Int. J. Communications, Network and System Sciences, **2011, 4, 345-350**
doi:10.4236/ijcns.2011.45039 Published Online May 2011 (http://www.SciRP.org/journal/ijcns)

Permutation and Complementary Algorithm to Generate Random Sequences for Binary Logic

Jie Wan, Jeffrey Z. J. Zheng
School of Software, Yunnan University Kunming, Kunming, China
E-mail: {*wanjiech, conjugatesys*}*@gmail.com*

Received December 15, 2010; *revised March* 23, 2011; *accepted April* 3, 2011

Abstract

Randomness number generation plays a key role in network, information security and IT applications. In this paper, a permutation and complementary algorithm is proposed to use vector complementary and permutation operations to extend n-variable Logic function space from 2^{2^n} functions to $2^{2^n} \times 2^n!$ configurations for variant logic framework. Each configuration contains 2^{2^n} functions can be shown in a $2^{2^{n-1}} \times 2^{2^{n-1}}$ matrix. A set of visual results can be represented by their symmetric properties in W, F and C codes respectively to provide the essential support on the variant logic framework.

Keywords: Logic Function, Permutation and Complementary, Variant Logic, Symmetric Distribution, Random Sequence

1. Introduction

Random numbers play an important role in many network protocols and encryption schemas on various network security applications [1], for example: digital signatures, authentication protocols, key generation for PKI, RSA/AES [2], nonce frustrate, symmetric stream encryption. A better random number algorithm will enhance encryption schemas, to do other applications. To satisfy different requirements, the NIST has published a series of statistical tests as standards [3] to determine whether a random number generator is suitable for a cryptographic application. After using the vector complementary and the permutation operations on binary logic, the variant logical framework extends the traditional Logic function space from 2^{2^n} functions to $2^{2^n} \times 2^n!$ configurations [4]. Under the new extension conditions, it is possible to use simple transformation to generate huge numbers of random sequences for future applications.

Permutation and complementary algorithm is described in the paper to express different random properties through a series of binary image sequences undertaken typical recursive operations.

2. Method

Cellular automata perform a natural way to generate random sequence. The principle of binary cellular automata [5] [6] can be explained by an example as follow:

First, a sequence 001100 and a function *f*: $\{00 \to 0, 01 \to 1, 10 \to 1, 11 \to 0\}$ are selected.

Second, the sequence can be decomposed from left to right. The last bit is composed to the first bit

$$001100 \to \{00, 01, 11, 10, 00, 00\}$$

Third, according to the decomposed sequences and the generating function, a new sequence 010100 can be generated. *i.e.*:

f: $001100 \to 010100$

Followed the algorithm, the space of the generation function can be extended further; large numbers of random sequences can be generated. This mechanism can increase the complexity of code-breaking.

In variant logic framework, the logic function space has been extended from 2^{2^n} to $2^{2^n} \times 2^n!$ by the permutation and the complementary operations.

In 2 variable functions of cellular automata, there are 16 generated functions. And the 16 functions can be described in a truth table (**Figure 1(a)**) with 16 entries.

2.1. Permutation Operation

The bit string of states $\{00, 01, 10, 11\}$ in generating function can be converted to decimal number $\{0, 1, 2, 3\}$. An example in **Figure 1(b)** is shown to permute 3210 to

	P Status				
J	3	2	1	0	K
	11	10	01	00	
0	0	0	0	0	0
1	0	0	0	1	1
2	0	0	1	0	2
3	0	0	1	1	3
4	0	1	0	0	4
⋮	…	…	…	⋮	⋮
⋮	…	…	…	⋮	⋮
13	1	1	0	1	13
14	1	1	1	0	14
15	1	1	1	1	15

(a)

$$\xrightarrow{P\begin{pmatrix}3210\\1320\end{pmatrix}}$$

	P Status				
J	1	3	2	0	K
	01	11	10	00	
0	0	0	0	0	0
1	0	0	0	1	1
2	1	0	0	0	8
3	1	0	0	1	9
4	0	0	1	0	2
⋮	…	…	…	⋮	⋮
⋮	…	…	…	⋮	⋮
13	0	1	1	1	7
14	1	1	1	0	14
15	1	1	1	1	15

(b)

Figure 1. Permutation example, (a) The Truth Table of 3210; (b) The Permutation Table of 1320.

1320 of the table.

2.2. Complementary Operation

In the complementary operation, the complementary vector σ is applied operate to the truth table.

It can be describe as

$$y^{\delta} = \begin{cases} y, \delta = 1 \\ \bar{y}, \delta = 0 \end{cases}$$

In 2-variable variant logic, σ is a binary sequence of 4 bits in $\{0000, \cdots, 1111\}$. In the example the original table is $\sigma = 1111$ and shown in **Figure 2(a)** the given $\sigma = 1100$ to **Table 2** can be described as $1320^{(1100)} = 1^1 3^1 2^0 0^0$. Under such operation, the sequence values of state 1 and 3 columns are invariant. But the values of columns whose index is 0 and values of the permutation sequence in state 2 and 0 are changed to their revised values respectively.

