Robustness Test of SIMON-32, SPECK-32, and SIMECK-32 Algorithms Using Fixed-Point Attacks

B H Susanti\textsuperscript{a,1}, O J Permana\textsuperscript{b} and Amiruddin\textsuperscript{a}

\textsuperscript{a}Politeknik Siber dan Sandi Negara, Jalan Haji Usa Raya, Bogor, 16120, Indonesia
\textsuperscript{b}Badan Siber dan Sandi Negara, Jalan Harsono RM No 70, Ragunan, 12540, Indonesia

E-mail: bety.hayat@poltekssn.ac.id, oky.jati@bssn.go.id, amir@poltekssn.ac.id

Abstract. SIMON-32 and SPECK-32 are a family of lightweight block ciphers publicly released by the National Security Agency (NSA) in June 2013. Meanwhile, SIMECK-32 is a lightweight block cipher based on the combination of good design components from SIMON-32 and SPECK-32 block ciphers. In this paper, we tested the robustness of the SIMON-32, SPECK-32, and SIMECK-32 algorithms against fixed-point attacks. Block cipher fixed-points may facilitate fixed-point attacks if an adversary can control the block cipher key input. The results of fixed-point attacks on the SIMON-32, SPECK-32, and SIMECK-32 algorithms show that all three variants of the algorithm have fixed-points.

1. Introduction
Lightweight cryptography is an algorithm designed for implementation needs on devices with constrained resources [5]. In 2013, a new family of lightweight block ciphers was proposed, namely SIMON and SPECK [3]. In 2015, a new lightweight block cipher family was introduced called SIMECK [11]. SIMECK is the development of the SIMON and SPECK algorithm [6]. The robustness of an algorithm is an important factor and one of the parameters for assessing an algorithm. One method to determine the robustness of an algorithm is the application of fixed-point attacks to find the presence or absence of a key that can produce fixed-points using the underlined algorithm.

The fixed-point attack is an attack that utilizes the weakness of the iteration function in the hash function construction that uses the compression function construction $IV_i = f(IV_{i-1}, m_i)$ where $IV_{i-1}$ is the initial vector and $m_i$ is the $i$-th message in a compression function [4]. The implications of certain properties of block ciphers, which may be of no practical concern when used for encryption, must be carefully examined when such ciphers are used to construct iterated hash functions. The general danger is that such properties may facilitate adversarial manipulation of compression function inputs so as to allow prediction or greater control of outputs or relations between outputs of successive iterations. A fixed-point on a block cipher facilitates an attacker in carrying out a fixed-point attack on a scheme that uses the block cipher algorithm by controlling the key used [7]. Fixed-point attacks are performed on block cipher algorithms with certain key inputs such that $E_k(x) = x$ [7]. From the best of our knowledge, fixed-point attacks have been applied to several block cipher algorithms, including KeeLoq [2], SIMON-32 and SIMECK-32 [10], GOST2 [8], and Skinny [1] as well as several hash function

\textsuperscript{1} Corresponding author
algorithms. In this study, a fixed-point attack on the SIMON, SPECK and SIMECK algorithms with a 32-bit block size is used to determine the comparison of the robustness of the three algorithms to fixed-point attacks.

2. Basic Concepts

2.1. SIMON

The SIMON algorithm is a lightweight block cipher algorithm that has a block size variation of 32, 48, 64, 96, and 128 bits [3]. In addition, the SIMON algorithm has three types of key schedules whose usage depends on variations in block size. The SIMON algorithm is represented by SIMON $2n$ where $n$-bit is the word size. For $k \in GF(2)^n$, key-dependent on round function of SIMON $2n$ is two stages with the structure of Feistel $R_k : GF(2)^n \times GF(2)^n \rightarrow GF(2)^n \times GF(2)^n$ defined as follows: $R_k(x_{i+1}, x_i) = (x_i \oplus (S^1x_{i+1} \& S^8x_{i+1}) \oplus S^2x_{i+1} \oplus k, x_{i+1})$. For further detailed of SIMON, see [3]. Scheme of round function of SIMON is given in Figure 1.

