A Key Scheduling Algorithm Based on Dynamic Quasigroup String Transformation and All-Or-Nothing Key Derivation Function

Abdulkadir Hassan Disina, Sapiee Jamel, Muhammad Aamir, Zahraddeen A. Pindar,
Mustafa Mat Deris and Kamaruddin Malik Mohamad
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia,
86400 Parit Raja, Batu Pahat, Johor, Malaysia.
academicdisina@gmail.com

***Abstract*—Cryptographic ciphers depend on how quickly the key affects the output of the ciphers (ciphertext). Keys are traditionally generated from small size input (Seed) to a bigger size random key. Key scheduling algorithm (KSA) is the mechanism that generates and schedules all sub-keys for each round of encryption. Researches have suggested that sub-keys should be generated separately to avoid related-key attack. Similarly, the key space should be disproportionately large to resist any attack meant for secret keys. To archive that, some algorithms adopt the use of matrixes such as quasigroup, Hybrid cubes and substitution box (S-box) to generate the encryption keys. Quasigroup has other algebraic property called "Isotopism", which literally means Different quasigroups that has the same order of elements but different arrangements. This paper proposed a Dynamic Key Scheduling Algorithm (KSA) using Isotope of a quasigroup as the dynamic substitution table. The proposed algorithm is a modification and upgrade to All-or-nothing Key Derivation Function (AKDF). To minimize the complexity of the algorithm, a method of generating Isotope from a non-associative quasigroup using one permutation is achieved. To validate the findings, non-associativity of the generated isotopes has been tested and the generated isotopes appeared to be non-associative. Furthermore, the proposed KSA algorithm will be validated using the Randomness test proposed and recommended by NIST, Avalanche and Correlation Assessment test.**

Index Terms—Key Scheduling Algorithm; Quasigroup; Random Number Generator; String Transformation

I. INTRODUCTION

The modern-day cryptography is not only to provide confidentiality, but simultaneously authenticates and verifies the integrity of the message and the sender respectively (Authenticated encryption). That ability can also be attributed to cryptographic keys as most ciphers rely on the keys [1]. Key scheduling algorithm (KSA) is a cryptographic algorithm that generates and manages session keys for all the rounds of encryption and decryption. Researches were conducted to provide a powerful key to withstand related key attack and increase the difficulty to cryptanalyze and recover secret keys [2][3]. It leads to the use of matrices and groups to proportionately enlarge the key space so as to make a brute force attack harder. Hybrid cube, cubicle hybrid cube and encryption based on rotation of Magic cube are some of the recent encryption algorithms that are based on matrices [1] [4]. The key space depends solely on the size of the matrix or cube. Generating those matrices empirically requires a high-speed processing capacity which can be very costly to resource-constrain environments. However, static matrices can be vulnerable to attack, an Adversary may recover the encryption key by determining the exact matrix used. Cubicle Hybrid Cube proposed cube rotations to convert the static nature of Hybrid Cube to dynamic, thus increases the complexity of the algorithm [5]. This paper proposed a dynamic Key Scheduling Algorithm from a highly non-associative non-commutative quasigroup. The proposed algorithm is primitively based on All-Or-Nothing Key Derivation Function. The proposed algorithm uses the user-given key alongside predefined quasigroup to generate Isotope as a dynamic substitution table. A new method of generating isotope of non-associative quasigroup is achieved using one permutation. The non-associativity and non-commutativity of the generated isotope is analyzed. The output of the KSA algorithm will be analyzed based on correlation assessment, avalanche effect and NIST test suit.

II. PRELIMINARIES

A quasigroup is n × n matrix that contains a set of positive integers arranged in rows and columns of the matrix, such that each integer occurs once in each row and column [6][7].

Definition[8]: let Q = {a1,a2…….an} be a finite set of n elements. A quasigroup (Q,*) is a groupoid (Algebra with one binary operation) satisfying the law (∀u, v ∈Q) (∃! x, y ∈ Q) u*x=v & y*u=v.

However, shapeless quasigroup with non-associative property proved to be more useful in crypto systems. Therefore, the totally non-associative quasigroup of order 16 is adopted in this research paper from the work of Meyer [9].

A. Isotope

Isotopism refers to two or more quasigroups that has the same order but different arrangement of elements, in such a way that either of the quasigroup can be transformed to the other. Several methods of generating isotope have been developed over the years, some of those methods can be found in [10][11]. Isotope can be generated and used from existing quasigroup as a dynamic substitution table. The definition of Isotopism is given in the next heading.

