Espresso: A Stream Cipher for 5G Wireless Communication Systems

Elena Dubrova[1] and Martin Hell[2]

[1] Royal Institute of Technology, Electrum 229, 164 40 Stockholm, Sweden
dubrova@kth.se
[2] Lund University, Box 117, SE-221 00 Lund, Sweden
martin.hell@eit.lth.se

Abstract. The demand for more efficient ciphers is a likely to sharpen with new generation of products and applications. Previous cipher designs typically focused on optimizing only one of the two parameters - hardware size or speed, for a given security level. In this paper, we present a methodology for designing a class of stream ciphers which takes into account both parameters simultaneously. We combine the advantage of the Galois configuration of NLFSRs, short propagation delay, with the advantage of the Fibonacci configuration of NLFSRs, which can be analyzed formally. According to our analysis, the presented stream cipher Espresso is the fastest among the ciphers below 1500 GE, including Grain-128 and Trivium.

Keywords: Stream cipher, encryption, FSR, wireless, 5G

1 Introduction

The importance of designing efficient and secure cryptographic systems is hard to overestimate. On one hand, with the growth of Internet-of-Things applications, more everyday-life products become security-critical and require high levels of assurance. On the other hand, these products typically have very limited resources available for the implementation of security mechanisms. In addition, the required computational effort and data rates are expected to significantly increase with new generations of products. The 5G is envisioned to have 1000 times higher traffic volume compared to current LTE deployments while providing a better quality of service [1]. Consumer data rates of hundreds of Mbps are expected to be available in a general scenario and multi-Gbps in specific scenarios [2]. Furthermore, 5G needs to support a low latency of a few milliseconds to address use cases such as safety or control mechanisms in the process industry, in the electrical-distribution grid, or for traffic safety [2].

To design a secure cipher which satisfies requirements of the most demanding products and applications, we have to find a best trade-off between hardware size and speed for a given security level. Previous stream cipher designs have either too high propagation delay (e.g. Grain [3]) or use too many flip-flops

(e.g. Trivium [4]) for a given security level. Thus, they optimize only one of the two important parameters - hardware size or speed. In this paper, we present a methodology for designing a class of stream ciphers which takes into account both parameters simultaneously, thus minimizing the hardware footprint and maximizing the throughput of the design. We combine the advantage of the Galois configuration of NLFSRs, short propagation delay, with the advantage of the Fibonacci configuration of NLFSRs, which can be more easily analyzed formally. A careful choice of taps for the output Boolean function allows us to perform a security analysis for linear approximation attacks. According to our evaluation, the presented stream cipher is the fastest among the ciphers below 1500 GE, including Grain-128 and Trivium.

The paper is organized as follows. Section 2 gives basic notation used in the sequel. Section 3 describes previous work. Section 4 presents the new stream cipher Espresso. Section 5 analyses its hardware cost. Section 6 presents the security analysis. Section 7 concludes the paper.

2 Preliminaries

Throughout the paper, we use "$\oplus$" and "$\cdot$" to denote addition and multiplication in $GF(2)$, respectively.

The Boolean functions $GF(2^n) \to GF(2)$ are represented using the *Algebraic Normal Form (ANF)* which is a polynomial over $GF(2)$ of type

$$f(x) = \sum_{i=0}^{2^n-1} c_i \cdot x_0^{i_0} \cdot x_1^{i_1} \cdot \ldots \cdot x_{n-1}^{i_{n-1}},$$

where $c_i \in \{0,1\}$, $(i_0 i_1 \ldots i_{n-1})$ is the binary expansion of i and $x = (x_0, x_1, \ldots, x_{n-1})$ [5].

An n-bit *Feedback Shift Register (FSR)* consists of n binary storage elements, called *stages*. Each stage $i \in \{0, 1, \ldots, n-1\}$ has an associated *state variable* x_i which represents the current value of the stage i and a *feedback function* $f_i : GF(2^n) \to GF(2)$ which determines how the value of i is updated.

A *state* of an FSR is a vector of values of its state variables. At each clock cycle, the next state of an FSR is determined from its current state by simultaneously updating the value of each stage i to the value of the corresponding feedback function f_i, $\forall i \in \{0, 1, \ldots, n-1\}$.

The *period* of an FSR is the length of the longest cyclic output sequence it produces.

If all feedback functions of an FSR are linear, then it is called a *Linear Feedback Shift Register* (LFSR). Otherwise, it is called a *Non-Linear Feedback Shift Register* (NLFSR).

An FSR can be implemented either in the *Fibonacci* or in the *Galois* configuration [6]. In the former, the feedback is applied to the input stage of the shift register only. All remaining feedback functions are of type $f_i = x_{i+1}$, for $i \in \{0, 1, \ldots, n-2\}$. In the latter, the feedback can potentially be applied to

every stage. Thus, the Fibonacci configuration is a special case of the Galois configuration. Due to its conceptual simplicity, the Fibonacci configuration has been studied much more thoroughly.

Two NLFSRs are called *equivalent* if sets of their output sequences are equal.

3 Previous Work

For encryption purposes, there are two types of ciphers, namely *block* and *stream ciphers*. Block ciphers have been studied for over 50 years [7]. Collected knowledge about their design and cryptanalysis made it possible to develop the Advanced Encryption Standard (AES) algorithm which is widely accepted and has strong resistance against various kind of attacks [8].

On the other hand, an active public investigation of stream ciphers began only about 20 years ago [9]. A common type of stream cipher is the *binary additive stream cipher*, in which the keystream, the plaintext, and the ciphertext are binary sequences. The keystream is produced by a *keystream generator* which takes a secret key and an initial value (IV) as a seed and generates a long pseudo-random sequence of 0s and 1s. The ciphertext is then obtained by the bit-wise addition of the keystream and the plaintext.