After the complementary operation, **Figure 2(a)** changes to **Figure 2(b)**.

2.3. Visualization

For function f applies once applied again the sequence 001100 output 010100.then applied again the 010100 to output 111100.For such binary sequence, select black for 1 and white for 0 to generate the visual patterns as follows (**Figure 3**).

2.4. Matrix Representation

For example, (**Figure 2(b)**) the truth value of 3th function is 1010. It can be converted to a binary coordinate $\langle 10|10 \rangle$ distinguished Left two and tight two bits respectively. So the decimal coordinate is $\langle 2|2 \rangle$. Then the (**Figure 2(b)**) can be converted to **Table 1**.

Under such converting the 2D matrix can be represented in **Table 2**.

3. Algorithm and Properties

3.1. Permutation and Complementary Algorithm

Using permutation and complementary operations, an algorithm is extended to express the n-ary variant logic functional space.

Algorithm: Permutation and Complementary:
Input: variable n
Output: a set of truth table of P^{σ}, $\forall P \in S\left(2^n\right)$, $\forall \sigma \in B_2^{2^n}$.

	σ				
	1	1	1	1	
J	P Status				K
	1	3	2	0	
	01	11	10	00	
0	0	0	0	0	0
1	0	0	0	1	1
2	1	0	0	0	8
3	1	0	0	1	9
4	0	0	1	0	2
⋮	⋯	⋯	⋯	⋮	⋮
⋮	⋯	⋯	⋯	⋮	⋮
13	0	1	1	1	7
14	1	1	1	0	14
15	1	1	1	1	15

(a)

$\xrightarrow{\sigma = 1100}$

	σ				
	1	1	0	0	
J	P Status				K
1		3	2	0	
	01	11	10	00	
0	0	0	1	1	3
1	0	0	1	0	2
2	1	0	1	1	11
3	1	0	1	0	10
4	0	0	0	1	1
⋮	⋯	⋯	⋯	⋯	⋮
⋮	⋯	⋯	⋯	⋯	⋮
13	0	1	0	0	4
14	1	1	0	1	13
15	1	1	0	0	12

(b)

Figure 2. Complementary Example, (a).The Permuatation Table of 1320$^{(1111)}$; (b).The Complementary Table of 1320$^{(1100)}$.

Method:

Step 1. Initial $\mathrm{T} = \left\{2^n - 1 \cdots 1\ 0\right\}$

Step 2. Generate a permutation *P* for *T*

Step 3. From $\sigma = 000\cdots0$ to 111 ⋯ 1 do vector complementary operation.

Step 4. Any new permutation?

Yes go to Step 2.

Step 5. End

where *S*(*N*) is a symmetry group with *N* member and B_2^M is a *M* variable Boolean structure with 2^M members.

3.2. Representation Scheme

Every truth table has a 2D matrix to arrange visual results of random sequence. The $\left\langle X, Y \right\rangle$ is the coordinate to allocate each visual result. So for n-ary logic function space, the 2D matrix has a size of $2^{2^{n-1}} \times 2^{2^{n-1}}$ shown (**Table 3**).

3.3. W, F and C Code

Three coding schemes can be distinguished in the algorithm.

W code [4] is a binary sequence of 2^n bits. It separates into two parts $\left(\mathrm{J}^1 \middle| \mathrm{J}^0\right)$. Each part has 2^{n-1} bits

F code is a subset of W code. And it is a symmetry code. In F code, if the I-th Meta state in J^1 is 1 or 0, the I-th meta state in J^0 is the negative state.

If a code is F code and the I-th meta state in J^1 have the same value. Besides four corners of its matrix are included in $\left\{0, x, \overline{x}, 1\right\}$, it is C code [4].

E.g. $\left\langle 32 \middle| 10 \right\rangle \left\langle 1110 \middle| 0100 \right\rangle$ is an element of *W* code. In the sequence 1 isn't the negative sequence of 3. And the 0 isn't also the negative sequence of 2.

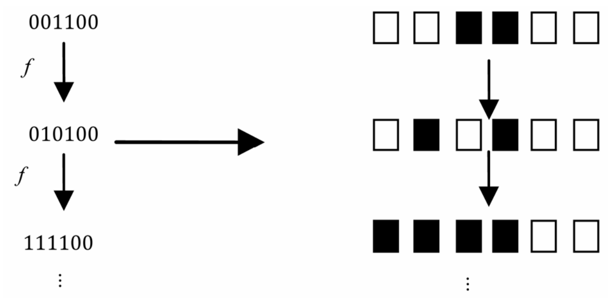

Figure 3. Visualize the Random sequence.

