Reducing the Space Complexity of BDD-based Attacks on Keystream Generators

Matthias Krause and Dirk Stegemann

Theoretical Computer Science
University of Mannheim, Germany
{krause,stegemann}@th.informatik.uni-mannheim.de

Abstract. The main application of stream ciphers is online-encryption of arbitrarily long data, for example when transmitting speech data between a Bluetooth headset and a mobile GSM phone or between the phone and a GSM base station. Many practically used and intensively discussed stream ciphers such as the E_0 generator used in Bluetooth and the GSM cipher A5/1 consist of a small number of linear feedback shift registers (LFSRs) that transform a secret key $x \in \{0,1\}^n$ into an output keystream of arbitrary length. In 2002, Krause proposed a Binary Decision Diagram (BDD) based attack on this type of ciphers, which in the case of E_0 is the best short-keystream attack known so far. However, BDD-attacks generally require a large amount of memory. In this paper, we show how to substantially reduce the memory consumption by divide-and-conquer strategies and present the first comprehensive experimental results for the BDD-attack on reduced versions of E_0, A5/1 and the self-shrinking generator.

Keywords: stream cipher, cryptanalysis, BDD, Bluetooth E_0, GSM A5/1, self-shrinking generator

1 Introduction

The main purpose of LFSR-based keystream generators is online encryption of bitstreams $p \in \{0,1\}^*$ that have to be sent over an insecure channel, e.g., for encrypting speech data to be transmitted from and to a mobile phone over the air interface. The output keystream $y \in \{0,1\}^*$ of the generator is bitwise XORed to the plaintext stream p in order to obtain the ciphertext stream $c \in \{0,1\}^*$, i.e., $c_i = p_i \oplus y_i$ for all i. Based on a secret initial state $x \in \{0,1\}^n$, which has to be exchanged between sender and legal receiver in advance, the receiver can compute the keystream y from x in the same way as the sender computed it and decrypt the message using the above rule.

We consider the special type of LFSR-based keystream generators that consist of a linear bitstream generator with a small number of Linear Feedback Shift Registers (LFSRs) and a non-linear compression function $C : \{0,1\}^* \to \{0,1\}^*$. From the secret key x, the LFSRs produce an internal bitstream $z \in \{0,1\}^*$, which is then transformed into the output keystream y via $y = C(z)$. Practical examples for this design include the E_0 generator, which is used as a building block for the cipher used in the Bluetooth standard for short-range wireless communication [4], the A5/1 generator from the widely used GSM standard for mobile telephones [5], and the self-shrinking generator [14].

Currently, the best attacks on E_0 in terms of time and memory requirements are algebraic attacks [1, 6] and correlation attacks [12, 11], but both types rely on the rather unrealistic assumption that a large amount of output keystream is available. The correlation attacks presented in [12, 11] additionally depend on the linearity of the key-schedule and other specific properties of the Bluetooth encryption system that could easily be altered in future versions of the cipher. Particularly, in [2] it was shown that small changes of the cipher design could completely avert the correlation attack in [12] and significantly worsen the efficiency of algebraic attacks on E_0.

The attack by Krause [10], which we consider in this paper, is a generic attack in the sense that it does not depend on specific design properties of the respective cipher. It only relies on the assumptions that the generator's output behaves pseudorandomly and that the test whether a given internal bitstream z produces a sample keystream can be represented in a Free Binary Decision Diagram (FBDD) of size polynomial in the length of z. In the case of E_0, the BDD-attack can easily be extended to an attack on the whole Bluetooth cipher. Another major advantage of the attack is that it reconstructs the secret key from the shortest information-theoretically possible prefix of the keystream; in the case of E_0 and A5/1, the first keystream frame already suffices to obtain all the information that is needed to compute the initial state, whereas both algebraic attacks and correlation attacks depend on the unrealistic number of at least 2^{23} available keystream frames. In fact, the BDD-attack is the best short-keystream attack on E_0 that is known so far.

Unlike both algebraic and correlation attacks, BDD-attacks can also be applied to irregularly clocked keystream generators like the A5/1 generator, for which the BDD-attack is one of the best generic attacks that do not depend on special properties of the GSM encryption system.

However, one drawback of the BDD-attack is its high memory consumption. We will approach this problem by presenting various efficiently parallelizable divide-and-conquer strategies (DCS) for E_0 and A5/1 that substantially reduce the memory requirements and allow us to tackle much larger keylengths with fixed computational resources. In the case of E_0, our DCS lowers the attack's memory requirements by a factor of 2^{25} and additionally yields a slight theoretical improvement of the theoretical runtime. Hence, we obtain the best attack on E_0 under realistic assumptions.

In [10], the application of the basic BDD-based attack to E_0, A5/1 and the self-shrinking generator were already theoretically described, but with rather pessimistic assumptions on the time and space requirements. We present the first comprehensive experimental results for the BDD-attack on reduced versions of these ciphers, showing that the performance in practice does not substantially deviate from the theoretical figures.

This paper is organized as follows. In Sect. 2, we introduce some notations, give an overview of Binary Decision Diagrams and their algorithmic properties, and review the original BDD-based attack presented in [10]. The impact of the BDD-attack on the keystream generators E_0, A5/1 and the self-shrinking generator are described together with their basic definitions in Sect. 3. Section 4 introduces our divide-and-conquer strategies for the attacks on E_0 and A5/1, and Sect. 5 presents our experimental results. Finally, Sect. 6 concludes the paper.

2 Preliminaries

2.1 LFSR-based Keystream Generators

In order to establish a consistent notation, we restate the definitions of linear feedback shift registers, linear bitstream generators and LFSR-based keystream generators and their basic properties.

Definition 1. *A* Linear Feedback Shift Register *(LFSR) of length n with a coefficient vector $c = (c_1, \ldots, c_n) \in \{0,1\}^n$ takes an initial state $x = (x_0, \ldots, x_{n-1}) \in \{0,1\}^n$ as input and produces a bitstream $l(x) = l_0(x), l_1(x), \ldots, l_i(x), \ldots$ according to*

$$l_i(x) := \begin{cases} x_i & \text{for } 0 \le i \le n-1 \\ \bigoplus_{k=0}^{n-1} c_{k+1} \cdot l_{i-n+k}(x) & \text{for } i > n-1 \end{cases} .$$

Note that each output bit of an LFSR is a linear combination of the initial state bits and that for each position i, there exists a subset $D(i) \subseteq \{0, \ldots, n-1\}$ such that $l_i(x) = \bigoplus_{j \in D(i)} x_j$. We call $D(i)$ the *domain* of i.

