# Ciphertext-Only Attack on Linear Feedback Shift Register-Based Esmaeili-Gulliver Cryptosystem

Yongwoo Lee, Young-Sik Kim, and Jong-Seon No

***Abstract*— Esmaeili and Gulliver recently proposed a secret key cryptosystem based on error-correcting codes in which a codeword modified by random insertions, deletions, and errors is used as a ciphertext. The secret keys used in this cryptosystem consist of random numbers generated by synchronized random number generators that are implemented using two distinct linear feedback shift registers (LFSRs) in each encryptor and decryptor, respectively. In this letter, we propose a ciphertext-only attack to break the Esmaeili-Gulliver cryptosystem based on LFSRs. The proposed attack requires $O(n)$ consecutive ciphertexts, where $n$ is the number of shift registers in the LFSR, which is the secret key size. The proposed attack consists of two steps, and the time complexity of the first step is linear in the secret key size while the second step is a polynomial-time algorithm.**

***Index Terms*— Code-based cryptosystem, cryptography, linear feedback shift register (LFSR), random number generator, secret key cryptosystem.**

## I. Introduction

POLYNOMIAL-TIME quantum algorithms for prime factorization and for the discrete logarithm problem have been proposed [1]. Hence, RSA and elliptic curve cryptosystems will no longer be used after the advent of quantum computing. Meanwhile, the public key cryptosystem based on algebraic coding theory and first proposed by McEliece [2] has resistance against attacks on quantum computers. Hence, the use of error correcting codes in cryptosystems has been extensively studied. Esmaeili and Gulliver recently proposed a code-based secret key cryptosystem [3]. In this cryptosystem, the states of random number generators, which are implemented by two synchronous linear feedback shift registers (LFSRs), are treated as shared secret keys. Code-related information such as the generator matrix and error correction capability does not need to be secret. Therefore, this cryptosystem is distinctive because of its small key size.

In this letter, it is shown that the Esmaeili-Gulliver cryptosystem is vulnerable when random numbers are generated from LFSRs as in [3]. We propose a ciphertext-only attack to reveal the secret keys of the Esmaeili-Gulliver cryptosystem by using consecutive ciphertexts. The keys revealed by this attack include the states and the generator polynomials of the two LFSRs. We also suggest methods that provide resistance against the proposed attack on the Esmaeili-Gulliver cryptosystem.

Manuscript received November 6, 2016; revised December 26, 2016 and January 7, 2017; accepted January 9, 2017. Date of publication January 17, 2017; date of current version May 6, 2017. This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (R-20160229-002941, Research on Lightweight Post-Quantum Crypto-Systems for IoT and Cloud Computing) and this research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2014R1A2A2A01006870). The associate editor coordinating the review of this letter and approving it for publication was Mark F Flanagan. *(Corresponding author: Y.-S. Kim.)*

Y. Lee and J.-S. No are with the Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea.

Y.-S. Kim is with the Department of Information and Communication Engineering, Chosun University, Gwangju 61452, South Korea (e-mail: iamyskim@chosun.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2017.2654238

## II. Esmaeili-Gulliver Cryptosystem

In the Esmaeili-Gulliver cryptosystem [3], the ciphertext is a codeword which is modified by random puncturing, insertion of random numbers, and addition of random errors. The random numbers are generated by two random number generators, which are secretly shared between the encryptor and the decryptor. Esmaeili and Gulliver particularly proposed the use of two LFSRs in the encryptor and decryptor as the random number generator. Although it is mentioned that other random number generators can be employed, the paper's main focus is on the case where the random numbers are generated using LFSRs.

Alice has two LFSRs of different lengths and Bob also has the same synchronized LFSRs, and they have the same initial conditions of these LFSRs as shared secrets. Consequently, Bob can find the exact locations of inserted and deleted bits and completely remove the deliberately added errors based on the states of the two LFSRs. Therefore, the Esmaeili-Gulliver cryptosystem is a secret key cryptosystem, while the usual code-based cryptosystems are public key cryptosystems.