Figure 1. Round function of SIMON [3]

2.2. SPECK

The SPECK algorithm is designed with the aim to make excellent performance on hardware and software. The notation for each different variant in SPECK is completely the same as that used for SIMON. For $k \in GF(2)^n$, key-dependent on round function of SPECK $2n$ is two stages with the structure of Feistel $R_k : GF(2)^n \times GF(2)^n \rightarrow GF(2)^n \times GF(2)^n$ and defined as follows: $R_k(x_{i+1}, x_i) = ((S^{(-a)}x_{i+1}x_i) \oplus k, S^3x_i \oplus (S^{(-a)}x_{i+1})x_i \oplus k)$, where $x_{i+1}$ is input value of 16 bits of the left part of each round, $x_i$ is input value of 16 bits of the right part of each round, $k_i$ is key of each round, $x_{i+2}$ is the result of 16 bits of the right part of each round, $x_{i+3}$ is the result of 16 bits of the left part of each round. For further detail of SPECK, see [3]. Scheme of round function of SPECK is given in Figure 2.

Figure 2. Round function of SPECK [3]

2
2.3. SIMECK
SIMECK algorithm is an algorithm designed as an improvement of the SIMON and SPECK algorithm. The notation for each different variant in SIMECK is the same as that used for SIMON and SPECK, it is just that the number of variations in the block size varies. The representation of the round SIMECK function is defined as follows: \( R(k_i)(x_{i+1}, x_i) = (x_i \oplus (x_{i+1} \& S^5x_{i+1}) \oplus S^1x_{i+1} \oplus k_i, x_{i+1}) \), where \( x_{i+1} \) is input value of 16 bits of the left part on each round, \( x_i \) is input value of 16 bits of the right part of each round, \( k_i \) is key used of each round, and \( x_{i+2} \) is the result of 16 bits of the left part of each round. Further detail of SIMECK, see [11]. Scheme of round function of SIMECK is given in Figure 3.

\[ \text{Figure 3. Round function of SIMECK [11]} \]

2.4. Fixed-point attack
The fixed-point attack is an attack that utilizes the weakness of the iteration function in the construction of hash functions which aims to find \( IV_{(i-1)} \) and \( x_i \) such that \( f(IV_{(i-1)}, x_i) = IV_{(i-1)} \) where \( IV_{(i-1)} \) is the initial vector and \( x_i \) is the \( i \)-th message in a compression function [9]. The existence of the message \( x_i \) will not affect the calculation of hashing message \( X \), because the message \( x_i \) that is inserted in the message \( X \) will produce \( IV_{(i-1)} \) which is the same as the previous initial vector [4]. Fixed-point attacks can also be performed on the block cipher algorithm. If a block cipher algorithm has a fixed-point value, then the fixed-point value can be used when carrying out a fixed-point attack on the hash function algorithm that uses the block cipher algorithm as its basic construction [7].

3. Research Method
This research used literature review and experimental methods. The literature review was carried out by reviewing several related papers, while the experimental method was conducted by applying the SIMON-32, SPECK-32, and SIMECK-32 algorithms using the C programming language. The implementation of these algorithms was carried out to simplify the calculation process. Next, an attack on the scheme was carried out using a fixed-point attack.

Briefly, the stages of this research are as follows:

a) Conducting a literature review.

b) Generating as many \( 2^{32} \) key inputs as long as 64 bits and performing the encryption process for each message with the key that has been generated with the message input value used is the initial value.

c) Comparing the ciphertext value with the message input value used. If \( E_k(x) = x \) is obtained, the key input value used is the fixed-point message key.
4. Implementing Fixed-Point Attacks
In this research, we experimented $2^{32}$ possibilities of inputs using 16 random plaintexts and 16 non-random plaintexts. Table 1, Table 2, and Table 3 summarize the result of fixed-point attacks experiments on SIMON-32, SPECK-32, and SIMECK-32, respectively, using both random and non-random plaintexts.

