The mapping with little-endian order can be treated as a Boolean function $f : \mathbb{Z}_2^{16} \mapsto \mathbb{Z}_9$.

**Dilithium-5:**
This is the final version of Dilithium and security parameter $\eta$ is 2, therefore the sampling function for this version is $S_2$. The function ExpandS, used for generating the secret vectors in key generation, maps a seed $\rho'$ to $(s_1, s_2) \in S_2^8 \times S_2^7$.

Similar to the explanation in Dilithium-2, the mapping without little-endian byte order input can be treated as a generalized Boolean function $f : \mathbb{Z}_2^4 \mapsto \mathbb{Z}_5$. The mapping with little-endian order can be treated as a generalized Boolean function $f : \mathbb{Z}_2^{16} \mapsto \mathbb{Z}_5$.

Now we explore the balancedness property of a generalized Boolean function coined from the rejection sampling *SampleInBall* in all three cases of the Dilithium- $2, 3$ and 5. However, unbalanced generalized Boolean functions are only bent functions and they achieve the highest nonlinearity. It refers to the idea of nonlinearity analysis of all these Boolean functions for cryptanalysis of this algorithm.

# 3 XOF function and neutrosophic Boolean function

XOF or extendable output functions are hash functions where we can change the length of the output based on our choice. There are many places in PQC algorithms Kyber and Dilithium where XOF is used for random sampling. The technique of sampling mentioned in section 2.2 is based on the uniformity of output of XOF functions. therefore, the efficiency of PQC algorithms depends on the efficient implementation of these XOF functions.

The performance of rejection sampling for the selection of vectors $s_1$ and $s_2$ depends on the SHAKE256. In every version of the Dilithium, the rejection sampling *SampleInBall* can be treated as a neutrosophic Boolean function. Here in the Dilithium algorithm steps 9 is for the sampling of a vector $c$. In the final proposed algorithm from [20], this $c$ is sampled from another vector $\hat{c}$. The function named *SampleInBall*$(\hat{c})$ gives the output $c$. After signing the message $\hat{c}$ becomes the part of the signature and $c$ will be safe as a secret. The proposed algorithm fixed security parameters for the sampling of $c$. The condition is directly related to frequency distribution in a vector of length 256 having entry $\{1, -1, 0\}$. More specifically the security parameter $\tau$ is the total count of 1 and $-1$ in $c$. The value of $\tau$ in all three versions of Dilithium is presented in figure 1. We made an analogy of Boolean function to *SampleInBall*$(\hat{c})$ function based on the distribution of $1, -1$, and 0 in the outcome of that function which is $c$. Since the outcomes of the *SampleInBall* function are of 256 length, therefore, we can treat them as a truth table of a Boolean function, in the next definition we present the idea to analyze the *SampleInBall* function as a Boolean function, However, The inputs are not of the size of eight bits but the length of output $c$ is 256, therefore, we have taken it as an eight-bit Boolean function.

**Definition 3.1.** A *SampleInBall* is a generalized Boolean function $f$ from $\mathbb{Z}_2^8$ to $\mathbb{Z}_3$.

Now the Walsh transformation of Boolean function $f$ from $\mathbb{Z}_2^8$ to $\mathbb{Z}_3$ can be defined as,

**Definition 3.2.** The Walsh Transformation of SampleInBall is defined as

$$\mathcal{WT}_f(a) = \sum_{x \in \mathbb{Z}_2^8} (-1)^{<x,a>} (\omega)^{f(x)}, \tag{1}$$

where $\omega$ is $3^{rd}$ root of unity which is $\frac{-1+\sqrt{3}i}{2}$.

A generalized Boolean $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$ function is called balanced if the cardinality of every sets $\{x : f(x) = i\}$ is $3^{n-1}$ for every $0 \leq i \leq 2$. The flat Walsh spectrum i.e. $|\mathcal{WT}_f(a)| = 2^n$ for all $a \in \mathbb{Z}_2^n$ is the condition for $f$ to be bent function. This can be generalized for a Boolean function $f$ from $\mathbb{Z}_q^n$ to $\mathbb{Z}_3$ for any prime $q$.

In the case of a generalized Boolean function $f$ from $\mathbb{Z}_q^n$ to $\mathbb{Z}_q$, where $q$ is a prime number, the generalized Walsh transformation is defined as

$$\mathcal{WT}_f(a) = \sum_{x \in \mathbb{Z}_q^n} (\zeta)^{f(x)+a.x}, \tag{2}$$

where $\zeta$ is $q^{th}$ root of unity. Following is an example of a generalized Boolean function $f : \mathbb{Z}_3^2 \mapsto \mathbb{Z}_3$ with its Walsh spectrum

| $x$ | (0,0) | (1,0) | (1,1) | (-1,1) | (0,-1) | (-1,0) | (-1,-1) | (1,0) | (0,1) |
|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1 | -1 | -1 | -1 | -1 | 0 | 0 | 0 | 0 |
| $\lvert\mathcal{WT}_f(x)\rvert$ | $2.9\overline{9}$ | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Table 2: A Boolean function $f : \mathbb{Z}_3^2 \mapsto \mathbb{Z}_3$

A generalized Boolean $f : \mathbb{Z}_q^n \mapsto \mathbb{Z}_q$ function is called balanced if the cardinality of every sets $\{x : f(x) = i\}$ is $q^{n-1}$ for every $0 \leq i \leq q-1$. Now in the next section, we present the results for the existence of bent functions with various other generalized properties of neutrobalancedness.