In the eSTREAM initiative, many stream ciphers, including Grain [3] and Trivium [4], were designed following the belief that stream ciphers can be made both faster and smaller than block ciphers. In recent years, however, several block ciphers have been presented which are comparable in size to Grain and Trivium. Some well-known examples include KATAN [10], LED [11], KLEIN [12], PRESENT [13], Piccolo [14] and TWINE [15]. The throughput for these is often given for 100KHz clock frequency since this is typical for RFID tags [16]. Yet, they can often be clocked faster and [17] reports some implementations reaching about 1Gbps using slightly more than 3000 GE and 90nm CMOS technology. For higher throughput and more compact design it appears that stream ciphers are the best choice.

However, the confidence in stream ciphers' security has been tapered by many broken systems. For example, the popular stream ciphers A5/1 and A5/2 used in the Global System for Mobile communications (GSM) standard and E0 used in Bluetooth have been found susceptible to a number of attacks [18]. As a result, A5/1 was replaced by a block cipher based A5/3 and A5/2 was prohibited. Another well-known stream cipher RC4 used in the original IEEE 802.11 standard to secure wireless networks has been shown especially vulnerable when the beginning of the output keystream is not discarded, non-random or related keys are used, or a single keystream is used twice [19]. As a result, it was replaced by the AES in the newest standard, IEEE 802.11i.

The confidence in stream ciphers is typically higher and their acceptance is faster if they are built from well-defined components whose security can be formally analyzed. One of the most studied components for stream ciphers is a *filter generator* which consists of an FSR [20] and a nonlinear output function taking its inputs from the selected stages of the FSR. It is known how

to make design choices for the size of internal state and the output function (number and position of inputs, nonlinearity, resiliency, algebraic degree, etc) so that the resulting filter generator is resistant to known attacks with a sufficient security margin [21–23]. Techniques that can guarantee long FSR period are also known [24, 25]. Examples of stream ciphers based on the idea behind filter generators include Grain [3] and Trivium [4].

4 Design Description

This section motivates and describes the presented stream cipher Espresso.

4.1 Design Methodology

The Grain family of stream ciphers [3] uses FSRs in the Fibonacci configuration. This adds simplicity to the security analysis, but has a drawback that the propagation delay through the feedback function is large due to the large size of the function. Trivium [4] uses much simpler feedback functions, but it has 288 flip-flops which are more area consuming compared to gates.

First, by using FSRs implemented in the Galois configuration, we can make the feedback functions smaller. This allows us to reduce the propagation delay compared to Grain while at the same time decrease the size compared to Trivium. Due to the large number of feedback functions in the presented design, its maximum degree of parallelization cannot be made as high as in both Grain and Trivium. Still, by carefully choosing feedback functions, we are able to guarantee the maximum degree of parallelization 4 and a maximum-length FSR.

Second, to enable security analysis of the presented design, we transform the original Galois NLFSR to an NLFSR whose configuration resembles the Fibonacci configuration. The core idea of our method is to assure that all of the most biased linear approximations of the output Boolean function take inputs only from those stages of the Galois NLFSR which have a corresponding equivalent stage in the transformed NLFSR. As a result, traditional cryptanalysis techniques can be applied to our design as well.

4.2 Design Details

The two main building blocks of Espresso are a 256-bit NLFSR G in the Galois configuration and a 20-variable nonlinear output function. To avoid confusion between the feedback functions of G and the feedback functions of the transformed NLFSR F introduced later, we denote a feedback function of the stage i of G by g_i, for all $i \in \{0, 1, \ldots, 255\}$.

The feedback functions of the NLFSR G are specified as follows:

$$
\begin{aligned}
g_{255}(x) &= x_0 \oplus x_{41}x_{70} \\
g_{251}(x) &= x_{252} \oplus x_{42}x_{83} \oplus x_8 \\
g_{247}(x) &= x_{248} \oplus x_{44}x_{102} \oplus x_{40} \\
g_{243}(x) &= x_{244} \oplus x_{43}x_{118} \oplus x_{103} \\
g_{239}(x) &= x_{240} \oplus x_{46}x_{141} \oplus x_{117} \\
g_{235}(x) &= x_{236} \oplus x_{67}x_{90}x_{110}x_{137} \\
g_{231}(x) &= x_{232} \oplus x_{50}x_{159} \oplus x_{189} \\
g_{217}(x) &= x_{218} \oplus x_3x_{32} \\
g_{213}(x) &= x_{214} \oplus x_4x_{45} \\
g_{209}(x) &= x_{210} \oplus x_6x_{64} \\
g_{205}(x) &= x_{206} \oplus x_5x_{80} \\
g_{201}(x) &= x_{202} \oplus x_8x_{103} \\
g_{197}(x) &= x_{198} \oplus x_{29}x_{52}x_{72}x_{99} \\
g_{193}(x) &= x_{194} \oplus x_{12}x_{121}
\end{aligned}
$$

All remaining feedback functions of G are of type $g_i(x) = x_{i+1}$.

The output function $z(x)$ is specified as follows:

$$
\begin{aligned}
z(x) = {} & x_{80} \oplus x_{99} \oplus x_{137} \oplus x_{227} \oplus x_{222} \oplus x_{187} \oplus x_{243}x_{217} \oplus x_{247}x_{231} \\
& \oplus x_{213}x_{235} \oplus x_{255}x_{251} \oplus x_{181}x_{239} \oplus x_{174}x_{44} \oplus x_{164}x_{29} \\
& \oplus x_{255}x_{247}x_{243}x_{213}x_{181}x_{174}
\end{aligned}
$$

The function $z(x)$ consists of a linear function of 6 variables and a bent function of 14 variables. Therefore, $z(x)$ is balanced, has nonlinearity $2^6(2^{13} - 2^6) = 520192$ and resiliency 5. The algebraic degree of $z(x)$ is 6 since its largest ANF monomial contains 6 variables.