Table 1. Result of fixed-point attack on SIMON-32 algorithm using random plaintext and non-random plaintext

| No | Random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP | Non-random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP |
|----|------------------|--------------------------------------|-----------|----------------------|---------------------------------------|-----------|
| 1  | 3c50 ceb2        | 5296 5399 0000 0000 0000 7997 055d 0000 0000 | 2         | 0000 0000           | 0571 2006 0000 0000                   | 1         |
| 2  | 5f6a ce23        | 01bd 5544 0000 0000                    | 1         | 1111 1111           | -                                     | 0         |
| 3  | 7ab6 de34        | -                                    | 0         | 2222 2222           | -                                     | 0         |
| 4  | 76cb f12c        | 0294 217b 0000 0000 0000 0000 84ee 0000 0000 | 2         | 3333 3333           | cd81 7ff0 0000 0000                   | 1         |
| 5  | 8263 1735        | -                                    | 0         | 4444 4444           | -                                     | 0         |
| 6  | a3b4 930c        | -                                    | 0         | 5555 5555           | -                                     | 0         |
| 7  | a004 4bc2        | e875 9e00 0000 0000                   | 1         | 6666 6666           | cd61 afe9 0000 0000                   | 1         |
| 8  | a15c 234b        | a0af cfaf 0000 0000                   | 1         | 7777 7777           | eade 34d3 0000 0000                   | 1         |
| 9  | b7ae 46ec        | 723e 1a2f 0000 0000                   | 2         | 8888 8888           | -                                     | 0         |
| 10 | b324 76fe        | -                                    | 0         | 9999 9999           | 412e 3ec6 0000 0000                   | 1         |
| 11 | bc3c 662b        | d86f 84a5 0000 0000                   | 1         | aaaa aaaa           | -                                     | 0         |
| 12 | cc52 dc3b        | 27be 2701 0000 0000                   | 2         | bbbb bbbb           | 412e 3ec6 0000 0000                   | 1         |
| 13 | ceba 773b        | 437a a5a2 0000 0000                   | 3         | cccc cccc           | 6fa5 8eb8 0000 0000                   | 1         |
| 14 | e34c a80f        | e06a 73ce 0000 0000                   | 1         | dddd dddd           | 4382 8354 0000 0000                   | 1         |
| 15 | e78a c45f        | 1682 16ee 0000 0000                   | 1         | eeee eeee           | -                                     | 0         |
| 16 | f20a be73        | cc48 2861 0000 0000                   | 1         | ffff ffff           | -                                     | 0         |
|    | Total            | 18                                   |           | 8                    |                                        |           |

Table 1 shows that using both random and non-random plaintexts in conducting fixed-point attacks on the SIMON-32 algorithm, we found several keys producing a fixed-point. For random plaintexts, we found a total of 18 keys, whereas for non-random plaintexts, we found a total of 8 keys. These mean that the SIMON-32 is not resistant to fixed-point attacks.

Similar to Table 1, Table 2 also shows that using both random and non-random plaintexts in conducting fixed-point attacks on the SPECK-32 algorithm, we found several keys producing a fixed-point. For random plaintexts, we found a total of 15 keys, whereas for non-random plaintexts, we found a total of 17 keys. These mean that the SPECK-32 is not resistant to fixed-point attacks.

Similar to Table 1 and Table 2, Table 3 also shows that using both random and non-random plaintexts in conducting fixed-point attacks on the SIMECK-32 algorithm, we found several keys producing a fixed-point. For random plaintexts, we found a total of 8 keys, whereas for
Table 2. Result of fixed-point attack on SPECK-32 algorithm using random plaintext and non-random plaintext

