Indonesian Journal of Combinatorics **4** (1) (2020), 21–26

INDONESIAN JOURNAL OF COMBINATORICS

# Randomness of encryption keys generated by super $H$-antimagic total labeling

Antonius Cahya Prihandoko[a], Yudha Alif Auliya[a], Diksy Media Firmansyah[b], Slamin[b]

[a] *Department of Information Technology, Universitas Jember, Jl. Kalimantan 37 Jember, Indonesia*
[b] *Department of Informatics, Universitas Jember, Jl. Kalimantan 37 Jember, Indonesia*

antoniuscp.ilkom@unej.ac.id, yudha.alif@unej.ac.id, diksy@unej.ac.id, slamin@unej.ac.id

**Abstract**

Super $H$-antimagic total labeling (SHATL) can be utilized to generate encryption keys. The keys are then used to establish the improved block and stream ciphers. In these ciphers, different blocks were encrypted by the different keys, but all block keys were connected one another. These conditions make the developed cryptosystems more secure and require less keys storage capacity compared to the ordinary block and stream cipher. The randomness of the generated keys, however, still need to be tested. The test is necessary to ensure that there is no specific pattern that can be utilized by any intruder to guess the keys. This paper presents the randomness tests applied to all key sequences generated by both the improved block scheme and the stream based scheme.

*Keywords:* randomness, encryption keys, super H-antimagic total labeling, crypto-graphy, block cipher, stream cipher
Mathematics Subject Classification: 05C78
DOI: 10.19184/ijc.2020.4.1.3

## 1. Introduction

A characteristic that is essential in the context of information security is confidentiality. Information is said to be confidential when it is protected from disclosure to unauthorized persons or systems. Cryptography is a popular approach to achieve information confidentiality. In this approach, information is initially encrypted before it is delivered through unsecure channels. The strength of encryption protocols depends on the encryption-decryption keys management: the keys

Received: 31 May 2019, Revised: 14 April 2020, Accepted: 27 May 2020.

have to be kept secret to unauthorized parties. Indeed, keeping the keys from being accessible to unauthorized parties is the major challenge for many cryptographic schemes.

Researches on the encryption-decryption keys management are continuously undertaken and focused to achieve information confidentiality according to the required security level. In previous works [7, 8], encryption keys were generated using super $H$-antimagic total labeling (SHATL). A bijective function $f$ is called an $(a, d)$-$H$-antimagic total labeling of graph $G$ if $f : V(G) \cup E(G) \rightarrow \{1, 2, \ldots, |V(G)| + |E(G)|\}$ such that for all subgraphs of $G$ isomorphic to $H$, the total $H$-weights $w(H) = \sum_{v \in V(H)} f(v) + \sum_{e \in E(H)} f(e)$ form an arithmetic sequence $\{a, a + d, a + 2d, ..., a + (n - 1)d\}$, where $a$ and $d$ are positive integers and $n$ is the number of all subgraphs of $G$ isomorphic to $H$. Additionally, if $f : V(G) \rightarrow \{1, 2, \ldots, |V(G)|\}$, then the $(a, d)$-$H$-antimagic total labeling $f$ is called super.

The encryption keys were constructed from the SHATL of a *generalized shackle* of graph. A *shackle* of graph $H$, symbolized by $G = shack(H, v, n)$, is a graph $G$ developed by non-trivial graphs $H_1, H_2, \ldots, H_n$, such that for every $1 \leq s, t \leq n$, with $|s - t| \geq 2$, $H_s$ and $H_t$ have no common vertex, but for every $1 \leq i \leq n - 1$, $H_i$ and $H_{i+1}$ have precisely one common vertex $v$, called *connecting vertex*, and all $n - 1$ connecting vertices are different. A *generalized shackle* of graph, denoted by $G = gshack(H, K \subset H, n)$, is the graph obtained from $G = shack(H, v, n)$ by substituting the connecting vertex by any subgraph $K \subset H$. The existence of super $(a, d)$-$H$ antimagic total labeling of *generalized shackle* of graph was proved using an *integer set partition technique* [2, 4]. This proof guarantee that constructing encryption keys using SHATL is feasible.

The constructed keys were utilized to establish the improved block and stream ciphers. The developed cryptosystems have been proved to be more secure and require less keys storage capacity compared to the ordinary block and stream cipher [7, 8]. A randomness test, however, still needs to be applied to the generated keys. This kind of test is needed to ensure that there is no specific pattern that can be utilized by any intruder to guess the key. This paper presents the randomness tests applied to all key sequences generated using SHATL both in the improved block scheme and the stream based scheme.