Definition [12]**:** A quasigroup (Q,\) is said to be isotopic to

another quasigroup (Q,*) if and only if there are bijection α,β,γ from K onto Q such that γ (x * y) = α (x) • β (y) for each x, y ∈ K. Then the triple (α,β,γ) is called an Isotopism (Q,*) to (Q, •).

B. Non-Commutativity

The word commutative originated from "commute" which literally means to move things around. Commutative property is the ability to switch two operands with each other and produce the same result. Commutative quasigroups produces recognizable patterns that crypto system strives to avoid. On the other hand, non-commutative quasigroups play a huge role in crypto system in generating non-linear sequence, but generating non-commutative quasigroup reciprocates with computational cost. Therefore in this paper the non-commutative quasigroup of order 16 is adopted from the work of Mayer [9][13]. The non-commutative property can be verified by the use of the following definition.

Definition [14]: let Q be the quasigroup of order n with binary operation * and x, y ∈ Q such that x ≠ y. if x * y = y * x then the binary operation is commutative. And if y * x ≠ x * y then the quasigroup with binary operation (Q,*) is non-commutative.

C. Non-associativity

Associative property in a group means an operation between groups of quantities produce the same result as long as the order is the same. The word "Non-associative" means not necessarily associative. It produces a highly non-linear sequence if properly utilized. Therefore, this paper adopted a highly non-associative quasigroup for generating non-associative Isotope as a dynamic substitution table. The non-associativity has been tested for the generated isotopes. The following definition is used for the test [8][12][13].

Definition [9]: let Q be the quasigroup of order n with binary operation * and x, y *and z* ∈ Q such that x ≠ y ≠ z. if (x * y) * z = y * (x * z), then the binary operation is Associative. And if (x * y) * z ≠ y * (x * z), then the quasigroup with binary operation (Q,*) is non-associative.

D. Generating non-associative non-commutative quasigroup

Non-commutative, non-associative quasigroup plays a vital role in developing cryptographic primitives due to its unpredictable nature, but they are quite difficult to generate if computational cost is considered. There are many methods of generating shapeless quasigroups, such methods include but not limited to Using Feistel network and Non-affine complete mapping. Table 1 and Table 2 show the tabular representation of the mapping, interested readers should check Mayer for detail [8][9][13].

Table 1
θ and i ⊕ θ on the integer of group [8]

x	0	1	2	3	4	5	6	7
θ(x)	2	0	6	4	7	5	3	1
(i⊕ θ)(x)	2	1	4	7	3	0	5	6

The quasigroup is defined as: $x*y=\theta(x \oplus y) \oplus y$.

Table 2
Tabular representation of the mapping [8]

x	θ(x)	(i ⊕ θ)(x)
(0,0,0)	(0,1,0)	(0,1,0)
(0,0,1)	(0,0,0)	(0,0,1)
(0,1,0)	(1,1,0)	(1,0,0)
(0,1,1)	(1,0,0)	(1,1,1)
(1,0,0)	(1,1,1)	(0,1,1)
(1,0,1)	(1,0,1)	(0,0,0)
(1,1,0)	(0,1,1)	(1,0,1)
(1,1,1)	(0,0,1)	(1,1,0)

In a related development, another method of generating multiple quasigroups from existing one is achieved using "cyclic random permutation" (CRP). Eventually, generating the CRP from encryption key increases the complexity of the algorithm. Similarly, the structure of the quasigroup was not considered and that could be used to launch a quasigroup attack on the cipher [15].

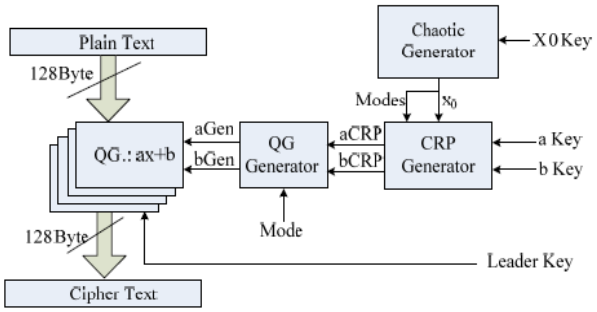

Figure 1: Cipher based on multiple quasigroups [15]

All these methods require to generate a permutation of order *n*, generating the permutation itself increases the complexity of the algorithm and the structure of the quasigroup depends solely on the nature of the permutation. To generate a shapeless quasigroup using the above-mentioned methods, a shapeless permutation is required. Therefore, this paper proposed a method of generating a shapeless quasigroup from predefined quasigroup. Each quasigroup of order *n* literally has other *n!* Quasigroups called Isotopes.