## 3.1 Classicalbalanced bent function

**Definition 3.3.** A function $f : \mathbb{Z}_q^n \mapsto \mathbb{Z}_3$ is said to be a Classicalbalanced function if it takes an equal number of 1's, 0's, and -1's.

**Theorem 3.4.** *A Classicalbalanced function $f : \mathbb{Z}_q^n \mapsto \mathbb{Z}_q$ never be a bent function.*

*Proof.* From the definition3.3 we know that a function $f : \mathbb{Z}_q^n \mapsto \mathbb{Z}_q$ is Classicalbalanced if and only if

$$\sum_{x \in \mathbb{Z}_q^n, f(x)=k} 1 = q^{n-1}, \tag{3}$$

for all $0 \leq k < q$. Using (3) it is proved in [26], that the Classicalbalanced function never be bent function. Now in the next section, we explore other generalizations of balancedness and bent functions. $\square$

In the case of a neutrosophic Boolean function $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$, it is obvious to say that no classical balanced function exists.

## 3.2 Neutrobalanced bent function

**Definition 3.5.** A function is said to be a Neutrobalanced function if the number of 1's, 0's, and -1's are not the same but exactly two of them are equal.

**Theorem 3.6.** *The Neutrobalanced bent function $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$ does not exist for odd $n$.*

*Proof.* Let $a, b$ and $c$ be the cardinality of $\{x \in \mathbb{Z}_2^n : f(x) = 1 = \alpha\}, \{x \in \mathbb{Z}_2^n : f(x) = -1 = \beta\}$ and $\{x \in \mathbb{Z}_2^n : f(x) = 0 = \gamma\}$ respectively. We can observe that

$$a + b + c = |\mathbb{Z}_2^n| = 2^n \tag{4}$$

Here $f$ is Neutrobalanced therefore any of two from $a, b$ and $c$ are always equal therefor taking symbolic equality of $a$ and $c$, we can write that

$$2a + b = 2^n. \tag{5}$$

From (1) the Walsh transformation of a Neutrobalanced function at $t \in \mathbb{Z}_3^n$ can be written as

$$\mathcal{WT}_f(t) = \quad \textstyle\sum_{x \in \mathbb{Z}_3^n} \omega^{f(x)}(-1)^{t.x}.$$

Walsh transformation at $t = 0$

$$\mathcal{WT}_f(0) = \sum_{x \in \mathbb{Z}_3^n} \omega^{f(x)} = a\omega^{\alpha} + b\omega^{\beta} + c\omega^{\gamma} \tag{6}$$

where $\omega$ is the $3^{rd}$ root of unity. Now since $f$ is Neutrobalanced and bent after taking the symbolic equality of $a$ and $c$

$$\mathcal{WT}_f(0) = a(\omega^{\alpha} + \omega^{\beta}) + b\omega^{\gamma} \tag{7}$$

and

$$|\mathcal{WT}_f(0)|^2 = 3^n. \tag{8}$$

Now using (7) and (8), we can characterize the necessary condition for the Neutrobalanced bent function in the Diophantine equations. For all values of $(\alpha, \beta, \gamma)$, (7) and (8) implies that

$$\begin{aligned} |a(1+\omega) + b\omega^2| &= 3^{\frac{n}{2}} \\ \implies (b-a)^2 &= 3^n \end{aligned} \tag{9}$$

Now from (5) and (9)

$$a = \frac{3^n - \sqrt{3^n}}{3} \ \textit{and}\ b = \frac{3^n + 2\sqrt{3^n}}{3}. \tag{10}$$

$a$ and $b$ are irrational numbers for odd values of $n$. Hence the theorem is proved. □

It is interesting to see the construction of symmetric and rotational symmetric Neutrobalanced bent Boolean functions for even values of $n$. In [26], for $n = 2$, the list of eight symmetric and rotational symmetric bent Boolean functions is presented. From the available list of bent functions, we can say that $f_5, f_6, f_7$ and $f_8$ presented in [26] are neutrobalanced symmetric and rotational bent Boolean functions. Their Algebraic Normal Form is presented as

**Neutrobalanced symmetric and rotational symmetric bent Boolean function** $f : \mathbb{Z}_3^2 \mapsto \mathbb{Z}_3$

$$f(x_1, x_2) = 1 - x_1x_2,$$
$$f(x_1, x_2) = 1 + x_1 + x_2 + x_1x_2,$$
$$f(x_1, x_2) = x_1 + x_2 - x_1x_2 - 1,$$
$$f(x_1, x_2) = x_1x_2 - 1.$$

### 3.3 Antibalanced bent function

**Definition 3.7.** A function is said to be an Antibalanced function if the number of 1's, 0's, and -1's are all distinct from each other.

