Constructing and Deconstructing Intentional Weaknesses in Symmetric Ciphers

Christof Beierle[1], Tim Beyne[2], Patrick Felke[3], and Gregor Leander[1]

[1] Ruhr University Bochum, Bochum, Germany `firstname.lastname@rub.de`
[2] imec-COSIC, KU Leuven, Leuven, Belgium `firsname.lastname@esat.kuleuven.be`
[3] University of Applied Sciences, Emden/Leer, Germany
`patrick.felke@hs-emden-leer.de`

Abstract. Deliberately weakened ciphers are of great interest in political discussion on law enforcement, as in the constantly recurring crypto wars, and have been put in the spotlight of academics by recent progress. A paper at Eurocrypt 2021 showed a strong indication that the security of the widely-deployed stream cipher `GEA-1` was deliberately and secretly weakened to 40 bits in order to fulfill European export restrictions that have been in place in the late 1990s. However, no explanation of how this could have been constructed was given. On the other hand, we have seen the MALICIOUS design framework, published at CRYPTO 2020, that allows to construct tweakable block ciphers with a backdoor, where the difficulty of recovering the backdoor relies on well-understood cryptographic assumptions. The constructed tweakable block cipher however is rather unusual and very different from, say, general-purpose ciphers like the AES.
In this paper, we pick up both topics. For `GEA-1` we thoroughly explain how the weakness was constructed, solving the main open question of the work mentioned above. By generalizing MALICIOUS we – for the first time – construct backdoored tweakable block ciphers that follow modern design principles for general-purpose block ciphers, i.e., more natural-looking deliberately weakened tweakable block ciphers.

Keywords: cryptanalysis · GPRS · GEA-1 · stream cipher · tweakable block cipher · LFSR · malicious · invariant attacks

1 Introduction

The design of deliberate and often hidden weaknesses in (symmetric) cryptographic primitives has a long history, both in practical examples as well as in academic constructions. For the former, among the most famous examples are the block cipher DES [31], for which the key size was deliberately weakened to 56 bits, and the pseudonumber generator Dual EC DRBG, which was equipped with a backdoor (see [27]) by a proper selection of its parameters. We refer to [8] for a detailed survey on the standardization and the weakness of Dual EC DRBG. We also like to mention the Russian cipher GOST (R 34.12-2015), aka Kuznyechik, where the S-box was shown to have undisclosed structures [28]. In [11] a nice argument was given that this structure is indeed very unlikely to appear by chance.

For academic constructions, we have seen approaches based on hiding a highly-biased linear approximation [32, 30] over a block cipher and approaches based on partitioning cryptanalysis [18], where the backdoor consists of a partition of the plaintext space that is preserved under the encryption function [26, 5, 17]. The latter approach is related to invariant subspace attacks [23] and non-linear invariant attacks [35]. In the case of hash functions, the work [1] showed how to design malicious variants of SHA-1 with built-in collisions.

For all of these academic constructions, the designers either do not claim security of the backdoor in the sense that it cannot be recovered even if its general form is known, or there is an attack which recovers the backdoor from the specification of the cipher (see e.g., [38]).

The interest into deliberately weakening symmetric primitives has been increased recently, again with respect to both aspects. On the one hand, the work [7] showed that there is a strong indication that the security of the widely deployed cipher GEA-1 was deliberately and secretly weakened to 40 bits in order to fulfill European export restrictions. On the other hand, we have seen the MALICIOUS design framework [29] that allows to construct a tweakable block cipher with a backdoor. One of the interesting features within this framework is that the difficulty of recovering the backdoor relies on well-understood cryptographic principles.

The MALICIOUS Framework. The authors defined the following four notions a cryptographic backdoor can fulfill, which we directly quote from [29].

– *Undetectability:* this security notion represents the inability for an external entity to realize the existence of the hidden backdoor.
– *Undiscoverability:* it represents the inability for an attacker to find the hidden backdoor, even if the general form of the backdoor is known.
– *Untraceability:* it states that an attack based on the backdoor should not reveal any information about the backdoor itself.
– *Practicability:* this usability notion stipulates that the backdoor is practical, in the sense that it is easy to recover the secret key once the backdoor is known.

