LUND UNIVERSITY

Using Coding Techniques to Analyze Weak Feedback Polynomials

Hell, Martin

Published in:
Proceedings

2010

Citation for published version (APA):
Hell, M. (2010). Using Coding Techniques to Analyze Weak Feedback Polynomials. In *Proceedings* (pp. 2523-2527)

Total number of authors:
1

LUND UNIVERSITY

PO Box 117
221 00 Lund
+46 46-222 00 00

Using Coding Techniques to Analyze Weak Feedback Polynomials

Martin Hell
Department of Electrical and Information Technology
Lund University, Sweden
E-mail: martin.hell@eit.lth.se

***Abstract*—We consider a class of weak feedback polynomials for LFSRs in the nonlinear combiner. When feedback taps are located in small groups, a distinguishing attack can sometimes be improved considerably, compared to the common attack that uses low weight multiples. This class of weak polynomials was introduced in 2004 and the main property of the attack is that the noise variables are represented as vectors. We analyze the complexity of the attack using coding theory. We show that the groups of polynomials can be seen as generator polynomials of a convolutional code. Then, the problem of finding the attack complexity is equivalent to finding the minimum row distance of the corresponding generator matrix. A modified version of BEAST is used to search all encoders of memory up to 13. Moreover, we give a tight upper bound on the required size of the vectors in the attack.**

I. Introduction

One of the most common building blocks in stream ciphers is the Linear Feedback Shift Register (LFSR). Two important design ideas based on LFSRs are the nonlinear combiner and the nonlinear filter generator. As an example, the E_0 stream cipher used in Bluetooth is based on the linear combiner, but adds some extra memory for increased security. In this paper we focus on the nonlinear combiner. It is well known that LFSRs with feedback polynomials of low weight should be avoided in stream ciphers. An attack exploiting low weight feedback polynomials is the fast correlation attack, proposed by Meier and Staffelbach [1]. Such polynomials provide a low weight parity check equation which can be exploited together with a bias in the output function.

Due to the fast correlation attack, low weight feedback polynomials can be considered weak in the context of stream cipher design. Much research has been put into the fast correlation attack and several improvements have been found, resulting in the fact that this attack is one of the most important cryptanalytic attacks on stream ciphers. In 2004, Englund, Hell and Johansson introduced a new class of weak feedback polynomials [2]. These polynomials were not necessarily of low weight, but instead had their feedback taps located in small groups, possibly very far apart. Each group can be represented by a polynomial, $g_i(x)$, with the first tap in a group being the constant term in $g_i(x)$. In 2008, Hell and Brynielsson [3] analyzed this attack further and showed that the Walsh transform could be used to efficiently find the complexity and the required vector length in the attack. However, the complexity of the algorithm limited the results to relatively modest vector lengths. An exhaustive search for all combinations of two groups, $g_0(x)$ and $g_1(x)$ with degree at most 8 was performed for all vector lengths up to 25. In this paper we consider the same problem, but from a coding theory point of view. We show that the polynomials $g_i(x)$ can be seen as generator polynomials for a convolutional code. Doing this, the problem of finding the best attack complexity is equivalent to the problem of finding the minimum row distance of the corresponding generator matrix. If the generator matrix is noncatastrophic, this in turn is equivalent to finding the free distance of the code. There are several well known algorithms dedicated to the problem of finding free distances and the currently most efficient is the Bidirectional Efficient Algorithm for Searching code Trees (BEAST) [4]. It is designed to find the weight spectrum of a code. We slightly modify BEAST to work also with catastrophic generator matrices and use it to exhaustively search all combinations of two groups, $g_0(x)$ and $g_1(x)$ with degree at most 13. With our approach there is practically no limitation on the size of the vectors, as there was in previous work. Moreover, we theoretically derive the largest possible vector length required in an attack for a given degree of $g_i(x)$, and also show the exact number of combinations requiring this vector length.

The paper is organized as follows. In Section II we give some preliminaries. In Section III we look at the previous work that is relevant to our analysis. The relation to coding theory is given in Section IV and in Section V we present the row distance search using our modified variant of BEAST. The theoretical results are given in Section VI and our results are concluded in Section VII.