### *A. Encryption*

Alice and Bob choose an $(n, k)$ linear code and an integer $\beta$ that determines how many sub-blocks a codeword is divided into. Note that neither the selected code nor $\beta$ is confidential.

Let $\boldsymbol{m}$ be the plaintext and $\boldsymbol{c}$ be the corresponding codeword. Let $\boldsymbol{H}$ be a parity check matrix of the code and $\boldsymbol{H}^{-1}$ be a right inverse of $\boldsymbol{H}$. In other words, $\boldsymbol{H}\boldsymbol{H}^{-1} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the $(n-k)\times(n-k)$ identity matrix. $\boldsymbol{H}^{-1}$ is not unique because $\boldsymbol{H}$ is not an invertible matrix. Therefore, Alice and Bob build $\boldsymbol{H}^{-1}$ using the same method explained in [4].

Alice and Bob securely share the initial states of the two LFSRs and their generator polynomials as secret keys. The $(n-k)$-bit state of the first LFSR with $(n-k)$ shift registers is used to generate random errors and the $\beta(\left\lfloor \log_2 \frac{n}{\beta} \right\rfloor + 1)$-bit state of the second LFSR with $\beta(\left\lfloor \log_2 \frac{n}{\beta} \right\rfloor + 1)$ shift registers is used to determine the locations of bit insertion or bit deletion in the ciphertext.

Let $\boldsymbol{s}$ denote the state of the first LFSR. Alice computes the modified codeword $\boldsymbol{c}_e = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{e} = \boldsymbol{s} \times (\boldsymbol{H}^{-1})^T$. Here, $\boldsymbol{e}$ has Hamming weight larger than the error correction capability. Therefore the ciphertext cannot be corrected by the usual decoding. If the Hamming weight of $\boldsymbol{e} = \boldsymbol{s} \times (\boldsymbol{H}^{-1})^T$ is less than or equal to the error correction capability, both

Alice and Bob simply discard it and use the next state of the first LFSR.

After $\boldsymbol{c}_e$ is generated, it is divided into $\beta$ sub-blocks. In addition, the current state of the second LFSR is divided into $\beta$ sub-blocks of $(\lfloor\log_2\frac{n}{\beta}\rfloor+1)$ bits. Each sub-block determines the bit insertion/deletion of the corresponding sub-block of $\boldsymbol{c}_e$. If the first bit in the sub-block of the state of the second LFSR is 0, the decimal equivalence of the remaining $\lfloor\log_2\frac{n}{\beta}\rfloor$ bits will be the bit position of the ciphertext sub-block to be deleted. Otherwise, a bit generated as in [3] will be inserted in the position. The total number of bit insertions and deletions is given as $\beta$. After the bit insertion/deletion process, the output $\boldsymbol{c}'$, the ciphertext, is sent to Bob.

## B. Decryption

To decrypt the ciphertext, Bob removes the inserted bits using the state of the synchronized second LFSR. Let $\boldsymbol{r}'$ denote the received ciphertext after the inserted bits are removed. Using the state of the first LFSR and $\boldsymbol{H}^{-1}$, Bob can construct the error vector $\boldsymbol{e}$ that is added during encryption. Next, he performs the same puncturing operation as the encryptor for the error vector; let $\boldsymbol{e}'$ denote the punctured error vector. Then, Bob can completely remove the errors from $\boldsymbol{r}'$ using $\boldsymbol{r}=\boldsymbol{r}'+\boldsymbol{e}'$, where $\boldsymbol{r}$ is a punctured version of the codeword that corresponds to the plaintext $\boldsymbol{m}$. Bob can recover $\boldsymbol{m}$ using the decoding process as he knows the positions of the punctured bits.

Note that there can be up to $\beta$ deletions in a ciphertext. Additionally, it is possible to have $t$ errors introduced by the channel. Thus, $\beta$ should be selected so as to satisfy $\beta \le d_{\min}-2t$, where $d_{\min}$ is the minimum distance of the code.