In practice, an LFSR is implemented in hardware with n binary register cells that are connected by a feedback channel.

Definition 2. *A* Linear Bitstream Generator *L consists of $k \ge 1$ parallel LFSRs L^r of length n_r, $r \in \{0, \ldots, k-1\}$, and $n_0 + \ldots + n_{k-1} = n$. L produces a bitstream $L(x) = L_0(x), L_1(x), \ldots, L_i(x), \ldots$ where*

$$L_i(x) := l_{s(i)}^{r(i)} \left(x^{r(i)} \right) \quad \text{where} \quad \begin{array}{l} r(i) = i \bmod k \\ s(i) = i \operatorname{div} k \end{array} ,$$

i.e., the i-th output bit of L corresponds to the $s(i)$-th output bit of LFSR $L^{r(i)}$. The initial states x^p of the LFSRs L^p, $p \in \{0, \ldots, k-1\}$, form the initial state $x \in \{0,1\}^n$ of L. For $i \ge 1$, we denote by $L_{\le i}(x)$ the i-extension *of x, i.e., the first i output bits $L_0(x), \ldots, L_{i-1}(x)$ that L produces from x.*

Definition 3. *An* LFSR-based (k,l)-keystream generator *(or (k,l)-combiner) $K = (L, C)$ consists of a linear bitstream generator L with k LFSRs and a nonlinear compression function $C : \{0,1\}^* \to \{0,1\}^*$ with l memory bits. From the secret key $x \in \{0,1\}^n$ that L is initialized with, K computes an internal bitstream $z = L(x)$ and transforms z into the output keystream via $y = C(z) =$*

$y_0, y_1, \ldots, y_i, \ldots$. *The compression function* C *computes the keystream in an online manner, i.e., there exists a function* $\delta : \mathbb{N} \to \mathbb{N}$ *with* $\delta(i) < \delta(j)$ *for* $i < j$*, such that* $y_i = C(z_0, \ldots, z_{\delta(i)-1})$*, i.e.,* y_i *only depends on the first* $\delta(i)$ *bits of* z*. Moreover,* C *reads the internal bits in the order in which they are produced by the LFSRs, i.e., for* $s > 0$ *and all* $r \in \{0, \ldots, k-1\}$*,* $L_{k \cdot s+r}(x)$ *is not read before* $L_{k\cdot(s-1)+r}(x)$*.*

We call a number $i \geq 1$ *a* key position *in* $L(x)$ *if* $L_i(x)$ *corresponds to one of the* x*-bits and a* non-key position *otherwise. Correspondingly, we denote by* $KP(i)$ *the set of key positions in* $\{0, \ldots, i-1\}$ *and by* $KB(z) \in \{0,1\}^{|KP(i)|}$ *the bits at the key positions in* $L(x)$*. Let* n_{min} *denote the maximum* i *for which all* $i' \leq i$ *are key positions and* $n_{\max}$ *the minimum* i *for which all* $i' > i$ *are non-key positions.*

In the context of the previous definitions, we can characterize the well-known regularly clocked combiners with memory (or shortly *regular* (k,l)*-combiners*), which consist of k LFSRs and an l-bit memory unit, in the following way.

Definition 4. *We call an LFSR-based* (k,l)*-keystream generator* regular*, if* y_i *only depends on the internal bits* $(z_{ki}, \ldots, z_{(k+1)i-1})$*, i.e., the* $(i+1)$*-st output bits of the LFSRs, and the state of the memory bits in iteration* i*.*

Definition 5. *Let* γ *denote the* best-case compression ratio $\gamma \in (0,1]$*, i.e.,* γm *is the maximum number of keybits that* C *produces from internal bitstreams of length* m*. For a randomly chosen and uniformly distributed internal bitstream* $Z^{(m)} \in \{0,1\}^m$ *and a random keystream* Y*, we define the average information that* Y *reveals about* $Z^{(m)}$ *as* $\alpha := \frac{1}{m} I\left(Z^{(m)}, Y\right) \in (0,1]$*.*[1]

For a randomly chosen and uniformly distributed internal bitstream $z \in \{0,1\}^m$, the probability of the keybits $C(z)$ being a prefix of a given keystream $y \in \{0,1\}^*$ can be expressed as

$$Prob_z[C(z) \text{ is prefix of } y] =$$

$$\sum_{i=0}^{\lceil \gamma m \rceil} Prob_{z \in \{0,1\}^m}[|C(z)| = i] \cdot Prob_{z \in \{0,1\}^m, |C(z)|=i}[C(z) = (y_0, \ldots, y_{i-1})] \ .$$

Concerning this probability, we will make the following assumption.

Assumption 1 (Independence Assumption). *For all* $m \geq 1$*, a randomly chosen, uniformly distributed internal bitstream* $z \in \{0,1\}^m$*, and all keystreams* $y \in \{0,1\}^*$*, we have* $Prob_z[C(z) \text{is prefix of } y] = p_C(m)$*, i.e., the probability of* $C(z)$ *being a prefix of* y *is the same for all* y*.*

As shown in [10], Assumption 1 yields $\alpha = -\frac{1}{m}\log_2 p_C(m)$.

From a straightforward calculation (c.f. [10] for details), we obtain

Observation 1. *For a regular* (k,l)*-combiner, it is* $\alpha = \gamma = \frac{1}{k}$*.*

[1] Recall that for two random variables A and B, the value $I(A,B) = H(A) - H(A|B)$ defines the information that B reveals about A.

Finally, we assume the keystream y to behave pseudorandomly.

Assumption 2 (Pseudorandomness Assumption). *For all keystreams y and all $m \leq \lceil \alpha^{-1} n \rceil$ it holds that $Prob_z[C(z)$ is prefix of $y] \approx Prob_x[C(L_{\leq m}(x))$ is prefix of $y]$, where z and x denote randomly chosen, uniformly distributed elements of $\{0,1\}^m$ and $\{0,1\}^{|KP(m)|}$, respectively.*

Note that a severe violation of Assumption 2 would imply a vulnerability of K via a correlation attack.

2.2 Binary Decision Diagrams (BDDs)

We briefly review the definitions of Binary Decision Diagrams and those algorithmic properties that are used in the BDD-based attack.