**Theorem 3.8.** *There does not exist any Antibalanced bent function $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$.*

*Proof.* Walsh transformation of the Antibalanced function $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$ at $t = 0$

$$\begin{aligned} \mathcal{WT}_f(0) &= \sum_{x \in \mathbb{Z}_3^n} \omega^{f(x)} \\ &= a\omega^\alpha + b\omega^\beta + c\omega^\gamma \end{aligned} \tag{11}$$

where $\omega$ is the $3^{rd}$ root of unity. Now for all possible values of $(\alpha, \beta, \gamma)$,11 implies that

$$\mathcal{WT}_f(0) = \frac{2a + b + c + i\sqrt{3}(b - c)}{2}. \tag{12}$$

Therefore the Walsh transformation of bent Antibalanced function satisfy

$$\begin{aligned} 4|\mathcal{WT}_f(0)| &= (2a + b + c)^2 + 3(b - c)^2 \\ &\mathit{or} \\ (a + 3^n)^2 + 3(b - c)^2 &= 4(3^n). \end{aligned} \tag{13}$$

Using (4) and (13) we can get a quadratic equation in $b$,

$$b^2 - (3^n + c)b + 3^{2n} + c^2 - (3^n)c - 3^n = 0.$$

After solving this quadratic equation for $b$, we get

$$b = \frac{(3^n + c) \pm i\sqrt{3^{2n+1} + 3c^2 + 2c(3^n) + 4(3^n)}}{2}. \tag{14}$$

Here $b$ is a positive integer, therefore,

$$3^{2n+1} + 3c^2 + 2c(3^n) + 4(3^n) = 0. \tag{15}$$

Now solving (15), for the integer values of $c$, we found that the real roots do not exist. Hence no Anitibalanced bent function $f : \mathbb{Z}_2^n \mapsto \mathbb{Z}_3$ exist. $\square$

# 4 Conclusion: $SampleInnBall$ function and Neutrobalanced bent function

It is interesting to use the observation in the previous section to characterize the SampleInBall function in Dilithium based on the security parameter $\tau$. In all three versions of Dilithium, the value of $\tau$ is $39, 49$, and $60$. Therefore it is straightforward that in the security level, 2 and 3, the SampleInBall function never be a Neutrobalanced bent function and achieves the highest nonlinearity. Consequently SampleInBall for $\tau = 39, 49$ is an Antibalanced Boolean function, and from theorem 3.8, no Antibalanced Bent function exists, therefore SampleInBall has not achieved the highest nonlinearity.

In the case of Dilithium of security level, 5 the value of $\tau$ is 60, therefore the sampling may or may not be highly nonlinear based on the frequency distribution of 1 and $-1$ in $c$. In case of $\tau = 60$ SampleInBall is Neutrobalanced if $\#1 = 30 = \# - 1$. Now from theorem 3.6, if SampleInBall is bent then it is sampled through an even number of bits only to achieve bentness. SampleInBall for $\tau = 60$ is Antibalanced if $\#1 \neq \# - 1$ and from theorem 3.8, SampleInBall is not a bent function.

We have done experiments through the available implementation of Dilithium in [27] and found that out of 122 test vectors, $\hat{c}$ in Dilithium-5 only 22 are neutrobalanced therefore they are not with the highest nonlinearity. It would be interesting to study another rejection sampling with respect to the idea of corresponding generalized Boolean function and neutrosphy.

# References


[1] Project Gutenberg's An Investigation of the Laws of Thought, by George Boole

[2] Mathematics of the 19th century, Mathematical logic Algebra Number Theory Probability theory, A.N Kolmogorov and A.P. Yushkevich, Springer Basel AG

[3] C. Carlet. Boolean Functions for Cryptography and Coding Theory. Cambridge University Press, Cambridge, 2021.

[4] J. F. Dillon. Elementary Hadamard difference sets. PhD thesis, Univ. of Maryland, 1974.

[5] S. Kavut. Boolean functions with excellent cryptographic properties in autocorrelation and walsh spectra. PhD thesis, Middle East Technical University, 2008.

[6] T. W. Cusick and P. Stanica (Year). Cryptographic Boolean Functions and Applications. Boca Raton, FL: CRC Press.