## III. Attack on the Esmaeili-Gulliver Cryptosystem

In this section, we propose a ciphertext-only attack to discover the states and the two generator polynomials of the two LFSRs, which are the secret keys of the LFSR-based Esmaeili-Gulliver cryptosystem. The proposed attack consists of two steps. First, we find the generator polynomial and the states of the second LFSR which are utilized for insertion/deletion from consecutive ciphertexts. Next, using the same ciphertexts, another ciphertext-only attack is carried out against the first LFSR which generates the error vector whose Hamming weight is greater than the code's error correction capability. For convenience, we refer to the state of the second LFSR as the *outer key* and that of the first LFSR as the *inner key*.

In [3], a specific way in which LFSRs are clocked between two consecutive instances is not explicitly suggested as the study only considers the case of individual plaintext-ciphertext samples. Since the states of LFSRs are secret, it is not possible to use complicated ways of updating the states in the application of the Esmaeili-Gulliver cryptosystem for long messages. In this case, the 'key scheduling' should either be specified or additional secret information related to key updating should be securely shared. In this case, the key size will be large. Therefore, it would be a natural choice to update states normally, which means that the LFSRs are clocked once or a constant number of times. We will assume that the LFSRs are clocked exactly once between consecutive instances for simplicity. Note however that essentially the same method of attack works if they are clocked any larger (constant) number of times between consecutive instances; the only difference lies in choosing the ciphertext pairs to be used in the attack.

## A. Attack on Outer Key

To find the outer key, the attacker collects consecutive ciphertexts denoted by $\boldsymbol{c}'_1,\boldsymbol{c}'_2,\boldsymbol{c}'_3,\ldots$ in order. The $l$-th sub-block of ciphertext $\boldsymbol{c}'_i$ is represented as $\boldsymbol{c}'_{i,l}$, and thus $\boldsymbol{c}'_i=(\boldsymbol{c}'_{i,1},\boldsymbol{c}'_{i,2},\ldots,\boldsymbol{c}'_{i,\beta})$. Let $|\boldsymbol{c}'_i|$ denote the number of bits in $\boldsymbol{c}'_i$, which is referred to as the length of $\boldsymbol{c}'_i$.

Let $\boldsymbol{K}^i_{out}$ be the outer key used in $\boldsymbol{c}'_i$. Note that the length of $\boldsymbol{K}^i_{out}$ is $\beta(\left\lfloor\log_2\frac{n}{\beta}\right\rfloor+1)$, and $\boldsymbol{K}^i_{out}$ can be expressed as $(d_{i,1},\boldsymbol{L}_{i,1},d_{i,2},\boldsymbol{L}_{i,2},\ldots,d_{i,\beta},\boldsymbol{L}_{i,\beta})$, where $d_{i,l}$ is a binary value which indicates whether a bit has been inserted or deleted and $\boldsymbol{L}_{i,l}$ is $\lfloor\log_2\frac{n}{\beta}\rfloor$-bit information to specify the location of an inserted or deleted bit in the $l$-th sub-block of the ciphertext.

Remember that this cryptosystem uses the outer key $\boldsymbol{K}^i_{out}$ to determine bit insertion/deletion and the bit locations for $\beta$ sub-blocks of $\boldsymbol{c}'_i$. In particular, the $(1+(l-1)(\lfloor\log_2\frac{n}{\beta}\rfloor+1))$-th bit of $\boldsymbol{K}^i_{out}$ for $1\le l\le\beta$, which is denoted by $d_{i,l}$, determines bit insertion/deletion for the given ciphertext sub-block $\boldsymbol{c}'_{i,l}$. Since the state of LFSR is left-shifted at a time, we can see that, after $\lfloor\log_2\frac{n}{\beta}\rfloor+1$ cyclic shifts from $i$, $d_{j,l}=d_{i,l+1}$ for $1\le l\le\beta-1$ with $j=i+(\lfloor\log_2\frac{n}{\beta}\rfloor+1)$. In other words, the relationship between $\boldsymbol{K}^i_{out}$ and $\boldsymbol{K}^j_{out}$ is given by $\boldsymbol{K}^j_{out}=(d_{j,1},\boldsymbol{L}_{j,1},d_{j,2},\boldsymbol{L}_{j,2},\ldots,d_{j,\beta},\boldsymbol{L}_{j,\beta})=(d_{i,2},\boldsymbol{L}_{i,2},\ldots,d_{i,\beta},\boldsymbol{L}_{i,\beta},d_{j,\beta},\boldsymbol{L}_{j,\beta})$. This means that the lengths of some sub-blocks of the $i$-th and $j$-th ciphertexts, $|(\boldsymbol{c}_{i,2},\boldsymbol{c}_{i,3},\ldots,\boldsymbol{c}_{i,\beta})|$ and $|(\boldsymbol{c}_{j,1},\boldsymbol{c}_{j,2},\ldots,\boldsymbol{c}_{j,\beta-1})|$, are equal. Therefore, we can determine the exact values of $d_{i,1}$ and $d_{j,\beta}$ by comparing the length of the two ciphertexts $\boldsymbol{c}'_i$ and $\boldsymbol{c}'_j$ as follows.