The basic idea of the MALICIOUS framework is to construct a tweakable block cipher such that for a particular malicious tweak pair (t, t'), the instance of the cipher for this tweak pair exhibits a differential property that allows for a practical cryptoanalytic attack. The tweak pair (t, t') is secured by being a pair of preimages for outputs of an extendable-output function (XOF) H.

With such a construction, the backdoor fulfills the notions of practicability, undiscoverability, and undetectability, but not untraceability.

The instances given in [29] are based on the block cipher LowMC [2]. The drawback is that the round function needs to be constructed by using a rather complex (basically random) linear layer and a partial S-box layer. As suggested for future work in [29], it would be interesting to find similar constructions which are based on cryptoanalytic attacks other than differential cryptanalysis, as this might lead to more natural instances.

Deliberate Weakness in `GEA-1`. General Packet Radio Service (GPRS) is a mobile data standard based on the GSM (2G) technology, and was widely deployed during the late 1990s and the early 2000s. At the cryptographic level, the data processed by the GPRS protocol is protected by a stream cipher. In 1998, ETSI Security Algorithms Group of Experts (SAGE) initially designed the proprietary 64-bit encryption algorithm `GEA-1` for this purpose. The cipher `GEA-1` is depicted in Figure 2 and consists of three LFSRs with different lengths and a non-linear Boolean function combining their outputs to produce the keystream.

Although classical algebraic attacks on `GEA-1` (e.g., those based on linearization) are hard to conduct in practice because of the limit on the data available to an adversary, in [7] the authors showed that `GEA-1` does not achieve an adequate security level. Indeed, they presented an attack on `GEA-1` with complexity corresponding to a security level of 40 bits. It is based on a simple but remarkable observation: After the linear initialization procedure, the joint state of two of the LFSRs have a joint entropy of only 40 bits, whereas their joint size adds up to 64 bits. This loss of entropy directly leads to a classical meet-in-the-middle attack with time complexity 2^{40}. Recently, in [3], the authors presented an attack on `GEA-1` with the same time complexity but a reduced memory complexity of only 4 MiB (instead of 44.5 GiB).

The authors of [7] further analyzed how frequently this surprising observation occurs for randomly chosen LFSRs. For this, they replaced the (two) LFSRs used in `GEA-1` by primitive LFSRs in Galois mode of the corresponding size chosen uniformly at random and computed the loss of entropy. After roughly one million trials, the maximal loss that was observed was at most 9 bits,[4] demonstrating that this behavior is (i) very rare and thus (ii) most likely built in to keep the ciphers effective strength at 40 bits.

One important question was not answered in [7], namely: How was this configuration of LFSRs constructed? By extrapolating the experimental observations given in [7, Table 2], we estimate the cost of constructing this simply by randomly picking primitive LFSRs to be in the range of roughly 2^{47} trials, summing up to around 2^{65} binary operations in total.[5] Taking into account that the design

[4] When considering all possible combinations of two of the three registers in `GEA-1`, the maximal observed loss was 11 bits.

[5] We estimate the number of expected solutions to be $s \cdot 2^{-2d+1}$, where s denotes the sample space and d the desired entropy loss. For each sample, one has to solve a linear system of dimension 64 to compute the entropy loss.

is already more than 20 years old, the cost of this would have been prohibitive. This strongly indicated that there must be a more elaborated and efficient way of achieving the desired setting.

1.1 Our Contribution

Deconstructing `GEA-1`: Choosing LFSRs with a Hidden Weakness. In the case of `GEA-1` we answer the open question on how to construct a `GEA-1`-like cipher with such a reduced security. Our observations and analysis, relying mainly on the polynomial representation of the involved LFSRs, imply that the actual `GEA-1` instance could have been obtained from our construction.