The rest of this paper is outlined as follows. Section 2 presents the mechanism of constructing encryption keys both in the block scheme and the stream cipher using SHATL. Section 3 describes the randomness of the keys sequences generated by SHATL.

## 2. Utilizing SHATL to Construct Encryption Keys

In order to simulate the randomness of key sequences generated using SHATL, let us recall the SHATL algorithm for constructing encryption key in a block cipher [7] and the algorithm for generating key stream using SHATL [8]. For the former, assume that the cryptosystem is working on 26 English alphabets. Constructing encryption keys using super $(a, d) - H$ antimagic total labeling of *generalized shackle* of graph is undertaken through the algorithm 1.

*Algorithms* 1. **SHATL Algorithm for constructing encryption keys**

```
1. Assign f as label of the graph elements
2. If f is bijection, do 3, otherwise back to 1
3. Take a certain d for super (a,d)-HATL
4. Take z = sum of the number of vertices and 26
```

```
5. Draw the layered diagram by ignoring all labels greater than z
6. Place all edge labels in sequence from left to right
   and start from the top to the bottom layer.
7. Use the sequence of labels as the encryption keys
```

Generating a key stream is undertaken by modifying algorithm 1. Assume that the cryptosystem is working on 26 English alphabets. The key stream construction is proceed through the algorithm 2.

*Algorithms* 2. **Algorithm for generating key stream**

```
1. Define f for labeling the graph elements
2. If f is bijection, do 3, otherwise back to 1
3. Take a certain d for super (a,d)-HATL
4. Take z = the number of vertices plus 26
5. Draw the layered diagram by ignoring all labels greater than z
6. Place all edge labels in sequence from left to right
   and start from the top to the bottom layer.
7. Name the sequence by s and let t = length of s
8. Use the sequence s of labels as the source of key stream
9. Determine b = length of block
10. Determine i, such that $1 \leq i \leq t-b$
11. Take $k = s_i, s_{i+1}, s_{i+2}, ..., s_{i+b-1}$ as initial block key
12. Determine stream function $k_{j+b} = g(k_j, k_{j+1}, ..., k_{j+b-1})$
```

Outputs of algorithm 2 are the initial block key $k$ and the stream function $g(k)$.

## 3. Randomness of the Constructed Key Sequences

A single SHATL can be used to construct a key sequence in a block cipher and multiple key sequences in a stream cipher. For the simulation, the randomness test is applied to all sequences generated from Super $(a, 12) - H$ ATL of the graph $G$ that was provided in [7]. The labeling is illustrated in Figure 1. It shows that the vertex and edge labels start from 1 to 30 and 31 to 79, respectively.

A layered diagram rooted at label 1 is then drawn by ignoring the labels greater than 56 (Figure 2). The sequence obtained from the diagram is 31, 39, 48, 52, 54, 40, 50, 49, 51, 53, 55, 47, 32, 36, 56, 43, 46, 33, 37, 42, 45, 34, 38, 41, 44, 35, or (in its equivalence modulo 26) 5, 13, 22, 0, 2, 14, 24, 23, 25, 1, 3, 21, 6, 10, 4, 17, 20, 7, 11, 16, 19, 8, 12, 15, 18, 9.

We then use sequence $s$ = 5, 13, 22, 0, 2, 14, 24, 23, 25, 1, 3, 21, 6, 10, 4, 17, 20, 7, 11, 16, 19, 8, 12, 15, 18, 9 as the base sequence. As the randomness test requires a sequence with minimum length is 40, then $s$ can be enlarged by repeating previous components in the context of block cipher key. Therefore, the block cipher key that meet the requirement should be $s_b$ = 5, 13, 22, 0, 2, 14, 24, 23, 25, 1, 3, 21, 6, 10, 4, 17, 20, 7, 11, 16, 19, 8, 12, 15, 18, 9, 5, 13, 22, 0, 2, 14, 24, 23, 25, 1, 3, 21, 6, 10.