Let $u_i=|\boldsymbol{c}'_i|-|\boldsymbol{c}'_{i+(\lfloor\log_2\frac{n}{\beta}\rfloor+1)}|$. Then we have the following three cases:

Case 1) If $u_i>0$, $\boldsymbol{c}'_{i,1}$ is an inserted sub-block and $\boldsymbol{c}'_{j,\beta}$ is a deleted sub-block.

Case 2) If $u_i<0$, $\boldsymbol{c}'_{i,1}$ is a deleted sub-block and $\boldsymbol{c}'_{j,\beta}$ is an inserted sub-block.

Case 3) Otherwise, both $\boldsymbol{c}'_{i,1}$ and $\boldsymbol{c}'_{j,\beta}$ are inserted or deleted sub-blocks.

Therefore, $d_{i,1}$ and $d_{j,\beta}$ are disclosed in Case 1) and Case 2). Consequently, the success probability of determining the values of $d_{i,1}$ and $d_{j,\beta}$ is 1/2. The undetermined bits in Case 3) can be obtained by determining whether $\boldsymbol{c}'_{i+\beta(\lfloor\log_2\frac{n}{\beta}\rfloor+1),1}$ and $\boldsymbol{c}'_{i+\beta(\lfloor\log_2\frac{n}{\beta}\rfloor+1)+(\lfloor\log_2\frac{n}{\beta}\rfloor+1),\beta}$ are inserted or deleted sub-blocks using $u_{i+\beta(\left\lfloor\log_2\frac{n}{\beta}\right\rfloor+1)}$ based on the above three cases. Here, $d_{i+\beta(\lfloor\log_2\frac{n}{\beta}\rfloor+1),1}$ and $d_{j,\beta}$ are equal for $j=i+(\lfloor\log_2\frac{n}{\beta}\rfloor+1)$. In other words, we have $d_{i+\beta(\lfloor\log_2\frac{n}{\beta}\rfloor+1),1}=d_{j,\beta}=d_{i,1}$ when $u_i=0$. If we have $u_{i+\beta(\left\lfloor\log_2\frac{n}{\beta}\right\rfloor+1)}=0$ again with a probability of 1/2, we need to check $u_{i+2\beta(\left\lfloor\log_2\frac{n}{\beta}\right\rfloor+1)}$ to find all the previous undetermined bits. That is, we can find $d_{i,1}, d_{i+\beta(\lfloor\log_2 \frac{n}{\beta}\rfloor+1),1}, d_{i+2\beta(\lfloor\log_2 \frac{n}{\beta}\rfloor+1),1}, \ldots$ by iteratively using $u_{i+m\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor+1)}$ for $m > 0$. Thus, we can determine $m+1$ bits at the same time because the equality in Case 3) gives us all the previous undetermined bits when $\boldsymbol{c}'_{i+m\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor+1),1}$ is finally determined as either an 'inserted' or a 'deleted' sub-block. It is easy to verify that the expected number of operations to obtain the bits is $\sum_{k=1}^{\infty} k \times \frac{1}{2^k} = 2$.

