Extended version of the paper that will appear under the same title in the proceedings of SAC 2004,
*Lecture Notes in Computer Science*, Springer-Verlag, 2004.

# Extending the Resynchronization Attack (extended version) *

Frederik Armknecht[1]** and Joseph Lano[2]*** and Bart Preneel[2]

[1] Universität Mannheim
Theoretische Informatik
68131 Mannheim, Germany
armknecht@th.informatik.uni-mannheim.de
[2] Katholieke Universiteit Leuven
Dept. Elect. Eng.-ESAT/SCD-COSIC,
Kasteelpark Arenberg 10, 3001 Heverlee, Belgium
{joseph.lano,bart.preneel}@esat.kuleuven.ac.be

**Abstract.** Synchronous stream ciphers need perfect synchronization between sender and receiver. In practice, this is ensured by a resync mechanism. Daemen et al. [9] first described attacks on ciphers using such a resync mechanism. In this paper, we extend their attacks in several ways by combining the standard attack with cryptanalytic techniques such as algebraic attacks and linear cryptanalysis. Our results show that using linear resync mechanisms should be avoided, and provide lower bounds for the nonlinearity required from a secure resync mechanism.

## 1 Introduction

Synchronous stream ciphers generate a key stream independently from the plaintext. They typically consist of a finite state machine from which at each iteration a key stream bit is generated by an output function. Synchronous stream ciphers have the advantage that there is no error propagation. On the other hand, perfect synchronization between sender and receiver is required. In order to prevent synchronization loss or to restore synchronization after synchronization loss is detected, a resynchronization mechanism is used. Such a mechanism generates a new initial state for the finite state machine from the secret key and a unique

* This work was supported by the Concerted Research Action (GOA) Mefisto-2000/04 of the Flemish Government.
** This work has been supported by grant 620307 of the DFG (German Research Foundation)
*** Research financed by a Ph.D. grant of the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT-Vlaanderen)

initialization vector $IV$ and thus prevents the reuse of key stream. For the sake of efficiency the resynchronization mechanism should be as fast as possible.

Daemen, Govaerts and Vandewalle [9] observed that this resynchronization mechanism can lead to a new type of attacks on synchronous stream ciphers. They also showed an efficient attack on nonlinearly filtered systems using a linear resynchronization mechanism and using an output Boolean function with few inputs. Golic and Morgari [12] extended this attack to the case where the output function is unknown. Borissov et al. [6] showed that a ciphertext-only attack is also possible in some cases.

In this paper, we extend the resynchronization attack to overcome some limitations of the original attack of Daemen *et al.* We achieve this by further refining the original resynchronization attack and by combining the attack with other attack methodologies, notably with algebraic attacks.

The paper is organized as follows. In Sect. 2, we present some preliminary notions: Boolean functions, the general framework of a stream cipher, algebraic attacks and resynchronization attacks. In Sect. 3 we present the Daemen *et al.* attack and its limitations. In Sect. 4 we show how to perform the Daemen *et al.* attack in real-time by precomputation. In Sect. 5 we describe several methods to mount a resync attack when the number of resyncs is small. Section. 6 describes methods to mount attacks when the output function has many inputs. In Sect. 7, we describe attacks on stream ciphers with memory and in Sect. 8 we discuss attacks on stream ciphers with nonlinear resynchronization mechanism. In Sect. 9, we apply some of the attacks described on some stream ciphers: $E0$ from the Bluetooth standard, Toyocrypt and the summation generator. We conclude in Sect. 10.

## 2 Preliminaries

### 2.1 Boolean Functions and Related Inputs

In this section we repeat some definitions and known facts about Boolean functions. Additionally, we provide some theorems about Boolean functions and related inputs. All calculations are done over the finite field GF(2).

**Definition 1** *For* $\alpha = (\alpha_1, \ldots, \alpha_n)$ *and* $x = (x_1, \ldots, x_n) \in \{0,1\}^n$*, we define* $m_\alpha(x) := \prod_i x_i^{\alpha_i}$ *and the degree* $\deg m_\alpha := |\alpha| := \#\{i|\alpha_i = 1\}$.

**Theorem 2** *(Algebraic Normal Form) Let* $f : \{0,1\}^n \to \{0,1\}$ *be a Boolean function. Then,* $f(x)$ *can be written as* $f(x) = \bigoplus_{\alpha \in \{0,1\}^n} c_\alpha \cdot m_\alpha(x)$ *for unique coefficients* $c_\alpha \in \{0,1\}$*. Hence, the value* $\max\{\deg m_\alpha | c_\alpha \neq 0\}$ *is unique and is called the degree* $\deg f$ *of* $f$.

**Definition 3** *For* $\alpha = (\alpha_1, \ldots, \alpha_n), \alpha' = (\alpha'_1, \ldots, \alpha'_n) \in \{0,1\}^n$ *we say that* $\alpha' \leq \alpha$ *if* $\forall i :\ \alpha'_i \leq \alpha_i$ *(treated as integers). Consequently, we say that* $\alpha' < \alpha$ *if* $\alpha' \leq \alpha$ *but* $\alpha' \neq \alpha$*. For* $\alpha' \leq \alpha$ *we define* $\alpha - \alpha' := (\alpha_1 - \alpha'_1, \ldots, \alpha_n - \alpha'_n)$.

Obviously, $\alpha' \leq \alpha$ (resp. $\alpha' < \alpha$) implies $\deg m_{\alpha'} \leq \deg m_\alpha$ (resp. $\deg m_{\alpha'} < \deg m_\alpha$).

**Lemma 4** *Let $\alpha, \delta^{(1)}, \delta^{(2)} \in \{0,1\}^n$ be arbitrary. For $i = 1, 2$ it holds that $m_\alpha(x \oplus \delta^{(i)}) = \bigoplus_{\alpha' \leq \alpha} m_{\alpha'}(x) m_{\alpha-\alpha'}(\delta^{(i)})$ and $m_\alpha(x \oplus \delta^{(1)}) \oplus m_\alpha(x \oplus \delta^{(2)})$ has a degree $\leq \deg m_\alpha - 1$.*

*Proof.* The first equation is obvious. The second one is because of:
$m_\alpha(x + \delta^{(1)}) + m_\alpha(x + \delta^{(2)})$
$= \sum_{\alpha' \leq \alpha} \left( m_{\alpha'}(x) m_{\alpha-\alpha'}(\delta^{(1)}) + m_{\alpha'}(x) m_{\alpha-\alpha'}(\delta^{(2)}) \right)$
$= \underbrace{m_\alpha(x) + m_\alpha(x)}_{=0} + \sum_{\alpha' < \alpha} \left( m_{\alpha'}(x) m_{\alpha-\alpha'}(\delta^{(1)}) + m_{\alpha'}(x) m_{\alpha-\alpha'}(\delta^{(2)}) \right).$ □

**Theorem 5** *Let $f$ be a Boolean function with $\deg f = d$. Then, $f(x \oplus \delta^{(1)}) \oplus f(x \oplus \delta^{(2)})$ has a degree $\leq d - 1$.*

*Proof.* By theorem 2 we can write $f(x)$ as $\sum_{\alpha, |\alpha| \leq d} c_\alpha m_\alpha(x)$. Then, by lemma 4 it is $f(x \oplus \delta^{(1)}) \oplus f(x \oplus \delta^{(2)}) = \bigoplus_{\alpha, |\alpha| \leq d} c_\alpha \underbrace{\left( m_\alpha(x \oplus \delta^{(1)}) \oplus m_\alpha(x \oplus \delta^{(2)}) \right)}_{\deg \leq d-1}$. □

The following corollary is obvious:

**Corollary 6** *For any even number $m$ and any vectors $\delta^{(1)}, \ldots, \delta^{(m)}$, the degree of the function $\bigoplus_{i=1}^m f(x \oplus \delta^{(i)})$ is $\leq \deg f - 1$.*

Theorem 7 is a special case of theorem 5 and will be of use in this paper:

**Theorem 7** *Let $f$ be a Boolean function with $\deg f = d$. Let $\boldsymbol{e}_i \in \{0,1\}^n$ be the unit vector with its only $1$ in position $i$. Then the function $f_1(x') = f(x) \oplus f(x \oplus \boldsymbol{e}_i)$ has degree $\leq d - 1$, where $x' = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \{0,1\}^{n-1}$.*

*Proof.* We first split the function $f(x)$ into two parts, the first part consisting of the monomials containing $x_i$, and the second part consisting of the monomials not containing $x_i$ as a factor:

$$f(x) = x_i \cdot f_1(x') \oplus f_2(x'), \tag{1}$$

where it is straightforward to see that $\deg f_1 \leq d - 1$ and $\deg f_2 \leq d$. We do the same for the function $f(x \oplus \boldsymbol{e}_i)$:

$$f(x \oplus \boldsymbol{e}_i) = (x_i \oplus 1) \cdot f_1(x') \oplus f_2(x'). \tag{2}$$

Taking the XOR of the equations (1) and (2) and eliminating terms occurring twice yields: $f(x) \oplus f(x \oplus \boldsymbol{e}_i) = f_1(x')$. □

## 2.2 General Framework for Synchronous Stream Ciphers

We consider a synchronous stream cipher with $n$-bit state $S$ updated by a linear function represented by a matrix $L$ (*e.g.*, one or more LFSRs) over $\mathbb{Z}_2$, and with a nonlinear output function $f$ that takes $\varphi$ input bits coming from $S$ to produce one output bit $z_t$. Some designs (*e.g.*, the combiners with memory) also include a $m$-bit memory $M$ that has a nonlinear update function $h$. This results in the following general framework for a synchronous stream cipher:

$$\begin{cases} z_t = f(S_t, M_t) \\ c_t = p_t \oplus z_t \\ S_{t+1} = L \cdot S_t \\ M_{t+1} = h(S_t, M_t)\,, \end{cases} \tag{3}$$

where $p_t$, $z_t$ and $c_t$ are respectively the plaintext, the key stream and the ciphertext at time $t = 0, 1, 2 \ldots$

The initial state $(S_0, M_0)$ is determined by the resynchronization mechanism, which combines a $k$-bit secret key $K$ and a $\iota$-bit known initialization vector $IV^i$ with an initialization function $f_{init}$:

$$(S_0, M_0) = f_{init}(K, IV^i)\,. \tag{4}$$

## 2.3 Algebraic Attacks

In this section we repeat some facts about algebraic attacks against LFSR-based key stream generators. We describe the general attack on combiners with memory introduced in [2] as this includes the special case of memoryless combiners [7].

An algebraic attack works as follows: first find a Boolean function $F \not\equiv 0$ (called an ad-hoc equation) such that for all $t$ it holds

$$0 = F(L^t \cdot K, \ldots, L^{t+r-1} \cdot K, z_t, \ldots, z_{t+r-1})\,. \tag{5}$$

Such a function $F$ can be found with the algorithm of [2] if $\varphi \cdot r$ is not too large.

