IMT4124 - correlation attack

Magnus Lien Lilja 2020, IMT4124, MIS
Email: magnull@stud.ntnu.no

September 25, 2020

1 Introduction

This paper presents the report for the project in IMT4124 - Cryptology, Spring 2020. section 2 presents the theory behind the correlation attack (task 2), as presented in Siegenthalers paper "*Decrypting a Class of Stream Ciphers Using Ciphertext Only,*" where this type of attack was originally presented [1]. Next, section 3 describes the implementation of the attack in order to break Geffe's generator (task 3). The results obtained from using different parameters in the correlation attack are also discussed. Lastly section 4 presents a different boolean combiner than Geffe's generator and according discussion (task 4).

2 Task 2 - Siegenthaler's correlation attack

This section presents the theory behind the correlation attack.

Random numbers are important in many aspects of cryptography, for example in stream ciphers [2]. It can be challenging to produce a sequence $z \in \{0, 1\}$, which is hard to predict and require low effort to produce and distribute. A true random sequence have good unpredictability properties and often use a source of entropy from natural sources as input. They are thus quite slow to generate and require the same amount of bits as the plaintext to distribute, if used in stream ciphers. Because of this pseudo-random bit generators are often used in practice. While this does not offer theoretical security, the produced pseudo-random sequence of l bits is easy to generate and distribute, because it is deterministic. To be unpredictable enough, the sequence have to be impossible to distinguish from a true random sequence in polynomial time [2]. If the pseudo-random sequence is not fulfilling this requirement the probability of guessing the next bit will be different from 0.5. This presents statistical dependencies in the sequence, which is not good.

Figure 1 shows an illustration of a pseudo-random bit generator, called Geffe's Generator. It consists of three left shift feedback registers (LFSR) and a nonlinear combiner function f:

$$f(x_1, x_2, x_3) = x_3 + x_1x_2 + x_2x_3$$

The function f is deterministic, so the output sequence depends on an initial state (seed) [2]. Because of this, the key in Geffe's generator consists of three seeds as well as the feedback polynomials of the LFSRs.

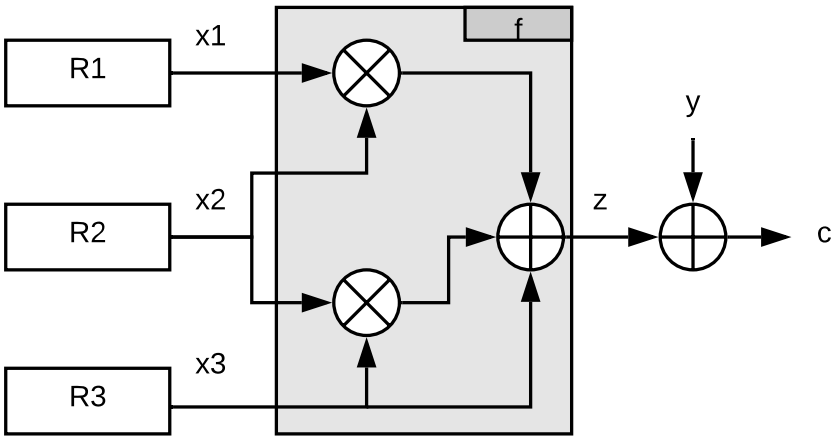

Figure 1: Geffe's Generator

In this paper, the feedback polynomials is considered a part of the cryptosystem and according to Kerchoffs principle, known to the cryptanalyst together with the whole cryptosystem except the key [3]. If the key is known to the cryptanalyst, he will be able to decipher all the previous and future messages enciphered with the same key. One possible method of getting all the seeds is to try every combination, also called a brute-force attack. The number of possible keys (keyspace) for Geffe's generator, assuming that the primitive feedback connections are known, can be calculated with Equation 1.

$$\prod_{i=1}^{s}(2^{L_i}-1) \tag{1}$$

Here, s represents the number of LFSRs and L_i their length (highest degree). In Geffe's generator, the keyspace is determined by L_i, because the number of LFSRs are limited to three. If there are no other weaknesses in the cryptosystem, the security level is dependent on the size of the keyspace which is considered the best case. However, this is not the case with Geffe's generator. The generator has a weakness which will reduce the number of tries to find the key significantly, by exploiting a statistical flaw in the running key sequence z.