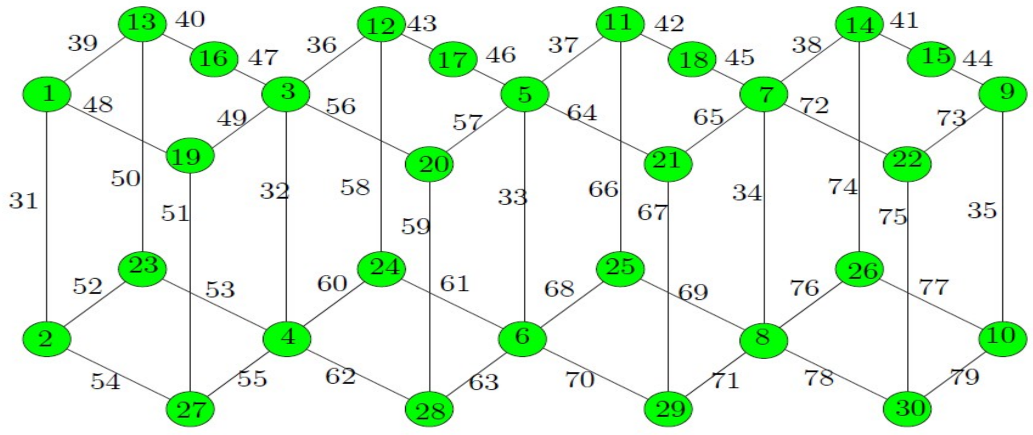

Figure 1. Super $(a, 12) - H$ ATL of a *generalized shackle* of graph

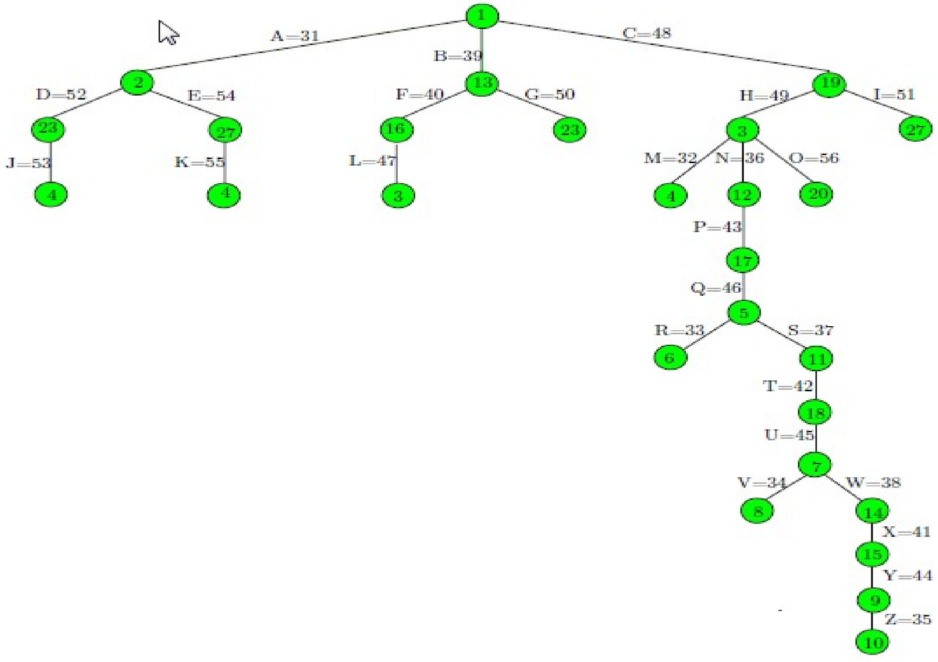

Figure 2. The layered diagram rooted at label 1

In the context of stream cipher key, $s_b$ can generate multiple key streams. For a single key stream construction, the stream function can be executed repeatedly until the required sequence length fulfilled. Suppose $i = 1$, $b = 5$, and the stream function is defined as $k_{j+5} = k_j + k_{j+1} \bmod 26$. We have the initial block key $k = 5, 13, 22, 0, 2$ and, thus the key stream is $st_1 =$ — 5, 13, 22, 0, 2 — 18, 9, 22, 2, 20 — 1, 5, 24, 22, 21 — 6, 3, 20, 17, 1 — 9, 23, 11, 18, 10 — 6, 8, 3, 2, 16 — 14, 11, 5, 18, 6 — 25, 16, 23, 24, 5.

For the same $b$ and stream function, multiple keystreams can be generated. Table 1 presents some keystreams generated from the previously produced base sequence. We then apply the MATLAB function, `runstest`, to these keystreams. The function returns a test decision for the null hypothesis that the values in the keystreams come in random order. Applying `runstest` to these keystreams returns value of $h = 0$. This test results indicate that the `runstest` does not reject the null hypothesis. This means that the values in all generated keystreams are in random order.