E. Quasigroup String transformation

Quasigroup String Transformation is an algebraic function used in cryptographic as a primitive that transform an input into a nonlinear formatted form, using a quasigroup as the substitution table [7] [11]. This technique is widely used in many cryptographic applications, such includes hash functions key scheduling algorithm and ciphers [15]. The diagram in Figure 2 visually describes a generic string transformation of a given input.

Definition [11]**:** given a quasigroup (Q,*) with elements {a1, a2, …..an} where ai ∈Q, i = 1, 2, to n. let l be a leader where l∈ Q. b={b1,b2….bn} is obtained.

$$Tr(a_1,a_2..a_n)=(b_1,b_2.b_n)=\begin{cases} l * a1(a2)\ j=1,\ldots.n \\ bj = bj-1 * aj * (aj+1)\ 1 \le j \le n \\ bn = bn-1 * an(l) \end{cases}$$

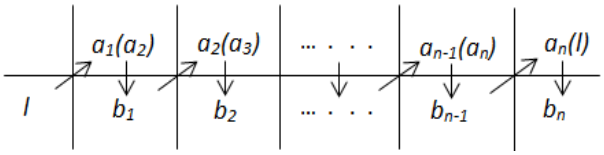

Figure 2: Graphical representation of generic *E* Transformation [16]

The technique has been transformed to forward and reverse, each round has different leader (*l*) to achieve significant avalanche effect [11].

F. Key Derivation Function

Key derivation function (KDF) is used to generate proportionately large cryptographic keys from small private string. KDFs used two main sub functions (extractor and expander) to transform the small input into cryptographically pseudorandom output. KDFs and KSAs works together to generate keys or have a lot in common. The following sub function describes the KSA.

Definition: let KDF be the function G: {0,1}s → {1,0}n using an operation that transform and expand short length user key {0,1}s to arbitrary length {1,0}n random string, where n > s.

G. Key Scheduling Algorithm

A cryptographic algorithm requires a mechanism that generates random encryption keys. The key is used to maintain the security (confidentiality and integrity) of data and information while on transit. The size and number of sub-keys depends on the design of the cipher. The key is either generated or extracted from a bunch of key material. Hybrid cubes is an example of KSAs that select encryption keys from already generated key space [1]. Some encryption keys are generated from small size user key using KDFs.

Definition: let KSA be the function F: $\{0,1\}^n \rightarrow \{1,0\}^n_1$ $\{1,0\}^n_2$,$\{1,0\}^n_3$, $\{1,0\}^n_s$ that generates or extract sub keys s from generated encryption key n. each sub key must be highly random and distinguishable from all sub keys.

Algebraic functions have been used to generate nonlinear variable length cryptographic keys through certain process. Conventional key derivation function has phases of processes, single phase concatenates the private string from the user with the predefined public string to generate the key [17][18]. On the other hand, two phases KDF become the improved version and to increase flexibility and randomness. There are two main sub-functions in the KDF; extractor and expander which works together to generate the cryptographic key [17]. The extractor generates the variables from private and predefined public string while the expander take input from the extractor and perform the expansion until the key has achieved the maximum number of bits required for the encryption [18]. To align the proposed algorithms to current standard, the key is generated in excess and then reduced to *128*, *192* and *256* at the last stage. The following section will explain the process.

III. PROPOSED METHODS

This paper proposed the use of one permutation to generate an isotope from existing non-associative non-commutative quasigroup. The user key determines the permutation that will be used to generate the isotope as a substitution table. It adopted the All-Or-Nothing Key Derivation Function (AKDF) as primitive to the proposed KSA. The proposed algorithm (KSA) takes arbitrary length input from user, process it and bring out a random encryption key(s). The graphic representation of the whole idea can be visualized in Figure 3.