Table 1. Coordinate map of 1320$^{(1100)}$.

	σ				
	1	1	0	0	
J	P Status				Transformed bracket
	1	3	2	0	
	01	11	10	00	
0	0	0	1	1	<0,3>
1	0	0	1	0	<0,2>
2	1	0	1	1	<2,3>
3	1	0	1	0	<2,2>
4	0	0	0	1	<0,1>
⋮	…	…	…	⋮	⋮
⋮	…	…	…	⋮	⋮
13	0	1	0	0	<1,0>
14	1	1	0	1	<3,1>
15	1	1	0	0	<3,0>

Table 2. 2D Matrix of the 1320$^{(1100)}$.

<0,0> 5	<0,1> 4	<0,2> 1	<0,3> 0
<1,0> 13	<1,1> 12	<1,2> 9	<1,3> 8
<2,0> 7	<2,1> 6	<2,2> 3	<2,3> 2
<3,0> 15	<3,1> 14	<3,2> 11	<3,3> 10

Table 3. 2D Matrix for n-ary logic functions.

<0,0>	…	…	$\left\langle 0, 2^{2^{n-1}}-1\right\rangle$
<1,0>	…	…	$\left\langle 1, 2^{2^{n-1}}-1\right\rangle$
⋮	⋮	⋮	⋮
$\left\langle 2^{2^{n-1}}-2, 0\right\rangle$	…	…	$\left\langle 2^{2^{n-1}}-2, 2^{2^{n-1}}-1\right\rangle$
$\left\langle 2^{2^{n-1}}-1, 0\right\rangle$	…	…	$\left\langle 2^{2^{n-1}}-1, 2^{2^{n-1}}-1\right\rangle$

$\langle 32|01\rangle\langle 1110|00\ 01\rangle$ is an F code. It has the symmetry property. In the sequence 0 is the negative sequence of 3 and 1 is the negative sequence of 2.

$(13|20)(01\ 11|10\ 00)$ is a C code. It has the symmetry property of *F* code. And four comers of 1320's matrix are included in $\{0, x, \bar{x}, 1\}$.

The further definition of *W*, *F* and *C* code can be found in the [4].

From the exhaustive of the binary variant function space, the number of *W*, *F*, *C* code in binary variant function space [7]shown in **Table 4**.

4. Coding Simples

W Code:

Permutation sequence: 3210

The value of σ : 1011

F Code:

Permutation sequence: 3201

The value of σ : 1111

C Code:

Permutation sequence: 1320

The value of σ : 1100

5. Result Analysis

In **Figure 4**, W code is shown a general code. Majority W code doesn't have apparent symmetry property. W code covers all the code space which form from binary input variable. These properties can be seen in **Figure 4**.

All the F codes have overall symmetry in 2D distribution. Obvious symmetry among functions in the 2D matrix can be observed in **Figure 5**.

Simple is shown in a C code in **Figure 6**. It is a small set of F code with complete symmetry property, C code have the four constant vertex property. The group of the four vertex in C code are located by 0, 15, 10, 5 functions respectively.

In the n-ary logical function permutation and complementary algorithm, the permutation is operated for $2^n!$; the complementary exhaustive needs 2^{2^n} operation for each permutation operation. A total of computational complexity of an n-ary variant logical function using Permutation and Complementary algorithm is $O\left(2^n! \times 2^{2^n}\right)$.

6. Conclusions

A permutation and complementary algorithm has been proposed for n-ary logical function. And sample results are visualized. The visual results of *W*, *F* and *C* code in the variant and invariant properties support the variant

Table 4. The number of *W*, *F* and *C* code in 2-ary variant functional space.

Code System	No
W	384
F	128
C	16

<0,0> 4	<0,1> 5	<0,2> 6	<0,3> 7
<1,0> 0	<1,1> 1	<1,2> 2	<1,3> 3
<2,0> 12	<2,1> 13	<2,2> 14	<2,3> 15
<3,0> 8	<3,1> 9	<3,2> 10	<3,3> 11

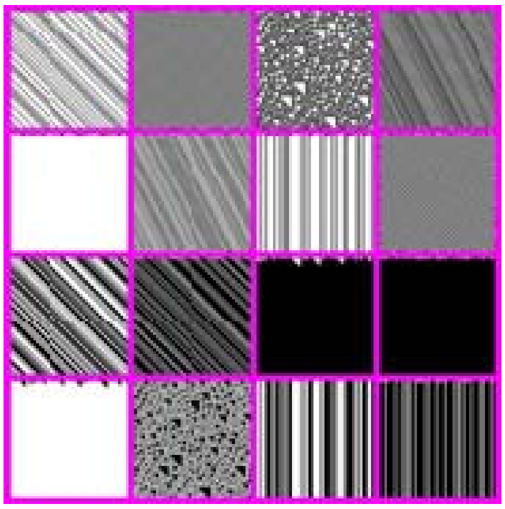

Figure 4. The 2D matrix diagram and the visual result of 3210^{1011}.

