Permutation and Complementary Algorithm to Generate Random Sequences for Binary Logic

Jie Wan and Jeffrey Zheng

Abstract Randomness number generation plays a key role in network, information security, and IT applications. In this chapter, 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 that 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, and 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

Project supported by Yunnan Advanced Overseas Scholar Project, NSF of China (61362014).

J. Wan
Yunnan University, Kunming, China
e-mail: wanjiech@163.com

J. Zheng (✉)
Key Laboratory of Yunnan Software Engineering, Yunnan University, Kunming 650091, Yunnan, China
e-mail: conjugatelogic@yahoo.com

© The Author(s) 2019

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 chapter to express different random properties through a series of binary image sequences undertaking 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 follows:

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

$$001100 \rightarrow \{00,01,11,10,00,00\}.$$

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 two variable functions of cellular automata, there are 16 generated functions, and the 16 functions can be described in a truth table (Fig. 1a) 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 Fig. 1b is shown to permute 3210 to 1320 of the table.

(a).The Truth Table of 3210

	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

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

(b).The Permutation Table of 1320

	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

Fig. 1 Permutation example

2.2 *Complementary Operation*

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

It can be described as

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

In two-variable variant logic, σ is a binary sequence of 4 bits in {0000, . . . , 1111}. In the example, the original table is σ = 1111 and shown in Fig. 2a given σ = 1100 in Table 2 which 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, Fig. 2a changes to Fig. 2b.

2.3 *Visualization*

For function f, once applied on the sequence 001100 to output 010100, then this function can be applied on the sequence 010100 to output 111100. For such binary sequence, select black for 1 and white for 0 to generate the visual patterns as follows (Fig. 3).

(a).The Permuatation Table of $1320^{(1111)}$

	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

$\xrightarrow{\sigma = 1100}$

(b).The Complementary Table of $1320^{(1100)}$

	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

Fig. 2 Complementary example

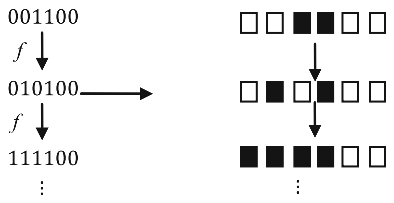


Fig. 3 Visualize the random sequence

2.4 *Matrix Representation*

For example (Fig. 2b), the truth value of third function is 1010. It can be converted to a binary coordinate ⟨10|10⟩ distinguished by left two and right two bits, respectively. So the decimal coordinate is ⟨2|2⟩. Then Fig. 2b can be converted to Table 1.

Under such conversion, 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.

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

	σ				Transformed bracket
	1	1	0	0	
J	P Status				
	1	3	2	0	
	01	11	10	00	
0	0	0	1	1	$\langle 0, 3\rangle$
1	0	0	1	0	$\langle 0, 2\rangle$
2	1	0	1	1	$\langle 2, 3\rangle$
3	1	0	1	0	$\langle 2, 2\rangle$
4	0	0	0	1	$\langle 0, 1\rangle$
⋮	…	…	…	⋮	⋮
⋮	…	…	…	⋮	⋮
13	0	1	0	0	$\langle 1,0\rangle$
14	1	1	0	1	$\langle 3,1\rangle$
15	1	1	0	0	$\langle 3,0\rangle$