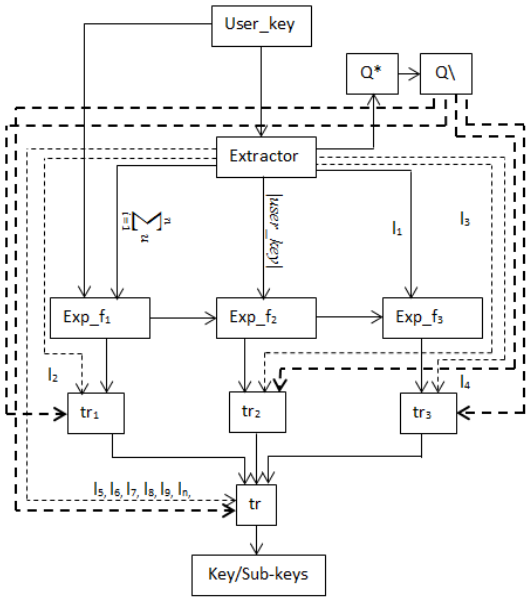

Figure 3: Dynamic Key Scheduling Algorithm

The proposed KSA accept *user_key* as the input from user that the random encryption key will be extracted from. the extractor therefore extract all the required variables for all the processes and instantations. The algorithm contains a pre-determined quasigroup (*Q*,*) known to the public and it is not part of Alice and Bob's secret key. It is used to generate the isotophe *(Q,\)* based on the user key. All other functions are adopted from All-Or-Nothing Key Derivation Function AKDF. Based on the AKDF, the expansion functions are responsible for algebraically dilating or expanding the input key from small size to big. the size and summation of the user private string are used to expand the key (*Exp_f1* and *Exp_f2*), any changes in the private string will result a measure change in the output key. if an attaker predicts all the private string exept for 1, the key cannot be recovered. the expantion functions are very sensitive to changes in the input key. The *Exp_f3* use both output of *Exp_f1* and *Exp_f2* as input to generate the third block. In this function, quasigroup string transformation is used for the expansion. The use of summation and size of the private string from the user play a vital role in detecting any changes. With this algorithm, adversary cannot predict the initial key by observing the behaviour of the ciphertext.

The transformation function *tr1*, *tr2* and *tr3* are to transform the tree output blocks from expansion function seperatly while *tr* is the last transformation function that puts all the three blocks together and transform them to a random string to ensure security. the sub-keys will be produced from the

output of *tr* at the last stage.

A. Description of the proposed KSA

In this section, instantiation of all the function and variables have been created to demonstrate the process. The functions are, Extraction, Expansion and Transformation. All the sub-functions of this algorithm will be explained with some examples in the next section.

Algorithm DKSA: $\{0,1\}^n \rightarrow \{1,0\}_1^n \{1,0\}_2^n, \{1,0\}_s^n$

1. $Q,\backslash = \pi(Q,*)$
2. $exp_{f\text{-}1} \leftarrow (x_i = \text{size} + u_i \bmod n)$
3. $exp_{f\text{-}2} \leftarrow y_j = \text{sum} + p_j \bmod n$
4. $exp_{f\text{-}3} \leftarrow T_A(b)$.
5. $tr1 \leftarrow T_{E,D}(l_2(p))$.
6. $tr2 \leftarrow T_{E,D}((l_3 y))$.
7. $tr3 \leftarrow T_{E,D}((l_4 a))$.
8. $tr \leftarrow T_{E,D}(l_n(l_6(l_5(x))))$.

return $\{1,0\}_1^n, \{1,0\}_2^n, \{1,0\}_3^n, \{1,0\}_c^n$

B. Dynamic Quasigroup (Q,|)

A method of generating quasigroup from existing quasigroup is proposed. Each row and column of a quasigroup of order *n* is a permutation of order *n* and each quasigroup of order *n* has $n \times 2$ permutations (rows and columns) of order *n*. therefore in this paper a method of generating quasigroup from a pre-defined quasigroup using single permutation is achieved. The permutation is selected from the predefined quasigroup based on the user key. The shapeless permutation will be used to generate a dynamic quasigroup with 100% commutativity and associativity inheritance. The following definition describes the proposed method.

Definition: Let (Q,\) be the Isotope of non-associative non-commutative quasigroup(Q,*). Such that, (Q,\) = π(Q,*).where π is a permutation in (Q,*).

Lemma: the quasigroups has direct mapping with each other ((Q,*) → (Q,\)). Each element x in quasigroup (Q*) mapped directly to element z in (Q,\). The isotope inherits the shape of the parent quasigroup.