In $z(x)$, 15 out of 20 indexes of variables are taken from the following full positive difference set[3]:

$$\{255, 247, 243, 227, 222, 213, 187, 181, 174, 164, 137, 99, 80, 44, 29\}.$$

In seven two-variable monomials of $z(x)$, the difference between the first and the second indexes of variables is taken from the following full positive difference set:

$$\{26, 16, 22, 4, 58, 130, 135\}.$$

NLFSRs in the Fibonacci configuration are much more studied and cryptanalyzed compared to the NLFSRs in the Galois configuration. To make use of the accumulated knowledge, we can transform the NLFSR G into an equivalent NLFSR F whose configuration resembles the Fibonacci configuration (see Figure 1) and therefore is easier to cryptanalyze. The NLFSR F has only two

[3] A set is called full positive difference set if the positive pairwise differences between its elements are distinct [21]

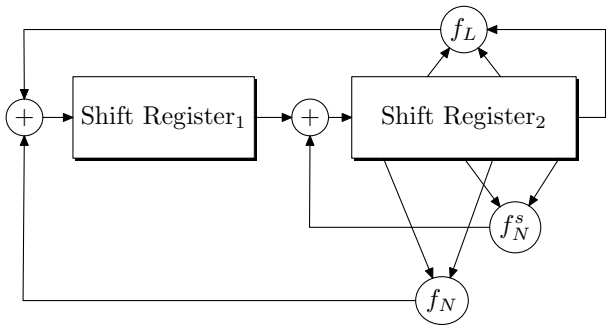

Fig. 1. The structure of the transformed NLFSR F.

non-trivial feedback functions:

$$\begin{aligned} f_{255}(x) &= x_0 \oplus x_{12} \oplus x_{48} \oplus x_{115} \oplus x_{133} \oplus x_{213} \oplus x_{41}x_{70} \oplus x_{46}x_{87} \\ &\quad \oplus x_{52}x_{110} \oplus x_{55}x_{130} \oplus x_{62}x_{157} \oplus x_{74}x_{183} \oplus x_{87}x_{110}x_{130}x_{157} \\ f_{217}(x) &= x_{218} \oplus x_3x_{32} \oplus x_8x_{49} \oplus x_{14}x_{72} \oplus x_{17}x_{92} \oplus x_{24}x_{119} \oplus x_{36}x_{145} \\ &\quad \oplus x_{49}x_{72}x_{92}x_{119} \end{aligned}$$

and all remaining feedback functions are of type $f_i(x) = x_{i+1}$. We can see that the function $f_{255}(x)$ of F is of type $f_{255}(x) = f_L(x) \oplus f_N(x)$ where $f_L(x)$ is a linear function of 6 variables and $f_N(x)$ is a nonlinear bent function of 12 variables. The function $f_{217}(x)$ is of type $f_{217}(x) = x_{218} \oplus f_N^s(x)$ where $f_N^s(x)$ is the "shifted" version of $f_N(x)$, in which each variable x_i is replaced by x_{i-38}. $f_N^s(x)$ cancels the effect of non-linearity introduced by $f_N(x)$. So, stages 217 to 0 of F generate the linear sequence induced by the primitive polynomial

$$1 + x^{12} + x^{48} + x^{115} + x^{133} + x^{213} + x^{256} \tag{1}$$

which is induced by the function $f_L(x)$. It is known that NLFSRs constructed in this way have the period $2^n - 1$ where n is the size of the state [24].

The equivalence of G and F can be shown by applying the Fibonacci-to-Galois transformation [6]. The set of sequences generated by the stage 231 of G is equivalent to the set of sequences generated by the stage 255 of F. The set of sequences generated by the stage 193 of G is equivalent to the set of sequences generated by the stage 217 of F. Since G is equivalent to F, its period is $2^{256}-1$.

The function $f_{255}(x)$ is balanced, has nonlinearity $2^6(2^{11} - 2^5) = 129024$, resiliency 5, and algebraic degree 4. Since F and G are equivalent, the function $g_{231}(x)$ of G has the same properties.

Indexes of variables of $f_L(x)$ form the full difference set

$$\{0, 12, 48, 115, 133, 213\}$$

and indexes of variables of $f_N(x)$ form the full difference set

$$\{41, 46, 52, 55, 62, 70, 74, 87, 110, 130, 157, 183\}.$$

4.3 Key and IV Initialization

The cipher Espresso is initialized as follows. Let k_i denote the bits of the key k, $0 \leq i \leq 127$, and IV_i denote the bits of the initialization value IV, $0 \leq i \leq 95$. The key and IV bits are loaded into the shift register as follows:

$$\begin{aligned} x_i &= k_i, && 0 \leq i \leq 127 \\ x_i &= IV_{i-128}, && 128 \leq i \leq 223 \\ x_i &= 1, && 224 \leq i \leq 254 \\ x_i &= 0, && i = 255 \end{aligned}$$

The initialization phase consists of clocking the cipher 256 times, XORing the produced output bit with the stages x_{255} and x_{217}. Thus, in this phase the feedback functions $g_{255}(x)$ and $g_{217}(x)$ of the NLFSR G are given by

$$\begin{aligned} g_{255}(x) &= x_0 \oplus x_{41}x_{70} \oplus z(x) \\ g_{217}(x) &= x_{218} \oplus x_3x_{32} \oplus z(x) \end{aligned}$$

After initialization, the cipher is clocked for three more cycles (due to the pipelining of the output function and additional logic required for switching between the initialization and the keystream generation phases, as explained in Section 5) and then the keystream is produced.

5 Hardware Cost Analysis

In order to reduce the propagation delay of the circuit implementing the output function $z(x)$, we can pipeline it as follows:

$$\begin{aligned} z_1(x) &= x_{80} \oplus x_{99} \oplus x_{137} \oplus x_{227} \\ z_2(x) &= x_{222} \oplus x_{187} \oplus x_{243}x_{217} \\ z_3(x) &= x_{247}x_{231} \oplus x_{213}x_{235} \\ z_4(x) &= x_{255}x_{251} \oplus x_{181}x_{239} \\ z_5(x) &= x_{174}x_{44} \oplus x_{164}x_{29} \\ z_6(x) &= x_{255}x_{247}x_{243}x_{213}x_{181}x_{174} \\ z_7(x) &= z_1(x) \oplus z_2(x) \oplus z_3(x) \oplus z_4(x) \\ z_8(x) &= z_5(x) \oplus z_6(x) \\ z(x) &= z_7(x) \oplus z_8(x) \end{aligned}$$

A circuit diagram implementing the pipelined version of $z(x)$ is showed in Figure 2. As a consequence of the pipelining, the output of the stream cipher is delayed by two clock cycles, increasing the latency. In addition, the pipelining increases the area by 8 flip-flops. However, it allows us to increase the throughput 1.7 times. In our opinion, the substantial gain in throughput outweighs the minor increases in areas and latency.