A measurement of the correlation between x_i and z is defined as $P[x_i = z] = q_i$. From an analysis of the truth table in Table 1 it is observed that $q_1 = q_2 = 0.75$. This probability can be taken advantage of to guess the initial states of each of the two LFSRs independent of the others, and thus limit the possible initial states to try. Because one of the registers having a correlation probability of $q_2 = 0.5$ the initial states of R_2 cannot be guessed by the same type of attack. The number of keys can therefore be reduced to the calculation in Equation 2, which is a lot less than the keyspace by exhaustive search of all the registers.

$$\sum_{i=1}^{s}(2^{L_i}-1) \tag{2}$$

x_1	x_2	x_3	z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Table 1: Truth table - Geffe's generator

The correlation attack starts out with defining some variables and constraints. The ciphertext only attack assumes that the plain text source y have a $Pr[y = 0] = p_0 \neq 0.5$. Next, two hypotheses H_0 and H_1 are defined as normal distributions Q. H_0 is the null hypothesis and accepted if the correlation measurement α is less than the threshold T. In contrast, H_1 is accepted if $\alpha \geq T$, which means that it is likely that the initial-state V_i can produce the intercepted ciphertext sequence. It is preferred to have as little overlap as possible between the two distributions. If they are equal, which means either $p_0 = 0.5$ or $q_i = 0.5$, the attack cannot be performed because it is impossible to tell whether the current initial state belongs to either of the two probability-distributions. To keep them further separated, the mean of H_1 needs to be far apart from the mean of H_0, $\mu_0 = 0$. The mean value of H_1 is defined as

$$\mu_1 = l(2p_e - 1)$$

A longer ciphertext (higher l) is needed in order to achieve a higher mean value in practice. It is therefore assumed that a longer ciphertext will reduce the number of false positives. Figure 2 shows the probability density function for the two distributions H_1 and H_0. It illustrates that a variation of T will increase or decrease the probability of false positives and negatives.

α is the correlation measurement between the LFSR output x_i and the ciphertext bits c_i. It is calculated as shown in Equation 3. The alpha value is calculated with the difference between the length of the ciphertext l and two times the hamming/edit distance. This means that two bit sequences $z, -z$ of length l, will obtain a low α value. Two identical sequences will get the highest alpha value possible, which is l.

$$\alpha = l - (2\sum_{i=1}^{l} c_i \oplus x_i) \tag{3}$$

In Siegenthalers paper the probability for false positives (p_f) is calculated like this.

$$p_f = Q\left(\left|\frac{T}{\sqrt{l}}\right|\right) \tag{4}$$

Which can be used to determine T from knowing p_f and the length of the observed ciphertext l.