Proof: Let $x*y$ be a binary operation in (Q,*). where π = {0,C,6,7,1,8,F,E,A,5,3,D,4,9,2,B} and (Q,\) = π(Q,*(x*y)). The quasigroup (Q,*) and (Q,\) has a direct mapping to each other. (Q,\(x*y))= π(Q,*(x*y)).

C. Generating the Dynamic Quasigroup

The dynamic quasigroup is generated from a predefined quasigroup of order 16 based on the user inputs as explained earlier. The following definition and example demonstrate the process.

Definition: Let (Q,*) be the non-associative non-commutative quasigroup of order n.

i=sum % order of the quasigroup (Q,*)
π = {a1, a2….an}
a1=x*y1
a2= x*y2
an= x*yn
x is constant and y is from 0 to n.

Where x and y are locations in quasigroup (Q,*)
j=0 to15
$Q,| = \pi(x_i, y_j)$

Example

From the predefined non-associative non-commutative quasigroup of order 16, the isotope is generated. The example below, describes the process in details.

x=36 % 16
$\pi = \{a_1, a_2 \ldots a_n\}$
$a_1 = 4*0 = 0$
$a_2 = 4*1 = c$
$a_{16} = 4*16 = B$
π = {0,C,6,7,1,8,F,E,A,5,3,D,4,9,2,B}
$Q,| = \pi(x,y)$

The generated isotope is expected to inherit all the properties of the parent quasigroup.

IV. ANALYSIS AND RESULT

In this paper, the method of generating shapeless quasigroup from existing quasigroup with complete inheritance has been tested. It has been analyzed to verify whether the generated quasigroup has inherited the non-commutative and non-associative property from the parent quasigroup or not. Similarly, the final output of the algorithm (encryption keys) has also been tested to verify the correlation and avalanche effect between all sub keys. The randomness of the generated key(s) is also tested using the randomness test suit proposed by National Institute for Standard and Technology (NIST).

A. Analysis of the quasigroup

In this section, the properties of the pre-defined and generated quasigroups are analyzed to measure the associativity and commutativity. For the quasigroup of order *16*, there are *12*0 instances to compare the commutativity of *x*y* with *y*x*. the favorable result is to obtain equal amount of instances where $x*y \neq y*x$ in both predefined quasigroup and the isotope. The adopted quasigroup is first examined. It turned out to be only eight (8) instances appears to be commutative. Almost all the permutations in the predefined quasigroup have been used to create an isotope. All generated isotopes inherited the same commutativity from the parent quasigroup as presumed.

B. Test data

In this test, 3 different sizes of encryption keys (128, 256 and 512) were generated and the following result is obtained from the test. Each key has 3 other sub keys (Sub_key_1, Sub_key_2 Sub_key_3 Sub_key_4.). Table 3, Table 4, Table 5 and Table 6 contains input, output size and the output of each generated keys. Each set of keys are denoted with an alphabet (a, b, c and d), While each sub key is numbered from 1 to 4.

Table 3
Generated 128 bit keys

Keys ID	Input: 12345678	Output Size: 128
Sub_key_1a	A B 9 7 4 5 8 0 6 B 7 3 4 F 8 0	
Sub_key_2a	9 8 B 7 B E 9 7 1 A 5 4 F B 8 4	
Sub_key_3a	0 E B 0 D F E A 2 2 C 9 7 D 4 A	
Sub_key_4a	F D 4 4 5 9 E 8 4 3 2 B 4 6 6 E	

Table 4
Generated 128 bit keys

Output keys	Input: 0234567	Output Size: 128
Sub_key_1b	D 6 9 9 9 B 1 6 B 3 3 F D 2 0 7	
Sub_key_2b	E 2 9 2 7 9 6 A 9 1 B D 5 3 F 8	
Sub_key_3b	9 3 B E 2 1 5 4 F 9 5 B 9 2 1 8	
Sub_key_4b	B 1 2 D E 5 6 B 3 0 6 E B 0 9 3	

Table 5
Generated 256 bit keys

Output keys	Input: 12345679	Output Size: 256
Sub_key_1c	E 2 3 C B 4 0 A A F 1 2 2 0 E A 0 9 7 3 F A C 5 6 B 7 E 8 F 0 A	
Sub_key_2c	C 1 B 0 E A F 0 2 A 7 B 5 2 9 9 8 5 6 4 E C C 5 0 7 E 9 2 2 4 B	
Sub_key_3c	0 B F 9 1 7 A 9 A E C 4 D 5 2 4 6 E 4 5 9 2 9 B 3 6 5 F 5 2 1 2	
Sub_key_4c	E 6 2 D 1 D 3 6 1 F E 3 9 6 0 9 1 4 1 C B 6 7 4 3 6 F 0 F E 9 F	