Secondly, recover the secret key $K$ by solving this system of equations. For this purpose, several methods (Linearization, XL, XSL, Groebner bases, ...) exist. Amongst them only the linearization method allows a general estimation of the work effort. We give now a description of the linearization method. Due to the linearity of $L$, all equations (5) have degree $\leq d := \deg F$. Therefore, the number $\mathcal{M}$ of different monomials occurring is limited by:

$$\mathcal{M} \leq \binom{k}{0} + \ldots + \binom{k}{d} =: \beta(k, d) \in \mathrm{O}(k^d). \tag{6}$$

By replacing each monomial by a new variable, the attacker gets a linear system of equations in $\mathcal{M}$ unknowns. It can be solved by Gaussian elimination or more refined methods like the one by Strassen [26]. As $\beta(k, d) \in O(k^d)$, the lower the degree $d$, the faster the attack.

## 2.4 Resynchronization Attacks

For a synchronous stream cipher, perfect synchronization between sender and receiver is required. The aim of the resynchronization mechanism is to achieve this in a secure fashion.

A first solution is *fixed resync*. In this scenario, the message is divided into frames, and each frame $i$ is encrypted with a unique $IV^i$, a frame counter updated in a deterministic way. An attack under this scenario is called a *known* $IV$ resynchronization attack. The frequency at which resynchronization should occur depends on the risk of synchronization loss. Examples of stream ciphers that use fixed resync are the $E0$ algorithm used in the Bluetooth [5] wireless communication standard, and A5 [1] used in GSM cellular voice and data communication.

A second scenario is that the receiver sends a resynchronization request to the sender as soon as synchronization loss is detected. This is called *requested resync*. In this scenario, the receiver may be allowed to choose the nonce $IV^i$ used in the frame. This may enable a *chosen* $IV$ resynchronization attack, as described *e.g.* by Joux and Muller [15]. Security under the chosen $IV$ attack scenario implies security under the known $IV$ attack scenario. Hence, a good resynchronization mechanism should be resistant against a chosen $IV$ attack.

Table 1 shows the general setting for a resync attack as used throughout this paper. For $R$ frames, the attacker knows the first $T$ key stream bits.[3] He also knows the initial value $IV^i$ of each frame.

| Frame | Clock 0 | | ... | Clock $T-1$ | |
|---|---|---|---|---|---|
| | State | Output | | State | Output |
| 0 | $(S_0^0, M_0^0) = f_{init}(K, IV^0)$ | $z_0^0 = f(S_0^0, M_0^0)$ | ... | $(S_{T-1}^0, M_{T-1}^0)$ | $z_{T-1}^0 = f(S_{T-1}^0, M_{T-1}^0)$ |
| 1 | $(S_0^1, M_0^1) = f_{init}(K, IV^1)$ | $z_0^1 = f(S_0^1, M_0^1)$ | ... | $(S_{T-1}^1, M_{T-1}^1)$ | $z_{T-1}^1 = f(S_{T-1}^1, M_{T-1}^1)$ |
| $\vdots$ | $\vdots$ | $\vdots$ | ... | $\vdots$ | $\vdots$ |
| $R$-1 | $(S_0^{R-1}, M_0^{R-1}) = f_{init}(K, IV^{R-1})$ | $z_0^{R-1} = f(S_0^{R-1}, M_0^{R-1})$ | ... | $(S_{T-1}^{R-1}, M_{T-1}^{R-1}0)$ | $z_{T-1}^{R-1} = f(S_{T-1}^{R-1}, M_{T-1}^{R-1})$ |

**Table 1.** The situation of a frequently reinitialized key stream generator

We now describe a first resynchronization attack by Daemen *et al.* [9] on a simplified version of this framework and point out its limitations.

[3] For most attacks described in this paper, the observed key stream bits need not be successive. Where this is really a requirement, we will specify this.

## 3 The Daemen *et al.* Resynchronization Attack

### 3.1 Description

The resynchronization attack of Daemen *et al.* is a known plaintext attack for the special case of a simple memoryless combiner with a linear resynchronization mechanism.

The framework of the attack can be described as follows:

$$\begin{cases} S_0^i = A \cdot K \oplus B \cdot IV^i \\ z_t^i = f(\Pi \cdot S_t^i) \\ S_{t+1}^i = L \cdot S_t^i \, . \end{cases} \tag{7}$$

In these equations, $A$, $B$, $L$ and $\Pi$ are known binary matrices, $A \in \mathbb{Z}_2^{n\times k}$, $B \in \mathbb{Z}_2^{n\times \iota}$, $L \in \mathbb{Z}_2^{n\times n}$ and $\Pi \in \mathbb{Z}_2^{\varphi\times n}$. The matrices $A$ and $B$ represent the fact that the resync mechanism is linear, the projection matrix $\Pi$ shows that the output function $f$ uses only a subset of $\varphi$ bits of $S_t^i$. The initialization vector of the $i$th frame is $IV^i$, and $z_t^i$ is the key stream bit at time $t$ of the $i$th frame.

We introduce the key-dependent unknown values $\textsc{k}_t$ and the known values $\textsc{iv}_t^i$ as follows:

$$\begin{cases} \textsc{k}_t = \Pi \cdot L^t \cdot A \cdot K \\ \textsc{iv}_t^i = \Pi \cdot L^t \cdot B \cdot IV^i \, . \end{cases} \tag{8}$$

Using the setting from Table 1, the attacker can then rewrite this system into a system of equations built as follows:

$$z_t^i = f(\textsc{k}_t \oplus \textsc{iv}_t^i) \quad \text{for } 0 \le i \le R-1,\ 0 \le t \le T-1. \tag{9}$$

Now we try to find a solution of this system of equations for each $t$. Assume without loss of generality that $t = 0$. If $\varphi$ is not too large, we can perform an exhaustive search over $\textsc{k}_0$, and check whether the guess satisfies the $R$ equations for $t = 0$. If $R \ge \varphi$, it is expected that a unique solution for $\textsc{k}_0$ exists. Hence the correct value of $\textsc{k}_0$ has been found, and thus $\varphi$ linear equations in the bits of the secret key. This is repeated for $t = 1, 2, ..., p-1$, such that $p = \lceil k/\varphi \rceil$, until the entire secret key is deduced.

The complete attack requires on average $\lceil k/\varphi \rceil \cdot 2^{\varphi}$ evaluations of the function $f$, at least $\varphi$ resyncs and about $k$ bits of key stream in total ($\varphi$ frames of length $\lceil k/\varphi \rceil$). Note that the computational complexity of the attack increases exponentially with the number $\varphi$ of inputs of the Boolean function $f$.

### 3.2 Limitations

The Daemen *et al.* attack can be seen as a divide-and-conquer attack. Standard cryptanalytic attacks such as correlation and algebraic attacks work chronologically on a key stream, which corresponds to a row in Table 1. On the contrary, the Daemen *et al.* attack tries to solve the system column by column. One of the motivations of the paper is to combine both approaches. Of special interest is the combination of the resynchronization attack with algebraic attacks.

In many cases, it is not obvious whether the column by column approach by Daemen *et al.* works or not. We have identified the following limitations:

1. The attack does not work in real time.
2. The number of resyncs $R$ has to be at least the number $\varphi$ of input bits of the output function $f$.
3. The complexity is prohibitively large for large values of $\varphi$.
4. The divide-and-conquer approach does not work if the key stream generator uses additional memory.
5. The initialization function $f_{init}$ has to be linear.

In this paper, we will address each of these limitations and show how to overcome them.

## 4 Real-time Attack

The attack of Daemen *et al.* shows that ciphers that use a linear resynchronization mechanism and that have a Boolean function $f$ with few inputs are insecure. This enables a passive attack on such designs. A real-time attack in which the attacker can discover the plaintext immediately (and even modify it in a controlled way) is not possible, because the time required to perform the $p$ exhaustive searches will be too high. Here we show how to easily replace this iterated exhaustive search by a precomputation step. This enables real-time active attacks on such ciphers.

We start from the realistic assumption that the $IV^i$s are chosen in a deterministic way, for instance by a counter or a fixed update mechanism. In the precomputation step, we first calculate the values $\mathrm{IV}_t^i$ for $0 \leq i \leq \varphi - 1$ and $0 \leq t \leq T - 1$. Then we calculate the following $\varphi$ bits, and repeat this for all values of $\mathrm{G}_t$ (a guess for $\mathrm{K}_t$) going from 0 to $2^\varphi - 1$, and this for all $t$.

$$\begin{cases} f(\mathrm{G}_t \oplus \mathrm{IV}_t^0) = b^0_{\mathrm{G}_t,t} \\ f(\mathrm{G}_t \oplus \mathrm{IV}_t^1) = b^1_{\mathrm{G}_t,t} \\ \dots \\ f(\mathrm{G}_t \oplus \mathrm{IV}_t^{\varphi-1}) = b^{\varphi-1}_{\mathrm{G}_t,t} \,. \end{cases} \tag{10}$$

For each time $t$, we obtain $2^\varphi$ sequences $b^0_{\mathrm{G}_t,t} b^1_{\mathrm{G}_t,t} \dots b^{\varphi-1}_{\mathrm{G}_t,t}$. Because the length of these sequences is $\varphi$, every value of $\mathrm{G}_t$ is expected to correspond with a unique sequence $b^0_{\mathrm{G}_t,t} b^1_{\mathrm{G}_t,t} \dots b^{\varphi-1}_{\mathrm{G}_t,t}$. We then sort the $\mathrm{G}_t$ values based on the numerical value of the corresponding sequence, and store this in memory.

The attack now goes as follows. We group the outputs observed at say $t = 0$ in a sequence $z_0^0 z_0^1 \dots z_0^{\varphi-1}$. We jump to this position in our table built for $t = 0$, and the value found there is the correct value of $\mathrm{K}_0$. We do the same for the times $t = 1, 2, \dots p - 1$, and we have then found the necessary $\mathrm{K}_i$ to directly determine the secret key $K$.

The total complexity of the precomputation step is about $k \cdot 2^\varphi$ evaluations of $f$ (but this can of course be replaced by $2^\varphi$ evaluations of $f$ and $k \cdot 2^\varphi$ table look-ups). The memory requirement is about $k \cdot 2^\varphi$ bits, which is feasible for many stream ciphers (*e.g.*, for a secret key of $k = 256$ bits, and a Boolean function with $\varphi = 20$ inputs, 32 Mbyte is required).