By repeating this procedure, we can find the consecutive $2\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$ bits, which are a sufficient number of bits for the Berlekamp–Massey algorithm [5], [6] to discover the generator polynomial of the second LFSR. Hence, after obtaining $2\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$ bits of the second LFSR output, the attacker can determine the generator polynomial and the initial condition for the outer key.

In the proposed attack on the outer key, $2\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$ operations and less than $3\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$ ciphertexts are needed to determine $2\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$ consecutive LFSR outputs on average. From simulations, and using the same parameters as in [3], i.e., $n = 900, k = 810$, and $\beta = 20$, it is found that the expected number of required ciphertexts is about $328 < 3\beta(\left\lfloor\log_2 \frac{n}{\beta}\right\rfloor + 1)$. This result is obtained from $10^5$ experiments that are carried out by using primitive polynomials in [8] to generate LFSRs.

## *B. Attack on Inner Key*

The attack on the inner key is done after obtaining the outer key. To simplify the analysis, we assume that there is no random channel error at first. Suppose that the attacker knows the consecutive ciphertexts $\boldsymbol{c}'_1, \boldsymbol{c}'_2, \ldots$. From the recovered outer keys, the attacker can completely remove the randomly inserted bits and fill the deleted bits with zeros. Suppose that $\boldsymbol{v}_i$ denotes the partially recovered codeword from $\boldsymbol{c}'_i$ which is recovered by removing randomly inserted bits and filling the deleted bits with zeros. We can then calculate $\boldsymbol{H}\boldsymbol{v}_i^T$ as

$$\begin{aligned}\boldsymbol{H}\boldsymbol{v}_i^T &= \boldsymbol{H}(\boldsymbol{c}_i + \boldsymbol{\epsilon}_i + \boldsymbol{e}_i)^T \\ &= \boldsymbol{H}\boldsymbol{c}_i^T + \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{H}\boldsymbol{e}_i^T = \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{H}\boldsymbol{e}_i^T \\ &= \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{H}(\boldsymbol{s}_i(\boldsymbol{H}^{-1})^T)^T = \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{s}_i^T\end{aligned}$$

where $\boldsymbol{\epsilon}_i$ is the error vector given by filling zeros in the deleted bit positions. Note that the maximum Hamming weight of $\boldsymbol{\epsilon}_i$ is $\beta$ because the maximum number of deleted bits is $\beta$.

For two consecutive, partially recovered codewords $\boldsymbol{v}_i$ and $\boldsymbol{v}_{i+1}$, we have $\boldsymbol{v}_i = \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{s}_i$ and $\boldsymbol{v}_{i+1} = \boldsymbol{H}\boldsymbol{\epsilon}_{i+1}^T + \boldsymbol{s}_{i+1}$. It should be noted that $\boldsymbol{s}_{i+1}$ is obtained from $\boldsymbol{s}_i$ by 1-bit left-shifting and then adding a newly generated bit at the right-most side as they are consecutive states of the first LFSR. That is, for $\boldsymbol{s}_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,n-k})$ and $\boldsymbol{s}_{i+1} = (s_{i+1,1}, s_{i+1,2}, \ldots, s_{i+1,n-k})$, we have $s_{i,m} = s_{i+1,m-1}$ for $2 \le m \le n-k$.

In order to carry out the proposed attack on inner keys, we need to select two consecutive ciphertexts $\boldsymbol{c}_i$ and $\boldsymbol{c}_{i+1}$ such that there is no overlapping of the deleted positions of the two consecutive ciphertexts. It is not difficult to verify that for sub-blocks of the consecutive ciphertexts, overlapping of the deleted positions occurs only when the corresponding sub-blocks of the outer keys are all zero. Hence, the probability of a pair of consecutive ciphertexts having a nonzero overlap of deleted positions, denoted by $p_{\text{overlapped}}$, is $1 - \left(1 - \frac{1}{2^{1+\lfloor\log_2 n/\beta\rfloor}} \times \frac{1}{2}\right)^{\beta}$. For $n = 900$ and $\beta = 20$, as suggested by Esmaeili and Gulliver, $p_{\text{overlapped}} = 0.145$. Therefore, there are enough ciphertext pairs with no overlapping of the deleted bit positions that can be used for the proposed attack.