Table 6
Generated 512 bit keys

Output keys	Input: 12345679	Output Size: 512
Sub_key_1d	0 9 3 5 9 4 2 8 3 3 9 8 A D 1 3 C 4 9 9 4 6 D 9 E 9 0 E 9 A 4 3 E 8 9 E E 6 E 8 F 3 9 2 2 5 1 F 7 9 6 9 2 9 A D 0 1 7 2 E 7 E C	
ub_key_2d	C 7 1 6 2 7 B 1 F 7 2 5 D D 6 E 9 0 1 8 6 E 7 E F 8 E E D 0 9 7 0 C F C 7 4 8 F 7 5 C A 7 D 3 0 4 B 6 6 D 9 2 E 1 1 D 4 4 A 0 C	
Sub_key_3d	7 5 C E 9 A A 5 9 D E B 5 3 9 5 0 A 1 4 2 2 4 D F 4 6 2 9 4 A 7 1 5 C B 8 4 8 0 A 4 C 1 4 3 9 C 8 7 D 3 A 0 C 3 8 2 D B 6 7 A 0	
Sub_key_4d	8 7 6 7 E B A A C 5 3 C E 9 9 8 4 C 3 1 D 0 D 7 C 6 4 A 7 2 D D F 4 F 9 8 3 5 F A 6 B C 6 0 0 D 1 1 8 5 2 2 E 9 E A 1 9 0 B 6 C	

C. Correlation Assessment

This test is to examine how each sub key relates to other sub keys. In this test, the favorable result is usually close to *0* (*< 1*) and if the result is close to *1* then the elements that can be used to predict other sub keys exist. Any predictable pattern found between sub keys could be used by crypto analyst to launch an attack. Two set of keys are tested in this test and all the *4* sub keys are being compared with each other to determine whether they are related or not. The test require two variables (*x* and *y*) and for each comparison, one of the operant is *x* and the other is considered as *y* [19]. The obtained favorable result of this test can be verified from Table 7 and Table 8.

Table 7
Correlation between one set of sub keys

x	y	Correlation
Sub_key_1a	Sub_key_2a	0.268
Sub_key_1a	Sub_key_3a	-0.080
Sub_key_1a	Sub_key_4a	-0.116
Sub_key_2a	Sub_key_3a	0.279
Sub_key_2a	Sub_key_4a	-0.132
Sub_key_3a	Sub_key_4a	0.204

Table 8
Correlation between different sub keys

x	y	Correlation
Sub_key_1d	Sub_key_2d	0.022
Sub_key_1d	Sub_key_3d	-0.074
Sub_key_1d	Sub_key_4d	0.067
Sub_key_2d	Sub_key_3d	-0.080
Sub_key_2d	Sub_key_4d	-0.001
Sub_key_3d	Sub_key_4d	0.098

D. Avalanche Effect

This test examines how small changes in the input significantly affect the output. A scheme is said to have an effective Avalanche property if small changes in the input affectively changed the output [20] [21]. Ineffective avalanche property in cryptographic scheme could be exploited by the crypto analyst as vulnerability. The proposed scheme appears to have a favorable avalanche effect as in Table 9, Table 10, Table 11 and Table 12.

Table 9
Avalanche property between sub keys

Sub Keys		Avalanche
Sub_key_1a	Sub_key_2a	87.5%
Sub_key_1a	Sub_key_3a	100%
Sub_key_1a	Sub_key_4a	93.75%
Sub_key_2a	Sub_key_3a	87.5%
Sub_key_2a	Sub_key_4a	100%
Sub_key_3a	Sub_key_4a	100%

Table 10
Avalanche property between different sub keys

Sub Keys		Avalanche
Sub_key_1b	Sub_key_2b	93.75%
Sub_key_1b	Sub_key_3b	93.75%
Sub_key_1b	Sub_key_4b	100%
Sub_key_2b	Sub_key_3b	93.75%
Sub_key_2b	Sub_key_4b	93.75%
Sub_key_3b	Sub_key_4b	100%