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

$\langle 0, 0\rangle$ 5	$\langle 0, 1\rangle$ 4	$\langle 0, 2\rangle$ 1	$\langle 0, 3\rangle$ 0
$\langle 1, 0\rangle$ 13	$\langle 1, 1\rangle$ 12	$\langle 1, 2\rangle$ 9	$\langle 1, 3\rangle$ 8
$\langle 2, 0\rangle$ 7	$\langle 2, 1\rangle$ 6	$\langle 2, 2\rangle$ 3	$\langle 2, 3\rangle$ 2
$\langle 3, 0\rangle$ 15	$\langle 3, 1\rangle$ 14	$\langle 3, 2\rangle$ 11	$\langle 3, 3\rangle$ 10

```
Algorithm: Permutation and Complementary:
Input: variable n
Output: a set of truth table of $P^\sigma$, $\forall P \in S(2^n)$, $\forall \sigma \in B_2^{2^n}$.
Method:
Step 1. Initial $T = \{2^n 2^n - 1 \cdots\cdots 10\}$
Step 2. Generate a permutation P for T
Step 3. From $\sigma = 000\ldots0$ to $111\ldots1$ 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 an M variable Boolean structure with 2^M members.

Table 3 2D matrix for n-ary logic functions

$\langle 0, 0\rangle$	…	…	$\langle 0, 2^{2^{n-1}}-1\rangle$
$\langle 1, 0\rangle$	…	…	$\langle 1, 2^{2n-1}-1\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\langle 2^{2n-1}-2, 0\rangle$	…	…	$\langle 2^{2n-1}-2, 2^{2n-1}-1\rangle$
$\langle 2^{2n-1}-1, 0\rangle$	…	…	$\langle 2^{2n-1}-1, 2^{2n-1}-1\rangle$

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

Code system	No
W	384
F	128
C	16

3.2 Representation Scheme

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

3.3 W, F, and C

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, $\left(J^1|J^0\right)$. 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 Ith meta-state in J^1 is 1 or 0, the Ith meta-state in J^0 is the negative state.

If a code is F code, the Ith meta-state in J^1 has the same value. Besides, four corners of its matrix are included in $\{0, x, \bar{x}, 1\}$; it is C code [4].

For example, (32|10)(1110|0100) is an element of W code. In the sequence, 1 is not the negative sequence of 3, and the 0 is not also the negative sequence of 2. (32|01)(1110|0001) 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)(0111|1000) 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 codes can be found in [4].

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

4 Coding Simples

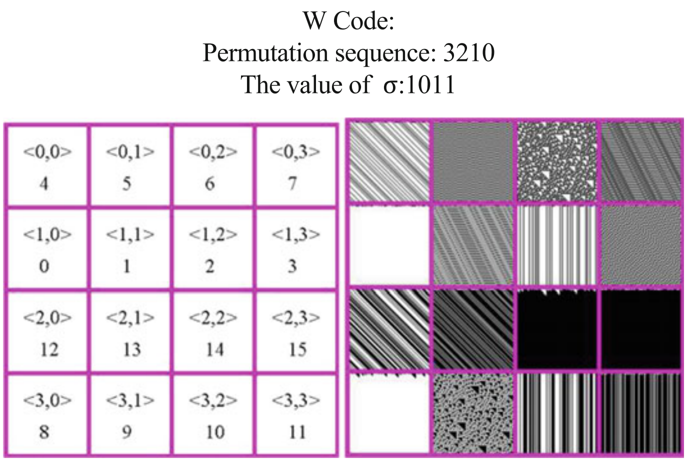


Fig. 4 The 2D matrix diagram and the visual result of 3210^{1011}

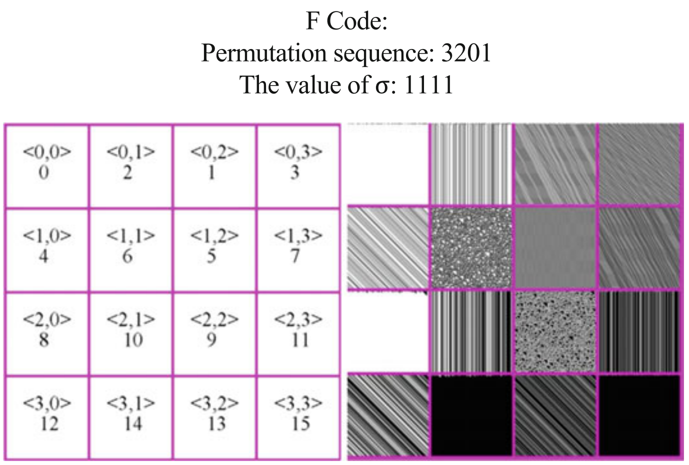
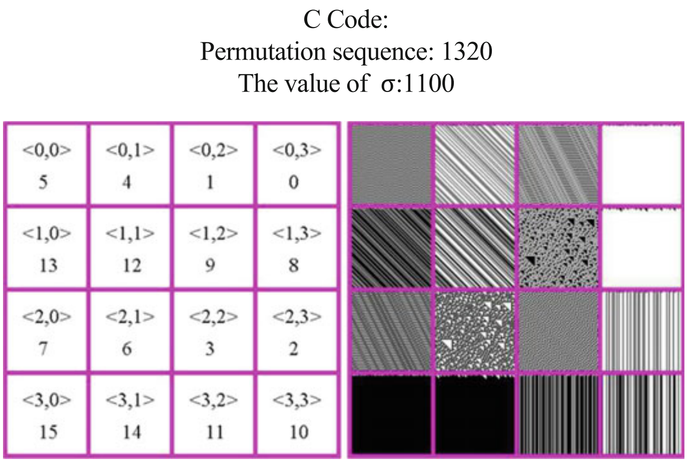


Fig. 5 The 2D matrix diagram and the visual result of 3201^{1111}



<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

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

5 Result Analysis

In Fig. 4, W code is shown as a general code. Majority W code does not have apparent symmetry property. W code covers all the code spaces which are formed from binary input variable. These properties can be seen in Fig. 4.

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

Simple is shown in a C code in Fig. 6. It is a small set of F code with complete symmetry property. C code has the four-constant vertex property. The group of the four vertexes in C code are located by 0, 15, 10, and 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 Conclusion

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 codes in the variant and invariant properties support the variant logic system through experimentation to use an algorithmic mechanism to generate a series of huge random number sequences.

References

1. W. Stallings, *Cryptography and Network Security: Principles and Practice* (Pearson Education, 2006)
2. J. Soto, L. Bassham, *Randomness Testing of the Advanced Encryption Standard Finalist Candidates* (NIST, 2000)
3. *Random number generation* (NIST, 2008), http://csrc.nist.gov/groups/ST/toolkit/rng/index.html
4. J.Z.J. Zheng, C.H. Zheng, A framework to express variant and invariant functional spaces for binary logic. Front. Electr. Election. Eng. China **5**(2) (2010) (Higher Education Press & Springer Press)
5. S. Wolfram, *Theory and Applications of Cellular Automata* (Word Scientific, Singapore, 1986)
6. S. Wolfram, Cellular automata as models of complexity. Nature **311** (4 October 1984)
7. J. Wan, J. Zheng, Showing exhaustive number sequences of two logic variables for variant logic functional space, in *Proceedings of Asia-Pacific Youth Conference on Communication* (APYCC), p. 4 (October 2010)

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (http://creativecommons.org/licenses/by/4.0/), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.