## 5 Attack with a Small Number of Resynchronizations

In [9], the authors made the assumption that the number of solutions converges to 1 if $R \gtrsim \varphi$. Actually, the number of required resyncs depends on the cipher and the observed public parameters $IV^i$. In [14], Golic and Morgari discussed the number of $IV$s that are needed for the Daemen *et al.* attack to work. They showed that with a non-negligible probability more than $\varphi$ known $IV$s are necessary. This results in an increased attack complexity, both for the original attack and for the precomputation attack.

We will here follow different approaches. We want the attacks to work in any case and with a minimal number of known $IV$s. Simulations on various Boolean functions have confirmed that the standard resynchronization attack does not always work in practice with $\varphi$ $IV$s. This is due to two reasons, the first being imperfect behavior of the function $f$. However, this effect is not very important because in most stream ciphers the function $f$ has good statistical properties, which typically include balancedness and high nonlinearity. A second reason is that sometimes collisions occur between two values $\text{IV}_t^a$ and $\text{IV}_t^b$ where $a \neq b$. We will show several ways to overcome this problem.

### 5.1 Computational Approach

**Two-phase attack.** We implement the algorithm in two steps. The first step, the resynchronization attack, retains a set of values for each of the $\text{K}_0, \text{K}_1 \ldots \text{K}_{p-1}$. In a second step, we then search through all possible combinations until we have found the correct secret key.

Simulations have shown that for $\varphi$ (or more) known $IV$s, the time complexity of the second step is negligible. In other words, the resynchronization attack (extended with the fast search step) is always successful under realistic assumptions with $\varphi$ known $IV$s.

Using this two-step algorithm, one can also mount a resynchronization attack with $R < \varphi$ known $IV$s. The time complexity of the second step can then be shown to be about $2^{(\varphi-R)\cdot\frac{k}{\varphi}}$. Even if this complexity increases exponentially with decreasing $R$, this shows that a resynchronization attack is still feasible for $R$ smaller than (but close to) $\varphi$.

**Overlapping bits.** There is also another interesting way to perform a resynchronization attack when $R < \varphi$. Let's take the case $R = \varphi - 1$. For $\text{K}_t$, we will get two possibilities after the exhaustive search. But looking at the bits of these two possibilities $\text{K}_{t,1}$ and $\text{K}_{t,2}$, about half of these will be equal, and will therefore certainly be the correct values for these bits of $\text{K}_t$. This implies that we have still found $\varphi/2$ linear equations in $K$, and we will just need frames that are twice as long as in the standard attack, *i.e.*, have length $T \geq 2k/\varphi$ each. This is still very realistic in most cases. We can develop a similar reasoning for smaller values of $R$, but the length of the frames and the complexity required increases rapidly: they can be shown to be $2^{2^{\varphi-R}-1} \cdot k/\varphi$ and $2^{2^{\varphi-R}-1} \cdot k/\varphi \cdot 2^{\varphi}$ respectively (see App. A).

In Table 2, we summarize the complexities of the attacks. We also give a realistic example for a secret key of $k = 300$ bits and $\varphi = 30$. This clearly shows that limiting the number of allowed resyncs per secret key to less than $\varphi$ will not prevent a resynchronization attack. We will also give an example of this later on when we discuss the attack on Bluetooth.

**Table 2.** The complexities of the practical resynchronization attacks

| Attack | Resyncs needed | Length of each frame | Total complexity |
|---|---|---|---|
| Daemen *et al.* | $\varphi$ | $k/\varphi$ | $/\varphi \cdot 2^{\varphi}$ |
| | 30 | 10 bit | $2^{33}$ |
| Two-phase attack | $R(<\varphi)$ | $k/\varphi$ | $k/\varphi \cdot 2^{\varphi} + 2^{(\varphi-R)\cdot k/\varphi}$ |
| | 29 | 10 bit | $2^{33}$ |
| | 28 | 10 bit | $2^{33}$ |
| | 27 | 10 bit | $2^{33}$ |
| | 26 | 10 bit | $2^{40}$ |
| Overlapping bits | $R(<\varphi)$ | $2^{2^{\varphi-R}-1} \cdot k/\varphi$ | $2^{2^{\varphi-R}-1} \cdot k/\varphi \cdot 2^{\varphi}$ |
| | 29 | 20 bit | $2^{34}$ |
| | 28 | 80 bit | $2^{36}$ |
| | 27 | 1280 bit | $2^{40}$ |
| | 26 | 328 kb | $2^{48}$ |

## 5.2 Using Algebraic Attacks

As we have pointed out, Table 1 implies the following system of equations:

$$z_t^i = f(\mathrm{K}_t \oplus \mathrm{IV}_t^i), \quad 0 \le i \le R-1,\ 0 \le t \le T-1. \tag{11}$$

Hence, another possibility is to try to solve it as a whole instead of "column-by-column." The linearization method described in 2.3 requires that the number of linearly independent equations exceeds the number of occurring monomials $\mathcal{M}$. This requires $T \cdot R \ge \mathcal{M}$. As $\mathcal{M}$ is upper bounded by $\beta(k, d') \in O(k^{d'})$ with $d' = \deg f$, the lower the degree of the equations the faster the attack. In the literature [7, 8, 18, 4], several conditions and methods are described for transforming (11) into a new system of equations

$$g(\mathrm{K}_t \oplus \mathrm{IV}_t^i, z_t^i) = 0, \quad 0 \le i \le R-1,\ 0 \le t \le T-1. \tag{12}$$

with $d := \deg g < \deg f$. Next, we will show how to use the resync setting to decrease the degree of (12) further.

**The Degree-*d*-1 attack.** The first approach is to construct new equations of degree $\leq d-1$. We express $g$ by

$$g(\mathrm{K}_t \oplus \mathrm{IV}_t^i, z_t^i) = \bigoplus_j g_j(\mathrm{K}_t \oplus \mathrm{IV}_t^i) \cdot \tilde{g}_j(z_t^i)\,. \tag{13}$$

Observe that the functions $g_j$ and $\tilde{g}_j$ depend only on $g$ and are all known to the attacker. The idea is to find appropriate linear combinations of (13) to reduce the degree. Let $I := \{j \mid \deg g_j = d\}$ and rewrite (13) to

$$g = \underbrace{\bigoplus_{j \in I} g_j \cdot \tilde{g}_j}_{\deg g_j \;=\; d} \oplus \underbrace{\bigoplus_{j \notin I} g_j \cdot \tilde{g}_j}_{\deg g_j < d}\,. \tag{14}$$

Theorem 8 provides a method for decreasing the degree at least by 1:

**Theorem 8** *Let $g$ be expressed as described in (13). For* any *known* $\mathrm{IV}_t^0, \ldots, \mathrm{IV}_t^{|I|}$ *and corresponding known outputs $z_t^i$, coefficients $c_0, \ldots, c_{|I|} \in \{0,1\}$ with at least one $c_i \neq 0$ can be computed such that the degree of $\bigoplus_{i=0}^{|I|} c_i \cdot g(\mathrm{K}_t \oplus \mathrm{IV}_t^i, z_t^{(i)})$ is $\leq \deg g - 1$.*

*Proof.* We set $g_j^i := g_j(\mathrm{K}_t \oplus \mathrm{IV}_t^i)$ and $\tilde{g}_j^i := \tilde{g}_j(z_t^i) \in \{0,1\}$. With (13), we can write

$$\bigoplus_{i=0}^{|I|} c_i \cdot g(\mathrm{K}_t \oplus \mathrm{IV}_t^i, z_t^i) = \bigoplus_{j \in I} \bigoplus_{i=0}^{|I|} c_i \cdot \tilde{g}_j^i \cdot g_j^i \;\oplus\; \bigoplus_{j \notin I} \bigoplus_{i=0}^{|I|} c_i \cdot \tilde{g}_j^i \cdot g_j^i$$

The second part of the right hand side has a degree $\leq d-1$ by definition of $I$. The idea is to find coefficients $c_0, \ldots, c_{|I|} \in \{0,1\}$ such that the first part of the right hand side has degree $\leq d-1$ too. By Corollary 6, it is sufficient that $\sum_{i=0}^{|I|} c_i \cdot \tilde{g}_j^i$ (treated as an integer) is an even number for all $j \in I$.

We show now that it is always possible. For each $i$ we define the vector $\overrightarrow{V_i} := \left(\tilde{g}_1^{(i)}, \ldots, \tilde{g}_{|I|}^{(i)}\right) \in \{0,1\}^{|I|}$. Then the assumption above is equivalent to $\bigoplus_i c_i \cdot \overrightarrow{V_i} = \overrightarrow{0}$. By the theory of linear algebra, the $|I|+1$ vectors of the $|I|$-dimensional vector space $\{0,1\}^{|I|}$ are linearly dependent. Therefore, such coefficients $c_i$ exist. □

Let $\mathcal{M}_e$ be the number of monomials of degree $\leq e$ occurring in (12). The attack complexity is as follows. First we have to calculate (for a fixed clock $t$) the coefficients $c_i$. This requires $O(|I|^3)$ operations. Then, the computation of the function of degree $\leq d-1$ is equivalent to the summation of (several) vectors of size $\mathcal{M}_{d-1}$. The two steps have to be repeated about $\mathcal{M}_{d-1}$ times to get enough linearly independent equations of degree $\leq d-1$. The final step is to use Gaussian elimination to solve the linearized system of equations $\approx (\mathcal{M}_{d-1})^3$. Therefore the overall number of operations is about

$$\left(|I|^3 + M_{d-1}\right) \cdot M_{d-1} + (M_{d-1})^3. \tag{15}$$

Because of $\mathcal{M}_e \leq \beta(k,e)$, an upper bound is

$$\left(|I|^3 + \beta(k,d-1)\right) \cdot \beta(k,d-1) + \beta(k,d-1)^3. \tag{16}$$

Note that it may happen that $\bigoplus_i c_i \cdot g(\textsc{k}_t \oplus \textsc{iv}_t^i, z_t^i)$ is equal to zero for some $t$.[4]

As opposed to fast algebraic attacks [8, 3], this approach does not require the highest-degree monomials to be independent of the key stream bits. Moreover, the number of key stream bits required is $\leq \beta(k,d-1) + |I|$ instead of $\leq \beta(k,d)$. On the other hand, fast algebraic attacks benefit from the fact that the most time consuming part can be sourced out in a precomputation step. This is not possible here. Another advantage is that its applicability is independent of the values of $\textsc{iv}_t^i$ and $z_t^i$ and that it does not require $\varphi$ to be low.

**The Degree-$e$ attack.** So far, we concentrated only on decreasing the degree by 1. But clever combinations may reduce the degree even further. In the worst case these combinations may be linear, even if the degree of $g$ is high. This is for example the case for the $E_0$ key stream generator.[5] We develop now the theory how to compute the lowest possible degree. In the following, we treat $\textsc{k}$ as $\varphi$ unknowns.