| No | Random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP | Non-random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP |
|----|------------------|----------------------------------------|-----------|----------------------|----------------------------------------|-----------|
| 1  | 3e50 ccb2        | -                                      | 0         | 0000 0000            | 6f9b 90ab 0000 0000                    | 1         |
| 2  | 5fca ce23        | -                                      | 0         | 1111 1111            | 6b72 2194 0000 0000                    | 2         |
| 3  | 7ab6 dc34        | 6d52 872d 0000 0000                    | 3         | 2222 2222            | 1191 5520 0000 0000                    | 1         |
| 4  | 76cb f12c        | c7d0 9589 0000 0000                    | 1         | 3333 3333            | 786a 2846 0000 0000                    | 1         |
| 5  | 8263 1735        | 107d 5512 0000 0000                    | 3         | 4444 4444            | -                                      | 0         |
| 6  | a3b4 930c        | 5ad9 f6c7 0000 0000                    | 1         | 5555 5555            | 4621 7796 0000 0000                    | 2         |
| 7  | a004 4bc2        | 3517 20a8 0000 0000                    | 1         | 6666 6666            | e5d3 6093 0000 0000                    | 1         |
| 8  | a45c 234b        | 2b31 1fc3 0000 0000                    | 1         | 7777 7777            | e630 061d 0000 0000                    | 1         |
| 9  | b7ae 46ec        | -                                      | 0         | 8888 8888            | -                                      | 0         |
| 10 | b324 76fe        | -                                      | 0         | 9999 9999            | 789e 49a0 0000 0000                    | 4         |
|    |                  |                                        |           | 7588 6000 0000 0000   | 94f7 fb11 0000 0000                    |           |
|    |                  |                                        |           | 9f1b 8e16 0000 0000   | -                                      |           |
|    |                  |                                        |           |                      |                                        |           |
|    |                  |                                        |           |                      |                                        |           |
|    |                  |                                        |           |                      |                                        |           |
|    |                  |                                        |           |                      |                                        |           |
| 11 | bc3c 662b        | 34bf f564 0000 0000                    | 3         | aaaa aaaa            | -                                      | 0         |
| 12 | cc52 dc3b        | -                                      | 0         | bbbb bbbb            | a32b 2980 0000 0000                    | 1         |
| 13 | ceba 773b        | 87d3 3c31 0000 0000                    | 1         | eeee eeee            | cf97 aed1 0000 0000                    | 1         |
| 14 | e34c a80f        | 8ab4 6bf5 0000 0000                    | 1         | dddd dddd            | 341f 0237 0000 0000                    | 2         |
| 15 | e78a e45f        | -                                      | 0         | eeee eeee            | -                                      | 0         |
| 16 | f20a be73        | -                                      | 0         | ffff ffff            | -                                      | 0         |

non-random plaintexts, we found a total of 16 keys. These mean that the SIMECK-32 is not resistant to fixed-point attacks.

5. Effect Analysis of Algorithm Component Against Fixed-Point Occurrence

Based on the results described in the previous section, a traceability analysis of the SIMON-32, SPECK-32 and SIMECK-32 block cipher algorithms were performed. The search process was carried out to look for sufficient algorithm conditions that affect the occurrence of fixed-points in the three algorithms. In the encryption process, all three algorithms have modular operations and rotations. The list of operations used at the encryption stage using SIMON-32, SPECK-32, and SIMECK-32 algorithms can be seen in Tables 4, 5, and 6, respectively.

Information on one round encryption process on the SIMON-32, SPECK-32, and SIMECK-32 algorithms is defined into four stages and the process is differentiated in 16-bit left (L) and right
Table 3. Result of fixed-point attack on SIMECK-32 algorithm using random plaintext and non-random plaintext