Table 11
Avalanche property between sub keys

Sub Keys		Avalanche
Sub_key_1c	Sub_key_2c	93.75%
Sub_key_1c	Sub_key_3c	100%
Sub_key_1c	Sub_key_4c	93.75%
Sub_key_2c	Sub_key_3c	97%
Sub_key_2c	Sub_key_4c	91%
Sub_key_3c	Sub_key_4c	91%

Table 12
Avalanche property between different sub keys

x	y	Avalanche
Sub_key_1d	Sub_key_2d	92%
Sub_key_1d	Sub_key_3d	96%
Sub_key_1d	Sub_key_4d	97%
Sub_key_2d	Sub_key_3d	91%
Sub_key_2d	Sub_key_4d	96%
Sub_key_3d	Sub_key_4d	93%

E. The NIST Test

This is a standard test proposed by NIST to measure the randomness of key scheduling algorithms, and any crypto system that has to do with random numbers. Minimum of *100* bits is required as input for all the tests and if the P-value obtained from each result is *<0.01* then the sequence is not random, otherwise, it's random [22]. Out of the *16* different tests, this paper focuses on only *3* most important ones. The Frequency (Mono bit) Test, Frequency Test within Block and Run Test. All subsequent tests in the NIST test suit depend on the Frequency (Mono bit) test. The purpose of this is to check the proportion of ones and zeros in the generated random. Similarly, the Block test is to examine the frequency of each block within the sequence. The length of each block and how many blocks within a sequence will be determined by this test. The Run test is to check the uninterrupted sequence of like bits and to check how often those bits repeat themselves in the generated pseudo random numbers [22].

Table 13 and Table 14 contain the obtained results of this test.

Table 13
The NIST test

Key ID	Freq_Test	Input Block	Run Test
Sub_key_1a	0.6171	0.1512	0.9750
Sub_key_2a	0.6171	0.2017	0.9750
Sub_key_3a	0.8026	0.2317	0.4481
Sub_key_4a	1.0000	0.4335	0.6171

Table 14
The NIST test

Key ID	Freq Test	Input Block	Run Test
Sub_key_1b	0.8597	0.1985	0.7215
Sub_key_2b	0.4795	0.9134	0.3515
Sub_key_3b	0.8597	0.7089	0.0514
Sub_key_4b	0.4795	0.6728	0.2309

V. CONCLUSION

Key scheduling algorithm (KSA) is the mechanism that generates encryption keys and all other round keys for each round of encryption. An adversary may study the behavior and relationship between all the round keys as a way to launch an attack on that algorithm Researches have suggested that sub-keys should be generated separately and disproportionately large to avoid related-key attack. The use of matrices has been adapted to achieve maximum security. Generating those matrices could be time consuming and the pattern of the matrices could compromise the secrecy of the algorithm. This paper introduced the use of highly shapeless quasigroup as a dynamic substitution table for generating encryption key(s). In the proposed scheme, all generated Isotopes appears to have inherited all the properties of the parent quasigroup. The proposed KSA adopted the AKDF as the mechanism for expanding the key. All generated keys and sub keys have been tested and proved to have a reasonable randomness property. The future work of this research is to compare with other schemes to practically validate the proposed algorithm.

ACKNOWLEDGMENT

This research is supported by the Office for Research, Innovation, Commercialization and Consultancy (ORICC), Universiti Tun Hussein Onn Malaysia (UTHM) under Project Vot No. U195.

REFERENCES

[1] S. Jamel, M. M. Deris, I. T. R. Yanto, and T. Herawan. "The hybrid cubes encryption algorithm (HiSea)," in *Advances in Wireless, Mobile Networks and Applications. Communications in Computer and Information Science*, S. S. Al-Majeed, C. L. Hu, and D. Nagamalai, Eds. Berlin, Heidelberg: Springer, 2011, pp. 191-200.

[2] J. Kelsey, and B. Schneier. "Key-schedule cryptanalysis of DEAL," in *Selected Areas in Cryptography* (SAC 1999). *Lecture Notes in Computer Science*, H. Heys, and C. Adams, Eds. Berlin, Heidelberg: Springer, 2000, pp. 118–134.

[3] J. Kelsey, B. Schneier, and D. Wagner, "Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2 and TEA," in *Information and Communications Security*, Y. Han, T. Okamoto, and S. Qing, Eds. Berlin, Heidelberg: Springer, pp. 233–246, 1997.