II. Preliminaries

Consider the fast correlation attack [1]. The ideas behind this attack, originally given as a key recovery attack on a nonlinear combiner, can easily be turned into a distinguishing attack on the same nonlinear combiner. The nonlinear combiner uses a set of T LFSRs, preferably with primitive feedback polynomials, and a nonlinear Boolean output function. We denote the ith LFSR by R_i and its size by L_i. The output of R_i at time t is denoted $x_i(t)$.

Let $z(t)$ be the keystream bit at time t. The correlation attack [5] relies on the fact that there is always a subset of the shift registers such that

$$\Pr(z(t) = x_{i_1}(t) \oplus x_{i_2}(t) \oplus \ldots \oplus x_{i_b}(t)) = \frac{1}{2}(1+\varepsilon), \quad (1)$$

with nonzero ε. The size b of this subset depends on the *correlation immunity* of the Boolean function [6].

The characteristic polynomial $f(x)$ of the sequence $s(t) = x_{i_1}(t) \oplus \ldots \oplus x_{i_b}(t)$ is given by

$$\begin{aligned} f(x) &= lcm(f_{i_1}(x), f_{i_2}(x), \ldots, f_{i_b}(x)), && (2)\\ &= c_0 + c_1 x + c_2 x^2 + \ldots c_L x^L, && (3)\end{aligned}$$

where f_{i_j} is the characteristic polynomial of the sequence generated by R_{i_j} and *lcm* is the least common multiple. If the polynomials are primitive, (2) is reduced to the product of the involved polynomials and the degree of $f(x)$ is $L = \sum_{j=1}^{b} L_{i_j}$. In the fast correlation attack, low weight multiples of $f(x)$ are searched, giving us a parity check equation of low weight that holds with probability $\neq 0.5$. In the remainder, we will assume that the set of LFSRs has been replaced by one LFSR according to (2) and for clarity of presentation we will from now on use the notation s_t to denote the value of sequence s at time instance t.

Hypothesis testing is a central component in a distinguishing attack. It can be used to decide if an observed collection of samples are drawn from a biased distribution, here called the cipher distribution P_C, or from a uniform distribution P_0. For an overview of hypothesis testing we refer to [7]. Its application to cryptanalysis is treated in e.g., [8], [9]. We will use the Kullback-Leibler distance

$$D(P_C \| P_0) = \sum_x \Pr_{P_C}(x) \log \frac{\Pr_{P_C}(x)}{\Pr_{P_0}(x)}, \quad (4)$$

also known as divergence or relative entropy, between two distributions to measure the efficiency of the attack. The number of samples needed in the hypothesis test is in the order of $O(1/D(P_C \| P_0))$.

III. Previous work

In this section we review the results given in [2] and [3]. Consider Eq. (1). Replace the LFSRs used in the nonlinear combiner by $f(x)$ as given by (2). Then, we can write $z_t = s_t \oplus e_t$, where e_t is the noise introduced by the approximation (1). We see that

$$\Pr(e_t = 0) = \frac{1}{2}(1+\varepsilon), \quad (5)$$

assuming that the Boolean output function is balanced. The polynomial $f(x)$ defines the recurrence

$$\bigoplus_{i=0}^{L} c_i s_{t+i} = 0, \quad t \geq 0, \quad (6)$$

and we can write

$$\bigoplus_{i=0}^{L} c_i z_{t+i} = \bigoplus_{i=0}^{L} c_i s_{t+i} \oplus \bigoplus_{i=0}^{L} c_i e_{t+i} = \bigoplus_{i=0}^{L} c_i e_{t+i}. \quad (7)$$

We will throughout the paper assume that all noise variables e_i are independent. Then, according to the Piling-up Lemma [10],

$$\Pr\left(\bigoplus_{i=0}^{L} c_i z_{t+i} = 0\right) = \Pr\left(\bigoplus_{i=0}^{L} c_i e_{t+i} = 0\right) = \frac{1}{2}(1+\varepsilon^w), \quad (8)$$

where w is the Hamming weight of $(c_0, c_1, \ldots, c_L)$. This results in a distinguishing attack requiring $O(L + 1/\varepsilon^{2w})$ keystream bits. L is the distance between the first and last keystream bit in each sample and $1/\varepsilon^2$ is a common rule of thumb widely used in cryptanalysis to approximate the number of samples needed to detect the bias ε in (5). The complexity of the attack is highly dependent on the weight w of $f(x)$ and thus, it is usually favourable to consider low weight multiples of $f(x)$ instead. This attack will be referred to as the *basic attack*. The idea proposed in [2] was to generalize this attack and consider the case when $f(x)$ can be written as