| No | Random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP | Non-random Plaintext | Keys Producing Fixed-Point Found (KFP) | Total KFP |
|----|------------------|----------------------------------------|-----------|----------------------|----------------------------------------|-----------|
| 1  | 3c50 ceb2        | -                                      | 0         | 0000 0000            | ff92 4f27 0000 0000                    | 1         |
| 2  | 5fe0 ce23        | -                                      | 0         | 1111 1111            | 4f6b 6f4b 0000 0000                    | 3         |
|    |                  |                                        |           |                      | 54c6 d9fa 0000 0000                    |           |
|    |                  |                                        |           |                      | 9620 34fb 0000 0000                    |           |
| 3  | 7ab6 dc34        | 55cc 2338 0000 0000                    | 3         | 2222 2222            | 34c0 1140 0000 0000                    | 1         |
|    |                  | 9671 4535 0000 0000                    |           |                      |                                        |           |
|    |                  | 99c2 1a5d 0000 0000                    |           |                      |                                        |           |
| 4  | 76cb f12c        | 7276 f9e9 0000 0000                    | 1         | 3333 3333            | 8878 22ca 0000 0000                    | 1         |
| 5  | 8263 1735        | -                                      | 0         | 4444 4444            | -                                      | 0         |
| 6  | a3b4 930c        | -                                      | 0         | 5555 5555            | 18d4 5134 0000 0000                    | 2         |
|    |                  |                                        |           |                      | 19c6 7fe6 0000 0000                    |           |
| 7  | a004 4bc2        | f4d8 33ff 0000 0000                    | 1         | 6666 6666            | 45f7 fbe4 0000 0000                    | 1         |
| 8  | a45c 234b        | -                                      | 0         | 7777 7777            | -                                      | 0         |
| 9  | b7ae 46ec        | -                                      | 0         | 8888 8888            | -                                      | 0         |
| 10 | b324 76fe        | -                                      | 0         | 9999 9999            | c299 cd79 0000 0000                    | 1         |
| 11 | be3c 662b        | -                                      | 0         | aaaa aaaa            | 20fb 00d5 0000 0000                    | 3         |
|    |                  |                                        |           |                      | 98ae f259 0000 0000                    |           |
|    |                  |                                        |           |                      | e25f 37ea 0000 0000                    |           |
| 12 | cc52 dc3b        | -                                      | 0         | bbbb bbbb            | 62dc 21ab 0000 0000                    | 1         |
| 13 | ceba 773b        | dcc1 a4fb 0000 0000                    | 1         | cccc cccc            | e983 4cea 0000 0000                    | 1         |
| 14 | e34c a80f        | 37a8 7831 0000 0000                    | 2         | dddd dddd            | -                                      | 0         |
|    |                  | a86a 1776 0000 0000                    |           |                      |                                        |           |
| 15 | e78a e45f        | -                                      | 0         | eeee eeee            | 233b bae7 0000 0000                    | 1         |
| 16 | f20a be73        | -                                      | 0         | ffff ffff            | -                                      | 0         |
|    |                  |                                        |           |                      |                                        |           |
|    |                  |                                        |           |                      | Total 8                               | 16        |

(R). Operation \( S^j(x) \) has the meaning of shifting to the left \( j \) times with the value of \( x \) being 16-bit left (L) or right (R), the value of \( i \) indicates that the operation is found in round \( i \).

Table 4. Operation of modular arithmetic and rotation on SIMON-32

| Operation       | \( L_i \)                                                                 | \( R_i \)                                                                 |
|-----------------|---------------------------------------------------------------------------|---------------------------------------------------------------------------|
| **Stage1**      | \( L_i \)                                                                  | \( (S^1(L_i) \odot S^S(L_i)) \oplus R_i \)                                |
| **Stage2**      | \( L_i \)                                                                  | \( ((S^1(L_i) \odot S^S(L_i)) \oplus R_i) \odot S^2(L_i) \)               |
| **Stage3**      | \( L_i \)                                                                  | \( ((S^1(L_i) \odot S^S(L_i)) \oplus R_i) \odot S^2(L_i) \oplus k_i \)   |
| **Stage4**      | \( (((S^1(L_i) \odot S^S(L_i)) \oplus R_i) \odot S^2(L_i)) \oplus k_i \) | \( L_i \)                                                                  |

In the SIMON-32 algorithm, Stage 1 represents the sum of modulo 16 between 16 left bits, which are rotated left once, with 16 left bits rotated left eight times. The results of the operation are then used for XOR operations with 16-bit right. On Stage 1, the encryption of the SIMON-
Table 5. Operation of modular arithmetic and rotation on SPECK-32

| Operation | $L_i$ | $R_i$ |
|-----------|------|------|
| Stage 1   | $L_i$ | $R_i$ |
| Stage 2   | $L_i$ | $R_i$ |
| Stage 3   | $L_i$ | $R_i$ |
| Stage 4   | $L_i$ | $R_i$ |