For this we describe the states, the initialization and the intersection of states by polynomials over $\mathbb{F}_2$. This description allows to formulate the conditions for a set-up that enables an attack as the one on `GEA-1` in terms of divisibility of state polynomials by the characteristic polynomial of the LFSRs. In a second step, we explain a possible construction of pairs of LFSRs with the desired entropy loss. The general idea here is to turn the problem around by starting with elements in the kernel and then searching for suitable LFSRs.

We show by decomposing the kernel of `GEA-1` that it can be easily constructed using our approach. As `GEA-1` is then only one example of the (re?)-discovered design strategy, we elaborate about other possible parameter choices in Section 3.2 and discuss the limits of this approach.

As a side remark, the above mentioned use of LFSRs in Galois mode can now also be justified: A Fibonacci LFSR that is based on a random characteristic polynomial, and thus very likely has many taps, is unfavorable to implement in software and thus unusual. For an LFSR in Galois mode, the choice of a random characteristic polynomial resulting in many taps is desirable.

In contrast to the MALICIOUS framework, the weakness in `GEA-1` is based on *hiding* a cryptoanalytic attack. This approach has the drawback that in principle everybody can recover this attack and can decipher messages encrypted with this system. Thus, this design fails to fulfill three of the four conditions for a backdoor proposed by [29], i.e., undiscoverability, undetectability and untraceability.

Constructing Backdoors: Trivial and Natural Variants of MALICIOUS. As described above, the original MALICIOUS framework was formulated in terms of differential cryptanalysis only. For our results, we use a naturally generalized version of the framework.

Our contribution is twofold. First, we show that *any* (tweakable) block cipher can be tweaked in a very simple way in order to comply with the MALICIOUS framework. In a nutshell, the idea of constructing such an instance is to check if the message hashes to a certain fixed value and if so, return the key instead of the encrypted message. If the hash does not match, the cipher is executed unchanged.

While this example shows that the initial goals of the MALICIOUS framework can be achieved in a trivial way, this artificial construction does not give further insights on how to construct *hidden* weaknesses. Ideally, a malicious designer would aim for constructing a rather *natural* instance which follows modern design principles in symmetric cryptography and for which a sound design rationale can be formulated. Towards achieving this goal, we propose two new instances of the MALICIOUS framework as our second contribution. While the instances presented in [29] rely on the existence of a high-probability differential, we base our construction on an invariant subspace, resp., nonlinear invariant, for the malicious tweak value. As we argue, this allows for more natural instances. In particular, in Section 5, we show how to use the round function of the Advanced Encryption Standard (AES) and to adapt the key schedule in order to embed a backdoor based on an *invariant subspace* over the round function. More precisely, we exploit an invariant subspace that is already known since 2004, see [22]. In Section 6, we construct a dedicated cipher, called Boomslang, that embeds a backdoor based on a *nonlinear invariant* over two consecutive round functions.

Compared to previous constructions not based on the MALICIOUS framework, in particular [26] and [30], our constructions also directly improve upon the usability of the backdoor as they enable significantly more practical key-recovery attacks.

Our constructions constitute the first backdoored ciphers that follow modern design principles of general-purpose block ciphers, and are expected to achieve competitive performance characteristics.

Ethical Aspect of Our Research. We (the authors) do not have the goal to support people in building intentionally weakened ciphers. A first step in order to prevent the use of intentionally weakened designs in the future is to investigate the design space of such constructions. Only when knowing what potential attackers, in this case acting as malicious developers of cryptographic primitives, are capable of doing, we are able to prevent them from doing so.

2 GEA-1 and its Cryptoanalytic Properties

In this section we recall the description of the stream cipher GEA-1 as well as its weakness, both as presented in [7]. Before doing so, we define and recall the basic mathematics behind the cipher.

2.1 Preliminaries

As usual, a matrix $A \in \mathbb{F}_2^{m \times n}$ corresponds to a linear map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^m$ by matrix-vector multiplication from the right. Thus the i^{th} column of A is the image of $e_i := (0, 0, \dots, 1, 0 \dots, 0) \in \mathbb{F}_2^n$, i.e., the i^{th} canonical unit vector.