$$f(x) = g_0(x) + x^{M_1} g_1(x) + x^{M_2} g_2(x) + \ldots + x^{M_\ell} g_\ell(x), \quad (9)$$

where $g_i(x)$ are polynomials of small degree ($\leq k$) and $M_1 < M_2 < \ldots < M_\ell$. It is also possible to consider multiples of this form. The details concerning these multiples will not be addressed in this paper, but can be found in [2], [3]. The polynomial (9) corresponds to an LFSR with taps placed in small groups. Each group has taps at most k shift register cells apart and groups are located far away from each other.

Now, consider the variable Q_i,

$$Q_i = \underline{g_0} \cdot e[i, i+k] \oplus \ldots \oplus \underline{g_\ell} \cdot e[M_\ell + i, M_\ell + i + k] \quad (10)$$

where $e[i,j] = (e_i, \ldots, e_j)^T$ and $\underline{g_i} = (g_{i,0}, g_{i,1}, \ldots, g_{i,k})$, $g_{i,j}$ is the jth coefficient in the polynomial $g_i(x)$. The main observation here is that even though consecutive noise variables e_i are independent, variables Q_i close together will be dependent. The reason is that the same noise variable will be used in Q_i's close together. Hence, we can consider the noise vector of length N given by

$$E_i = (Q_{N\cdot i}, Q_{N\cdot i+1}, \ldots, Q_{N\cdot(i+1)-1}). \quad (11)$$

E_i can also be written as

$$E_i = \bigoplus_{j=0}^{\ell} \mathbf{G}_N^{(j)} \cdot (e_{N\cdot i+M_j}, \ldots, e_{N\cdot(i+1)+M_j+k-1})^T, \quad (12)$$

where $\oplus$ denotes bitwise xor of binary vectors, $M_0 = 0$ and $\mathbf{G}_N^{(j)}$ is the size $N \times (N+k)$ matrix

$$\mathbf{G}_N^{(j)} = \begin{pmatrix} g_{j,0} & g_{j,1} & \cdots & g_{j,k} & & & \\ & g_{j,0} & g_{j,1} & \cdots & g_{j,k} & & \\ & & \ddots & \ddots & & \ddots & \\ & & & g_{j,0} & g_{j,1} & \cdots & g_{j,k} \end{pmatrix}. \quad (13)$$

The efficiency of the distinguishing attack depends on the distribution of E_i. The different $g_i(x)$ are assumed to be far apart so their contribution to the total noise vector can be computed independently. In [2], different combinations of g_i's were tested and it was noted that for some combinations, the number of keystream bits needed in the distinguishing attack was significantly lower than in the basic binary attack.

Further analysis, based on the Walsh transform of probability distributions, were done in [3]. First, consider only one polynomial $g_0(x)$ and introduce the size $N+k$ random variable vector $X = (x_0, x_1, \dots, x_{N+k-1})^T$ where x_i are independent binary random variables corresponding to the noise introduced by the linear approximation of the Boolean output function. Thus, we have $\Pr(x_i = 0) = \frac{1}{2}(1+\varepsilon)$. Further, the size N random variable vector $Y = (y_0, y_1, \dots, y_{N-1})^T$ is given by $Y = \mathbf{G}_N^{(0)} X$ and the number of samples needed in the distinguisher is in the order of $1/D(Y\|P_0)$. We can write

$$D(P_Y\|P_0) = \frac{1}{2\ln 2}\sum_{\omega\neq 0} W_{P_Y}(\omega)^2, \tag{14}$$

where

$$W_{P_Y}(\omega) = \Pr(\omega\mathbf{G}_N^{(0)}X=0) - \Pr(\omega\mathbf{G}_N^{(0)}X=1) = \varepsilon^{\|\omega\mathbf{G}_N^{(0)}\|_1}, \tag{15}$$

Note that since ε is small, the sum in (14) will be dominated by the term with smallest $\|\omega\mathbf{G}_N^{(0)}\|_1$ (Hamming weight). Combining (14) and (15) and generalizing this to several $(\ell+1)$ polynomials, we get the relative entropy

$$D(P_Y\|P_0) \approx \frac{1}{2\ln 2}\varepsilon^{2\cdot\min_\omega\left(\sum_{i=0}^{\ell}\|\omega\mathbf{G}_N^{(i)}\|_1\right)}. \tag{16}$$

Thus, the complexity of the attack depends on the smallest sum of the Hamming weights $\|\omega\mathbf{G}_N^{(i)}\|_1$.