Table 6. Operation of modular arithmetic and rotation on SIMECK-32

| Operation | $L_i$ | $R_i$ |
|-----------|------|------|
| Stage 1   | $L_i$ | $(L_i \circ S^5(L_i)) \oplus R_i$ |
| Stage 2   | $L_i$ | $((L_i \circ S^5(L_i)) \oplus R_i) \oplus S^4(L_i)$ |
| Stage 3   | $L_i$ | $(((L_i \circ S^5(L_i)) \oplus R_i) \oplus S^4(L_i)) \oplus k_i$ |
| Stage 4   | $L_i$ | $((L_i \circ S^5(L_i)) \oplus R_i) \oplus S^4(L_i) \oplus k_i$ |

Table 7. Tracing the effect of the SIMON-32 algorithm on the occurrence of fixed-points

| Round | Operation | $L$ | $R$ |
|-------|-----------|-----|-----|
| 1     | Stage 1   | $L_0$ | $(S^1(L_0) \circ S^8(L_0)) \oplus R_0$ |
|       | Stage 2   | $L_0$ | $((S^1(L_0) \circ S^8(L_0)) \oplus R_0) \oplus S^2(L_0)$ |
|       | Stage 3   | $L_0$ | $(((S^1(L_0) \circ S^8(L_0)) \oplus R_0) \oplus S^2(L_0)) \oplus k_0$ |
|       | Stage 4   | $L_0$ | $((S^1(L_0) \circ S^8(L_0)) \oplus R_0) \oplus S^2(L_0) \oplus k_0$ |
|       | ...       | ...  | ... |
| 31    | Stage 1   | $L_{30}$ | $(S^1(L_{30}) \circ S^8(L_{30})) \oplus R_{30}$ |
|       | Stage 2   | $L_{30}$ | $((S^1(L_{30}) \circ S^8(L_{30})) \oplus R_{30}) \oplus S^2(L_{30})$ |
|       | Stage 3   | $L_{30}$ | $(((S^1(L_{30}) \circ S^8(L_{30})) \oplus R_{30}) \oplus S^2(L_{30})) \oplus k_0$ |
|       | Stage 4   | $L_{30}$ | $((S^1(L_{30}) \circ S^8(L_{30})) \oplus R_{30} \oplus S^2(L_{30})) \oplus R_0$ |
| 32    | Stage 1   | $R_0$  | $(S^1(L_{31}) \circ S^8(L_{31})) \oplus R_{31}$ |
|       | Stage 2   | $R_0$  | $((S^1(L_{31}) \circ S^8(L_{31})) \oplus R_{31}) \oplus S^2(L_{31})$ |
|       | Stage 3   | $R_0$  | $(((S^1(L_{31}) \circ S^8(L_{31})) \oplus R_{31}) \oplus S^2(L_{31})) \oplus k_0$ |
|       | Stage 4   | $R_0$  | $((S^1(L_{31}) \circ S^8(L_{31})) \oplus R_{31} \oplus S^2(L_{31})) \oplus R_0$ |
32 algorithm results in changes to the right 16-bit value. Meanwhile, Stage 2 represents the XOR process between the results of Phase 1 on the right 16-bit to the left 16-bit value which is rotated to the left twice. Furthermore, Stage 3 represents the XOR process between the results of Stage 2 on the right 16-bit with the $i$-th subkey. Stage 4 represents the swap process between the left 16-bit value and the right 16-bit value.