Galois mode LFSRs. We recall some basic facts about linear feedback shift registers (LFSRs) in Galois mode, as depicted in Figure 1. For further reading we refer to [34, pp. 378 ff.] and [19, p. 227].

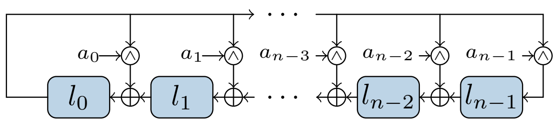

Fig. 1: An LFSR in Galois mode.

Given an LFSR L in Galois mode of degree n with entries in $\mathbb{F}_2$, clocking the state $l = (l_0, \ldots, l_{n-1})$ is equivalent to the matrix-vector multiplication

$$G_L \cdot l = \begin{bmatrix} a_0 & 1\,0 \cdots 0 \\ a_1 & 0\,1 \cdots 0 \\ \vdots & \vdots\,\vdots\,\ddots\,\vdots \\ a_{n-2} & 0\,0 \cdots 1 \\ a_{n-1} & 0\,0 \cdots 0 \end{bmatrix} \begin{bmatrix} l_0 \\ l_1 \\ \vdots \\ l_{n-2} \\ l_{n-1} \end{bmatrix} = \begin{bmatrix} a_0 l_0 + l_1 \\ a_1 l_0 + l_2 \\ \vdots \\ a_{n-2} l_0 + l_{n-1} \\ a_{n-1} l_0 \end{bmatrix} .$$

The characteristic polynomial of G_L is

$$g \coloneqq \det(X I_n + G_L) = X^n + a_0 X^{n-1} + \cdots + a_{n-2} X + a_{n-1} \in \mathbb{F}_2[X]$$

and is a multiple of the minimal polynomial of G_L. Here, I_n denotes the $n \times n$ identity matrix. Although LFSRs in Galois mode can be defined more generally, we only consider the case where g is primitive. That is, g is the minimal polynomial of an element $\alpha \in \mathbb{F}_{2^n}$ that generates the multiplicative group $\mathbb{F}_{2^n}^\times$. In this case, g is also the minimal polynomial of G_L. Only primitive polynomials are of cryptographic interest as they correspond to LFSRs with a maximal period of $2^n - 1$. Since primitive polynomials are necessarily irreducible, we have $a_{n-1} \neq 0$, which is equivalent to the fact that G_L is invertible. Conversely, any primitive polynomial g corresponds to an (invertible) companion matrix of an LFSR in Galois mode with minimal polynomial g.

Galois matrices. In the sequel, the matrix G_L will be called a *Galois matrix* of degree n and the corresponding minimal polynomial the *Galois polynomial.* Moreover, given an LFSR L in Galois mode with minimal polynomial g, we also denote the Galois matrix by G_g if appropriate.

To describe our construction, we need the following properties of Galois matrices. These are well-known facts from classical ring or field theory respectively (e.g., see [24, 36]). To keep the paper self-contained and to enhance readability we include a proof below.

Theorem 1. *Given a Galois matrix G_g of degree n with primitive Galois polynomial g, let $\mathbb{F}_2[G_g] := \{\sum_{i=0}^m t_i\, G_g^i \mid m \in \mathbb{N}, t \in \mathbb{F}_2^{m+1}\}$ be the subring of $\mathbb{F}_2^{n\times n}$ generated by G_g and let (g) denote the ideal generated by $g \in \mathbb{F}_2[X]$. Then the following statements are true.*

1. *The map $\psi : \mathbb{F}_2[X]/(g) \to \mathbb{F}_2[G_g]$ defined by $\psi(\sum_{i=0}^m t_i X^i) = \sum_{i=0}^m t_i G_g^i$ is a ring isomorphism.*[6]

[6] Note that we use $\sum_{i=0}^m t_i X^i$ as a shorthand for the corresponding coset of $\mathbb{F}_2[X]$.