Definition 6. *A* Binary Decision Diagram *(BDD) over a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a directed, acyclic graph $G = (V, E)$ with $E \subseteq V \times V \times \{0,1\}$. Each inner node v has exactly two outgoing edges, a 0-edge $(v, v_0, 0)$ and a 1-edge $(v, v_1, 1)$ leading to the 0-successor v_0 and the 1-successor v_1, respectively. A BDD contains exactly two nodes with outdegree 0, the sinks s_0 and s_1. Each inner node v is assigned a label $v.label \in X_n$, whereas the two sinks are labeled $s_0.label = 0$ and $s_1.label = 1$. There is exacly one node with indegree 0, the root of the BDD. We define the size of a BDD to be the number of nodes in G, i.e., $|G| := |V|$. Each node $v \in V$ represents a Boolean Function $f_v \in B_n = \{f | f : \{0,1\}^n \rightarrow \{0,1\}\}$ in the following manner: For an input $a = (a_1, \ldots, a_n) \in \{0,1\}^n$, the computation of $f_v(a)$ starts in v. In a node with label x_i, the outgoing edge with label a_i is chosen, until one of the sinks is reached. The value $f_v(a)$ is then given by the label of this sink.*

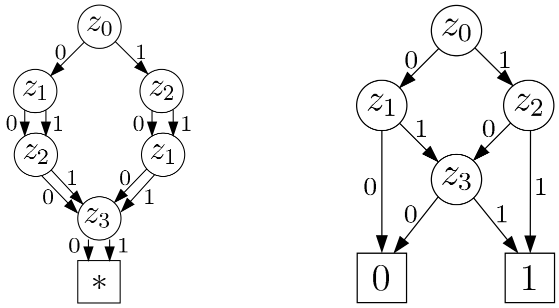
Definition 7. *For a BDD G over X_n, let $G^{-1}(1) \subseteq \{0,1\}^n$ denote the set of inputs accepted by G, i.e., all inputs $a \in \{0,1\}^n$ such that $f_{root}(v) = 1$.*

Definition 8. *An* oracle graph *$G_0 = (V, E)$ over a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a modified BDD that contains only one sink s, labeled $*$, and for all $x_i \in X_n$ and all paths P from the root in G to the sink, there exists at most one node in P that is labeled x_i.*

Definition 9. *A* Free Binary Decision Diagram *with respect to an oracle graph G_0 (a G_0-FBDD for short) over a set of variables $X_n = \{x_1, \ldots, x_n\}$ is a BDD in which the following property holds for all inputs $a \in \{0,1\}^n$. Let the list $G_0(a)$ contain the variables from X_n in the order in which they are tested on the path defined by a in G_0. Similarly, let the list $G(a)$ contain the variables from X_n in the order of testing in G. If x_i and x_j are both contained in $G(a)$, then they occur in $G(a)$ in the same order as in $G_0(a)$. We call a BDD G an FBDD, if there exists an oracle graph G_0 such that G is a G_0-FBDD.*

Figure 1 shows examples for an oracle graph G_0 and a G_0-FBDD.

Fig. 1. An oracle graph G_0 over $\{z_0, \ldots, z_3\}$ and a G_0-FBDD



Definition 10. *An FBDD G is called* Ordered Binary Decision Diagram *(OBDD) if there exists an oracle graph G_0 such that G is a G_0-FBDD and G_0 is degenerated into a linear list.*

FBDDs possess several algorithmic properties that will prove useful in our context. Let G_0 denote an oracle graph over $X_n = \{x_1, \ldots, x_n\}$ and let the G_0-FBDDs G_f, G_g and G_h represent Boolean functions $f, g, h : \{0,1\}^n \to \{0,1\}$. Then, there exists an algorithm MIN that computes in time $O(|G_f|)$ the (uniquely determined) minimal G_0-FBDD G of size $|G| \leq n \cdot |G_f^{-1}(1)|$ that represents f. In time $O(|G_0| \cdot |G_f| \cdot |G_g| \cdot |G_h|)$, we can compute a minimal G_0-FBDD G with $|G| \leq |G_0| \cdot |G_f| \cdot |G_g| \cdot |G_h|$ that represents the function $f \wedge g \wedge h$. Additionally, it is possible to enumerate all elementes in $G_f^{-1}(1)$ in time $O\left(n \cdot |G_f^{-1}(1)|\right)$. We refer the reader to [18] for details on the corresponding algorithms.

2.3 BDD-based Attack

The original BDD-based attack in [10], which we are going to describe in this section, assumes a known-plaintext scenario, i.e., the attacker manages to obtain a few plaintext-ciphertext pairs $(p_1, c_1), \ldots, (p_t, c_t) \in \{0,1\}^2$. Since the c_i were computed as $c_i = p_i \oplus y_i$ based on the output $y_0, \ldots, y_t \in \{0,1\}$ of an LFSR-based keystream generator $K = (L, C)$, he can compute the first t keybits as $y_i = p_i \oplus c_i$. From this prefix of the keystream, he wants to reconstruct the secret initial state x of L.

We observe that for any internal bitstream $z \in \{0,1\}^m$ that yields a prefix of the observed keystream-piece y, the following conditions must hold.

Condition 1. *z is an m-extension of the key bits in z, i.e., $L_{\leq m}(KB(z)) = z$.*

Condition 2. *$C(z)$ is a prefix of y.*

We call any z of length m that satisfies these conditions an m-candidate. The idea is now to start with $m = n_{min}$ and to dynamically compute the m-candidates for $m > n_{min}$, until only one m-candidate is left. The smallest m for which this will be most likely the case follows directly from the following Lemma.

Lemma 1. *Under Assumption 2, it holds for all keystreams y and all $m \leq \lceil \alpha^{-1} n \rceil$ that $|\{x \in \{0,1\}^n : C(L_{\leq m}(x))$ is prefix of $y\}| \approx 2^{n^*-\alpha m} \leq 2^{n-\alpha m}$, where $n^* = |KP(m)|$. Hence, there exist approximately $2^{n-\alpha m}$ m-candidates.*

Lemma 1 implies that there will be only one m-candidate for $m \geq \lceil \alpha^{-1} n \rceil$. The key bits in this m-candidate form the secret initial state that the attacker is looking for.