Table 8. Tracing the effect of the SPECK-32 algorithm on the occurrence of fixed-points

| Round | Operation | $L$ | $R$ |
|-------|-----------|-----|-----|
| 1     | $S^{-7}(L_0)$ | $R_0$ | $R_0$ |
|       | $S^{-7}(L_0) \circ R_0$ | $S^2(R_0)$ | $S^2(R_0)$ |
|       | $(S^{-7}(L_0)) \circ R_0 \oplus k_0$ | $S^2(R_0) \oplus ((S^{-7}(L_0)) \circ R_0) \oplus k_0$ | $S^2(R_0) \oplus ((S^{-7}(L_0)) \circ R_0) \oplus k_0$ |
| ...   | ...       | ... | ... |
| 31    | $S^{-7}(L_{20})$ | $R_{20}$ | $R_{20}$ |
|       | $S^{-7}(L_{20}) \circ R_{20}$ | $S^2(R_{20})$ | $S^2(R_{20})$ |
|       | $(S^{-7}(L_{20})) \circ R_{20} \oplus k_{20}$ | $S^2(R_{20}) \oplus ((S^{-7}(L_{20})) \circ R_{20}) \oplus k_{20}$ | $S^2(R_{20}) \oplus ((S^{-7}(L_{20})) \circ R_{20}) \oplus k_{20}$ |
| 32    | $S^{-7}(L_{21})$ | $R_{21}$ | $R_{21}$ |
|       | $S^{-7}(L_{21}) \circ R_{21}$ | $S^2(R_{21})$ | $S^2(R_{21})$ |
|       | $(S^{-7}(L_{21})) \circ R_{21} \oplus L_0$ | $(R_0 \oplus L_0) \oplus L_0$ | $(R_0 \oplus L_0) \oplus L_0$ |

In the SPECK-32 algorithm, Stage 1 represents the right-hand rotation process $\alpha$ times on the left 16-bit with the $\alpha$ value in this study is 7. Then, Stage 2 represents the XOR process between the results of Stage 1 on the 16-bit left with a value of 16-right bit. Stage 3 represents the XOR process between the results of Stage 2 on the left 16-bit which is rotated to the left once. Stage 4 represents the swap process between the left 16-bit value and the right 16-bit value.

In searches that had been previously done, operations that affect the occurrence of fixed-points are marked in red text. A summary of the results of the analysis of the encryption process for the SIMON-32, SPECK-32, and SIMECK-32 algorithms are given in Table 7, Table 8, and Table 9 respectively.
Table 9. Tracing the effect of the SIMECK-32 algorithm on the occurrence of fixed-points

| Round | Operation | L        | R            |
|-------|-----------|----------|--------------|
| 1     | Stage1    | $L_0$    | $(L_0 \oplus S^5(L_0)) \oplus R_0$ |
|       | Stage2    | $L_0$    | $((L_0 \oplus S^5(L_0)) \oplus R_0) \oplus S^1(L_0)$ |
|       | Stage3    | $L_0$    | $((L_0 \oplus S^5(L_0)) \oplus R_0) \oplus S^1(L_0) \oplus k_0$ |
|       | Stage4    | $((L_0 \oplus S^5(L_0)) \oplus R_0) \oplus S^1(L_0) \oplus k_0$ |
| ...   | ...       | ...      | ...          |

Stage 31

| Round | Operation | L        | R            |
|-------|-----------|----------|--------------|
|       | Stage1    | $L_{30}$ | $(L_{30} \oplus S^5(L_{30})) \oplus R_{30}$ |
|       | Stage2    | $L_{30}$ | $((L_{30} \oplus S^5(L_{30})) \oplus R_{30}) \oplus S^1(L_{30})$ |
|       | Stage3    | $L_{30}$ | $((L_{30} \oplus S^5(L_{30})) \oplus R_{30}) \oplus S^1(L_{30}) \oplus k_0$ |
|       | Stage4    | $R_0$    | $((L_{30} \oplus S^5(L_{30})) \oplus R_{30}) \oplus S^1(L_{30}) \oplus k_0$ |

Stage 32

| Round | Operation | L        | R            |
|-------|-----------|----------|--------------|
|       | Stage1    | $R_0$    | $(L_{31} \oplus S^5(L_{31})) \oplus R_{31}$ |
|       | Stage2    | $R_0$    | $((L_{31} \oplus S^5(L_{31})) \oplus R_{31}) \oplus S^1(L_{31})$ |
|       | Stage3    | $R_0$    | $((L_{31} \oplus S^5(L_{31})) \oplus R_{31}) \oplus S^1(L_{31}) \oplus k_0$ |
|       | Stage4    | $L_0$    | $((L_{31} \oplus S^5(L_{31})) \oplus R_{31}) \oplus S^1(L_{31}) \oplus L_0$ |