<0,0> 0	<0,1> 2	<0,2> 1	<0,3> 3
<1,0> 4	<1,1> 6	<1,2> 5	<1,3> 7
<2,0> 8	<2,1> 10	<2,2> 9	<2,3> 11
<3,0> 12	<3,1> 14	<3,2> 13	<3,3> 15

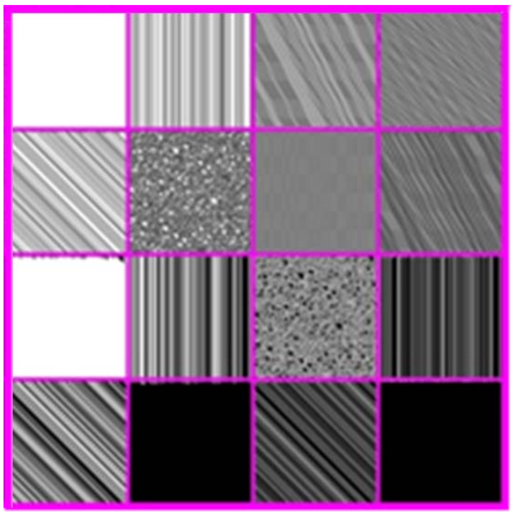

Figure 5. The 2D matrix diagram and the visual result of 3210^{1111}.

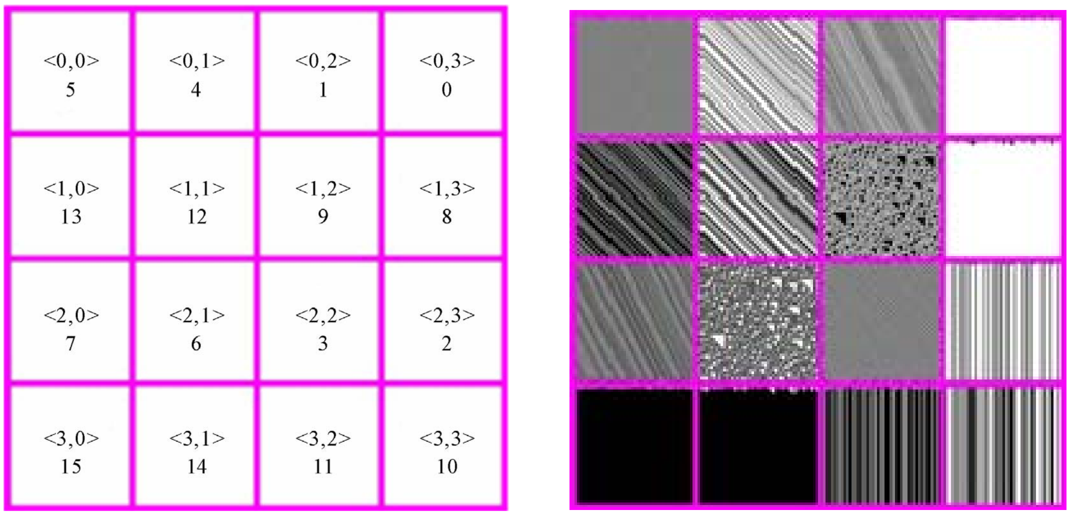

Figure 6. The 2D matrix diagram and the visual result of 1320^{1100}.

logic system through experimentation to use an algorithmic mechanism to generate a series of huge random number sequences.

7. References

[1] W. Stallings, "Cryptography and Network Security: Principles and Practice," Prentice Hall, 2005.

[2] J. Soto and L. Bassham, "Randomness Testing of the Advanced Encryption Standard Finalist Candidates," National Institute of Standards and Technology (NIST), 2000.

[3] National Institute of Standards and Technology (NIST), "Random number generation," 2008. Internet Avail- able: http://csrc.nist.gov/groups/ST/toolkit/rng/index.html

[4] J. Zhi, J. Zheng and C. H. Zheng, "A framework to express variant and invariant functional spaces for binary logic," Frontiers of Electrical and Electronic Engineering in China, Vol. 5, No. 2, 2010, pp. 163-172.

[5] S. Wolfram, "Theory and Applications of Cellular Automata," Singapore: Word Scientific, 1986.

[6] S. Wolfram, "Cellular automata as models of complexity," *Nature*, Vol. 311, 1984, pp. 419-424.

[7] J. Wan and J. Zheng, "Showing Exhaustive Number Sequences of Two Logic Variables for Variant Logic Functional Space," *Proceedings of Asia-Pacific Youth Conference on Communication (APYCC)*, October 2010, p. 4.