Table 1. Some constructed stream keys

| $i$ | $st_i$ |
|---|---|
| 1 | 5, 13, 22, 0, 2, 18, 9, 22, 2, 20, 1, 5, 24, 22, 21, 6, 3, 20, 17, 1, 9, 23, 11, 18, 10, 6, 8, 3, 2, 16, 14, 11, 5, 18, 6, 25, 16, 23, 24, 5 |
| 6 | 14, 24, 23, 25, 1, 12, 21, 22, 0, 13, 7, 17, 22, 13, 20, 24, 13, 9, 7, 18, 11, 22, 16,25, 3, 7, 12, 15, 2, 10, 19, 1, 17, 12, 3, 20, 18, 3, 15, 23 |
| 11 | 3, 21, 6, 10, 4, 24, 1, 16, 14, 2, 25, 17, 4, 16, 1, 16, 21, 20, 17, 17, 11, 15, 11, 8, 2, 0, 0, 19, 10, 2, 0, 19, 3, 12, 2, 19, 22, 15, 14, 21 |
| 16 | 17, 20, 7, 11, 16, 11, 1, 18, 1, 1, 12, 19, 19, 2, 13, 5, 12, 21, 15, 18, 17, 7, 10, 7, 9, 24, 17, 17, 16, 7, 15, 8, 7, 23, 22, 23, 15, 4, 19, 19 |
| 21 | 19, 8, 12, 15, 18, 1, 20, 1, 7, 19, 21, 21, 8, 0, 14, 16, 3, 8, 14, 4, 19, 11, 22, 18, 23, 4, 7, 14, 15, 1, 11, 21, 3, 16, 12, 6, 24, 19, 2, 18 |
| 26 | 9, 5, 13, 22, 0, 14, 18, 9, 22, 14, 6, 1, 5, 10, 20, 7, 6, 15, 4, 1, 13, 21, 19, 5, 14, 8, 14, 24, 19, 22, 22, 12, 17, 15, 18, 8, 3, 6, 7, 0 |
| 31 | 2, 14, 24, 23, 25, 16, 12, 21, 22, 15, 2, 7, 17, 11, 17, 9, 24, 2, 2, 0, 7, 0, 4, 2, 7, 7, 4, 6, 9, 14, 11, 10, 15, 23, 25, 21, 25, 12, 22, 20 |
| 36 | 1, 3, 21, 6, 10, 4, 24, 1, 16, 14, 2, 25, 17, 4, 16, 1, 16, 21, 20, 17, 17, 11, 15, 11, 8, 2, 0, 0, 19, 10, 2, 0, 19, 3, 12, 2, 19, 22, 15, 14 |

## 4. Conclusion

This work simulates randomness test to key sequences generated using SHATL. This kind of test is needed to ensure that there is no specific pattern of the key stream that can be utilized by an attacker to guess the key. The results of the randomness test show that the values in the encryption keys generated using SHATL, both in block and stream ciphers, come in random order. This condition indicates that the SHATL based encryption keys are eligible to increase security of the block and stream ciphers protocols.

**Acknowledgement**

Thanks to all colleagues at the Combinatorics, Graph Theory, and Network Topology (CGANT) research group, the University of Jember, Indonesia, for establishing this collaboration research.

## References

[1] P. M. Alcover, A. Guillamon, and M. C. Ruiz, A new randomness test for bit sequence, *Inform.*, **24** (3) (2013), 339–356.

[2] M. Bača, L. Brankovic, M. Lascsáková, O., Phanalasy, and A. Semaničová-Feňovčíková, On $d$-antimagic labelings of plane graphs, *Electron. J. Graph Theory Appl.*, **1** (1) (2013), 28–39.

[3] Dafik, A. K. Purnapraja, and R. Hidayat, Cycle-super antimagicness of connected and disconnected tensor product of graphs, *Procedia Comput. Sci.*, **74** (2015), 93–99.

[4] Dafik, Slamin, and D. Tanna, A. Semaničová-Feňovčíková, M. Bača, Constructions of $H$-antimagic graphs using smaller edge-antimagic graphs, *Ars Combin.*, **100** (2017), In Press.

[5] Dafik, M. Hasan, Y. N. Azizah, and I. H. Agustin, A generalized shackle of any graph $H$ admits a super $H$-antimagic total labeling, *Math. Comput. Sci. J.*, (2016). Submitted.

[6] P. L'Ecuyer, Testing random number generators, *Proc. 1992 Winter Simulation Conf.*, IEEE Press, Dec. (1992), 305–313.

[7] A. C. Prihandoko, Dafik, I. H. Agustin, D. Susanto, A. I. Kristiana, and Slamin, The construction of encryption key by using a super $H$-antimagic total graph, *Program Abstr. Asian Math. Conf.*, **AMC** (2016), 408, ISBN 978-602-74668-0-7.

[8] A. C. Prihandoko, Dafik, and I. H. Agustin, Implementation of super $H$-antimagic total graph on establishing stream cipher, *Indones. J. Combin.*, **3** (1) (2019), 14–23.

[9] M. E. Whitman and H. J. Mattord, *Principles Inf. Secur.*, (2012), Boston: Course Technology.