IV. Relation to Coding Theory

In this section we show that the efficiency of the attack is closely related to coding theory. In particular, the number of keystream bits needed in the distinguisher is related to the minimum distance of a linear, zero-tail terminated convolutional code.

For a thorough overview of coding theory we refer to [11] and [12].

Denote the information sequence by $\mathbf{u} = u_0, u_1, \dots$ and the code sequence by $\mathbf{v} = v_0, v_1, \dots$. Then the code sequence is given by $\mathbf{v} = \mathbf{uG}$ where $\mathbf{G}$ is the generator matrix.

Consider the matrix $\mathbf{G}^{(j)}$ given by (13). This matrix can be seen as a generator matrix for a rate $R = 1$ linear convolutional code truncated after its first N rows. Similarly, a zero-tail terminated generator matrix for a rate $R = 1/c$ linear convolutional code can be constructed as

$$\mathbf{G_N} = \begin{pmatrix} G_0 & G_1 & \dots & G_k & & & \\ & G_0 & G_1 & \dots & G_k & & \\ & & \ddots & \ddots & & \ddots & \\ & & & G_0 & G_1 & \dots & G_k \end{pmatrix}, \tag{17}$$

where each G_j is a $1\times(\ell+1)$ matrix corresponding to the jth coefficient of the $\ell+1$ polynomials $g_i(x), 0 \le i \le \ell$,

$$G_j = (g_{0,j} \quad g_{1,j} \quad \dots \quad g_{\ell,j}). \tag{18}$$

Let $u_{[0,n]} = u_0, u_1, \dots, u_n$. The jth order row distance of a generator matrix is defined as the minimum Hamming weight of a codeword resulting from $j+1$ information symbols $u_{[0,j]} \neq \mathbf{0}$, followed by the sequence forcing the state to the zero state. This can be written as

$$d_j^r = \min_{u_{[0,j]}\neq\mathbf{0}}\{u_{[0,j]}\mathbf{G_{j+1}}\}. \tag{19}$$

Comparing this to (16), we see that the number of samples in our distinguisher is related to the row distance of the generator matrix. In particular, when using vectors of length N, the efficiency of the distinguisher is related to the row distance of order $N-1$. For vectors of length N we have

$$D(P_Y\|P_0) \approx \frac{1}{2\ln 2}\varepsilon^{2\cdot d_{N-1}^r}. \tag{20}$$

It is easy to see that

$$d_j^r \ge d_{j+1}^r, \tag{21}$$

since the sequence $u_{[0,j+1]} \neq \mathbf{0}$ can be chosen as $0u_{[0,j]} \neq \mathbf{0}$ which will give the same Hamming weight in the code sequence as the sequence $u_{[0,j]}$. The free distance of a code is defined as the minimum Hamming distance between two differing codewords,

$$d_{\text{free}} = \min_{\mathbf{v}\neq\mathbf{v}'}\{d_H(\mathbf{v},\mathbf{v}')\}. \tag{22}$$

The free distance is upper bounded by the row distance and for *noncatastrophic* generator matrices we have

$$\lim_{j\to\infty} d_j^r = d_{\text{free}}. \tag{23}$$

Since the row distance is a monotonically decreasing function that approaches the free distance of the code, this metric is very useful when searching for codes with optimum (largest) free distance. Since d_j^r can be found very efficiently for small j, it can serve as a rejection rule for candidate encoders. For a catastrophic generator matrix we have

$$\lim_{j\to\infty} d_j^r \ge d_{\text{free}}. \tag{24}$$

This is due to the fact that codewords in a convolutional code are of infinite length and the trellis can enter a loop with only zeros in the output. Thus, it will not merge with the all-zero codeword. When computing the row distance we need to, by definition, return to the all-zero state. Catastrophic generator matrices are not interesting in error control coding since a finite number of errors in the estimated codeword can result in an infinite number of errors in the estimated information sequence.