In each of Table 7, 8, and 9, $L_0$ is 16-bit initial left input value, $R_0$ is 16-bit initial right input value, $IV_L$ is 16-bit left IV input value, and $IV_R$ is 16-bit right IV input value. The stage 2 value for the right part of round 31 is $a$ and the value of stage 2 for the right part of round 32 is $b$. It can be seen that the fixed-point starts at the 31<sup>rd</sup> round for the SIMON-32 and SIMECK-32 algorithms, while in the SPECK-32 algorithm, it occurs at the 32<sup>nd</sup> round of the 3<sup>rd</sup> stage. In the SIMON-32 and SIMECK-32 algorithm, if $k_{31} = a \oplus R_0$ and $k_{32} = b \oplus L_0$, then $E_K(x) = x$, while in the SPECK algorithm, if $k_{22} = a \oplus L_0$ and $k_{23} = L_0 \oplus R_0$, then $E_K(x) = x$.

6. Conclusion
In this study, we have conducted fixed-point attack experiments on three Feistel-structured block cipher algorithms i.e. SIMON-32, SPECK-32, and SIMECK-32. The results showed that in the three studied algorithms there are some certain messages or keys which when used in the encryption process would produce the same ciphertext value as the plaintext input used. Additionally, in the three algorithms, it is also found some keys that produce fixed-points, but in the SIMECK-32 algorithm, the number of fixed-point producing keys is less than those in the SIMON-32 and SPECK-32 algorithms.

Acknowledgments
This research was supported by Politeknik Siber dan Sandi Negara.
References

[1] Ahmadi S and Aref M R 2017 Improved Fixed Point Attack on GOST2 ISCIIS:54-57

[2] Bard G V 2009 The Fixed-point attack in Algebraic Cryptanalysis (Boston: Springer)

[3] Beaulieu R, Shors D, Smith J, and Treatman-clark S 2013 The Simon and Speck Families of Lightweight Block Ciphers Cryptology ePrint Archive (National Security Agency, USA):1–42 Retrieved (http://eprint.iacr.org)

[4] Danda M and Renddy K 2007 Design and Analysis of Hash Function

[5] Katagi M and Moriai S 2008 Lightweight Cryptography for the Internet of Things Sony Corporation 7–10 Retrieved (http://www.iab.org/wp-content/IAB-uploads/2011/03/Kaftan.pdf).

[6] Köbl S and Arnab R 2015 “A Brief Comparison of Simon and Simeck Cryptology ePrint Archive: Report 2015/706 Retrieved (https://eprint.iacr.org/2015/706)

[7] Menezes A J, van Oorschot P C, and Vanstone S A 1997 Handbook of Applied Cryptography 1st ed (Boca Raton: CRC Press)

[8] Posteuca R and Negara G 2017 New Related-Key Attacks and Properties of Skinny 64-128 Cipher Proceedings of the Romanian Academy, Series A vol 18, pp 333-350

[9] Preneel B, Govaerts R, and Vandewalle J 1993 Hash Functions Based on Block Ciphers: A Synthetic Approach Advances in Cryptology-CRYPTO ’93 368–378 Retrieved (http://link.springer.com/chapter/10.1007/3-540-48329-2-31).

[10] Saputra S R Y 2016 Perbandingan Algoritma SIMON-32 Dan SIMECK-32 Serta Penerapannya Pada Skema Fungsi Hash Preneel-Govaerts-Vandewalle (PGV)-5 Berdasarkan Collision Test Dan Fixed Point Attack Sekolah Tinggi sandi Negara (Unpublished thesis)

[11] Yang G, Zhu B, Suder V, Aagaard M D, and Gong G 2015 The Simeck Family of Lightweight Block Ciphers Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 9293 (June):307—329