In order to compute and represent the intermediate m-candidates efficiently, we use the following BDD-based approach. For $m \geq 1$, let G_m^C denote the oracle graph over $\{z_0, \ldots, z_{m-1}\}$ that determines for each internal bitstream $z = (z_0', \ldots, z_{m-1}') \in \{0,1\}^m$ the order in which the bits of z are read by the compression function C. Bitstreams z fulfilling conditions 1 and 2 will be represented in the minimal G_m^C-FBDDs R_m and Q_m^y, respectively. Then, the G_m^C-FBDD $P_m^y = \mathrm{MIN}(Q_m^y \wedge R_m)$ accepts exactly the m-candidates.

The cost of this strategy essentially depends on the sizes of the intermediate results P_m^y, which can be determined as follows.

Assumption 3 (FBDD Assumption). *For all $m \geq n_{min}$, it holds that $|G_m^C| \in m^{O(1)}$, $|Q_m| \in m^{O(1)}$, and $|R_m| \leq |G_m^C| 2^{m-n^*}$.*

Lemma 2. *If K fulfills Assumption 3, then*

$$|P_m^y| \leq \max_{1 \leq m \leq \lceil \alpha^{-1} n \rceil} \left\{ \min \left\{ p(m) \cdot 2^{m-n^*}, m \cdot 2^{n^*-\alpha m} \right\} \right\} \leq p(m) \cdot 2^{r^*(m)} \quad ,$$

where $p(m) = |G_m^C|^2 \cdot |Q_m^y|$ and $r^*(m) = \frac{1-\alpha}{1+\alpha} n^*$.

From this bound on $|P_m^y|$, one can straightforwardly derive the time, space and data requirements of the BDD-based attack.

Theorem 1. *Let $K = (L, C)$ be an LFSR-based keystream generator with initial state $x \in \{0,1\}^n$, information ratio α and best-case compression ratio γ. If K fulfills the Independence Assumption, the Pseudorandomness Assumption and the FBDD Assumption, an initial state $\tilde{x}$ with $C(L(\tilde{x})) = y$ for a given keystream $y = C(L(x))$ can be computed in time and with space $n^{O(1)} 2^{\frac{1-\alpha}{1+\alpha} n}$ from the first $\lceil \gamma \alpha^{-1} n \rceil$ consecutive bits of y.*

3 Applications

We now survey the impact of the basic BDD-attack on the self-shrinking generator, the E_0 generator, and the A5/1 generator and compare it to other attacks on these ciphers.

The self-shrinking generator was introduced by Meier and Staffelbach [14]. It consists of only one LFSR and no memory. [10] showed that for the self-shrinking Generator, we have $\alpha \approx 0.2075$ and $\gamma = 0.5$ as well as $|Q_m| \leq m^2$ for $m \geq 1$.

Corollary 1. *From a prefix of length $\lceil 2.41n \rceil$ of a keystream $y = C(L(x))$ produced by a self-shrinking generator of keylength n, an initial state $\tilde{x}$ with $C(L(\tilde{x})) = y$ can be computed in time and with space $n^{O(1)} 2^{0.6563n}$.*

This is the best short-keystream attack on the self-shrinking generator known so far. It slightly improves the bounds of $2^{0.75n}$ and $2^{0.694n}$ that were obtained in [14] and [19], respectively. The long-keystream attack in [15] needs at least $2^{0.3n}$ keystream bits in order to compute the initial state in less than $2^{0.6563n}$ polynomial-time operations.

The E_0 keystream generator from the short-range wireless communication standard Bluetooth [4] is a regular $(4,4)$-combiner with key length 128; its LFSRs have lengths $39, 33, 31, 25$. Therefore, we have $\alpha = \gamma = \frac{1}{4}$, and [10] showed that $|Q_m| \leq 32m$. Hence, we obtain from Theorem 1:

Corollary 2. *From a prefix of length n of a keystream $y = C(L(x))$ produced by an E_0 keystream generator of keylength n, an initial state $\tilde{x}$ with $C(L(\tilde{x})) = y$ can be computed in time and with space $n^{O(1)}2^{0.6n} = n^{O(1)}2^{76.8}$ for $n = 128$.*

The attack on E_0 by Fluhrer and Lucks [8] trades off time and necessary keystream bits. For the minimum number of 132 available keystream bits the attack needs 2^{84} polynomial time operations. The best currently known long-keystream attacks against E_0 are algebraic attacks [1] and correlation attacks [12, 11]. These attacks all need a large amount of keystream (2^{28} to 2^{39} in the case of correlation attacks), and even in terms of time and memory requirements, [11] is the only feasible attack among them.

The A5/1 generator is used in the GSM standard for mobile telephones. According to [5], who obtained its design by reverse engineering, the generator consists of 3 LFSRs R_0, R_1, R_2 of lengths n_0, n_1, n_2, respectively, and a clock control ensuring that the keybits do not linearly depend on the initial states of the LFSRs. For each $r \in \{0,1,2\}$, a register cell q_{N^r}, $N^r \in \{\lceil \frac{n_r}{2} \rceil - 1, \lceil \frac{n_r}{2} \rceil\}$, is selected in LFSR R_r as input for the clock control. The GSM standard uses the parameters $(n_0, n_1, n_2) = (19, 22, 23)$ and $(N^0, N^1, N^2) = (11, 12, 13)$.

In order to write the generator in a $K = (L, C)$ fashion, we simulate its linear bitstream generator by six LFSRs $L^0, \ldots, L^5$. L^0, L^1, and L^2 are used exclusively for producing the keybits and correspond to R_0, R_1 and R_2 in the original generator, and the control values are computed from the outputs of L^3, L^4 and L^5, which correspond to L^0, L^1 and L^2 shifted by N^0, N^1 and N^2.

In [10], it was shown that in the case of A5/1, $\alpha = 0.2193$ and $\gamma = \frac{1}{4}$ as well as $|G_m^C| \in O(m^3)$ and $|Q_m| \in O(m^4)$. Plugging these values into the statement of Theorem 1 yields

Corollary 3. *From a prefix of length $\lceil 1.14n \rceil$ of a keystream $y = C(L(x))$ produced by an A5/1 keystream generator of keylength n, an initial state $\tilde{x}$ with $C(L(\tilde{x})) = y$ can be computed in time and with space $n^{O(1)}2^{0.6403n} = n^{O(1)}2^{41}$ for $n = 64$.*

We note that since $\lceil 1.14n \rceil = 73$ and the framelength in GSM is 114 Bits for each direction, we only need the first frame, i.e., the first around 4.6 milliseconds of a conversation in order to reconstruct the initial state.