In our attack, the $1\times(\ell+1)$ matrix G_0 consists of only ones. This corresponds to the fact that the constant term is 1 in our polynomials $g_i(x)$. From this it also follows that $d_j^r > \ell$ for all choices of j. Combining this with (21) it is clear that d_j^r becomes stationary as j increases. In coding this value is denoted d_∞^r. Since $j+1$ corresponds to the length N of the vectors in our attack, and the vectors should be chosen to be the smallest size which gives the best attack, we are interested in the value $\widehat{j}$ such that

$$\widehat{j} \doteq \min_{j\ge 0}\{j\} : d_j^r = d_\infty^r. \tag{25}$$

Assuming that we start and end in the all-zero state, a possible interpretation for noncatastrophic encoding matrices in coding theory could be that $\widehat{j}+1$ is the smallest length of an information sequence such that there are two codewords with Hamming distance d_{free}.

The vector length $N=j+1$ when j is chosen to fulfill (25) will be denoted N_{opt} reflecting the fact that it is the optimal choice for the vector length in the attack.

V. Computing the row distance

In this section we discuss algorithms that can be used to find d_j^r in general, and, more importantly in our attack, $d_{\widehat{j}}^r$. The most straight forward algorithm is to recursively search the tree resulting from the state transitions in the convolutional encoder. By leaving the all-zero state, we search for the path leading back to the all-zero state with smallest Hamming weight. This weight is equivalent to the smallest Hamming weight of all codewords. A recursive search will first compute d_0^r. For each leaf in the tree we check if the accumulated Hamming weight is larger than or equal to d_0^r. If this is the case, that path is abandoned. Since the corresponding generator matrix might be catastrophic, it is possible to end up in an infinite loop and never return to the all-zero state. By saving the states corresponding to the visited nodes as long as the output weight is zero, we can easily check if we are in such a loop.

A more efficient algorithm is BEAST [4]. It was proposed in 2004, as an efficient algorithm to compute the weight spectra for convolutional codes. The minimum nonzero value of the weight spectrum corresponds to the free distance of the code, assuming a noncatastrophic generator matrix. The algorithm is similar to the straight forward algorithm given above, but additionally takes advantage of the fact that we know which state to return to, i.e., the all-zero state. Assume that we want to find the number of codewords of weight α. Then the end state of all forward paths in the tree with Hamming weight $\alpha/2$ are saved in a list. A similar list is constructed for all backward paths of weight $\alpha/2$ and then the two lists are combined. Naturally, for a rate $1/c$ code, we have to compute c forward lists and c backward lists and combine the lists that will result in codewords of Hamming weight α. The BEAST algorithm is currently the fastest known algorithm to find d_{free} of a convolutional code. In [4], BEAST is described for noncatastrophic encoding matrices and a similar change as described above to detect loops of weight zero has to be implemented in order to apply it to our situation. Also, the algorithm has been modified to save $\widehat{j}$ when $d_{\widehat{j}}^r$ has been computed.

Using our slightly modified version of BEAST, we have computed the minimum row distance $d_{\widehat{j}}^r$ and $\widehat{j}$ for all possible combinations of two polynomials of degrees k_0 and k_1, $1 \leq k_0, k_1 \leq 13$.

We examine two main problems related to the current attack:

- What is the number of samples needed in the distinguisher?
- Which vector length is needed in order to get the most efficient attack in terms of number of samples needed in the distinguisher?

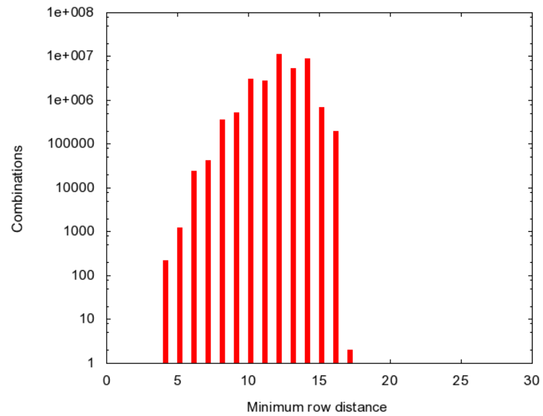

Fig. 1. The number of polynomial combinations with given $d_{\widehat{j}}^r$.