In order to further reduce the propagation delay of the presented design, we apply de Morgan rule to re-express ANFs of the feedback functions g_{235} and g_{197} of the NLFSR G as follows:

$$\begin{aligned} g_{235}(x) &= x_{236} \oplus x_{67}x_{90}x_{110}x_{137} = x_{236} \oplus ((x_{67}x_{90})' + (x_{110}x_{137})')' \\ g_{197}(x) &= x_{198} \oplus x_{29}x_{52}x_{72}x_{99} = x_{198} \oplus ((x_{29}x_{52})' + (x_{72}x_{99})')' \end{aligned}$$

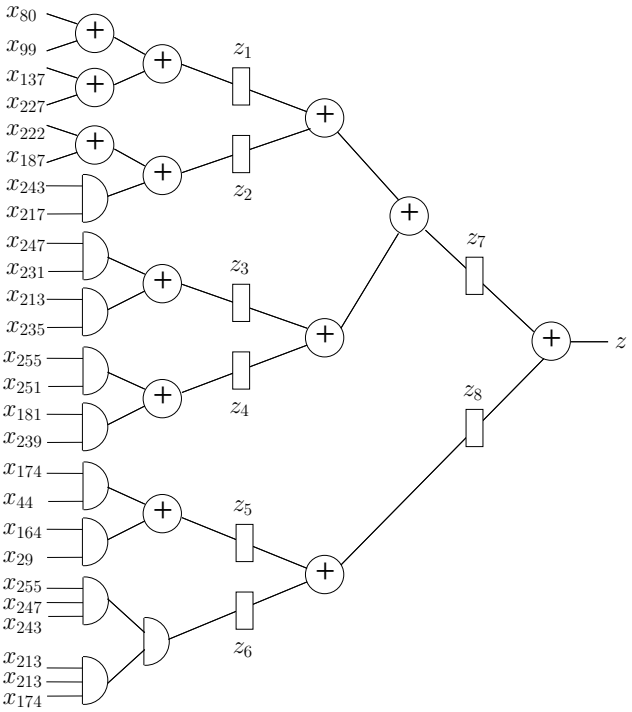

Fig. 2. A circuit implementing the pipelined version of $z(x)$.

where x' denotes the Boolean complement of x (defined as $x' = x \oplus 1$), and "+" denotes the Boolean OR. From Table 1, the reader can see that, in CMOS technology, NAND or NOR are much smaller and faster than AND. Therefore, we can decrease both, the area and the delay, by replacing a 4-input AND as shown above.

Finally, we need to take care of the propagation delay of the feedback functions g_{255} and g_{217} during the initialization phase. In this phase, the functions $g_{255}(x)$ and $g_{217}(x)$ are computed as

$$g_{255}(x) = x_0 \oplus x_{41}x_{70} \oplus z(x)$$
$$g_{217}(x) = x_{218} \oplus x_3x_{32} \oplus z(x)$$

Figure 3 shows how switching between the initialization and the keystream generation phases can be implemented for g_{255} without increasing the critical path (a circuit for the function g_{217} is similar). The output of $z(x)$ needs to be multiplexed and pipelined. Note that while the function describing a regular 2-input multiplexer (MUX) is $a \cdot b + a' \cdot c$, a multiplexer in which one input is fixed to 0 can be implemented as

$$a \cdot b + a' \cdot 0 = a \cdot b$$

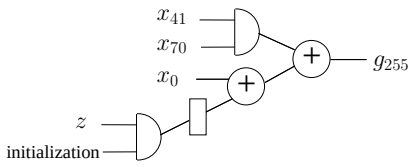

Fig. 3. A circuit implementing the switching between the initialization and the keystream generation phases.

Gate	Area, μm^2	Area, GE	Delay, ps
2-input NAND	3.7	1	33
2-input NOR	3.7	1	57
2-input AND	5	1.4	87
3-input AND	7	1.9	95
2-input XOR	10	2.7	115
flip-flop	19	5.1	221

Table 1. Parameters of gates and flip-flops for a typical 90nm CMOS technology.

Therefore, an AND gate can be used to implement the multiplexing of $z(x)$, as shown in Figure 3. Since the delay of an AND is smaller than the delay of an XOR, the proposed switching scheme does not increase the overall delay. However, it increases the latency by one clock cycle.

After these modifications, the NLFSR G requires 12 2-input ANDs, 4 2-input NANDs, 2 2-input NORs, 19 2-input XORs and 256 flip-flops to be implemented. The output function $z(x)$ requires 8 2-input ANDs, 2 3-input ANDs, 13 2-input XORs and 8 flip-flops. The additional logic for switching between the initialization and the keystream generation phases requires 2 ANDs, 2 XORs and 2 flip-flops.

If we use 90nm CMOS technology for implementing this NLFSR (see Table 1 for parameters of gates), then we can approximate the area and the propagation delay of the presented stream cipher Espresso as

Area of (22 2-input ANDs + 2 3-input ANDs + 4 2-input NANDs +
2 2-input NORs + 34 XORs + 266 flip-flops) = 5540 μm^2 = 1497 GE
Delay of (2 XORs + flip-flop) = 451 ps.

So, the presented design can support data rates of up to 2.22 Gbits/sec. Its latency is 232 ns (computed as (256+256+3) clock cycles $\times$ 451 ps). It can be parallelized to produce up to 4 bits per clock cycle, because three bits after each stage which is updated using a feedback are not used neither as state variables nor in the output function.

p	Espresso			
	Area, A_1, GE	Throughput, T_1, Gbits/s	Latency, L_1, ns	Security, bits
1	1497	2.22	232	128
2	1680	4.44	116	128
4	2045	8.88	59	128

Table 2. Hardware parameters of Espresso; p = degree of parallelization (number of bits generated per one clock cycle).