Suppose that a ciphertext pair $\boldsymbol{c}'_i$ and $\boldsymbol{c}'_{i+1}$ is selected. Let $D_i$ be the index set of the deleted bits in $\boldsymbol{c}'_i$. Note that the deleted bit positions can be obtained from the outer keys, which are completely obtained by the previous attack on the outer key. Let $\boldsymbol{H}'_i$ be a matrix generated by replacing the $l$-th column of $\boldsymbol{H}$ with an all-zero column vector for every $l \notin D_i$. Then from the equation for the $i$-th ciphertext given by

$$\boldsymbol{H}\boldsymbol{\epsilon}_i^T = \boldsymbol{H}\boldsymbol{v}_i^T + \boldsymbol{s}_i^T \tag{1}$$

$\boldsymbol{H}$ on the left-hand side can be replaced with $\boldsymbol{H}'_i$ because the $l$-th element of $\boldsymbol{\epsilon}_i$ is zero for $l \notin D_i$. Also,

$$\begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{\epsilon}_i^T = \begin{bmatrix}\boldsymbol{H}\boldsymbol{v}_i^T \\ 0\end{bmatrix} + \begin{bmatrix}\boldsymbol{s}_i^T \\ 0\end{bmatrix} \tag{2}$$

holds true, where $\boldsymbol{0}$ is an $n$-tuple zero vector and 0 is a zero bit. Similarly, for the $(i+1)$-th ciphertext, we have

$$\begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\boldsymbol{\epsilon}_{i+1}^T = \begin{bmatrix}0 \\ \boldsymbol{H}\boldsymbol{v}_{i+1}^T\end{bmatrix} + \begin{bmatrix}0 \\ \boldsymbol{s}_{i+1}^T\end{bmatrix}. \tag{3}$$

By adding (2) and (3), the left-hand side is given as

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{\epsilon}_i^T + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\boldsymbol{\epsilon}_{i+1}^T \\ &= \begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{\epsilon}_i^T + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\boldsymbol{\epsilon}_{i+1}^T + \begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix}\boldsymbol{\epsilon}_{i+1}^T + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\boldsymbol{\epsilon}_i^T \\ &= \left(\begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\right)(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})^T\end{aligned}$$

because $\boldsymbol{H}'_i\boldsymbol{\epsilon}_{i+1} = \boldsymbol{H}'_{i+1}\boldsymbol{\epsilon}_i = 0$ from $D_i \cap D_{i+1} = \varnothing$. The addition of the right-hand sides of (2) and (3) is given as

$$\begin{aligned}&\begin{bmatrix}\boldsymbol{H}\boldsymbol{v}_i^T \\ 0\end{bmatrix} + \begin{bmatrix}\boldsymbol{s}_i^T \\ 0\end{bmatrix} + \begin{bmatrix}0 \\ \boldsymbol{H}\boldsymbol{v}_{i+1}^T\end{bmatrix} + \begin{bmatrix}0 \\ \boldsymbol{s}_{i+1}^T\end{bmatrix} \\ &= \begin{bmatrix}\boldsymbol{H}\boldsymbol{v}_i^T \\ 0\end{bmatrix} + \begin{bmatrix}0 \\ \boldsymbol{H}\boldsymbol{v}_{i+1}^T\end{bmatrix} + \begin{bmatrix}s_{i,1} \\ 0 \\ \vdots \\ 0 \\ s_{i+1,n-k}\end{bmatrix}.\end{aligned}$$