**Definition 9** *We set $S(g) := \{g(\textsc{k} \oplus \textsc{iv}, z) \mid \textsc{iv} \in \{0,1\}^\varphi, z \in \{0,1\}\}$ and define by $< g >:=< S(g) >$ the linear span of $S(g)$ (i.e., all possible linear combinations). $<g>$ is a vector space over the finite field GF(2). By $\dim g$ we define the dimension of $<g>$ and by $B(g)$ an arbitrary basis of $<g>$. Further on we set $\mathcal{M}_d(g)$ to be all monomials of degree $\leq d$ which occur in $S(g)$.*

From the theory of linear algebra, the following theorem is obvious:

**Theorem 10** *A function of degree $\leq e$ exists in $< g >$ only if the vectors in $B(g) \cup \mathcal{M}_e(g)$ are linearly dependent.*

Let $\mathcal{S}$ be a set of Boolean functions. We now describe an algorithm to compute a linearly independent set of functions $< \mathcal{S} >$ with the lowest possible degree $e$: We treat the functions in $\mathcal{S}$ like rows of a matrix where each column reflects one occuring monomial. Then, we apply Gaussian elimination in such way that the monomials with the highest degree are eliminated first and so on. Finally, we just pick those functions in the result with the lowest degree.

If $\mathcal{S} = B(g)$, the algorithm computes the lowest possible value for $e$. Let $\tilde{B} := \{g(\textsc{k}_t \oplus \textsc{iv}_t^i, z_t^i) \mid 0 \leq i \leq R-1\}$ be the set of functions available to the attacker. $< \tilde{B} >$ might be only a subset of $< g >$. In this case, the lowest possible degree can be higher.

We try now to estimate the complexity of the Degree-$e$ attack. The first step is to find an appropriate linear combination in $\tilde{B}$. The effort is about $(\dim g)^2 \cdot |\mathcal{M}_d(g)|$ to find the linear combination and about $\mathcal{M}_e$ to compute

[4] For example, this can not be avoided if $g$ is linear. But in this case, the cipher is weak anyhow.

[5] The best before was a system of equations of degree 3 (see [8]).

the corresponding vector of size $\mathcal{M}_e$. This has to be repeated at least $\mathcal{M}_e$ times. Finally, a system of equations in $\mathcal{M}_e$ has to be solved ($\approx \mathcal{M}_e^3$). Hence, the overall number of operations is about

$$((\dim g)^2 \cdot |\mathcal{M}_d(g)| + \mathcal{M}_e) \cdot \mathcal{M}_e + \mathcal{M}_e^3. \tag{17}$$

Because of $\mathcal{M}_e \leq \beta(k,e)$ and $\dim g \leq \mathcal{M}_d \leq \beta(k,d)$, the following expression is an upper bound for the complexity

$$(\beta(\varphi,d)^3 + \beta(k,d)) \cdot \beta(k,e) + \beta(k,e)^3 \tag{18}$$

In the individual case, the applicability of this attack depends on many parameters: the function $g$, the number $R$ of accessible frames and the corresponding values of $\text{IV}^i$ and $z_t^i$. Hence the attack does not work in every case. On the other hand, it puts on the designer the responsibility of making sure that these attacks are not feasible.

Moreover, if the set $\tilde{B}$ is a basis of $< g >$, then an equation of the lowest possible degree can be constructed. What is the probability that this happens? Let $s := \dim g$ and $R \geq s$. If we assume that each expression $g(\text{K}_t \oplus \text{IV}_t^i, z_t^i)$ is a random vector in $\{0,1\}^s$ then by [29], the probability that $\tilde{B}$ is a basis of $< g >$ is

$$\text{Prob} = \prod_{i=R-s+1}^{m} \left(1 - \frac{1}{2^i}\right) \tag{19}$$

## 6 Resynchronization Attacks with Large $\varphi$

The Daemen *et al.* attack only works when the number $\varphi$ of inputs to the Boolean function is not too large. However, we will show in this section that using a linear resynchronization mechanism will inevitably induce weaknesses into stream ciphers, even when $\varphi$ is very large. We will show a chosen *IV* attack, a known *IV* attack and an algebraic attack.

### 6.1 A Chosen *IV* Attack

The standard attack has a large time complexity of $\lceil n/\varphi \rceil \cdot 2^\varphi$ evaluations of the function $f$, but it only requires $\varphi$ resyncs. We will now show that a tradeoff is possible.

Let $\text{K}_t$ in (9) consist of the bits $k_0, k_1, \ldots k_{\varphi-1}$. We make the reasonable assumption that in the chosen *IV* attack, the attacker can control the values of $\text{IV}_t^i$, consisting of the bits $iv_0, iv_1, \ldots iv_{\varphi-1}$. We now start the chosen *IV* attack. We first take a constant $C$. We then perform resyncs with all the values $\text{IV}_t^i = C \oplus i$, where we let $i$ take all values[6] going from 0 ($00\ldots0$) to $2^u - 1$ ($00\ldots011\ldots1$)

[6] The impact of this choice of $i$ is that the last $u$ input bits of $f$ will take all possible values. Of course we can do the same with any combination of $u$ bits by choosing $i$ as needed.

for some $u$. Let's consider the first two values of our resynchronization attack. We denote $k_i \oplus iv_i$ as $x_i$. We know that:

$$\begin{cases} f(x_0, x_1, \dots x_{\varphi-1}) = z_0^0 \\ f(x_0, x_1, \dots x_{\varphi-1} \oplus 1) = z_0^1 \,. \end{cases} \tag{20}$$

By XORing both equations and using Theorem 7 we get:

$$f_1(x_0, x_1, \dots x_{\varphi-2}) = z_0^0 \oplus z_0^1 \,, \tag{21}$$

where the Boolean function $f_1$ has many properties that are desirable for an attacker. The degree of $f_1$ is lower, it has fewer monomials and it depends on less variables than $f$. This makes many attacks much easier.

In our attack, we will apply this method with $2^u$ chosen $IV$s in an iterative way. As an illustration, these are the equations for $u = 2$.

$$\left.\begin{array}{l} \left.\begin{array}{l} f(\dots x_{\varphi-2}, x_{\varphi-1}) = z_0^0 \\ f(\dots x_{\varphi-2}, x_{\varphi-1} \oplus 1) = z_0^1 \end{array}\right\} \Rightarrow \begin{array}{l} f_1(\dots x_{\varphi-2}) = \\ z_0^0 \oplus z_0^1 \end{array} \\ \left.\begin{array}{l} f(\dots x_{\varphi-2} \oplus 1, x_{\varphi-1}) = z_0^2 \\ f(\dots x_{\varphi-2} \oplus 1, x_{\varphi-1} \oplus 1) = z_0^3 \end{array}\right\} \Rightarrow \begin{array}{l} f_1(\dots x_{\varphi-2} \oplus 1) = \\ z_1^2 \oplus z_0^3 \end{array} \end{array}\right\} \Rightarrow \begin{array}{l} f_2(x_0 \dots x_{\varphi-3}) = \\ z_0^0 \oplus z_0^1 \oplus z_0^2 \oplus z_0^3 \end{array} \tag{22}$$

The basic attack requires at every time $2^u \cdot \varphi$ resyncs, in order to obtain $\varphi - u$ equations in the Boolean function $f_u(x_0 \dots x_{\varphi-u-1})$ which can then be used in a normal resynchronization attack.

In practice, however, we note that the number of monomials, variables and the degree of the equation decreases very rapidly, making the attack work with very small complexity. We will give an example using the Boolean function of Toyocrypt in Sect. 9.2.

## 6.2 A Known *IV* Attack

We now describe another attack, which shows that the linear resynchronization mechanisms introduces weaknesses in the fixed resync setting for all Boolean functions.

The principle of the attack is similar to the linear cryptanalysis method, developed by Matsui for attacking block ciphers [17]. First we search for a linear expression for the Boolean function that holds with probability $p \neq 0.5$. We then collect sufficiently many resyncs such that we can determine key bits using a maximum likelihood method. We will now describe this in more detail.

Our starting point is the fact that for any $\varphi$-input nonlinear Boolean function $f$, we can always find a subset $S \subset \{0, 1, \dots \varphi - 1\}$ for which the equation

$$\bigoplus_{i \in S} x_i = f(x_0, \dots x_{\varphi-1}) \tag{23}$$

holds with probability $0.5 + \epsilon$, where $\epsilon \neq 0$. Suppose that the best bias we have found is $\epsilon$ $(0 < \epsilon \leq 0.5)$. For each time $t$, with $R$ known $IV$s, we get the following equations:

$$\begin{cases} \bigoplus_{i \in S} k_i = \bigoplus_{i \in S} iv_i^0 \oplus z_t^1 \\ \vdots \\ \bigoplus_{i \in S} k_i = \bigoplus_{i \in S} iv_i^{I-1} \oplus z_t^{I-1} , \end{cases} \tag{24}$$

each of which holds with probability $0.5 + \epsilon$. We now count for how many of these equations the right hand side is 1 respectively 0. We assume then that the correct right hand side is the value (0 or 1) that occurs most if $\epsilon > 0$, and the value that occurs least if $\epsilon < 0$.

We now have found one linear equation in the state bits that is true with some probability. This probability increases with the value of $R$ and is dependent on the magnitude of $\epsilon$. As in [17], the probability that the equation is correct, given $R$ resyncs and a bias $\epsilon$, is equal to:

$$\int_{-2 \cdot \sqrt{R} \cdot |\epsilon|}^{\infty} \frac{1}{\sqrt{2 \cdot \pi}} \cdot e^{-x^2/2} \cdot dx = 0.5 + 0.5 \cdot \mathit{erf}(\sqrt{2 \cdot R} \cdot |\epsilon|) , \tag{25}$$

where *erf* is the error function. If we want the probability of correctness to approach one, we need the number of resyncs $R$ to be $c \cdot \epsilon^{-2}$ for a small constant value $c$.

The output of a Boolean function is correlated to at least one linear function of the inputs, see Xiao and Massey [28]. The smallest bias $\epsilon$ that can be found[7] is at least $2^{-\frac{\varphi}{2}-1}$. This implies that any linear resynchronization mechanism with a $\varphi$-input Boolean function $f$ can be broken by this resynchronization attack using at most about $2^{\varphi+2}$ known $IV$s.

How to search for the best linear approximation has been well-studied. The Walsh-Hadamard transform can be used to find the best linear approximation, see [23] for a thorough treatment. In the context of correlation attacks, such linear approximations have been studied in the literature, both for memoryless combiners and nonlinear filter generators [23] as for combiners with memory [12, 19].