2. *Every nonzero element $v \in \mathbb{F}_2^n$ is G_g-cyclic, i.e., $\{v, G_g v, \ldots, G_g^{n-1} v\}$ is a basis for $\mathbb{F}_2^n$.*

Proof. By the definition of minimal polynomials, the set of all polynomials p with $p(M) = 0$ for a matrix M with minimal polynomial q is equal to the ideal (q). Since (q) is a maximal ideal, $\mathbb{F}_2[X]/(q)$ is a finite field of degree n over $\mathbb{F}_2$. As remarked above, g is primitive and the minimal polynomial of G_g. Thus, the canonical map $\phi : \mathbb{F}_2[X] \longrightarrow \mathbb{F}_2[G_g] : p \mapsto p(G_g)$ has kernel (g) and hence ψ is an isomorphism by the first isomorphism theorem.

For the second claim, suppose the vectors $v, G_g v, \ldots, G_g^{n-1} v$ are linearly dependent. Then there exist $t_0, \ldots, t_{n-1} \in \mathbb{F}_2$ such that not all t_i equal zero and

$$\textstyle\sum_{i=0}^{n-1} t_i\, G_g^i\, v = \big(\sum_{i=0}^{n-1} t_i\, G_g^i\big) v = 0 \; .$$

By applying the isomorphism ψ^{-1}, we get that $0 \neq \sum_{i=0}^{n-1} t_i X^i \in \mathbb{F}_2[X]/(g)$ as the degree of the polynomial $\sum_{i=0}^{n-1} t_i X^i$ is at most $n-1$ and not all t_i are equal to zero. As $\mathbb{F}_2[X]/(g)$ is a finite field, any nonzero element such as $\sum_{i=0}^{n-1} t_i X^i$ is invertible. Since any isomorphism maps invertible elements to invertible elements, it follows that $\sum_{i=0}^{n-1} t_i G_g^i$ is invertible. Hence $\left(\sum_{i=0}^{n-1} t_i G_g^i\right) v \neq 0$ as $v \neq 0$. This is a contradiction and therefore, v is G_g-cyclic. □

Remark 1. Note that $\mathbb{F}_2[X]/(g)$ is a field. The matrix G_g is the representation matrix of the linear mapping $p \mapsto Xp$ over the finite field $\mathbb{F}_2[X]/(g)$ with respect to the basis $X^{n-1}, X^{n-2}, \ldots, 1$. This connection to a central mapping in the theory of finite fields (also called Galois fields), the so-called left multiplication, leads to the name for these kind of LFSRs.

The following corollary is extensively used in the remainder of this paper. The proof is a straightforward application of Theorem 1 and therefore omitted.

Corollary 1. *Let e_0 denote the vector $e_0 := (1, 0, \ldots, 0) \in \mathbb{F}_2^n$ and G_g a Galois matrix of degree n for a primitive polynomial g. Then, for $m \geq 0, t_i \in \mathbb{F}_2, i = 0, \ldots, m$, we have $\sum_{i=0}^{m} t_i G_g^i e_0 = 0$ if and only if g divides $\sum_{i=0}^{m} t_i X^i$ in $\mathbb{F}_2[X]$.*

2.2 Description of `GEA-1`

We now turn to the description of `GEA-1`, and in particular the mechanism used to initialize the LFSR registers.

Keystream generation. The keystream is generated from three LFSRs over $\mathbb{F}_2$, called A, B and C, together with a 7-bit non-linear filter function f. The registers A, B and C have lengths $31, 32$ and 33, respectively and the LFSRs work in Galois mode. In particular, the Galois polynomials corresponding to LFSRs A,