Finally, we have the following linear equation on $(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})$ as

$$\begin{aligned}&\left(\begin{bmatrix}\boldsymbol{H}'_i \\ \boldsymbol{0}\end{bmatrix} + \begin{bmatrix}\boldsymbol{0} \\ \boldsymbol{H}'_{i+1}\end{bmatrix}\right)(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})^T \\ &= \begin{bmatrix}\boldsymbol{H}\boldsymbol{v}_i^T \\ 0\end{bmatrix} + \begin{bmatrix}0 \\ \boldsymbol{H}\boldsymbol{v}_{i+1}^T\end{bmatrix} + \begin{bmatrix}s_{i,1} \\ 0 \\ \vdots \\ 0 \\ s_{i+1,n-k}\end{bmatrix}.\end{aligned} \tag{4}$$

Since $\boldsymbol{H}'_i$, $\boldsymbol{H}'_{i+1}$, $\boldsymbol{H}$, $\boldsymbol{v}_i$, and $\boldsymbol{v}_{i+1}$ are already known, we have to find $(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})$ for the four cases of $\left(s_{i,1}, s_{i+1,n-k}\right) = (0,0), (0,1), (1,0)$, and $(1,1)$. This equation can be easily solved. For example, since the solutions whose $l$-th bits are 0 for all $l \notin D_i \cup D_{i+1}$ are valid, the equation can be reduced to a Gaussian elimination of an $((n-k)+1) \times (2\beta+1)$ binary matrix in the worst case.

After the determination of $(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})$ from the linear system of equations, $\boldsymbol{\epsilon}_i$ and $\boldsymbol{\epsilon}_{i+1}$ are uniquely determined from the given $(\boldsymbol{\epsilon}_i + \boldsymbol{\epsilon}_{i+1})$ because $D_i \cap D_{i+1} = \varnothing$. From the equation $\boldsymbol{H}\boldsymbol{v}_i^T = \boldsymbol{H}\boldsymbol{\epsilon}_i^T + \boldsymbol{s}_i^T$, $\boldsymbol{s}_i$ and $\boldsymbol{s}_{i+1}$ are obtained as $(\boldsymbol{v}_i - \boldsymbol{\epsilon}_i)\boldsymbol{H}^T$ and $(\boldsymbol{v}_{i+1} - \boldsymbol{\epsilon}_{i+1})\boldsymbol{H}^T$, respectively. We can finally determine $(n-k)+1$ bits of the state of the first LFSR. If there are multiple solutions of the $(\boldsymbol{s}_i, \boldsymbol{s}_{i+1})$ pair, we could find the correct values among the candidates using the same relationships between the other consecutive states such as $\boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_i$ or $\boldsymbol{s}_{i+1}$ and $\boldsymbol{s}_{i+2}$. Solutions of $(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i)$ are given by the similar equations on $(\boldsymbol{\epsilon}_{i-1} + \boldsymbol{\epsilon}_i)$. The candidates of $\boldsymbol{s}_{i-1}$, $\boldsymbol{s}_i$, and $\boldsymbol{s}_{i+1}$ are then narrowed down to the 3-tuple $(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i, \boldsymbol{s}_{i+1})$, which is obtained from $(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i)$ and $(\boldsymbol{s}_i, \boldsymbol{s}_{i+1})$ with common $\boldsymbol{s}_i$. When $(\boldsymbol{s}_{i-1}, \boldsymbol{s}_i, \boldsymbol{s}_{i+1})$ is not unique, we can narrow down the candidates of solutions using $\boldsymbol{s}_{i-2}$ and $\boldsymbol{s}_{i-1}$ or $\boldsymbol{s}_{i+1}$ and $\boldsymbol{s}_{i+2}$ until only one solution is left.

By performing this procedure once more, we can obtain at least $2(n-k)$ bits of the first LFSR outputs which are enough to run the Berlekamp-Massey algorithm; thus it is possible to find the generator polynomial and state of the first LFSR [5], [6].

In [3], Esmaeili and Gulliver mentioned that it is possible to add random channel errors during transmission and these errors can be fixed by the error correction code such that $d_{min} \geq 2t + \beta$.