The biases found in actual Boolean functions used in stream ciphers will be much higher than the lower bounds described above. This is due to several reasons: the functions have to be balanced, they have to be easily implementable, and for combiners they will also have to take into account the trade-off that has to be made between nonlinearity and resilience, see Sarkar and Maitra [25]. It can be expected that most Boolean functions used in practice are vulnerable to this known $IV$ attack on a linear resynchronization mechanism.

[7] This lower bound for $\epsilon$ follows from the universal nonlinearity bound for Boolean functions. Equality applies to the so-called bent functions. Stream ciphers typically do not use bent functions because they are not balanced. The size of the smallest bias to be found in balanced Boolean functions is still an open problem, but some bounds have been presented, see [11] for an overview. For simplicity, we take the bias of bent functions, but the bias for actual functions will be higher and therefore less resyncs will be needed in practice.

### 6.3 An Algebraic Attack

As said in Sect. 5.2, the goal is to find a solution to the equations:

$$g(\mathrm{K}_t \oplus \mathrm{IV}^i_t, z^i_t), \quad 0 \le i \le R-1,\ 0 \le t \le T-1 \tag{26}$$

Again, the approaches to reduce the degree as described in Sect. 5.2 can be also applied here.

If all bits of $\mathrm{K}_t$ are uniquely specified by the equations, use Gröbner bases or the linearization method to solve (26) clock by clock. If the degree of the equations is low (e.g., Toyocrypt), it might be faster and require less $IV$s than the approach described in 6.2.

### 6.4 The Degree-1 Attack

Another approach is to apply the methods described in Sect. 5.2 if $e = 1$ is possible. In this case, we get at least one linear equation in the bits of $\mathrm{K}_t$ directly. If we repeat this for enough values of $t$ the corresponding $K$ can be reconstructed by solving a system of linear equations.

The exact effort depends on many parameters: the function $g$, the number $R$ of frames, the corresponding values of $\mathrm{IV}^i$ and $z^i_t$ and so on (see also Sect. 5.2). Due to (17), the number of operations is about

$$((\dim g)^2 \cdot |\mathcal{M}_d(g)| + k) \cdot k + k^3 \tag{27}$$

or more general

$$(\beta(\varphi, d)^3 + k) \cdot k + k^3 \tag{28}$$

Expression (28) shows that the approach might be feasible if the degree $d$ is small.

## 7 Attacks on Combiners with Memory

Many stream ciphers use their linear state in conjunction with a (small) non-linear memory in order to avoid the trade-off between correlation immunity and nonlinearity for the combining function, see Rueppel [22].

In this section we demonstrate that resynchronization attacks can also be performed on stream ciphers with memory. Note that the known $IV$ attack of Sect. 6 can also be applied on combiners with memory.

### 7.1 A Standard Resynchronization Attack

We will use the following model based on the general case of the combiners with memory:

$$S^i_0 = A \cdot K \oplus B \cdot IV^i \tag{29}$$

$$M^i_0 = \text{const} \tag{30}$$

$$S^i_{t+1} = C \cdot S^i_t \tag{31}$$

$$M^i_{t+1} = h(D \cdot S^i_t, M^i_t) = h(\mathrm{K}_t \oplus \mathrm{IV}^i_t, M^i_t) \tag{32}$$

$$z^i_t = f(D \cdot S^i_t, E \cdot M^i_t) = f(\mathrm{K}_t \oplus \mathrm{IV}^i_t, E \cdot M^i_t)\,. \tag{33}$$

In practice, some designs only start outputting key stream when $t = \mu$; this results in an improved diffusion of the key and the initialization vector into the nonlinear state. We will discuss both the cases $\mu = 0$ and $\mu > 0$. Note that in some designs $M_0^i$ is also dependent on the key and the $IV$. This can in our model be treated in the same way as the case $\mu > 0$.

**$\mu = 0$.** In the case $\mu = 0$, the resynchronization attack can easily be adapted to work also with combiners with memory. We again describe the attack with $\varphi$ resyncs.

The first series of outputs can be written as $z_0^i = f(\text{K}_0 \oplus \text{IV}_0^i, M_0^i)$. Because $M_0^i$ is known, the attacker can recover $\text{K}_0$ by exhaustive search. He can then determine $M_1^i$ for all the resyncs using (32). Now he knows all inputs to (33) for $t = 1$ except $\text{K}_1$, which he can again recover through exhaustive search, and so on. All complexities of this attack are exactly the same as for the case without memory. The only difference is that each step now consists of one evaluation of $f$ and of $\varphi$ evaluations of $h$.

**$\mu > 0$.** The attacker does not know the initial contents of the $m$-bit memory $M$. Moreover, this memory is different for each resync. The attack now works exactly as above, except that the attacker will first have to guess the contents of $M$ at $t = \mu$. The time complexity of the attack now becomes $2^{\varphi \cdot m} \cdot \lceil n/\varphi \rceil \cdot 2^{\varphi}$ evaluations of the $f$ and $h$ function. As $\varphi$ and $m$ are quite small in most actual designs, the attacks are feasible.

Let's take as an example a combiner consisting of 5 LFSRs with total length 320 bits, and with 5 memory bits. The complexity of the resynchronization attack is then equal to $2^{36}$ function evaluations.

**Practical considerations of the attack.** As for the case without memory, we would like that the attack always works with $\varphi$ resyncs. At some times, we will have several possible values for $\text{K}_i$. In a second phase, we cannot use the exhaustive search method of the memoryless case, because we would then have to try all possible values for updating the memory bits, which would increase the complexity enormously.

This problem can be easily overcome by implementing the algorithm with a depth-first search. When at some time $t$ we have several possibilities for $\text{K}_t$, we pick the first one and go to $t + 1$. If we have no solution at time $t$, we go back to $t - 1$ and try the next possibility there. When we have arrived at time $\lceil k/\varphi \rceil$, we have found a sufficient number of values and we check if we have found the correct key. Simulations indicate that when the number of resyncs $R$ is equal to or larger than $\varphi$, the attack will find the correct values very quickly and has to search very few states.

The same approach can also be used when the attacker disposes of less than $\varphi$ resyncs, i.e., $R < \varphi$. In the case $\mu > 0$ this may even be advantageous from a complexity viewpoint, because we have to perform an exhaustive search over less than $\varphi$ initial memory states. But again the complexity of the search algorithm will increase exponentially with decreasing $R$, making this attack feasible only when $R$ is close to $\varphi$.

A particularity is the case when the Boolean output function $f$ is linear. In that case we don't get new information at each new resync, because all equations $z_0^i = f(\text{K}_0 \oplus \text{IV}_0^i, M_0^i)$ are equivalent. This problem can be easily overcome by using the memory update function $h$ to do the checks during the search. An example of such a linear output function is $E0$, which we will describe in Sect. 9.1.

### 7.2 Using Ad-hoc Equations

Another possibility is the use of ad-hoc equations which have been introduced in [2]. The authors showed that for a combiner with memory with $m$ memory bits $M_t$, an equation

$$F(S_t^i, \ldots, S_{t+m}^i, z_t^i, \ldots, z_{t+m}^i) = 0 \tag{34}$$

of degree $\leq \lceil \frac{\varphi \cdot (m+1)}{2} \rceil$ always exists which is completely independent of the memory bits. They also propose an algorithm to find ad-hoc equations

$$G(S_t^i, \ldots, S_{t+r-1}^i, z_t^i, \ldots, z_{t+r-1}^i) = 0 \tag{35}$$

with the lowest possible degree $d$ if $\varphi \cdot r$ is not too large. For example, an ad-hoc equation of degree 4 using $r = 4$ successive clocks exists for the $E_0$ key stream generator.

As (35) is independent of the memory bits, these equations can be used for *all* attacks described in the previous sections. An additional requirement is now that the attacker knows enough successive key stream bits.

If $\varphi \cdot r$ is small the Daemen *et al.* attack is applicable. The number of operations is about

$$\lceil k/(\varphi \cdot r) \rceil \cdot 2^{\varphi \cdot r} + k^3 \tag{36}$$

The methods described in Sect. 5.2 to reduce the degree of the equations can be easily adapted to the case of ad-hoc equations.

## 8 Attacks on Stream Ciphers with Nonlinear Resynchronization

In this section, we will show that stream ciphers with a nonlinear resynchronization mechanism can also be vulnerable to resynchronization attacks. A first attack is a chosen $IV$ attack; its principle is similar to that of Daemen *et al.* The second attack is a known $IV$ attack that uses the principle of linear approximations as used in the known $IV$ attack of Sect. 6.2. We will demonstrate these attacks on the two-level memoryless combiner. The framework of the attack is shown in Fig. 1. In a first level, the key and an $IV$ are linearly loaded into the LFSRs. The input to $f$ at time $t$ of the level 1 initialization is denoted $\text{X}_t$. The following holds:

$$\text{X}_t = (x_t^0, x_t^1, \ldots, x_t^{\varphi-1}) = \text{K}_t \oplus \text{IV}_t = (k_t^0 \oplus iv_t^0, k_t^1 \oplus iv_t^1, \ldots, k_t^{\varphi-1} \oplus iv_t^{\varphi-1}) \,. \tag{37}$$

The output $y_i$ of level 1 is collected and is used as the initial state for the level 2 generator. We take here the simplifying assumption that this is done as shown in the figure. Level 2 generates the key stream $z_i$.

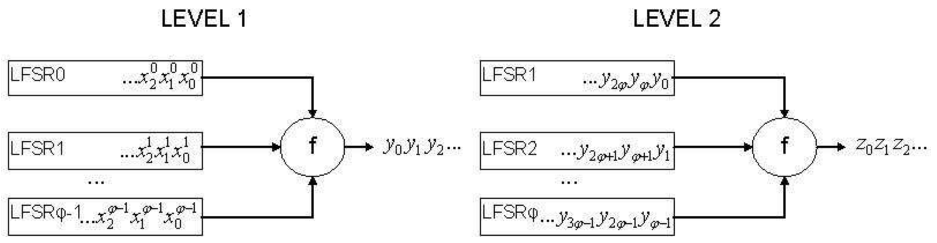
LEVEL 1
LEVEL 2
LFSR0 ...$x_2^0 x_1^0 x_0^0$
LFSR1 ...$x_2^1 x_1^1 x_0^1$
...
LFSRφ-1...$x_2^{\varphi-1} x_1^{\varphi-1} x_0^{\varphi-1}$
f
$y_0 y_1 y_2 \dots$
LFSR1 ...$y_{2\varphi} y_{\varphi} y_0$
LFSR2 ...$y_{2\varphi+1} y_{\varphi+1} y_1$
...
LFSRφ...$y_{3\varphi-1} y_{2\varphi-1} y_{\varphi-1}$
f
$z_0 z_1 z_2 \dots$

**Fig. 1.** Model for a two-level combiner

### 8.1 A Chosen *IV* Attack

We will show an attack scenario on this construction which holds under the assumption that the attacker can choose the value of the IVs which go into the Boolean function (this is for instance the case when the initial state equals the XOR of the key and the initialization vector).