$$T = Q(|1 - p_f|)\sqrt{l} \tag{5}$$

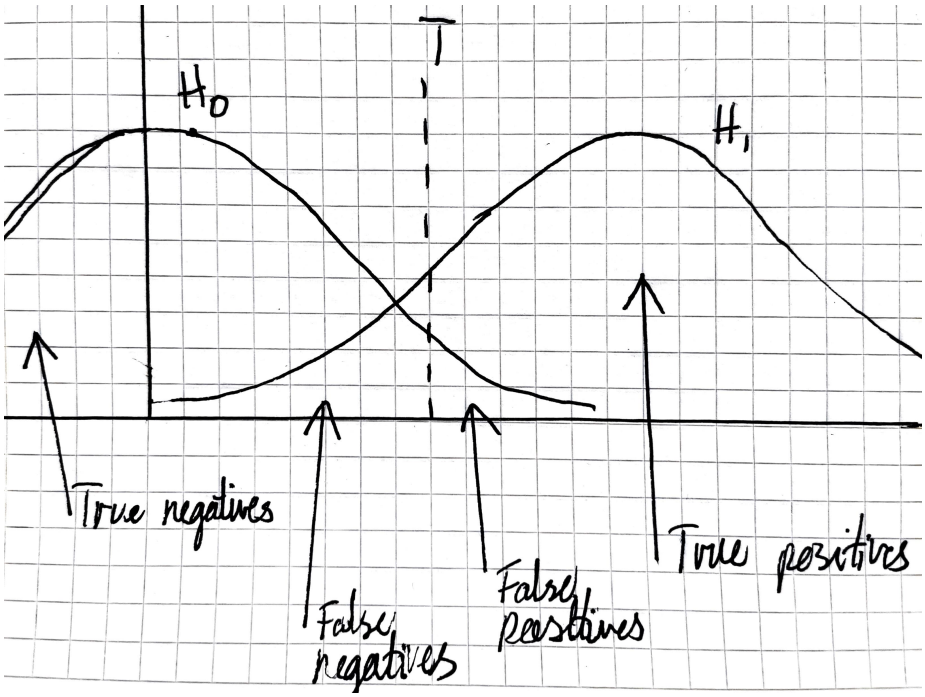

Figure 2: Probability density function - overview of details

The threshold T is affected by the length of the intercepted ciphertext sequence and p_f. Higher l, and lower p_f will lead to a higher T. Too high l can increase the runtime, because there are more bits which needs to be manipulated in order to calculate α. On the other hand, a too low l can possibly increase the likelihood of false negatives (p_m). When setting the p_f value there is also a balance between false positives and false negatives, which needs to be taken into account. A lower p_f value will decrease the amount of false positives, but increase the likelihood of missing the event (false negatives). An ideal situation is to have no false negatives or positives, but only one true positive as a result from the correlation attack. However, in practice it is preferred to have a few more false positives than a risk of missing the event. The likelihood for false negatives is defined by Siegenthaler as shown in Equation 6.

$$p_m = Q\left[\left|\frac{l(2pe-1)-T}{\sqrt{4lpe(1-pe)}}\right|\right] \tag{6}$$

Because primitive polynomials are used in LFSR R_i, the length of the period is equal to the number of different initial states $2^{L_i}-1$. After setting a reasonable value for p_f, as well as determine p_0 and q_i for R_i, the attack starts by trying out all the different initial states for the registers R_1 and R_3 in Geffes generator. For each of the initial states, α is calculated and compared to T. If $\alpha \geq T$ it is likely that the current initial state can produce the intercepted ciphertext sequence. After running through all of the initial states, a number of possible candidate initial states are obtained.

A correlation attack directly on R_2 cannot be done because of $q_2 = 0.5$. Siegenthaler mentions that a modified correlation attack can be performed after knowing the correct initial states of R_1 and R_2 [1]. Another possibility is exhaustive search, which is an option when some candidate initial states are obtained and the number of possible tries is dramatically reduced.

3 Task 3 - Implementation of the attack

This section presents the implementation of the attack and discuss the results obtained, without going too much into code specific functionality. Specifics about the code, how to use it and example output can be seen in the file "*README.txt*", included with the code following report. Task one which implements Geffes generator and the according LFSRs to generate ciphertext as well as task three which breaks the generator is implemented in the file "*main.py*".

In this implementation of Geffe's generator, there are three LFSRs (R_1, R_2, R_3) which implements the following linear recurrence relations. It can be found that all of the polynomials are primitive, which is required to get a maximum length period from the LFSR.

$$R_1 = x^{10} + x^7 + x^3 + x + 1$$
$$R_2 = x^{20} + x^{15} + x^{12} + x^8 + x^6 + x^5 + 1$$
$$R_3 = x^{11} + x^7 + x^3 + x + 1$$

The initial states for each of the registers represents the key, and is set to a value $V_i : 1 \geq V_i \leq 2^{L_i}$. In the current implementation of Geffe's generator V_i is set statically in the code. The keys would ideally be deviated from a password, but this is not the point of this demonstration. V_i are obviously not visible to the cryptanalyst, and it is the goal to obtain these values. The maximum amount of attempts when performing a brute-force attack with these three polynomials in Geffe's generator (keyspace), will be approximately $2.19 * 10^{12}$. On average the key is found after half the keyspace is tested. This means that it is quite possible to brute-force all the keys, given a large amount of resources available. However, the keyspace would increase with higher order polynomials, and make it harder to find the correct key with exhaustive search. When performing a correlation attack on Geffe's generator with the specified polynomials and only true positives obtained, the keyspace is reduced to approximately $1.05 * 10^6$.