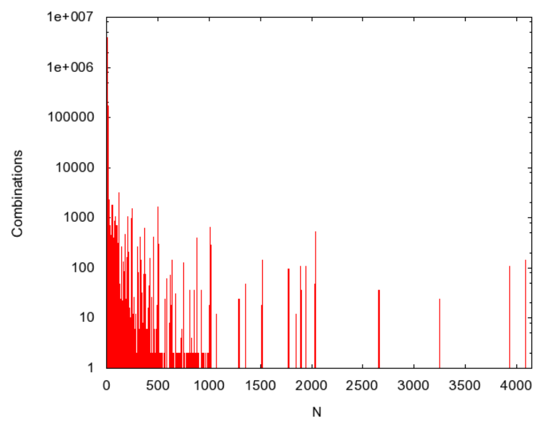

Fig. 2. The number of polynomial combinations requiring a given vector length N.

Our new analysis allows us to answer the first question, not only for vector lengths $N<30$ as previously done [3], but for the *optimal* vector length. Furthermore, the second question can now be answered with the *optimal* vector length, not the optimal with the restriction $N<30$.

Fig. 1 is a histogram that shows the number of polynomial combinations that results in a given $d_{\widehat{j}}^r$.

Since $d_{\widehat{j}}^r$ immediately tells us the number of samples needed in the distinguisher, this graph gives a rough idea on what performance we can expect from our attack. Note that the largest free distance for a convolutional code with memory 13 is known to be 16. The two encoders in Fig. 1 with $d_{\widehat{j}}^r = 17$ both corresponds to catastrophic encoding matrices.

Fig. 2 is a histogram showing the number of polynomial combinations that require a given vector length $N=\widehat{j}+1$ in order to mount the best attack.

We see that even though most combinations require a modest value of N, some combinations require a very high value, the largest being $N = 4084$. As with the free distance of a convolutional code, it is difficult to predict the efficiency of the attack, as well as the required vector length, by just looking at two arbitrary polynomials. However, for very large vector length it is possible to derive theoretical results.

VI. Theoretical results

In this section we give a tight bound on the vector length given the maximum degree of the involved polynomials $g_i(x)$. We give expressions for the number of polynomial combinations with maximum vector length and also characterize the form of these polynomials. Further, we show that for *any* degree $k > 3$ on the involved polynomials, there will *always* be combinations of weight > 6 that result in $d_j^r = 6$, i.e., a very efficient attack compared to the basic attack.

In the sequel, the following property will be used.

Property 1: In a rate $1/2$ encoder, if $g_1(x) = (1+x)g_0(x)$, then the output sequence generated by $g_1(x)$ in the encoder is the derivative of the sequence generated by $g_0(x)$,

$$v_i^{(1)} = v_i^{(0)} \oplus v_{i-1}^{(0)}. \tag{26}$$

Note that $v_{-1}^{(0)} = 0$ since we start in the all-zero state.

Now, assume that the polynomial $g_0(x)$ is primitive, and $g_1(x) = (1+x)g_0(x)$. Thus, the degree of $g_1(x) = k$, which is also the memory order of the encoder, and the degree of $g_0(x) = k-1$. Let $k > 3$ and the input sequence to the encoder be the m-sequence with characteristic polynomial being the reciprocal of $g_0(x)$, followed by two trailing zeros. Moreover, the sequence is chosen such that it ends with $k-2$ zeros. This necessarily means that it starts with a one. If the encoder starts in the all-zero state, $v_0^{(0)} = 1$. Then $v_1^{(0)} = 0, \ldots, v_{2^{k-1}-2}^{(0)} = 0$ because of the chosen input sequence. Since the sequence ended with $k-2$ zeros, we have to input 2 extra zeros in order to force the encoder back to the all-zero state. Thus, $v_{2^{k-1}-1}^{(0)} = 1$ and $v_{2^{k-1}}^{(0)} = 0$ since the last memory cell is not used by $g_0(x)$. Combining this with Property 1 we get

$$\begin{aligned} v^{(0)} &= 1\underbrace{00000\ldots00}_{2^{k-1}-2}10 \\ v^{(1)} &= 11\underbrace{0000\ldots00}_{2^{k-1}-3}11. \end{aligned}$$

From this it follows that the minimum row distance in this case is 6. The optimal vector length is the number of rows in the generator matrix (17). We know that the number of columns is $N+k$ and this equals the number of output symbols, $2^{k-1}+1$. Hence,

$$N = 2^{k-1} - (k-1). \tag{27}$$

Since $g_0(x)$ is primitive, this is a tight upper bound on the vector length for polynomials of degree $\leq k$. (Note that this analysis holds also for $k = 3$, with $g_0(x) = 1 + x + x^2$ and $g_1(x) = 1 + x^3$, but the total weight is here 5 meaning that the basic attack will be better.) Based on the analysis we give the following proposition.