We start with $t = 0$. We let $\text{IV}_0$ take $j$ different values, while keeping the other IVs constant. We obtain the following equations:

$$\begin{cases} f(f(\text{K}_0 \oplus \text{IV}_0^1), y_1, \dots y_{\varphi-1}) = z_0^1 \\ f(f(\text{K}_0 \oplus \text{IV}_0^2), y_1, \dots y_{\varphi-1}) = z_0^2 \\ \dots \\ f(f(\text{K}_0 \oplus \text{IV}_0^j), y_1, \dots y_{\varphi-1}) = z_0^j \,. \end{cases} \tag{38}$$

Denote the vector $(y_1, \dots y_{\varphi-1})$ by R. In half of the cases, we will see that all $z_0^i$ are equal (either all 0 or all 1). This is due to the fact that $f(0, \text{R}) = f(1, \text{R})$. In the other half of the cases, which is of interest here, the $z_0^i$ take both the values 0 and 1 in a random-looking way; which is due to the fact that $f(0, \text{R}) \neq f(1, \text{R})$.

Assume that we are in the latter case. We now guess the $\varphi - 1$ bits of R, let's call this guess G. If $f(0, \text{G}) = f(1, \text{G})$, then our guess is certainly not correct and we proceed to the next guess. If $f(0, \text{G}) \neq f(1, \text{G})$, the guess is possible; we now get $j$ equations in $\text{K}_0$ of the form:

$$f(\text{K}_0 \oplus \text{IV}_0^i) = z_0^i \oplus f(0, \text{G}) \,. \tag{39}$$

The equations we have obtained are exactly the same as in the case of the linear resynchronization mechanism. If $j$ is large enough, we expect to find a unique solution for $\text{K}_0$ over all the guesses for G. It can be shown that we need about $2 \cdot \varphi$ chosen $IV$s to achieve this.

In the same way, we can also recover $\mathrm{K}_1, \mathrm{K}_2, \ldots$ which gives us the whole secret key of the system. The attack requires a total of about $2 \cdot k$ chosen *IV*s and has a time complexity of $\lceil k/\varphi \rceil \cdot 2^{2 \cdot \varphi - 1}$. This attack has been implemented on various 8-bit Boolean functions, and we can easily recover the key.

### 8.2 A Known *IV* Attack

In this attack, we extend the approach in which we search a bias in the $\varphi$-input Boolean function (see the known *IV* attack in Sect. 6) in a straightforward way to the case of the two-level combiners. Each bit now goes twice through the function $f$, but we can show that a bias still persists. The equation we want to hold is as follows:

$$\bigoplus_{j \in BS} k_j = \bigoplus_{j \in BS} iv_j^i \oplus z_0^i \,, \tag{40}$$

where the set $BS$ consists of the bits of the set $S$ involved in the linear bias, for all times $t \in S$ of the first level of the combiner. Similar equations can be written for the next iterations. Let's denote the cardinality of the set $S$ by $s$. Of course it holds that $s \leq \varphi$. The cardinality of the set $BS$ is then evidently $s^2$.

The piling-up lemma [17] learns that the probability that this equation holds is equal to $1/2 + 2^s \cdot \epsilon^{s+1}$, where $\epsilon$ is the bias of the Boolean function. The bias of this equation is thus $2^s \cdot \epsilon^{s+1}$, which means we need $R = 2^{-2 \cdot s} \cdot \epsilon^{-2 \cdot s - 2}$ resyncs to break this system by a known *IV* attack.

We will show what this implies for actual Boolean functions. We take Boolean functions with $\varphi$ inputs and resilience $\rho$. We will use two well-known lower bounds for the bias $\epsilon$:

$$\begin{cases} \epsilon \geq 2^{-\varphi/2-1} \\ \epsilon \geq 2^{\rho+1-\varphi} \,, \end{cases} \tag{41}$$

where the first bound is due to Parseval's relation and the second to the trade-off between nonlinearity and resilience, see [27]. The cardinality of the set $S$ is now[8] $s = \rho + 1$. We can calculate an upper bound for the number of resyncs needed for a successful attack as a function of $\varphi$ and of the resilience $\rho$. This is shown in Fig. 2 for some values of $\varphi$. These graphs show that memoryless combiners with few inputs cannot be made resistant against resynchronization attacks on a two-level combiner. For larger functions it should be checked whether the Boolean function is strong enough to withstand the above attack. The bounds given here may be refined by a more careful examination of the properties of the various classes of Boolean functions.

[8] We conjecture that we will find the bias for $\rho + 1$ in practice. We will certainly find a bias for $\rho + 1$, as $f$ is $\rho$-th order correlation immune but not $\rho + 1$-th order correlation immune. As the cases we discuss are optimal from a designer's point of view, we expect the Walsh spectrum to be flattened as much as possible over the values with Hamming weight $> \rho$ and therefore to find a bias (very close) to $\epsilon$ for Hamming weight $\rho + 1$. This conjecture can be easily verified for a popular class of functions, the plateaued functions [30].

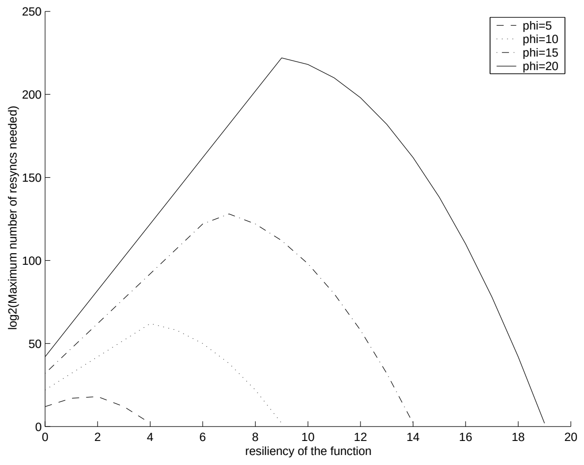

**Fig. 2.** Upper bound for the number of resyncs as a function of the resilience $\rho$ with as parameter the number $\varphi$ of input bits of the Boolean function

## 8.3 Implications of the Attacks

The known $IV$ attack described above for the two-level memoryless combiner can be easily extended to other nonlinear resynchronization mechanisms. It is also possible to apply the attack on other designs, such as combiners with memory, nonlinearly filtered generators and irregularly clocked shift registers. We can use techniques as described by Golic [12, 13] to find suitable linear approximations. Our attack can be used to evaluate the strength of any resynchronization mechanism, and resistance against this attack is a minimum requirement for any design. We are currently investigating the impact of this attack on some actual designs, such as the resynchronization mechanisms of $E0$ and the NESSIE [21] candidates.

## 9 Application to some Key stream Generators

In this section we will describe some actual implementions of the attacks described in the previous sections. We will describe attacks on $E0$, Toyocrypt and the summation generator.

### 9.1 Attacks on $E0$

**Description of $E0$.** The stream cipher $E0$ is used in the Bluetooth wireless standard [5] to encrypt data. The general design of $E0$ is inspired by the summation combiner of Rueppel. It is depicted in Fig. 3. $E0$ consists of 4 LFSRs ($\varphi = 4$), with total length of 128 bits (25 bits for LFSR $A$, 31 for LFSR $B$, 33 for LFSR $C$, 39 for LFSR $D$). $E0$ also contains 4 nonlinear memory bits, the so-called blender. At each iteration the LFSRs are clocked, the new state of the blender is calculated as a nonlinear function of its current state and of the 4 output bits of the LFSRs, and one bit of key stream is calculated as the XOR of the 4 output bits of the LFSRs and of the output bit of the blender.

We want to stress out that our results do not imply an immediate weakness. The reason is that the Bluetooth encryption consists of a two level encryption. Therefore any known differences at the beginning of level 1 do not produce known differences after level 2. On the other hand, the Bluetooth encryption scheme would be in danger if an ad-hoc equation would be known which depends only on the input of level 1 and the output of level 2 (= the key stream). The methods developed in [2] are not practical in this case[9] but other ways to find such an equation may exist. Further on, our attacks show that if a feasible attack on level 2 of $E0$ can be found, it is very easy to recover the actual secret key used in level 1.

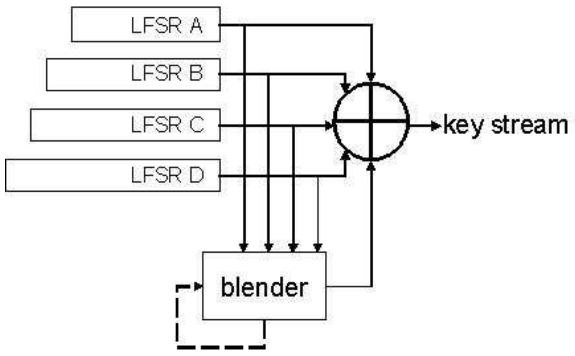

**Fig. 3.** Layout of the stream cipher $E0$

In level 1 the LFSRs are linearly loaded with the key and an initialization vector. The blender is set to zero. Then, $E0$ is clocked 200 times. The first 72 outputs are discarded, the next 128 outputs will be the initial state of the LFSRs in level 2. The initial state of the blender in the second level is set equal to the final state of the blender after the first level.

**Known *IV* attack on level 1 of *E*0.** In this section, we describe a resynchronization attack on level 1 of $E0$ such as described in Sect. 7.1. This attack is similar to the attack of S. Fluhrer described in [10]. We show that this attack can be put into the the general framework of resynchronization attacks and hence gets very efficient once $R \geq \varphi$.

In this attack we assume that we have recovered the state of $E0$ for some instances of level 2. This could have been done by implementing attacks on frames of level 2, *e.g.*, an algebraic attack as described by [3]. We now want to recover the

[9] The reason is that the output of level 1 is permuted before it is used as the input of level 2.

actual secret key which is used in level 1. In our attack, we will use two properties of $E0$. The first is the above mentioned fact that the final state of level 1 of the blender is also the initial state of level 2. This means that we will know the final state of the blender in level 1 in our resynchronization attack. The second is that the blender can be run backwards [24]. Namely, apart from the normal equation $M^i_{t+1} = h(\mathrm{K}_t \oplus \mathrm{IV}^i_t, M^i_t)$, we can also easily determine the function $h'$ such that $M^i_t = h'(\mathrm{K}_t \oplus \mathrm{IV}^i_t, M^i_{t+1})$.

Our attack is now a regular resynchronization attack on combiners with memory and $\mu = 0$, with as only particularity that the attack is executed on the level 1 generator running backwards in time. The attack has been implemented in the depth-first search manner described above. At each iteration we take the next state that fulfills all equations generated by the LFSRs, by $f$ and by $h'$.