When performing the correlation attack on R_1 and R_3, firstly the T value is calculated by empirically determining the appropriate p_f value as input. Next, the whole period $x_0..x_i$ of R_i is generated and the α value calculated by using the hamming distance of x_i and c_i. If $\alpha \geq T$ then H_1 is accepted and the current state of R_i is evaluated as a possible candidate initial state to generate the ciphertext. However, due to the probability of p_f, the candidate initial states can contain false positives. Siegenthaler proposes to use a different part of the observed ciphertext to again calculate α for each of the candidate initial states, withe the goal to reduce the possible amount of false positives. However, it is not certain whether this will eliminate the amount of false positives any more than using all the observed ciphertext for the calculation of α in the first place.

During the exhaustive search for R_2, after candidate initial states for R_1 and R_3 is obtained, a method for determining when the plain text is found have to be applied. The cryptanalyst does of course not know what the plain text is, and have to use an automated process of testing whether the correct seeds are found or not in a cipher text only attack. By guessing that the input is English ASCII text, the cryptanalyst can use entropy to find possible candidates which represents English text. An other alternative is to use known static patterns in the text, for example HTML tags, magic bytes or similar to stop the program when a possible candidate seed has been found. This can work well with for example compressed data or video and images as input. In this implementation, Shannon's entropy of the deciphered text have been used as a comparison of entropy of standard English text. Enciphered or seemingly random bits have a higher entropy, and will clearly separate the possible candidate seeds from the bad ones.

As mentioned, p_f can be determined by the cryptanalyst. Figure 3 shows the amount of false positives obtained after computing α for each of the initial states of R_1 and R_3 and the predicted amount of false positives p_f. The figure shows that the true amount of false positives follows the predicted amount reasonably well. Although, the true amount of false positives is slightly below what is predicted.

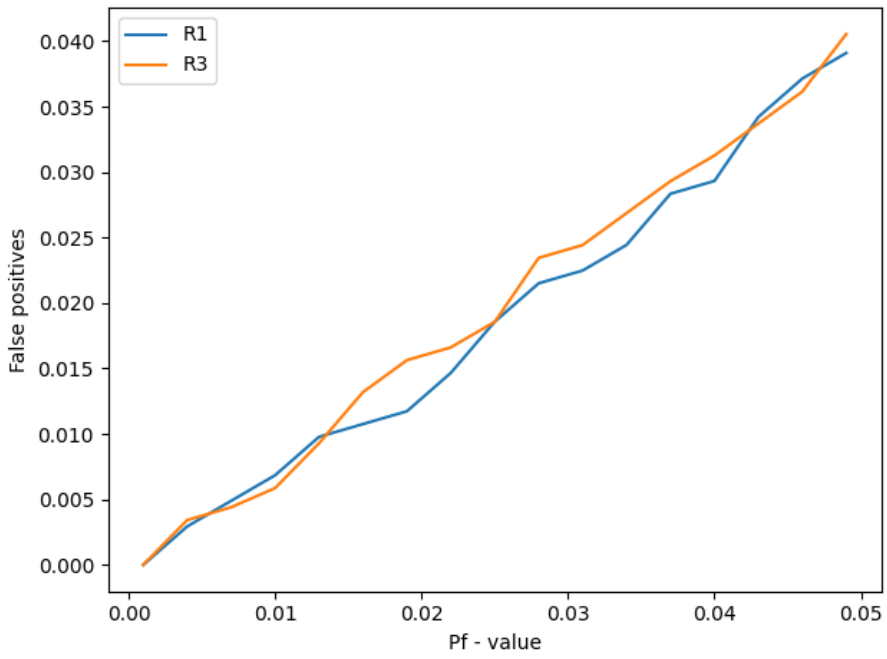

Figure 3: False positives with different false alarm probabilities p_f ($l = 8000$)