B and C are

$$\begin{aligned}
g_A &= X^{31} + X^{30} + X^{28} + X^{27} + X^{23} + X^{22} + X^{21} + X^{19} + X^{18} + X^{15} \\
&\quad + X^{11} + X^{10} + X^{8} + X^{7} + X^{6} + X^{4} + X^{3} + X^{2} + 1 , \\
g_B &= X^{32} + X^{31} + X^{29} + X^{25} + X^{19} + X^{18} + X^{17} + X^{16} + X^{9} + X^{8} \\
&\quad + X^{7} + X^{3} + X^{2} + X + 1 , \\
g_C &= X^{33} + X^{30} + X^{27} + X^{23} + X^{21} + X^{20} + X^{19} + X^{18} + X^{17} + X^{15} \\
&\quad + X^{14} + X^{11} + X^{10} + X^{9} + X^{4} + X^{2} + 1 ,
\end{aligned}$$

respectively. The function f belongs to a class of cryptographically strong Boolean functions that can be decomposed into two bent functions on 6 bits. For the considerations below, the choice of f is irrelevant and we omit it here.

When all registers have been initialized (see below), the actual keystream generation starts. This is done by taking the bits at seven specified positions in each register to be the input to f. The outputs of the three f-functions are XORed together to produce one bit of the keystream. Figure 2 shows the particular feedback positions of each register, as well as which positions form which input to f. After calculating a single keystream bit, all registers are clocked once each before the process is repeated to generate the next bit.

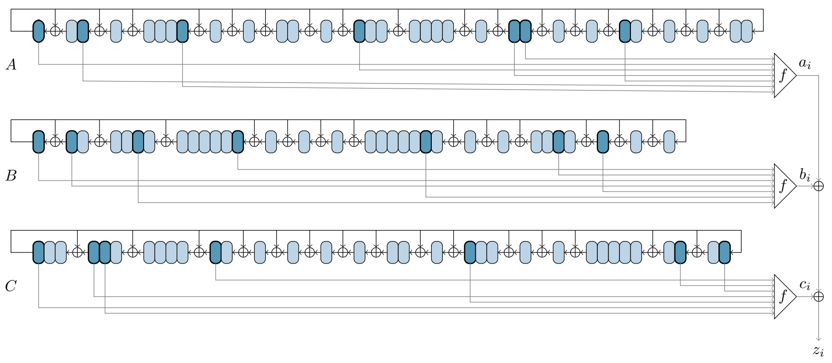

Fig. 2: Overview of the keystream generation of `GEA-1` [7].

Initialization. The cipher is initialized using a non-linear feedback shift register S of length 64. This register is filled with zeros at the start of the initialization process. The input for initializing `GEA-1` consists of a public 32-bit initialization vector iv, one public bit dir (indicating direction of communication/uplink or downlink in a cellular network), and a 64-bit secret key k. The initialization starts by clocking S for 97 times, feeding in one input bit with every clock. The input bits are introduced in the order $\mathsf{iv}_0, \mathsf{iv}_1, \ldots, \mathsf{iv}_{31}, \mathsf{dir}, k_0, k_1, \ldots, k_{63}$. When all input bits have been loaded, the register is clocked another 128 times with zeros as input. The feedback function consists of f, XORed with the bit that is shifted out, and XORed with the next bit from the input sequence.

After S has been clocked 225 times, the content of the register is taken as a 64-bit vector $s = (s_0, \ldots, s_{63})$. This string is taken as a seed for initializing A, B and C as follows. First, all three registers are initialized with zeros. Then, each register is clocked 64 times, with an s_i-bit XORed onto the bit that is shifted out before feedback. Register A inserts the bits from s in the natural order $s_0, s_1, ..., s_{63}$. The sequence s is cyclically shifted by 16 positions before being inserted to register B, so the bits are entered in the order $s_{16}, s_{17}, \ldots, s_{63}, s_0, \ldots, s_{15}$. For register C the sequence s is cyclically shifted by 32 positions before insertion starts. Figure 3 depicts the process for register B. If any of the registers A, B or C end up in the all-zero state, the bit in position zero of the register is forcibly set to one before keystream generation starts.

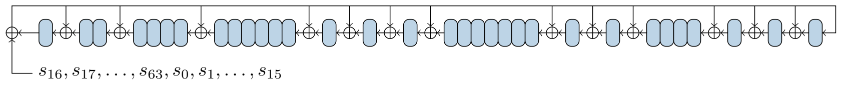

Fig. 3: Initialization of register B [7].