This attack has been implemented in C and turns out to be very efficient. Three resyncs already enable us to recover the secret key in several minutes. But it is only when the number of resyncs is equal to 4 (the number of LFSRs) that the attack gets very efficient. This is because the search that has to be done is very limited, as can be predicted by our framework for the resynchronization attack. The attack runs in a few milliseconds on a PC. The search algorithm only needs to go through a couple of thousands guesses for the states. If the number of resyncs is larger than 4, the algorithm is even faster because even more conditions are imposed and even less values need to be searched.

**Actual algebraic attacks.** The secret key $K = (a_0, \ldots, d_{38})$ is the initial state of four LFSRs $A$, $B$ $C$ and $D$. Let $a_t$, $b_t$, $c_t$ and $d_t$ denote their output bits at clock $t$. We define

$$\sigma^1_t := a_t \oplus b_t \oplus c_t \oplus d_t,\ \sigma^2_t := a_t b_t \oplus a_t c_t \oplus a_t d_t \oplus b_t c_t \oplus b_t d_t \oplus c_t d_t,$$
$$\sigma^3_t := a_t b_t c_t \oplus a_t b_t d_t \oplus a_t c_t d_t \oplus b_t c_t d_t,\ \sigma^4_t := a_t b_t c_t d_t$$

Then the ad-hoc equation found in [2] is

$$\begin{aligned} & G(a_{t-1}, \ldots, d_{t-1}, \ldots, a_{t+2}, \ldots, d_{t+2}, z_{t-1}, \ldots, z_{t+2}) \\ = & (z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2}) \oplus \sigma^1_t \cdot (z_{t-1} \oplus z_{t+1} \oplus z_{t+2}) \cdot (1 \oplus z_t) \oplus \\ & \oplus \sigma^2_t \cdot (z_{t-1} \oplus z_t \oplus z_{t+1} \oplus z_{t+2}) \oplus \\ & \oplus \sigma^1_{t+1} \cdot (1 \oplus \sigma^2_t) \cdot (z_{t+1}) \oplus \sigma^1_{t+1} \cdot \sigma^1_t \cdot (1 \oplus z_t) \cdot (z_{t+1}) \oplus \\ & \oplus \sigma^1_t \cdot (\sigma^2_{t+1} \oplus \sigma^1_{t+2} \oplus \sigma^1_{t-1}) \cdot (1 \oplus z_t) \oplus \sigma^4_t \oplus \sigma^2_{t+1} \cdot \sigma^2_t \end{aligned}$$

Obviously, it is $r = d = 4$ and $|I| = 1$ . We calculated that $\dim G = 39$, $|\mathcal{M}_d(G)| = 213$, $e = 1$ (the minimum possible degree) and $\mathcal{M}_1 = n = 128$, $\mathcal{M}_4 \approx 2^{23.07}$ and $\mathcal{M}_3 \approx 2^{18.32}$. Figure 4 shows the estimated efforts for the attacks.

Let $\tilde{B} := \{G(\mathrm{K}_t \oplus \mathrm{IV}^i_t, z^i_t, z^i_{t+1}, z^i_{t+2}, z^i_{t+3}) |\ 0 \leq i \leq R-1\}$ be the set of functions available to the attacker. Using (19), we calculated the probability of $\tilde{B}$ being a basis of $V(G)$, depending on the number $R$ of resyncs:

| $R$ | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 |
|---|---|---|---|---|---|---|---|---|
| Prob | 28.9 | 57.8 | 77 | 88 | 93.9 | 96.9 | 98.4 | 99.2 |

| Attack | Degree-$d$-1 Attack | Degree-1 attack Attack | Daemen *et al.* with ad-hoc eqs. |
|---|---|---|---|
| Equation | (15) | (27) | (36) |
| Effort | $2^{54.25}$ | $2^{25.38}$ | $2^{23.32}$ |

**Fig. 4.** Attacks on the $E_0$ key stream generator

Additionally, we've checked the estimation by repeating the following test 3000 times: For a fixed random value of $\text{K}_0$ we generated randomly $\text{IV}_0^i$ and corresponding expressions $G(\text{K}_0 \oplus \text{IV}_0^i, z_0^i, z_1^i, z_2^i, z_3^i)$, until the number of linearly independent expressions was equal to $\dim G = 39$. It turned out that the probabilities computed with (19) were too optimistic. The following table shows the results of our test:

| $R \leq$ | 39 | 40 | 41 | 42 | 43 | 44 | ... | 49 | 50 | ... | 58 | 59 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | 1.2 | 6.5 | 16 | 28.2 | 41.7 | 54.3 | ... | 89.1 | 91.8 | ... | 98.8 | 99.1 |

In more than 90% of all cases, the number $R$ of frames was between 39 and 50, and in more than 99% between 39 and 59. Hence, we assume that the Degree-$e$ attack and the Degree-1 attack work against $E_0$ with a high probability for about 59 resynchronizations. Additionally, we expect similar effects for other key stream generators.

We did similar tests concerning the Daemen *et al.* attack using ad-hoc equations. I.e., we did 6000 times the following test: We reinitialized the cipher for random IV's until 9 bit of information of $\text{K}_0$ were uniquely specified.[10] It turned out that $R$ was always $\geq 19$. The following table shows the results of our test:

| $R \leq$ | 19 | 20 | 21 | 22 | ... | 32 | 33 | ... | 43 | 44 | ... | 58 | 59 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| % | 0.1 | 0.3 | 1 | 1.9 | ... | 49.2 | 55.3 | ... | 89.5 | 91 | ... | 98.5 | 99.2 |

In more than 90% of all cases, the $R$ was between 19 and 44, and in more than 99% between 19 and 59. As expected, the Daemen *et al.* attack needed less frames than the Degree-1 attack.

We simulated the Degree-1 attack and the Daemen *et al.* attack on the whole $E_0$ keystream generator independently 500 times on a computer. I.e., for a fixed unknown key $K$ we generated new frames with random known differences until the bits $a_1, b_1, c_1, d_1, \ldots, a_{39}, b_{39}, c_{39}, d_{39}$ could be reconstructed.[11] Figure 5 displays the results. It shows how many times a given number of frames was needed. E.g., in 12 resp. 39 cases out of 500, the Degree-1 attack resp. the Daemen *et al.* attack needed 47 frames. In our simulations, the Degree-1 attack needed on average more frames. The same holds for the time effort: Degree-1 attack needed on average 239 s, Daemen *et al.* attack 4 only 91 s.

[10] In fact, due to the form of $G$, only 8 of the $\varphi \cdot r = 16$ bits can be reconstructed directly. The other 8 bits occur only as a linear combination = 1 bit information.

[11] The attack could be improved by using the fact that $a_1, \ldots, a_{25}$ uniquely determine $a_{26}, \ldots, a_{39}$ etc. or by reconstructing only a part of the key and guess the missing bits.

| # Frames | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Degree-1 attack | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 2 | 12 | 36 | 49 | 41 |
| Daemen *et al.* | 1 | 4 | 6 | 8 | 9 | 14 | 13 | 20 | 30 | 21 | 26 | 39 | 29 | 36 | 31 |
| # Frames | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 |
| Degree-1 attack | 55 | 48 | 44 | 41 | 26 | 22 | 18 | 12 | 15 | 10 | 6 | 10 | 9 | 9 | 7 |
| Daemen *et al.* | 23 | 23 | 28 | 14 | 9 | 19 | 9 | 8 | 8 | 9 | 6 | 9 | 3 | 7 | 7 |
| # Frames | 66 | 67 | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 78 | 83 | 98 | |
| Degree-1 attack | 3 | 1 | 5 | 2 | 3 | 2 | 3 | 1 | 0 | 2 | 1 | 3 | 1 | 0 | |
| Daemen *et al.* | 3 | 3 | 4 | 3 | 2 | 5 | 5 | 2 | 2 | 1 | 0 | 0 | 0 | 1 | |

**Fig. 5.** Computer simulations of the Degree-1 attack and the Daemen *et al.* attack using ad-hoc equations on the $E_0$ key stream generator

## 9.2 Attacks on Toyocrypt

**Description of Toyocrypt.** Toyocrypt was submitted to Cryptrec, the Japanese government call for cryptographic primitives, and was accepted to the second phase but has finally been rejected. A description can be found in [20]. At each clock $t$ the key stream bit $z_t$ is produced by

$$\begin{aligned} z_t = f(L^t(K)) = f(k_0^t, \ldots, k_{127}^t) = k_{127}^t \oplus \textstyle\sum_{i=0}^{62} k_i^t k_{\alpha_i}^t \oplus k_{10}^t k_{23}^t k_{32}^t k_{42}^t \oplus \\ \oplus k_1^t k_2^t k_9^t k_{12}^t k_{18}^t k_{20}^t k_{23}^t k_{25}^t k_{26}^t k_{28}^t k_{33}^t k_{41}^t k_{42}^t k_{51}^t k_{53}^t k_{59}^t \oplus \textstyle\prod_{i=0}^{62} k_i^t , \end{aligned} \tag{42}$$

with $\{\alpha_0, \ldots, \alpha_{62}\}$ being some permutation of the set $\{63, \ldots, 125\}$.

**Algebraic attacks.** As noticed in [7], multiplying $f$ with $(1 \oplus k_{23}^t)$ resp. $(1 \oplus k_{42}^t)$ provides two ad-hoc equations $G_1$ and $G_2$ of degree 3.

Due to the large number $\varphi \cdot r$ of key bits used in $G_1$ and $G_2$) ($\varphi = 126$, $r = 1$), the computation of $\dim G_1$ resp. $\dim G_2$ in the naive way is infeasible. Therefore, we concentrate only on the Degree-$d$-1 attack. As $|I| = 1$, $\mathcal{M}_3 \le \beta(128, 3) = 2^{18.42}$ and $\mathcal{M}_2 \le \beta(128, 2) = 2^{13.01}$, the number of operations is about is

$$\left(|I|^3 + \mathcal{M}_3\right) \cdot \mathcal{M}_2 + \mathcal{M}_2^3 \le (1 + 2^{18.42}) \cdot 2^{13.01} + (2^{13.01})^3 \approx 2^{39.04}$$

**Chosen *IV* attack.** We can see, for instance, that only the monomial $x_{24} \cdot x_{63}$ in (42) contains the variable $x_{63}$. We now perform a chosen $IV$ attack as in Sect. 6.1 with two $IV$s that only differ in bit $x_{63}$. Eliminating all terms as in (20) gives us the following simple equation for $t = 0$:

$$x_{24} = z_0^0 \oplus z_0^1 , \tag{43}$$

which is a simple linear equation in the linear state bits. We need just $n$ such equations to determine the entire linear state of the algorithm. We can obtain these easily by performing more resyncs as above for other times, and by doing resyncs at the same time but flipping another bit that also only appears in monomials of degree 2 (there are many of them here).