Another test was done with a static p_f level, while varying the length l of the intercepted sequence as seen in Figure 4. l was varied from 400 to 30000 bits, and the corresponding percentage of false positives observed in intervals

of 3000 bits. As expected, the trend of false positives generated from R_1 and R_3 decreases as the length of the ciphertext increases. However, there are some peaks around 10000 and 30000 bits when it comes to false positives in R_1. Another observation is that the amount of false positives in R_3 does not follow the rate of R_1 in any way. The lowest combined amount of false positives on R_1 and R_3, without missing the event is preferred. This will make the exhaustive search for V_2 as fast as possible, and is achieved around 6000 and 29000 ciphertext bits. Increasing the length of the ciphertext will also decrease the observed amount of false positives to below the set p_f value of 0.01.

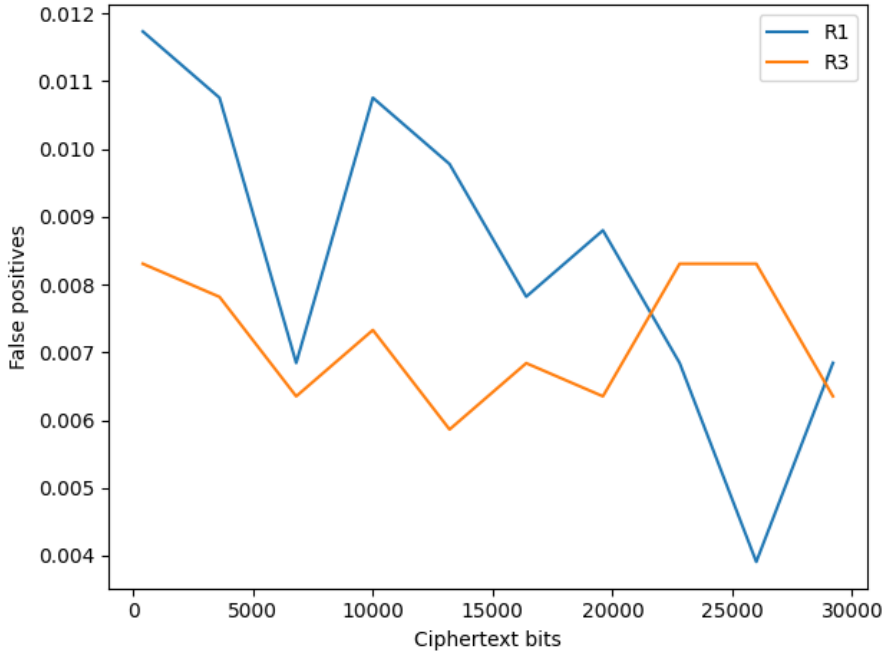

Figure 4: False positives with different input lengths ($p_f = 0.01$)

4 Task 4 - Combiner variations

This task was approached by truth table analysis to find the best boolean combiner. The program "*find_best_combiner.py*" checks all the $2^8 - 1$ different possibilities for the output of the combining function $f(x)$ in a truth table, according to several criterion.

Firstly a requirement is to have a non-linear order of two, because a linear combiner is easy to solve by algebraic attacks. Thus, the advantage of having a boolean combiner is pretty much gone, because it is easy to predict the rest of the keystream. Having a non-linear order of two means that a maximum correlation immunity of order two (pairs of input sequences) is not possible with three input bits [4].

Balancedness is another requirement and will reduce the likelihood of sta-

tistical attacks to succeed, because the sequence is not biased and thus harder to predict. Of course the main interest is to prevent the correlation attack, and thus seek to get a higher order correlation immunity. As shown later, the balancedness criterion will have an impact on the correlation as well. If balancedness is considered as a requirement, there exist no output sequence which is correlation immune of order one and non-linear. However, the second best then is to have two of the input sequences not correlating with the output sequence. 24 out of 255 sequences exists which also has one input sequence with $q_i = 0.5$ in correlation order two. One of them is shown in Table 2 as z_2, where good correlation properties are marked in bold. The visual form of the ANF can be seen in Figure 5. From the table, it is observed that z_2 has a non-linear order of two. If the correlation attack is applied with correlation order one it will take considerably longer amount of time to find the correct initial states. This is because two of the registers initial states have to be found with exhaustive search. Considering the same polynomials as in Task 3 (section 3), it would require $2.14 * 10^9$ tries maximum to find the correct initial states (keys), again assuming that there are only true positives resulting from the correlation attack. However, this combiner function is not really an improvement over Geffes generator, because two of the pairs of of input sequences (correlation order two), have statistical dependencies on the output sequence. A similar correlation attack as the one used to break Geffes generator can be used on this one as well, but with pairs of input sequences instead to perform a higher order correlation attack.