As already observed in [7], if we exclude the unlikely case that any of the three registers A, B or C is still in the all-zero state after the shifted insertion of s, the initialization process of the three registers with the string s is obviously linear and therefore there exist three matrices $M_A \in \mathbb{F}_2^{31\times 64}$, $M_B \in \mathbb{F}_2^{32\times 64}$ and $M_C \in \mathbb{F}_2^{33\times 64}$ such that $\alpha = M_A s, \beta = M_B s$, and $\gamma = M_C s$, where α, β and γ denote the states of the three LFSRs after the initialization phase.

2.3 The Attack on `GEA-1`

Let us consider the initialization matrices $M_A \in \mathbb{F}_2^{31\times 64}$, $M_B \in \mathbb{F}_2^{32\times 64}$ and $M_C \in \mathbb{F}_2^{33\times 64}$ such that $\alpha = M_A s, \beta = M_B s$, and $\gamma = M_C s$. We exclude here the unlikely case that α, β or γ is still in the all-zero state after the shifted insertion of s. These three matrices have full rank. This implies that the number of possible starting states after initialization is maximal when each LFSR is considered independently, i.e., there are $2^{31} - 1$ possible states for register A, $2^{32} - 1$ possible states for register B, and $2^{33} - 1$ possible states for register C, as should be expected. This corresponds to the linear mappings represented by M_A, M_B and M_C having kernels of dimension of at least 33, 32 and 31, respectively. However, when considering pairs of registers, one gets a decomposition of $\mathbb{F}_2^{64}$ as a direct sum of the kernels of the linear mappings. In [7] it was observed that if one denotes $T_{A,C} \coloneqq \ker(M_A) \cap \ker(M_C)$ and $U_B \coloneqq \ker(M_B)$, then

$$\dim(T_{A,C}) = 24, \quad \dim(U_B) = 32, \text{ and } \quad U_B \cap T_{A,C} = \{0\} .$$

From this, it directly follows that $\mathbb{F}_2^{64}$ can be decomposed into the direct sum $U_B \oplus T_{A,C} \oplus V$, where V is of dimension 8. Thus, for the key-dependent and secret string s, there exists a *unique* representation $s = u + t + v$ with $u \in U_B$, $t \in T_{A,C}$, $v \in V$ and

$$\begin{aligned}\beta &= M_B(u + t + v) = M_B(t + v)\\ \alpha &= M_A(u + t + v) = M_A(u + v)\\ \gamma &= M_C(u + t + v) = M_C(u + v)\ .\end{aligned}$$

Indeed, from this decomposition, s can be computed with a meet-in-the-middle attack with a complexity[7] of 2^{37} `GEA-1` evaluations to build (and sort) a table with 2^{32} entries of size 89 bit (65 keystream bits to reconstruct the key uniquely with high probability and 24 bits for t leading to this keystream) and a brute-force step of complexity 2^{40} `GEA-1` evaluations for each new session key $k_0, \dots, k_{63}$. For more details of the attack see [7]. Note that once s is recovered it is easy to recover $k_0, \dots, k_{63}$ by clocking the S-register backwards. Hence, the attack has to be conducted only *once per GPRS session* and is done in 2^{40} operations once the table has been computed. In other words, the joint state of A and C can be described with only 40 bits and thus can take only 2^{40} possible values. This is the key observation of the attack and in [7] computer simulations are used to argue that such a decomposition of the key space is highly unlikely to occur accidentally. The main question arising in this context is how to design such a system. As demonstrated by the experiments conducted in [7], a trial and error approach is elusive. This question will be answered in the next section.

3 Deconstructing `GEA-1`: Shifting Matters

In this section we will give a method to build ciphers of `GEA-1` type which are vulnerable to the attack described above and thereby answer the corresponding question in [7].