[7] O.S Rothaus, On “bent” functions, Journal of Combinatorial Theory, Series A, Volume 20, Issue 3,1976,Pages 300-305, ISSN 0097-3165, https://doi.org/10.1016/0097-3165(76)90024-8

[8] N. T. Coutois and Wili Meier, ”Algebraic attacks on stream ciohers with linear feedback” in Advances in Cryptology-EROCRYPT 2003 LNCS 2656, Berlin:Springer-verlag, pp. 346-359, 2003.

[9] C. Carlet, D. Dalai, K. Kupta and S. Maitra, ”Algebraic immunity for cryptographically significant Boolean functions: Analysisi and Construction”, IEEE Transactions on Information Theory, vol. IT-52, no. 7, pp. 3105-3121, 2006.

[10] Nicolae Tăndăreanu, On generalized Boolean functions, Discrete Mathematics, Volume 34, Issue 3, 1981, Pages 293-299, ISSN 0012-365X, https://doi.org/10.1016/0012-365X(81)90007-8.

[11] Martinsen, T., Meidl, W., Pott, A., Stănică, P. (2018). On Symmetry and Differential Properties of Generalized Boolean Functions. In: Budaghyan, L., Rodríguez-Henríquez, F. (eds) Arithmetic of Finite Fields. WAIFI 2018. Lecture Notes in Computer Science, vol 11321. Springer, Cham. https://doi.org/10.1007/978-3-030-05153-2-11

[12] C. Carlet, Vectorial Boolean functions for cryptography, In: Y. Crama, P. Hammer, Boolean Methods and Models, Cambridge Univ. Press, Cambridge. Available: http://www-roc.inria.fr/secret/Claude.Carlet/pubs.html

[13] Pante Stanica, Thor Martinsen, Octal Bent Generalized Boolean Functions, arXiv:1102.4812v2

[14] Smarandache, F., A Unifying Field in Logics: Neutrosophic Logic, Neutrosophy, Neutrosophic Set, Neutrosophic Probability, American Research Press, Rehoboth, 2003.

[15] Regev, O. 2009. On lattices, learning with errors, random linear codes, and cryptography. J. ACM 56, 6, Article 34 (September 2009), 40 pages. DOI = 10.1145/1568318.1568324 http://doi.acm.org/10.1145/1568318.1568324

[16] Decru, T., Panny, L., Vercauteren, F. (2019). Faster SeaSign Signatures Through Improved Rejection Sampling. In: Ding, J., Steinwandt, R. (eds) Post-Quantum Cryptography. PQCrypto 2019. Lecture Notes in Computer Science(), vol 11505. Springer, Cham. https://doi.org/10.1007/978-3-030-25510-7-15

[17] Zhongxiang Zheng, Anyu Wang, Lingyue Qin, "Rejection Sampling Revisit: How to Choose Parameters in Lattice-Based Signature", Mathematical Problems in Engineering, vol. 2021, Article ID 9948618, 12 pages, 2021. https://doi.org/10.1155/2021/9948618

[18] S. Agrawal, D. Stehlé, and A. Yadav. Round-optimal lattice-based threshold signatures, revisited. In ICALP, 2022.

[19] CRYSTALS-Kyber (version 3.02) – Submission to round 3 of the NIST post-quantum project. Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Specification document (update from August 2021). 2021-08-04

[20] CRYSTALS-Dilithium – Algorithm Specifications and Supporting Documentation (Version 3.1) Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Specification document (update from February 2021). 2021-02-08

[21] Vadiraja G. R, Pandey, S. K. ., P. R. Mishra, Poojary, P. . (2022). ClassicalBalanced, AntiBalanced and NeutroBalanced functions. Neutrosophic Sets and Systems, 48, 386-398. Retrieved from https://fs.unm.edu/NSS2/index.php/111/article/view/2111

[22] Smarandache, F. Introduction to NeutroAlgebraic Structures and AntiAlgebraic Structures (revisited). Neutrosophic Sets and Systems, 31, 2020; pp.1-16.

[23] Smarandache, Florentin, Introduction to NeutroAlgebraic Structures and AntiAlgebraic Structures, in his book "Advances of Standard and Nonstandard Neutrosophic Theories", Pons Ed., Brussels, European Union, 2019

[24] Smarandache, F., Neutrosophy. / Neutrosophic Probability, Set, and Logic, ProQuest Information & Learning, Ann Arbor, Michigan, USA, 105 p., 1998.

[25] Smarandache, F. NeutroAlgebra is a Generalization of Partial Algebra, International Journal of Neutrosophic Science, 2(1), 2020; pp. 08-17.

[26] Count and cryptographic properties of a generalized symmetric Boolean function, Shashi Kant Pandey, P.R. Mishra, B.K. Dass; Italian Journal of Pure and Applied Mathematics, 37, 2016

[27] https://github.com/GiacomoPope/dilithium-py