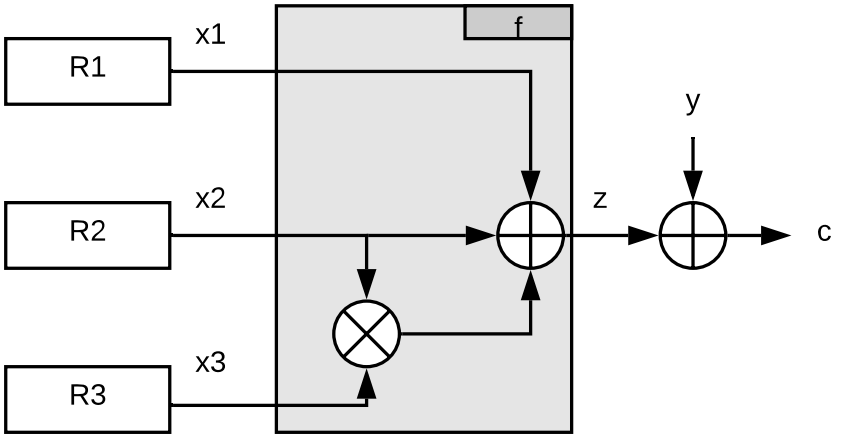

Figure 5: Modified Non-linear combiner function

The sequence can be correlation immune of order one if the balancedness criterion is not considered. This means that none of the input sequences are correlating with the output sequence. But then in correlation order two, all of the pairs of input sequences have statistical dependencies with the output. Which is actually worse than the correlation properties in Geffe's generator. There are eight such sequences in total out of 255 possible. One of them is shown in Table 2 as z_3. It could be faster to break such a sequence than a balanced, such

as the one produced by Geffe's generator. This is because during the exhaustive search for R_2 in Geffe's generator, all of the false positives obtained would have to be permutated and tried with R_2 in order to find the correct initial state. In practice, this means that the register without statistical dependencies ($q_i = 0.5$) could have to be run through as many times as the number of false positives obtained. Depending on the number of false positives ("false alarms") found, the runtime will increase with the time complexity $O(n)$. If an exhaustive search is not required, all of the registers only needs to be run through one time, and thus also more reliable, considering that the parameters to calculate T are set properly.

S	ANF	Correlation order 1			Correlation order 2		
		q_1	q_2	q_3	q_1	q_2	q_3
z_2	$x_2{+}x_2x_3{+}x_1$	6/8	**4/8**	**4/8**	6/8	2/8	**4/8**
z_3	$x_2{+}x_2x_3{+}x_1x_3{+}x_1x_2$	**4/8**	**4/8**	**4/8**	6/8	2/8	6/8

Table 2: Proposed boolean combiners - Correlation properties

References

[1] Siegenthaler, "Decrypting a Class of Stream Ciphers Using Ciphertext Only," *IEEE Transactions on Computers*, vol. C-34, no. 1, pp. 81–85, Jan. 1985. [Online]. Available: http://ieeexplore.ieee.org/document/1676518/

[2] D. R. Stinson and M. B. Paterson, *Cryptography: theory and practice*. CRC press, 2019.

[3] C. E. Shannon, "Communication Theory of Secrecy Systems*," *Bell System Technical Journal*, vol. 28, no. 4, pp. 656–715, Oct. 1949. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6769090

[4] T. Siegenthaler, "Correlation-immunity of nonlinear combining functions for cryptographic applications (Corresp.)," *IEEE Transactions on Information Theory*, vol. 30, no. 5, pp. 776–780, Sep. 1984, conference Name: IEEE Transactions on Information Theory.