The first short-keystream attack on A5/1 was given by Golić in [9] and needs 2^{42} polynomial time operations. Afterwards, several long-keystream attacks on A5/1 were proposed. [3] presents an attack that breaks A5/1 from 2^{15} known keystream bits within minutes after a preprocessing step of 2^{48} operations. Due to exploits of the linearity of the initialization procedure, the attack described in [7] and its refinement in [13] manage to break the cipher within minutes, requiring only few seconds of conversation and little computational resources.

4 Divide-and-Conquer Strategies

One obvious disadvantage of BDD-based attacks is the high memory consumption that is essentially determined by the size of the intermediate results P_m^y. For an LFSR-based keystream generator with keylength n, one possible approach to this problem is to divide the search space, more precisely the set $B_n = \{0,1\}^n$ of possible initial states of L, into segments and to apply BDD-based attacks to the segments individually. We denote a segmentation of B_n by the pair (f, T), where T is the finite set of segment labels and $f : \{0,1\}^* \to T$ a partial function that assigns a segment to each possible internal bitstream. For a given keystream y and each $t \in T$, we perform a BDD-based search on the set $B_n^t = \{x \in B_n | f(L(x)) = t\}$ in order to find an initial state $\tilde{x} \in B_n^t$ such that $C(L(\tilde{x})) = y$.

Similarly to the general attack described in the previous section, we denote by $Q_m^{y,t}$ the minimal G_0^C-FBDD that decides whether $C(z) = y$ and $f(z) = t$, and by R_m^t the minimal G_m^C-FBDD that accepts for $f(z) = t$ exactly those internal bitstreams z that are m-extensions of $KB(z)$. Moreover, let S_m^t be the minimal G_m^C-FBDD that decides for $f(z) = t$ whether $z_{m-1} = L_{m-1}(KB(z))$ and define $P_m^{y,t} := MIN(Q_m^{y,t} \wedge R_m^t)$. We can then apply the same algorithm for dynamically computing $P_{n_{min}}^{y,t}, P_{n_{min}+1}^{y,t}, \ldots$ as in the original case. Consequently, we obtain $n^{O(1)}2^{w^*}$ as time and space requirements for the BDD-based search on B_n^t, with w^* computed analogously to r^* in Lemma 2. For the overall attack, i.e., performing the BDD-based search on B_n^t for all $t \in T$, we get a memory consumption in the order of $n^{O(1)}2^{w^*}$ and a runtime of $n^{O(1)} \cdot |T| \cdot 2^{w^*}$ if the attacks are executed sequentially. Since the B_n^t are disjoint, the overall attack is efficiently parallelizable, and the $|T|$ factor can be further reduced.

We note that in general, we will only gain from a divide-and-conquer strategy (DCS) if $|T|$ is not too large and $w^* \leq r^*$. For the latter to be the case, the $|Q_m^{y,t}|$ have to be negligibly small and $|R_m^t|$ must be significantly smaller than $|R_m|$.

We consider now DCS that define a subset $V \subseteq KP(n_{\max})$ of the initial state bits of L to be constant. We call a position $m \geq 1$ a V-*determined* position if $m \in V$ or if its domain $D(m)$ is a subset of V. For an internal bitstream z, let $t \in \{0,1\}^{|V|}$ denote the values of z at the positions in V. Then, the segmentation of B_n is given by $(f_V, T(V))$, where $f_V(z) = t$ and $T(V) = \{0,1\}^{|V|}$.

The FBDDs $Q_m^{y,t}$ can be obtained from Q_m^y by setting constant the variables that correspond to the V-determined positions. Hence, $|Q_m^{y,t}| \leq |Q_m^y|$. Moreover, since the test whether $z_{m-1} = L_{m-1}(KB(z))$ can be omitted for the V-determined positions, we have $|R_m^t| \leq |G_m^C| 2^{r(m,V)}$, where $r(m, V)$ denotes the number of non-V-determined positions in $\{n_{min} + 1, \ldots, m\}$. Note that the original attack corresponds to the case $V = \oslash$ and therefore $r(m, V) \leq m - n^*$, hence $|R^t_m| \leq |R_m|$.

4.1 DCS for Regular (k, l)-Combiners

We consider two examples that are applicable to regular (k, l)-combiners like the E_0 keystream generator.

First, we define V to contain exactly the positions of the first s output bits of each LFSR. In the worst case, there are no V-determined positions besides the positions in V. We only need to consider the assignments to the positions in V that are consistent with y. By Lemma 1, we have $|T(V)| \approx |\{0,1\}^{(1-\alpha)ks}| = 2^{(k-1)s}$. For $t \in T(V)$, the effort of a BDD-based search of the corresponding segment is equivalent to the effort for the original BDD-attack on a (k, l)-combiner of keylength $(n - ks)$, i.e., $w^* = \frac{k-1}{k+1}(n - ks)$. For the overall runtime, we obtain

$$n^{O(1)} \cdot 2^{(k-1)s + \frac{k-1}{k+1}(n-ks)} \in n^{O(1)} 2^{\frac{k-1}{k+1}n + \frac{k-1}{k+1}s} \quad ,$$

which is by a factor of $2^{\frac{k-1}{k+1}s}$ worse than the original attack . On the other hand, the required memory is reduced by a factor of $2^{\frac{k-1}{k+1}ks}$.

As a second example, we choose as V the set of all key positions that belong to the shortest LFSR in L, which we assume w.l.o.g. to be the LFSR L^0. Let $n_0 \leq \frac{n}{k}$ be the length of L^0. Then, $T(V) = \{0,1\}^{n_0}$ is the set of all possible initial states of L^0. Since every k-th position of an internal bitstream z is V-determined, w^* corresponds to the performance of the original BDD-attack on a $(k-1, l)$ combiner of keylength $n - n_0$, i.e., $w^* = \frac{k-2}{k}(n - n_0)$. For the overall runtime, we obtain $2^{n_0 + \frac{k-2}{k}(n-n_0)}$. It is easy to see that for $n_0 \leq \frac{n}{k+1}$, we have

$$n_0 + \frac{k-2}{k}(n - n_0) \leq \frac{k-1}{k+1} n \quad ,$$

i.e., for sufficiently small n_0, we even obtain a runtime improvement in addition to the significantly reduced space requirements. In the case of the original E_0, we have $n_0 = 25 \leq 25.6 = \frac{128}{4+1}$. Hence, we obtain

Lemma 3. *For the E_0 keystream generator with keylength $n = 128$, choosing V to be the set of all key positions that belong to the shortest LFSR yields a runtime of the BDD-based attack of $n^{O(1)} 2^{25 + \frac{1}{2}103} = 2^{76.5}$ and a memory consumption of $2^{51.5}$.*

Compared to the original BDD-attack, we have improved the memory consumption by a factor of about 2^{25} and the runtime by a factor of $2^{0.3}$.