[4] M. F. Mushtaq, S. Jamel, and M. M. Deris, "Triangular coordinate extraction (TCE) for hybrid cubes," *Journal of Engineering and Applied Science*, vol.12, no. 8, pp. 2164-2169, 2017.

[5] D. Rajavel, and S. P. Shantharajah, "Cubical key generation and encryption algorithm based on hybrid cube's rotation," in *Proceedings of the International Conference on Pattern Recognition, Informatics and Medical Engineering*, 2012, pp. 183-187.

[6] V. Dimitrova, and J. Markovski, "On quasigroup pseudo random sequence generators," in *Proc. of the 1st Balkan Conf. on Informatics*, 2004, pp. 21–23.

[7] A. Krapež, "An application of quasigroups in cryptology," *Math. Maced*, vol. 8, pp. 47–52, 2010.

[8] S. Markovski, D. Gligoroski, and L. Kocarev, "Unbiased random sequences from quasigroup string transformations," in *the Proceedings of the 12th Int. Workshop on Fast Software Encryption (FSE 2005)*, vol. 3557, 2005, pp. 163–180.

[9] K. A. Meyer, *A New Message Authentication Code Based on the Non-Associativity of Quasigroups*. Retrospective Theses and Dissertations, 2006.

[10] O. Grošek, "Isotopy of Latin squares in cryptography," *Tatra Mountains Mathematical Publications*, vol. 45, pp. 27–36, 2010.

[11] Smile Markovski, "Design of crypto premitives based on quasigroup," *Quasigr. Relat. Syst.*, vol. 23, pp. 41–90, 2015.

[12] V. Bakeva, "Parastrophic quasigroup string processing," in *Proceedings of the Conference on Informatics and Information Technology*, 2011, pp. 19–21.

[13] A. Mileva and S. Markovski, "Shapeless Quasigroups Derived by Feistel Orthomorphisms," *Glasnik Matematički*, vol. 47, no. 67, pp. 333–349, 2012.

[14] H. Michael Damm, "Totally anti-symmetric quasigroups for all orders," *Discrete Math.*, vol. 307, no. 6, pp. 715–729, 2007.

[15] H. Zorkta and T. Kabani, "New Cipher Algorithm Based on Multiple Quasigroups," *International Journal of Machine Learning and Computing*, vol. 1, no. 5, pp. 454–459, 2011.

[16] Z. Pindar, S. H. Jamel, A. Disina, and M. M. Deris, "Compression function based on permutations quasigroups," *ARPN Journal of Engineering and Applied Sciences*, vol. 11, no. 12, pp. 1–8, 2015.

[17] A. H. Disina, S. Jamel, Z. A. Pindar, and M. M. Deris, "All-or-nothing key derivation function based on quasigroup string," in *International Conference on Information Science and Security (ICISS)*, 2006, pp. 6–10.

[18] C. W. Chuah, E. Dawson, and L. Simpson, "Key derivation function: the SCKDF scheme," in *Security and Privacy Protection in Information Processing Systems*, L. J. Janczewski, H. B. Wolfe, and S. Shenoi, Eds. Berlin, Heidelberg: Springer, 2013, pp. 125–138.

[19] H. Krawczyk, "Cryptographic extraction and key derivation: The HKDF scheme," in *Lect. Notes Comput. Sci.*, vol. 6223 LNCS, 2010, pp. 631–648.

[20] H. Ahmad, A. Hassan, M. Saeb, and H. D. Hamed, "The 'PYRAMIDS' block cipher," *International Journal of Network Security*, vol. 2, pp. 50–60, 2005.

[21] J. C. H. Castro, J. M. Sierra, A. Seznec, A. Izquierdo, and A. Ribagorda, "The strict avalanche criterion randomness test," *Math. Comput. Simul.*, vol. 68, no. 1, pp. 1–7, 2005.

[22] S. Ramanujam, and M. Karuppiah, "Designing an algorithm with high Avalanche Effect," *Int. J. Comput. Sci. Netw. Secur.*, vol. 11, no. 1, pp. 106–111, 2011.

[23] L. E. Bassham, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, S. D. Leigh, M Levenson, M. Vangel, N. A. Heckert, and D. L. Banks, *A Statistical Test Suite* for *Random* and *Pseudorandom Number Generators* for *Cryptographic Applications*. National Institute of Standard and Technology, Technology Administration, US Department of Commerce, 2010.