For a stable cryptosystem, we assume that a good error correcting code is used and the bit error rate is $P_e = 0.01$ in the channel as an example. Then, for $n = 900$, $k = 810$, and $\beta = 20$ as suggested by Esmaeili and Gulliver [3], the probability that a ciphertext cannot be corrected is about $10^{-11}$. In such condition, the probability of obtaining two consecutive ciphertexts with no channel error added is about $2^{-26}$. We can take the proposed attack on the inner key for these ciphertexts. This then reveals the $n-k+1$ bits of the first LFSR output. After the $n-k+1$ bits of the first LFSR output are revealed, there are only two possible inner keys for the next ciphertext. This allows the attacker to guess the inner key and decode every ciphertext that comes after, finally revealing $2(n-k)$ bits of the output of the first LFSR. The bottom line is that the channel error is not enough to prevent the proposed attack for the parameters suggested in [3].

## IV. Conclusions

In this letter, a ciphertext-only cryptanalysis of an LFSR-based Esmaeili-Gulliver cryptosystem is proposed. The attack on the outer key requires $2\beta(\left\lfloor \log_2 \frac{n}{\beta} \right\rfloor + 1)$ operations and less than $3\beta(\left\lfloor \log_2 \frac{n}{\beta} \right\rfloor + 1)$ ciphertexts on average. The attack on the inner key reuses the consecutive ciphertexts used in the attack on the outer key and its complexity is that of solving a set of binary linear equations. Hence, increasing the length of the ciphertext is not a solution against the proposed cryptanalysis.

Since the proposed attack relies on the relationship between LFSR outputs used in some ciphertext pair, the security level of the cryptosystem may be improved by preventing the attacker from finding the proper pair. To prevent the proposed attack while maintaining the key size and simplicity, it can be considered to use the state of LFSR by randomly skipping some of the consecutive states instead of using every clock. For example, the state of the LFSR can be clocked a number of times equal to the decimal equivalence of the first $\lfloor \log_2 l \rfloor$ bits, where $l$ is the number of shift registers in LFSR, i.e., $n-k$ or $\beta\left(\lfloor \log_2 \frac{n}{\beta} + 1 \rfloor\right)$. Then, 0 to $2^{\lfloor \log_2 l \rfloor}$ bits are randomly skipped. This scheme makes it difficult for the attacker to determine the proper pair of ciphertexts for an attack on the inner key and the outer key.

Although this approach would make the attacking procedure complex, the cryptographic random number generators that are based on a secure hash function or secure cryptographic algorithms as presented in NIST SP 800-90A [7], are a better choice in the Esmaeili-Gulliver cryptosystem. In addition, we suggest random padding for the ciphertext to prevent an attack using the length of ciphertexts such as the proposed attack on the outer key.

## Acknowledgment

The authors would like to thank anonymous reviewers and the associate editor for their valuable suggestions and comments that helped to improve the quality of this letter.

## References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] R. J. McEliece, "A public-key cryptosystem based on algebraic coding theory," Deep Space Netw., Barstow, CA, USA, Prog. Rep. 42-44, Jan. 1978, pp. 114–116.

[3] M. Esmaeili and T. A. Gulliver, "A secure code based cryptosystem via random insertions, deletions, and errors," *IEEE Common. Lett.*, vol. 20, no. 5, pp. 870–873, May 2016.

[4] A. I. Barbero and Ø. Ytrehus, "Modifications of the Rao–Nam cryptosystem," in *Proc. Int. Conf. Coding Theory Cryptogr. Relat. Areas*, Guanajuato, Mexico, 1998, pp. 1–13.

[5] E. R. Berlekamp, *Algebraic Coding Theory*. New York, NY, USA: McGraw-Hill, 2015.

[6] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Trans. Inf. Theory*, vol. 15, no. 1, pp. 122–127, Jan. 1969.

[7] E. Barker and J. Kelsey, "Recommendation for random number generation using deterministic random bit generators," NIST Special Pub. 800-90A, Revision 1, Jun. 2015.

[8] M. Živković, "A table of primitive binary polynomials," *Math. Comput.*, vol. 62, no. 205, pp. 385–386, Jan. 1994.