4.2 DCS for the A5/1 Generator

In the following, we compute the information rate of the A5/1 generator with respect to a family of choices for the set V, particularly those defined by setting one or several LFSRs or half-LFSRs to be constant. As stated in Sect. 3, in the unmodified definition of the A5/1 generator, each of the three LFSRs is divided into two, approximately equally long halfs, a value-half consisting of the output cell and the cells between output and clock-control cell and a control half consisting of the clock-control cell and the rest of the register. Since the value-LFSRs and the control-LFSRs in the modified setting correspond to the value-halfs and the control-halfs in the unmodified case, setting constant LFSRs or half-LFSRs in the original definition is equivalent to fixing the corresponding LFSRs in the modified case.

For all natural $i \geq 1$, let us denote by Y_i and Z_i the random variables corresponding to the i-th output bit and the number of internal bits processed for the production of the i-th output bit, respectively, taken over the probability space of all random internal bitstreams. In all cases, Y_i and Z_i will fulfill the following conditions.

– For all $i > 1$, Z_i is independent of $Z_1, \ldots, Z_{i-1}$, and Y_i is independent of $Y_1, \ldots, Y_{i-1}$.
– It holds that $Pr[Y_i = 0] = Pr[Y_i = 1] = \frac{1}{2}$.
– There are natural numbers $a > b > c$ and probabilities p, q and $r = 1-p-q$ such that $Pr[Z_i = a] = p$, $Pr[Z_i = b] = q$, and $Pr[Z_i = c] = r$.

Let us denote the situation that Y_i and Z_i fulfill the above conditions as case $[(p, a), (q, b), (r, c)]$. It can be easily checked that the unrestricted A5/1 generator corresponds to case $[(1/4, 6), (3/4, 4), (0, 0)]$. We will see below that all generators derived from the A5/1 generator by setting constant one or more of the six LFSRs correspond to $[(p, a), (q, b), (r, c)]$ for some p, q, r, a, b, c. In these cases, we can compute the information rate α with the help of the following Theorem.

Theorem 2. *In the case $[(p, a), (q, b), (r, c)]$, the information rate equals α, where $t = 2^\alpha$ is the unique positive real solution of $pt^a + qt^b + rt^c - 2 = 0$.*

A proof for Theorem 2 can be found in Appendix A. Note that for the special case $[(1, k), 0, 0]$ the information rate is $1/k$.

In the following, we compute the information rates for restrictions of type $(v_1v_2v_3|c_1c_2c_3) \in \{0, 1\}^6$, which means that those value-substreams i for which $v_i = 1$ and control-substreams j for which $c_j = 1$ are set constant. Note that the unrestricted case corresponds to $(000|000)$. We do not consider the case of 5 constant internal substreams as computing the remaining unknown half-LFSR from a given keystream can be done in linear time.

For symmerty reasons, the number of remaining cases resulting from setting constant 1,2,3,4 substreams can be reduced. Firstly, it is easy to see that for all permutations π of $\{1, 2, 3\}$ it holds that restriction $(v_1v_2v_3|c_1c_2c_3)$ is equivalent to restiction $(v_{\pi(1)}, v_{\pi(2)}v_{\pi(3)}|c_{\pi(1)}c_{\pi(2)}c_{\pi(3)})$. Furthermore, observe that with respect to restriction $(v|c)$, $v, c \in \{0, 1\}^3$, the number of internal bits $Z(u, V, C)$ processed for the production of the next output bit assuming the current values in the control-substreams are $u \in \{0, 1\}^3$ equals

$$Z(u, v, c) = \sum_{i, c_i=0} f_i(u) + \sum_{i, v_i=0} f_i(u) \ , \tag{1}$$

where for $i \in \{1,2,3\}$ the Boolean function $f_i : \{0,1\}^3 \rightarrow \{0,1\}$ is defined to output 1 on u iff the i-th LFSR will be clocked w.r.t. u, i.e.,

$$f_i(u) = (u_i \oplus u_{i+1 \mod 3} \oplus 1) \vee (u_i \oplus u_{i+2 \mod 3} \oplus 1) \ .$$

Relation (1) implies that for all $v, c, u \in \{0,1\}^3$ and $i \in \{1,2,3\}$, it holds that $Z(u,v,c) = Z(u,v',c')$, where v', c' are obtained from v, c by exchanging the i-th component. Hence, restriction $(v|c)$ is equivalent to restriction $(v'|c')$. It follows that the relevant cases are the restrictions (000|100), (100|100), (100|010), (100|110), (000|111), (100|111) and (110|110).

The information rates for these cases are summarized in Table 1. The computation of the values can be found in Appendix B.

Table 1. Information rates α

$\log\lvert T\rvert$	restriction	α	w^*
$\frac{2}{3}n$	(100|111)	0.6430	$0.2173n$
	(110|110)	0.6113	$0.2412n$
$\frac{1}{2}n$	(000|111)	0.4386	$0.3902n$
	(100|110)	0.4261	$0.4024n$
$\frac{1}{3}n$	(000|110)	0.3271	$0.507n$
	(100|100)	0.3215	$0.5134n$
$\frac{1}{6}n$	(000|100)	0.2622	$0.584n$
0	(000|000)	0.2193	$0.6403n$

5 Experimental Results

In order to provide a fast implementation of the FBDD algorithms, an FBDD-library was developed based on the publicly available OBDD package CUDD (see [17]). The experiments were conducted on a standard Linux PC with a 2.7 GHz Intel Xeon processor and 4 GB of RAM. All implementation was done in C using the gcc-compiler version 3.3.5.

Since the runtime of the cryptanalysis fundamentally depends on the maximum size of the intermediate FBDDs P_m^y, we investigate how much experimentally obtained values of $|P_m^y|$ deviate from the theoretical figures.