### 9.3 The Summation Generator with 4 Inputs

The summation generator was proposed by Rueppel [23]. The number $\varphi$ of inputs (i.e., the number of LFSRs) can be chosen arbitrarily. In [16], the authors proved that an ad-hoc equation of degree $\leq 2^{\lceil \log_2 \varphi \rceil}$ always exists if $r = \lceil \log_2 \varphi \rceil + 1$ clocks are regarded. We consider only the case of $\varphi = 4$ inputs and use the ad-hoc function $G$ of degree 4 given in [16]:

$$\begin{aligned} 0 = \sigma_t^4 \oplus (1 \oplus z_t) \cdot \sigma_t^3 \oplus \sigma_t^2 \cdot \sigma_{t+1}^1 \oplus (1 \oplus z_{t+1}) \cdot \sigma_t^2 \oplus \sigma_{t+1}^2 \oplus \sigma_{t+2}^1 \oplus z_{t+2} \\ \oplus (1 \oplus z_t) \cdot \sigma_t^1 \cdot \sigma_{t+1}^1 \oplus (1 \oplus z_t) \cdot (1 \oplus z_{t+1}) \cdot \sigma_t^1 \oplus (1 \oplus z_{t+1}) \cdot \sigma_{t+1}^1 \end{aligned}$$

We calculated that $|I| = 1$, $\dim G = 19$, $|\mathcal{M}_d(G)| = 69$, $e = 1$ and appreciated $\mathcal{M}_4$ resp. $\mathcal{M}_3$ by $\beta(128, 4) \approx 2^{23.32}$ resp. $\beta(128, 3) \approx 2^{18.43}$. Figure 6 displays our estimations for some attacks. To be comparable with the other results in this paper, we have chosen a key size of $n = 128$.

| Attack | Degree-$d$-1 Attack | Degree-$e$- Attack | Daemen *et al.* using ad-hoc eqs. |
|---|---|---|---|
| Equation | (15) | (27) | (36) |
| Effort | $2^{55.25}$ | $2^{22.34}$ | $2^{23.32}$ |

**Fig. 6.** The Degree-$d$-1 Attack, the Degree-$e$ attack and the Daemen *et al.* attackon the summation generator with 4 LFSRs and a key size of 128

## 10 Conclusions

In [9], Daemen, Govaerts and Vandewalle presented the original resynchronization attack on synchronous stream ciphers. In this paper, we have extended this resynchronization attack in several directions, by using new attack methods and by combining the attack with cryptanalytic techniques such as algebraic attacks and linear cryptanalysis.

Our attacks on linear resynchronization mechanisms show that a linear resynchronization mechanism should never be used in practice. Even if the system uses few resyncs, has an input function with many inputs and has a non-linear memory, it will still very likely contain weaknesses that can be exploited by one of our attack scenarios.

Nowadays, resynchronization mechanisms are typically designed in an *ad hoc* manner, by making them nonlinear to an extent that seems to be sufficient. Our attacks on nonlinear resynchronization mechanisms lead to a better understanding of the strength of such mechanisms and can be used to provide a lower bound for the nonlinearity required from a secure resynchronization mechanism. This allows designers to consider the trade-offs between the speed and the security of a resynchronization mechanism.

**Acknowledgements**

The authors would like to thank An Braeken, Joe Cho, Matthias Krause, Stefan Lucks, Erik Zenner and the anonymous referees for helpful comments and discussions.

## References

1. R. Anderson, *A5 (Was: Hacking Digital Phones)*, sci.crypt post, June 1994.
2. F. Armknecht, M. Krause, *Algebraic Attacks on Combiners with Memory*, Crypto 2003, LNCS 2729, D. Boneh, Ed., Springer-Verlag, pp. 162-176, 2003.
3. F. Armknecht, *Improving Fast Algebraic Attacks*, FSE 2004, LNCS 3017, B. Roy, W. Meier, Eds., Springer-Verlag, pp. 65–82, 2004.
4. F. Armknecht, *On the Existence of Low-degree Equations for Algebraic Attacks*, Cryptology ePrint Archive, Report 2004/185, 2004.
5. Bluetooth S.I.G., *Specification of the Bluetooth System, Version 1.2*, available from www.bluetooth.org/spec, 2003.
6. Y. Borissov, S. Nikova, B. Preneel, J. Vandewalle, *On a Resynchronization Weakness in a Class of Combiners with Memory*, SCN 2002, LNCS 2576, S. Cimato, C. Galdi, G. Persiano, Eds., Springer-Verlag, pp. 164–173, 2002.
7. N. Courtois, W. Meier, *Algebraic Attacks on Stream Ciphers with Linear Feedback*, Eurocrypt 2003, LNCS 2656, E. Biham, Ed., Springer-Verlag, pp. 345–359, 2003.
8. N. Courtois, *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, Crypto 2003, LNCS 2729, D. Boneh, Ed., Springer-Verlag, pp. 177–194, 2003.
9. J. Daemen, R. Govaerts, J. Vandewalle, *Resynchronization Weaknesses in Synchronous Stream Ciphers*, Eurocrypt 1993, LNCS 765, T. Helleseth, Ed., Springer-Verlag, pp. 159–167, 1993.
10. S. Fluhrer, *Improved key recovery of level 1 of the Bluetooth Encryption System*, Cryptology ePrint Archive, Report 2002/068, 2002.
11. C. Fontaine, *Contribution à la Recherche de Fonctions Booléennes Hautement Non Linéaires, et au Marquage d'Images en Vue de la Protection des Droits d'Auteur*, PhD Thesis, Paris University, 1998.
12. J. Golic, *Correlation via Linear Sequential Circuit Approximation of Combiners with Memory*, Eurocrypt 1992, LNCS 658, R. Rueppel, Ed., Springer-Verlag, pp. 113–123, 1992.
13. J. Golic, *Linear Cryptanalysis of Stream Ciphers*, FSE 1994, LNCS 1008, B. Preneel, Ed., Springer-Verlag, pp. 154–169, 1994.
14. J. Golic, G. Morgari, *On the Resynchronization Attack*, FSE 2003, LNCS 2887, T. Johansson, Ed., Springer-Verlag, pp. 100–110, 2003.
15. A. Joux, F. Muller, *A Chosen IV Attack against Turing*, SAC 2003, LNCS 3006, M. Matsui, R. Zuccherato, Eds., Springer-Verlag, pp. 194–207, 2003.
16. D. Lee, J. Kim, J. Hong, J. Han, D. Moon, *Algebraic Attacks on Summation Generators*, FSE 2004, LNCS 3017, B. Roy, W. Meier, Eds., Springer-Verlag, pp. 34–48, 2004.
17. M. Matsui, *Linear Cryptanalysis Method for DES Cipher*, Eurocrypt 1993, LNCS 765, T. Helleseth, Ed., Springer-Verlag, pp. 386–397, 1993.
18. W. Meier, E. Pasalic, C. Carlet, *Algebraic Attacks and Decomposition of Boolean Functions*, Eurocrypt 2004, LNCS 3027, C. Cachin, J. Camenisch, Eds., Springer-Verlag, pp. 474-491, 2004.

19. W. Meier, O. Staffelbach, *Correlation Properties of Combiners with Memory in Stream Ciphers (extended abstract)*, Eurocrypt 1990, LNCS 473, I. Damgard, Ed., Springer-Verlag, pp. 204–213, 1990.
20. M. Mihaljević, H. Imai, *Cryptanalysis of Toyocrypt-HS1 stream cipher*, IEICE Transactions on Fundamentals, vol. E85-A, pp. 66–73, Jan. 2002. Available at http://www.csl.sony.co.jp/ATL/papers/IEICEjan02.pdf.
21. New European Schemes for Signature, Integrity and Encryption, http://www.cryptonessie.org
22. R. Rueppel, *Correlation Immunity and the Summation Generator*, Crypto 1985, LNCS 218, H. Williams, Ed., Springer-Verlag, pp. 260–272, 1985.
23. R. Rueppel, *Analysis and Design of Stream Ciphers*, Springer-Verlag, Berlin, 1986.
24. M. Saarinen, *Bluetooth und E0*, sci.crypt post, February 2002.
25. P. Sarkar, S. Maitra, *Nonlinearity Bounds and Constructions of Resilient Boolean Functions*, Crypto 2000, LNCS 1880, M. Bellare, Ed., Springer-Verlag, pp. 515–532, 2000.
26. V. Strassen, *Gaussian Elimination is Not Optimal*, Numerische Mathematik, vol. 13, pp. 354-356, 1969.
27. Y. Tarannikov, *On Resilient Boolean Functions with Maximum Possible Nonlinearity*, Indocrypt 2000, LNCS 1977, B. Roy, E. Okamoto, Eds., Springer-Verlag, pp. 19–30, 2000.
28. G. Xiao, J. Massey, *A Spectral Characterization of Correlation-immune Combining Functions*, IEEE Trans. Inf. Theory, Vol. IT-34, pp. 569–571, 1988.
29. K. Zeng, C. Yang, T. Rao, *On the Linear Consistency Test (LCT) in Cryptanalysis with Applications*, Crypto 1989, LNCS 435, G. Brassard, Ed., Springer-Verlag, pp. 164–174, 1990.
30. Y. Zheng, X. Zhang, *Plateaued Functions*, ICICS 1999, LNCS 1726, V. Varadharajan, Y. Mu, Eds., Springer-Verlag, pp. 284–300, 1999.

## A Calculation of the Complexities for the Overlapping Bits Attack

Suppose we have done $R$ resynchronizations, where $R < \varphi$. This implies that on average $2^{\varphi-R}$ solutions remain for $\mathrm{K}_t$. The probability that one specific bit is the same in all these solutions is $(1/2)^{2^{\varphi-R}-1}$. We have found one bit of information if this is the case, else we didn't learn anything. This holds for all $\varphi$ bits of $\mathrm{K}_t$.

The average frame length required is now equal to the amount of information required (the $k$ bits of the key) divided by the average information learned at each time step:

$$\texttt{frame length} = \frac{k}{\frac{\varphi}{2^{2^{\varphi-R}-1}}} = \frac{k}{\varphi} \cdot 2^{2^{\varphi-R}-1}\,. \tag{44}$$

The time complexity of the attack now consists of performing an exhaustive search over $2^{\varphi}$ possibilities at each time $t$. Hence the time complexity is equal to $2^{\varphi}$ times the frame length, which gives $\frac{n}{\varphi} \cdot 2^{2^{\varphi-R}-1} \cdot 2^{\varphi}$.