For a comparison, the area and propagation delay of the stream cipher Grain-128 [3] are given by:

Area of (22 ANDs + 34 XORs + 258 flip-flops) = 5352 μm^2 = 1446 GE
Delay of (AND + 4 XORs + flip-flop) = 768 ps.

We assume that a logic similar to the one shown in Figure 3 is used for switching between the initialization and the operational phases. Otherwise, the delay of Grain-128 is considerably higher. Grain-128 can be parallelized to produce up to 32 bits per clock cycle. For the degree of parallelization one, its latency is 296 ns (computed as $(128 + 256 + 1)$ clock cycles $\times$ 768 ps).

For the stream cipher Trivium [4] we have:

Area of (3 ANDs + 11 XORs + 288 flip-flops) = 5597 μm^2 = 1513 GE
Delay of (AND + 2 XORs + flip-flop) = 538 ps.

Trivium can be parallelized to produce up to 64 bits per clock cycle. For the degree of parallelization one, its latency is 663 ns (computed as $(80 + 4 \times 288)$ clock cycles $\times$ 538 ps).

Note that, in all three ciphers, the latency can be reduced if the key and IV are loaded in parallel rather than sequentially. However, such a technique requires the addition of a MUX to each flip-flip of the FSRs, implying the increase in area by at least 3 GE $\times$ FSR size. Furthermore, the propagation delay of the presented design and Trivium would increase by at least twice the delay of a NAND, implying a reduction in the maximum data rate of 283 Mbits/s for the presented design and of 203 Mbits/s for Trivium.

Tables 2, 3 and 4 summarizes the area and throughput of the three ciphers for the degrees of parallelization 1, 2 and 4. For the degree of parallelization 1, Espresso is 3.4% larger and 71% faster than Grain-128. Its latency is 22% smaller than the one of Grain-128. Compared to Trivium, it is 1% smaller, 19% faster, and has 65% smaller latency. We can see that Espresso is the fastest among the designs below 1500 GE.

p	Grain-128				$\frac{A_1-A_2}{A_1}$	$\frac{T_1-T_2}{T_2}$	$\frac{L_1-L_2}{L_2}$
	Area, A_2, GE	Throughput, T_2, Gbits/s	Latency, L_2, ns	Security, bits			
1	1446	1.30	296	128	3.4%	71%	-22%
2	1578	2.60	147	128	6.1%	71%	-22%
4	1842	5.20	47	128	9.9%	71%	-22%

Table 3. Comparison of Espresso with Grain-128.

p	Trivium				$\frac{A_1-A_3}{A_1}$	$\frac{T_1-T_3}{T_3}$	$\frac{L_1-L_3}{L_2}$
	Area, A_3, GE	Throughput, T_3, Gbits/s	Latency, L_3, ns	Security, bits			
1	1513	1.86	663	80	-1.0%	19%	-65%
2	1547	3.72	332	80	7.9%	19%	-65%
4	1614	7.43	166	80	21.1%	19%	-65%

Table 4. Comparison of Espresso with Trivium.

6 Security Analysis

This section will give a security analysis of the presented stream cipher Espresso. Both attacks on the running key stream and attacks on the initialization procedure are discussed.

6.1 Linear Approximations

Attacks using linear approximations were successful against the initial version of Grain, resulting in key recovery attacks. Being an NLFSR with a nonlinear output function, the current design has similarities with Grain. This makes it important to determine the resistance against these attacks.

The security against linear attacks will be analyzed using the equivalent transformed configuration F of the shift register G. Note that there are no linear terms in any shift register stages that do not have an equivalent in both configurations, so the analysis is valid also for the Galois configuration. For clarity of the presentation, we divide the state register into two separate parts. The state variables in the nonlinear part (Shift Register$_1$ in Figure 1) are denoted b and the state variables in the linear part (Shift Register$_2$ in Figure 1) are denoted by s. Furthermore, let B and S denote the size of the nonlinear part and the linear part of the shift register respectively. Thus, we have

$$b_i = s_{i+S} \qquad 0 \leq i < B. \tag{2}$$

The linear stages $s_i, 0 \leq i < S$ satisfy the linear recurrence relation

$$s_{i+256} = s_i + s_{i+43} + s_{i+123} + s_{i+141} + s_{i+208} + s_{i+244}$$

which is induced by the polynomial (1). Define the bias ε of an approximation as $\varepsilon = 2 \cdot \Pr(X = Y) - 1$, simply written as $X \overset{\varepsilon}{=} Y$. Then, the nonlinear output function can be approximated with a linear function and we can write

$$z(t) \overset{\varepsilon_1}{=} \bigoplus_i b_0(t + \phi_i) \oplus \bigoplus_j s_0(t + \theta_j). \tag{3}$$

Denote the number of b-variables in the output function $w_b(z)$, i.e., $0 \leq i < w_b(z)$ in (3). Similarly, the nonlinear feedback function can be approximated by a linear function in bits from s since there are no b-variables in the feedback in order for the nonlinear compensation to s_{S-1} to work properly. Thus, we can also write

$$b_{B-1}(t + 1) = b_0(t + B) \overset{\varepsilon_2}{=} \bigoplus_k s_0(t + \mu_k) \tag{4}$$

Combining (3) and (4), we can write the output as a sum of only variables from the linear part of the shift register,

$$z(t) \overset{\varepsilon_1}{=} \bigoplus_i b_0(t + B - (B - \phi_i)) \oplus \bigoplus_j s_0(t + \theta_j) \tag{5}$$

$$\overset{\varepsilon_1 \varepsilon_2^{w_b(z)}}{=} \bigoplus_i \bigoplus_k s_0(t + \mu_k - (B - \phi_i)) \oplus \bigoplus_j s_0(t + \theta_j), \tag{6}$$

where the piling-up lemma has been used to combine linear approximations. Thus, an output variable can always be written as a biased sum of s-variables, which in turn satisfy a linear recurrence relation. If we denote the weight of this recurrence relation by $w(LR)$, we get a distinguishing attack with total bias

$$\varepsilon_{\text{tot}} = \left(\varepsilon_1 \varepsilon_2^{w_b(z)}\right)^{w(LR)}. \tag{7}$$