We first consider the basic BDD-based attack. For the self-shrinking generator, the E_0 generator and the A5/1 generator, we analyzed several thousands of reduced instances with random primitive feedback polynomials and random initial states for various keylengths. For each considered random generator, we computed the actual maximum BDD-size of the intermediate results

$$P_{\max}(n) = \max_{1 \leq m \leq \lceil \alpha^{-1} n \rceil} \{|P_m^y|\} \ ,$$

the theoretical upper bound

$$P^t_{\max}(n) = \max_{1 \le m \le \lceil \alpha^{-1} n \rceil} \left\{ \min \left\{ p(m) \cdot 2^{m-n^*}, m \cdot 2^{n^* - \alpha m} \right\} \right\}$$

that was obtained in Lemma 2, as well as the quotient $q(n) = \frac{\log(P_{\max}(n))}{\log(P^t_{\max}(n))}$.

Similarly, we tested for E_0 and A5/1 the divide-and-conquer strategy of setting constant the shortest LFSR (s1), and we considered fixing the first $s = \frac{n_0}{2} \le \frac{n}{8}$ bits of each of the four LFSRs in E_0 (s2), where n_0 denotes the length of the shortest LFSR. Since the q-values did not noticeably decrease with increasing n in all our simulations, we estimate the attack's performance in dependence of n by multiplying the theoretical figures by $2^{q(n)}$. Particularly, we can obtain conjectures about the attack's performance on real-life instances of E_0 and A5/1 by replacing n with the actual keylengths. Table 2 shows the results of these computations along with details about the conducted experiments.

On average, the attack based on DCS s1 took 87 minutes for E_0 with $n = 37$ and 54 minutes for A5/1 with $n = 30$. The longest keylengths that we were able to tackle with the resources described at the beginning of this section were $n = 46$ for E_0 and $n = 37$ for A5/1. These attacks used up almost all of the available memory and took 60.5 and 25.1 hours to complete on average.

Table 2. Performance of the BDD-based attack in practice

generator	DCS	keylength interval	avg $q(n)$	no. of samples	estimated practical performance			
					Time		Space	
E_0	–	$[19, 37]$	0.85	2000	$2^{0.51n}$	$2^{65.28}$	$2^{0.51n}$	$2^{65.28}$
E_0	s1	$[19, 37]$	0.95	2700	$2^{0.475(n+n_0)}$	$2^{72.68}$	$2^{0.475(n-n_0)}$	$2^{48.93}$
E_0	s2	$[19, 37]$	0.9	2700	$2^{0.54n+0.27n_0}$	$2^{75.87}$	$2^{0.54n-1.08n_0}$	$2^{42.12}$
A5/1	–	$[15, 30]$	0.9	3000	$2^{0.5763n}$	$2^{36.88}$	$2^{0.5763n}$	$2^{36.88}$
A5/1	s1	$[19, 37]$	0.77	2400	$2^{0.3953n+0.77n_0}$	$2^{39.93}$	$2^{0.3953n}$	$2^{25.30}$
SSG	–	$[10, 35]$	0.8	3300	$2^{0.525n}$		$2^{0.525n}$	

6 Conclusion

In this paper, we have presented the first comprehensive experimental results for the BDD-based attack on the self-shrinking generator, the E_0 and the A5/1. Our analysis shows that the performance of the BDD-attack on these generators in practice will not substantially drop below the theoretical upper bounds. We introduced divide-and-conquer strategies based on setting constant several initial state bits of the LFSRs and confirmed experimentally that in this way, the memory consumption of the attack may be reduced at the expense of slightly increasing the runtime. We have only applied a few examples of DCS to the E_0

and the $A5/1$ generator. In [16], an additional DCS for E_0 is reported which lowers the memory requirements to about 2^{23} while increasing the runtime to $O(2^{83})$. It is an interesting open question if there exist more efficient strategies that are able to simultaneously reduce the runtime by a significant amount.

7 Acknowledgement

We would like to thank Frederik Armknecht for valuable comments and discussions.

References

1. F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In *Proc. of CYPTO 2003*, volume 2729 of *LNCS*, pages 162–176. Springer, 2003.
2. F. Armknecht, M. Krause, and D. Stegemann. Design principles for combiners with memory. In *Proc. of INDOCRYPT 2005*, volume 3797 of *LNCS*, pages 104–117. Springer, 2005.
3. A. Biryukov, A. Shamir, and D. Wagner. Real time cryptanalysis of A5/1 on a PC. In *Proc. of Fast Software Encryption 2000*, volume 1978 of *LNCS*, pages 1–13. Springer, 2000.
4. The Bluetooth SIG. *Specification of the Bluetooth System*, February 2001.
5. M. Briceno, I. Goldberg, and D. Wagner. *A pedagogical implementation of A5/1*, May 1999. http://jya.com/a51-pi.htm.
6. N. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In *Proc. of CRYPTO 2003*, LNCS, pages 177–194. Springer, 2003.
7. P. Ekdahl and T. Johansson. Another attack on A5/1. In *Proc. of International Symposium on Information Theory*, page 160. IEEE, 2001.
8. S. R. Fluhrer and S. Lucks. Analysis of the E_0 encryption system. In *Proc. of SAC 2001*, volume 2259 of *LNCS*, pages 38–48. Springer, 2001.
9. J. Golić. Cryptanalysis of alleged A5 stream cipher. In *Proc. of EUROCRYPT 1997*, volume 1233 of *LNCS*, pages 239–255. Springer, 1997.
10. M. Krause. BDD-based cryptanalysis of keystream generators. In *Proc. of EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 222–237. Springer, 2002.
11. Y. Lu, W. Meier, and S. Vaudenay. The conditional correlation attack: A practical attack on bluetooth encryption. In *Proc. of CRYPTO 2005*, volume 3621 of *LNCS*, pages 97–117. Springer, 2005.
12. Y. Lu and S. Vaudenay. Cryptanalysis of the bluetooth keystream generator two-level E0. In *Proc. of ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 483–499. Springer, 2004.
13. A. Maximov, T. Johansson, and S. Babbage. An improved correlation attack on A5/1. In *Proc. of SAC 2004*, volume 3357 of *LNCS*, pages 1–18. Springer, 2004.
14. W. Meier and O. Staffelbach. The self-shrinking generator. In *Proc. of EUROCRYPT 1994*, volume 950 of *LNCS*, pages 205–214. Springer, 1994.
15. M. J. Mihaljević. A faster cryptanalysis of the self-shrinking generator. In *Proc. of ACISP 1996*, volume 1172 of *LNCS*, pages 192–189. Springer, 1996.
16. Y. Shaked and A. Wool. Cryptanalysis of the bluetooth E_0 cipher using OBDDs. Technical report, Cryptology ePrint Archive, Report 2006/072, 2006.