Proposition 1: If $g_0(x)$ is primitive and $g_1(x) = (1 + x)g_0(x)$. Then the optimal vector length in our attack is given by $N_{opt} = 2^{k-1} - (k-1)$. The minimum row distance of the corresponding encoder is $d_{N_{opt}-1}^r = 6$. Moreover, the number of samples needed in the attack is in the order of ε^{-12}.

For any given degree $k > 3$ of $g_1(x)$, the number of polynomial combinations with this property is given by the number of primitive polynomials of degree $k-1$, i.e.,

$$\frac{\varphi(2^{k-1}-1)}{k-1}, \tag{28}$$

where $\varphi(\cdot)$ is the totient function.

This analysis explains the largest vector length $N = 4084$ in Fig. 2. With $k = 13$ we have $\varphi(4095)/12 = 144$ polynomial combinations that require a vector length $N_{opt} = 2^{12} - 12 = 4084$. There are no two polynomials with degree $\leq k$ requiring a larger vector length for the best attack.

VII. Conclusion

We have shown a relation between a distinguishing attack based on weak polynomials and coding theory. The relation allows the use of efficient algorithms for weight spectra to be used to compute the complexity of the attack. Moreover, the optimal vector length used in the attack can be found very efficiently and we show that for any degree k of the polynomials $g_i(x)$, there are always very efficient attacks. It is crucial that these polynomials are avoided in nonlinear combiners.

References

[1] W. Meier and O. Staffelbach, "Fast correlation attacks on certain stream ciphers," *Journal of Cryptology*, vol. 1, no. 3, pp. 159–176, 1989.

[2] H. Englund, M. Hell, and T. Johansson, "Correlation attacks using a new class of weak feedback polynomials." in *Fast Software Encryption 2004*, ser. Lecture Notes in Computer Science, B. Roy and W. Meier, Eds., vol. 3017. Springer-Verlag, 2004, pp. 127–142.

[3] M. Hell and L. Brynielsson, "Another look at weak feedback polynomials in the nonlinear combiner," in *International Symposium on Information Theory—ISIT 2009*. IEEE, 2009.

[4] I. Bocharova, M. Handlery, R. Johannesson, and B. Kudryashov, "A BEAST for prowling in trees," *IEEE Transactions on Information Theory*, vol. 50, no. 6, pp. 1295–1302, June 2004.

[5] T. Siegenthaler, "Decrypting a class of stream ciphers using ciphertext only," *IEEE Transactions on Computers*, vol. 34, pp. 81–85, 1985.

[6] ——, "Correlation-immunity of nonlinear combining functions for cryptographic applications," *IEEE Transactions on Information Theory*, vol. 30, pp. 776–780, 1984.

[7] T. Cover and J. Thomas, *Elements of Information Theory*, ser. Wiley series in Telecommunication. Wiley, 1991.

[8] T. Baignères, P. Junod, and S. Vaudenay, "How far can we go beyond linear cryptanalysis?" in *Advances in Cryptology—ASIACRYPT 2004*, ser. Lecture Notes in Computer Science, vol. 3329. Springer-Verlag, 2004, pp. 432–450.

[9] M. Hell, T. Johansson, and L. Brynielsson, "An overview of distinguishing attacks on stream ciphers," *Cryptography and Communications*, 2008.

[10] M. Matsui, "Linear cryptanalysis method for DES cipher," in *Advances in Cryptology—EUROCRYPT'93*, ser. Lecture Notes in Computer Science, T. Helleseth, Ed., vol. 765. Springer-Verlag, 1994, pp. 386–397.

[11] R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding*, ser. IEEE Series on Digital and Mobile Communication. IEEE Press, 1999.

[12] S. Lin and D. J. Costello, *Error Control Coding, Second Edition*. Prentice-Hall, Inc., 2004.