From this it is clear that the complexity of the attacks relies on the biases of the two approximations and on the number of b-variables that are used in the linear approximation of the output function. Looking at the design, we have $\varepsilon_1 = 2^{-7}$ and $\varepsilon_2 = 2^{-6}$ and $w_b(z) = 6$ for all biased linear approximations. From this it follows that the approximation (6) has bias 2^{-43} which makes an attack similar to the one in [26] inefficient. Also, if we use a weight 3 multiple of the linear recurrence relation the number of samples needed would be in the order of $1/\varepsilon_{\text{tot}}^2 = 2^{43 \cdot 3 \cdot 2} = 2^{172}$ (with distance $2^{218/2} = 2^{109}$ between first and last keystream bit in each sample [27, 28]).

6.2 Algebraic Attacks

Algebraic attacks have been proved very efficient against nonlinear combiners with or without memory [29,30]. The success of the attack is due to the linearity of the shift register and the fact that the output function is the only nonlinear part of the register. It is always possible to write equations describing output

bits using initial state bits. Due to the linearity of the shift register, the algebraic degree of these equations will never exceed the degree of the output function. With enough equations, linearization, or other more advanced methods [31–33], can be used to recover the internal state. Moreover, annihilators [34] can be used to lower the degree of the functions even more. With a part of the state being nonlinearly updated, these attacks are no longer applicable since several nonlinear register stages are used in the output function. The degree of the equations in initial state bits will increase and is not limited by the degree of the output function.

6.3 Time-Memory-Data Trade-off Attacks

TMTO attacks on stream ciphers can be divided into two categories, those that attempt to reconstruct the internal state, see e.g., [35–37] and those that attempt to recover the key, see e.g., [38]. The algorithms used in the latter attacks are the same as those in the former, they just use a different one-way function as target of the attack. The algorithm used in [35,36] simply records input/output combinations and uses enough data in order to have a collision with a recorded value. The trade off curve is given by $TM = N$, $T = D$, and $P = M = N/D$. The algorithm used in [37] instead created tables similar to those used by Hellman in [39] and has the trade-off given by $TM^2D^2 = N^2$, $1 \leq D^2 \leq T$ and $P = N/D$. Both algorithms uses the observation that an increased amount of data can lower the precomputation time. Since the size of the internal state is 2^{2k}, it is clear that recovering the internal state is not possible with $T < 2^k$ and $M < 2^k$ using any of the algorithms. On the other hand, recovering the key would be possible with e.g., $T = 2^{112}$, $M = 2^{112}$ and $D = 2^{56}$ but will require a precomputation time of $P = 2^{168}$. Some might argue that this would be a valid (academic) attack while some would claim that $P = 2^{168}$ is too large to be interesting when key size is 128 bits.

Ad hoc improvements to the TMTO attacks can also be considered, where recovering a subset of bits will allow recovering other bits as well using algebraic relations in the output function. The success of these attacks are specific to the design, in particular to the output function chosen in the design. The idea, as proposed in [40, 41] and demonstrated on the Grain family of stream ciphers, is to identify a subset of state bits, which together with some output bits can be used to determine the remaining state bits. Using this observation, the TMTO attack can be improved by only considering the subset of state bits needed for recovering the rest. The normality of the output Boolean function will here play an important role as it determines how many shift register bits need to be fixed in order to recover remaining state bits. The normality order of this function in the design is 7, which means that $14 - 7 = 7$ variables need to be fixed in order to get linear equations for the recovery. The Galois configuration of the shift register G, together with the fact that not all bits have a corresponding bit in the transformed equivalent register F, will complicate this attack. Still, we do not rule out that some improvement over the generic TMTO attacks are possible using this approach. However, the required memory complexity of such

an attack will far exceed that of brute force and a parallelized brute force [42] is likely to be much more efficient.

6.4 Chosen IV Attacks

The complexity of the initialization function does not affect the attack complexities in the TMTO attacks. In this section we consider attacks that do depend on the initialization function. In a chosen IV scenario, the adversary can choose the initialization vector used in the initialization step. This is the basis for the Cube attack [43] and AIDA attack [44] and can lead to key recovery if the initialization is not carefully designed. The number of iterations in the initialization should be chosen such that all key and IV bits affect the keystream bits in a complex way.

To determine the resistance against these types of attacks, maximum degree monomial tests have been performed. Any keystream bit can be written as a function of key and IV bits

$$z_i = f_i(k_0, \ldots, k_{127}, iv_0, \ldots, iv_{95}). \tag{8}$$

All key bits are fixed to zero and a subset of the IV bits are fixed as well. Thus, running through all possible combinations of the non-fixed bits, the truth table of the function f_i is obtained, which can in turn be used to compute the ANF. This will lead to a d-monomial test [45] as we could check the presence of monomials of degree d and compare it to the expected number for a random Boolean function. Intuitively, the maximum degree monomial only exists if all bits have been properly mixed by the initialization function, so we focus on this monomial. The total number of bits that can be used is 96 requiring a complexity of 2^{96} in order to determine the presence of the monomial $iv_0, \ldots, iv_{95}$. This is not feasible, and we instead adopt the test in [46] in order to find a monomial with manageable degree and that will be absent for as many initialization rounds as possible. The algorithm starts with just a few bits and exhaustively finds the monomial that is absent the maximum number of rounds. Then it greedily adds one more bit to the set and continues. All non-used key and IV bits are set to zero. For a conservative estimate the algorithm is allowed to use also key bits. This turns the Chosen IV attack into a less powerful nonrandomness detector since an attacker is not assumed to be able to choose key bits. Figure 4 shows the number of initialization rounds that can be broken using a particular degree (bit set size) for the monomial.