17. F. Somenzi. *CUDD: CU decision diagram package.* University of Colorado, Boulder, CO, USA, March 2001. http://vlsi.colorado.edu/~fabio/.
18. I. Wegener. *Branching Programs and Binary Decision Diagrams: Theory and Applications.* SIAM Monographs on Discrete Mathematics and Applications, 2000.
19. E. Zenner, M. Krause, and S. Lucks. Improved cryptanalysis of the self-shrinking generator. In *Proc. of ACISP 2001*, volume 2119 of *LNCS*, pages 21–35. Springer, 2001.

A Proof of Theorem 2

In order to prove Theorem 2, we need the following technical result that was proved in [10].

Lemma 4. *For all natural $N \geq 1$, probabilities $p \in (0,1)$ and real $\beta > 0$ it holds that* $\sum_{i=0}^{N} \binom{N}{i} p^i (1-p)^{N-i} 2^{\beta i} = \left(1 - p + p2^{\beta}\right)^N$.

Since we can obtain the information rate α from $\alpha = -\frac{1}{m}\log_2 p_C(m)$, we now compute the probability $p_C(m) = Prob_z[C(z) \text{ is prefix of } y]$ for the cases that parts of the LFSRs are set constant.

Case $[(p,a),(q,b),(r,c)]$ implies that on all random internal bitstreams of length m, m divisible by a, at least m/a output bits are produced. The number of internal bits remaining from m internal bits after the production of m/a output bits can be computed as

$$m - aU - bV - c\left(\frac{m}{a} - U - V\right) = \frac{a-c}{a}m - (a-c)U - (b-c)V \ ,$$

where U and V denote the number of output bits among the first m/a output bits for which a, resp. b internal bits are processed. Note that U is $(p, m/a)$-binomially distributed and that V, under the condition that $U = i$, is $(q/(q+r), m/a - i)$-binomially distributed. We obtain the following relation for $p_C(m)$.

$$p_C(m) = 2^{-\frac{m}{a}} \sum_{i=0}^{\frac{m}{a}} \sum_{j=0}^{\frac{m}{a}-i} Pr[U=i, V=j]\, p\left(\frac{a-c}{a}m - (a-c)i - (b-c)j\right) \quad \text{,i.e.,}$$

$$2^{-\alpha m} = 2^{-\frac{m}{a}} \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i}$$

$$\sum_{j=0}^{\frac{m}{a}-i} \binom{\frac{m}{a}-i}{j} \left(\frac{q}{q+r}\right)^j \left(\frac{r}{q+r}\right)^{\frac{m}{a}-i-j} \cdot 2^{-\alpha(\frac{a-c}{a}m - (a-c)i - (b-c)j)} \quad \text{,i.e.,}$$

$$2^{(1 - a\alpha + (a-c)\alpha)\frac{m}{a}} = \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{(a-c)\alpha i}$$

$$\sum_{j=0}^{\frac{m}{a}-i} \binom{\frac{m}{a}-i}{j} \left(\frac{q}{1-p}\right)^j \left(\frac{r}{1-p}\right)^{\frac{m}{a}-i-j} \cdot 2^{(b-c)\alpha j} \ .$$

Now, we apply Lemma 4 to the inner sum and obtain

$$2^{(1-n\alpha)\frac{m}{a}} = \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{(a-c)\alpha i} \cdot \left(\frac{r}{1-p} + \frac{q}{1-p} 2^{(b-c)\alpha}\right)^{\frac{m}{a}-i}.$$

Setting $s = \frac{r}{1-p} + \frac{q}{1-p} 2^{(b-c)\alpha}$, we get

$$\left(\frac{2}{s2^{c\alpha}}\right)^{\frac{m}{a}} = \sum_{i=0}^{\frac{m}{a}} \binom{\frac{m}{a}}{i} p^i (1-p)^{\frac{m}{a}-i} \cdot 2^{((a-c)\alpha - \log(s))i} = \left(1 - p + p2^{(a-c)\alpha - \log(s)}\right)^{\frac{m}{a}}.$$

Consequently, setting $t = 2^\alpha$, we obtain

$$\frac{2}{st^c} = 1 - p + p\frac{t^{a-c}}{s} \quad \Leftrightarrow \quad 2 = (1-p)st^c + pt^a.$$

$s = \frac{r}{1-p} + \frac{q}{1-p} t^{b-c}$ implies $2 = rt^c + qt^b + pt^a$, which yields the Theorem.

B Computation of α for the considered DCS for A5/1

In order to compute the remaining α values, we only need to compute the corresponding cases of the form $[(p, a), (q, b), (r, c)]$ for the given restrictions on the LFSRs.

We first consider the restriction (100|100). If the actual content of the output cells of the two non-constant control substreams is 00 or 11, then 4 internal bits will be processed, otherwise 2 internal bits will be processed. Hence, the corresponding case is $[(1/2, 4), (1/2, 2), 0]$ and therefore $\alpha \approx 0.3215$.

Under restriction (100|010), 4 internal bits will be processed if the actual content of the output cell of the constant control substream is $b \in \{0, 1\}$ and the actual content of the two non-constant control substreams is bb. If it is $b\bar{b}$ then 2, and in all remaining cases 3 internal bits will be processed. Therefore, the corresponding case is $[(1/4, 4), (1/2, 3), (1/4, 2)]$ and $\alpha \approx 0.3271$.

If we assume restriction (110|110), 2 internal bits will be processed if the assignments to the output cells of the constant control substreams is 01 or 10 or if all 3 output cells of the control-substreams coincide. If the assignment to the output cells of the constant control substreams is bb for some $b \in \{0, 1\}$ and the random assignment to the remaining control is output cell is $\neg b$, then the next output bit depends only on the constant assignments, and no internal bit will be processed. This implies that, in contrast to the above cases, $p_C(m)$ and α are not independent of the constant substreams and the given keystream. Therefore, we compute only the average information rate over all possible assignments to the constant control and output substreams. According to the above observation, the probability that 2 internal bits are processed for the next output bit is 3/4, and the probability that 0 internal bits are processed for the next ouput bit is 1/4. In total, we obtain $[(3/4, 4), (1/4, 0), 0]$ and therefore $\alpha \approx 0.6113$.

We can handle the remaining cases with similar arguments.