By using dedicated hardware it would be possible to test a larger number of IV bits, i.e., larger degree monomials. However, from the results in Figure 4 we deduce that the number of initialization steps is adequate to resist these types of chosen IV attacks. With 159 rounds that fail the nonrandomness test, we conclude that the proposed 256 rounds provide an adequate security margin. For a comparison, this test applied to Grain-128 can find non-randomness in about 240 initialization rounds with bit set size 23. Using bit set size of 40 the full Grain-128 initialization using 256 shows nonrandomness.

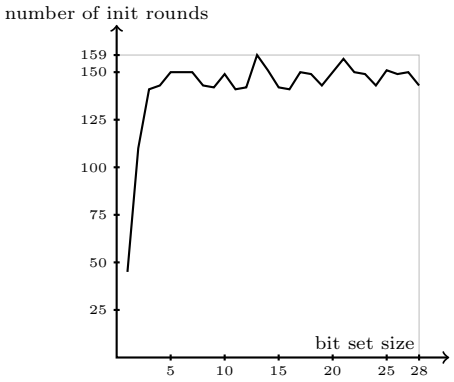

Fig. 4. The maximum number of initialization rounds that do not pass a maximum degree monomial test for a given monomial degree.

7 Conclusion

We presented a new stream cipher Espresso targeting 5G wireless communication systems. Its 1-bit per cycle version has 1497 GE area, 2.22 Gbits/sec throughput and 232 ns latency, meeting requirements of most 5G applications envisioned today. It is resistant to known attacks, including linear approximations, algebraic attacks, time-memory-data trade off attacks and chosen IV attacks.

8 Acknowledgements

This work carried out during the research visits of authors at the security group at Ericsson Research in 2013-2014 and supported by the grants SM12-0005 and SM12-0025 from the Swedish Foundation for Strategic Research. The authors would like to thank Mats Näslund and Ben Smeets from Ericsson Research for their help with this work, for sharing their expertise and providing many valuable comments and suggestions on the design.

References

1. M. Olsson, C. Cavdar, P. Frenger, S. Tombaz, D. Sabella, and R. Jantti, "5green: Towards green 5g mobile networks," in *Int. Conf. on Wireless and Mobile Computing, Networking and Communications*, pp. 212–216, Oct 2013.
2. Ericsson White Paper, "5G radio access," June 2013. http://www.ericsson.com/res/docs/whitepapers/wp-5g.pdf.
3. M. Hell, T. Johansson, A. Maximov, and W. Meier, "The Grain family of stream ciphers," *New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986*, pp. 179–190, 2008.
4. C. Cannière and B. Preneel, "Trivium," *New Stream Cipher Designs: The eSTREAM Finalists, LNCS 4986*, pp. 244–266, 2008.

5. R. Lidl and H. Niederreiter, *Introduction to Finite Fields and their Applications*. Cambridge Univ. Press, 1994.
6. E. Dubrova, "A transformation from the Fibonacci to the Galois NLFSRs," *IEEE Transactions on Information Theory*, vol. 55, pp. 5263–5271, November 2009.
7. B. Schneier, *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*. New York, NY, USA: John Wiley & Sons, Inc., 1995.
8. J. Daemen and V. Rijmen, "AES proposal: Rijndael," April 2003. National Institute of Standards and Technology.
9. M. Robshaw, "Stream ciphers," Tech. Rep. TR - 701, July 1994.
10. C. De Cannière, O. Dunkelman, and M. K. zević, "KATAN and KTANTAN — a family of small and efficient hardware-oriented block ciphers," in *Cryptographic Hardware and Embedded Systems—CHES 2009*, vol. 5747, pp. 272–288, Springer, 2009.
11. J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, "The led block cipher," in *Cryptographic Hardware and Embedded Systems, CHES 2011* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, pp. 326–341, Springer Berlin / Heidelberg, 2011.
12. Z. Gong, S. Nikova, and Y. Law, "Klein: A new family of lightweight block ciphers," in *RFID. Security and Privacy* (A. Juels and C. Paar, eds.), vol. 7055 of *Lecture Notes in Computer Science*, pp. 1–18, Springer Berlin Heidelberg, 2012.
13. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems—CHES 2007*, vol. 4727 of *Lecture Notes in Computer Science*, pp. 450–466, Springer Berlin Heidelberg, 2007.
14. K. Shibutani, T. Isobe, H. Hiwatari, A. Mitsuda, T. Akishita, and T. Shirai, "Piccolo: An ultra-lightweight blockcipher," in *Cryptographic Hardware and Embedded Systems - CHES 2011* (B. Preneel and T. Takagi, eds.), vol. 6917 of *Lecture Notes in Computer Science*, pp. 342–357, Springer, 2011.
15. T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE: A lightweight block cipher for multiple platforms," in *Selected Areas in Cryptography—SAC 2012* (L. Knudsen and H. Wu, eds.), vol. 7707 of *Lecture Notes in Computer Science*, pp. 339–354, Springer Berlin Heidelberg, 2013.
16. A. Juels, "RFID security and privacy: a research survey," *Selected Areas in Communications, IEEE Journal on*, vol. 24, pp. 381–394, Feb. 2006.
17. J. Borghoff, A. Canteaut, T. Gneysu, E. Kavun, M. Knezevic, L. Knudsen, G. Leander, V. Nikov, C. Paar, C. Rechberger, P. Rombouts, S. Thomsen, and T. Yaln, "Prince a low-latency block cipher for pervasive computing applications," in *Advances in Cryptology ASIACRYPT 2012* (X. Wang and K. Sako, eds.), vol. 7658 of *Lecture Notes in Computer Science*, pp. 208–225, Springer Berlin Heidelberg, 2012.
18. E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in *INDOCRYPT '00: Proceedings of the First International Conference on Progress in Cryptology*, (London, UK), pp. 43–51, Springer-Verlag, 2000.
19. E. Tews, R.-P. Weinmann, and A. Pyshkin, "Breaking 104-bit wep in under a minute." Cryptology ePrint Archive, Report 2007/120, 2007. http://eprint.iacr.org/.
20. S. Golomb, *Shift Register Sequences*. Aegean Park Press, 1982.
21. J. Golic, "On the security of nonlinear filter generators," in *Fast Software Encryption* (D. Gollmann, ed.), vol. 1039 of *Lecture Notes in Computer Science*, pp. 173–188, Springer Berlin / Heidelberg, 1996.

22. A. Braeken and J. Lano, "On the (im)possibility of practical and secure nonlinear filters and combiners," in *Proceedings of the 12th international conference on Selected Areas in Cryptography*, SAC'05, (Berlin, Heidelberg), pp. 159–174, Springer-Verlag, 2006.
23. T. W. Cusick and P. Stănică, *Cryptographic Boolean functions and applications*. San Diego, CA, USA: Academic Press, 2009.
24. E. Dubrova, "A scalable method for constructing Galois NLFSRs with period 2^n-1 using cross-join pairs," *IEEE Transactions on Information Theory*, vol. 1, no. 59, pp. 703–709, 2013.
25. E. Dubrova, "A method for generating full cycles by a composition of nlfsrs," *Design, Codes and Cryptography*, 2012.
26. C. Berbain, H. Gilbert, and A. Maximov, "Cryptanalysis of Grain," in *Fast Software Encryption 2006* (M. Robshaw, ed.), vol. 4047 of *Lecture Notes in Computer Science*, pp. 15–29, Springer, 2006.
27. D. Wagner, "A generalized birthday problem," in *Advances in Cryptology—CRYPTO 2002* (M. Yung, ed.), vol. 2442 of *Lecture Notes in Computer Science*, pp. 288–303, Springer, 2002.
28. J. D. Golić, "Computation of low-weight parity check polynomials," *Electronic Letters*, vol. 32, no. 21, pp. 1981–1982, 1996.
29. N. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," in *Advances in Cryptology—EUROCRYPT 2003* (E. Biham, ed.), vol. 2656 of *Lecture Notes in Computer Science*, pp. 345–359, Springer, 2003.
30. F. Armknecht and M. Krause, "Algebraic attacks on combiners with memory," in *Advances in Cryptology—CRYPTO 2003* (D. Boneh, ed.), vol. 2729 of *Lecture Notes in Computer Science*, pp. 162–176, Springer, 2003.
31. N. Courtois, A. Klimov, J. Patarin, and A. Shamir, "Efficient algorithms for solving overdefined systems of multivariate polynomial equations," in *Advances in Cryptology—EUROCRYPT 2003* (B. Preneel, ed.), vol. 1807 of *Lecture Notes in Computer Science*, pp. 392–407, Springer, 2000.
32. N. Courtois and J. Pieprzyk, "Cryptanalysis of block ciphers with overdefined systems of equations," in *Advances in Cryptology—ASIACRYPT 2002* (Y. Zheng, ed.), vol. 2501 of *Lecture Notes in Computer Science*, pp. 267–287, Springer, 2002.
33. J.-C. Faugére and A. Joux, "Algebraic cryptanalysis of Hidden Field Equation (HFE) cryptosystems using Gröbner bases," in *Advances in Cryptology—CRYPTO 2003* (D. Boneh, ed.), vol. 2729 of *Lecture Notes in Computer Science*, pp. 44–60, Springer, 2003.
34. W. Meier, E. Pasalic, and C. Carlet, "Algebraic attacks and decomposition of Boolean functions," in *Advances in Cryptology—EUROCRYPT 2004*, vol. 3027 of *Lecture Notes in Computer Science*, pp. 474–491, Springer, 2004.
35. J. Golić, "Cryptanalysis of alleged A5 stream cipher," in *Advances in Cryptology—EUROCRYPT 1997* (W. Fumy, ed.), vol. 1233 of *Lecture Notes in Computer Science*, pp. 239–255, Springer, 1997.
36. S. Babbage, "A space/time tradeoff in exhaustive search attacks on stream ciphers," in *European Convention on Security and Detection*, no. 408 in IEE Conference Publication, 1995.
37. A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Advances in Cryptology—ASIACRYPT 2000* (T. Okamoto, ed.), vol. 1976 of *Lecture Notes in Computer Science*, pp. 1–13, Springer, 2000.
38. J. Hong and P. Sarkar, "New applications of time memory data tradeoffs.," in *Advances in Cryptology—ASIACRYPT 2005* (B. Roy, ed.), vol. 3788 of *Lecture Notes in Computer Science*, pp. 353–372, Springer, 2005.

39. M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. IT-26, pp. 401–406, July 1980.
40. M. J. Mihaljevic, S. Gangopadhyay, G. Paul, and H. Imai, "Internal state recovery of Grain-v1 employing normality order of the filter function," *IET Information Security*, vol. 6, no. 2, pp. 55–64, 2012.
41. M. J. Mihaljevic, S. Gangopadhyay, G. Paul, and H. Imai, "Generic cryptographic weakness of k-normal Boolean functions in certain stream ciphers and cryptanalysis of Grain-128," *Periodica Mathematica Hungarica*, vol. 65, no. 2, pp. 205–227, 2012.
42. D. J. Bernstein, "Understanding brute force." eSTREAM, ECRYPT Stream Cipher Project, Report 2005/036, 2005. http://www.ecrypt.eu.org/stream.
43. I. Dinur and A. Shamir, "Cube Attacks on Tweakable Black Box Polynomials," in *Advances in Cryptology—EUROCRYPT 2009* (A. Joux, ed.), vol. 5479 of *Lecture Notes in Computer Science*, pp. 278–299, Springer, 2009.
44. M. Vielhaber, "Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential attack." Available at: http://eprint.iacr.org/2007/413/, 2007.
45. M.-J. O. Saarinen, "Chosen-IV statistical attacks on eStream stream ciphers," *Proc. Stream Ciphers Revisited (SASC'06)*, 2006.
46. P. Stankovski, "Greedy distinguishers and nonrandomness detectors," in *Progress in Cryptology—INDOCRYPT 2010* (G. Gong and K. C. Gupta, eds.), vol. 6498 of *Lecture Notes in Computer Science*, pp. 210–226, Springer, 2010.