It will become apparent (without giving a rigorous proof) that systems of `GEA-1` type with a keyspace that can be decomposed into a direct sum as above only appear for *very special choices* of the shift constants together with *very special choices* for the Galois polynomials. This is remarkable because one could intuitively expect that shifting only strengthens the system.

In general, this demonstrates that it is not recommended to modify the canonical way of feeding the key into the LFSRs. Indeed, only by modifying this initialization and by using the shifted key for the individual LFSRs, the attack becomes possible.

3.1 The Impact of Shifting

We first settle the question how to find two primitive Galois polynomials $g_1 \in \mathbb{F}_2[X]$ and $g_2 \in \mathbb{F}_2[X]$ of degree d_1 and d_2, respectively, such that for the corresponding Galois matrices G_{g_1} and G_{g_2}, the dimension of $T_{g_1,g_2,\mathsf{cs}} := \ker(M_{g_1}) \cap$

[7] The complexity is measured by the amount of operations that are roughly as complex as `GEA-1` evaluations (for generating a keystream of size ≤ 128 bit).

$\ker(M_{g_2,\mathsf{cs}})$ is at least ξ with $\mathsf{cs} \in \{0,1,\ldots,\kappa-1\}$ being the cyclic shift employed during the initialization and κ being the size of the session key (64 in case of `GEA-1`). Without loss of generality we focus on this case. It is a routine matter to extend the approach presented in the sequel to the case where in both initialization phases a shift is applied. First of all, note that the columns of M_{g_1} (the initialization matrix without a shift) consist of $G_{g_1}^{\kappa-i}e_0$, where $e_0 := (1,0,\ldots,0) \in \mathbb{F}_2^{d_1}$ and $i = 0,\ldots,\kappa-1$. Note that we have to clock the register i times before the first bit of s_i not equal to zero is plugged into the register and thus the state becomes non-zero. This explains the shape of $M_{G_{g_1}}$. Suppose $t \in \ker(G_{g_1})$, then

$$0 = M_{g_1} t = \sum_{i=0}^{\kappa-1} t_i\, G_{g_i}^{\kappa-i}\, e_0\,.$$

By Corollary 1, the above holds if and only if g_1 divides $\sum_{i=0}^{\kappa-1} t_i\, X^{\kappa-i}$. In the case of g_2, a similar reasoning takes into account the effect of cs such that the columns of $M_{g_2,\mathsf{cs}}$ (the initialization matrix with a shift) consist of $G_{g_2}^{\mathsf{cs}-i}\, e_0$ if $i < \mathsf{cs}$ and $G_{g_2}^{\kappa-i+\mathsf{cs}}\, e_0$ otherwise, where $e_0 := (1,0,\ldots,0)^\top \in \mathbb{F}_2^{d_2}$. Note that now we have to clock the register $\kappa + i - \mathsf{cs}$ times before the first non-zero bit of s_i is plugged into the register and thus the state is non-zero. This gives the first case. The second follows in the same way. In the same vein as above we get that

$$0 = M_{g_2,\mathsf{cs}}\, t = \sum_{i=0}^{\mathsf{cs}-1} t_i\, G_{g_2}^{\mathsf{cs}-i} e_0 + \sum_{i=\mathsf{cs}}^{\kappa-1} t_i\, G_{g_2}^{\kappa-i+\mathsf{cs}}\, e_0$$

if and only if g_2 divides $\sum_{i=0}^{\mathsf{cs}-1} t_i X^{\mathsf{cs}-i} + \sum_{i=\mathsf{cs}}^{\kappa-1} t_i\, X^{\kappa-i+\mathsf{cs}}$. Hence, a vector $t \in \mathbb{F}_2^\kappa$ lies in $T_{g_1,g_2,\mathsf{cs}}$ if and only if g_1 divides $\sum_{i=0}^{\kappa-1} t_i\, X^{\kappa-i}$ and g_2 divides $\sum_{i=0}^{\mathsf{cs}-1} t_i\, X^{\mathsf{cs}-i}$ $+\sum_{i=\mathsf{cs}}^{\kappa-1} t_i\, X^{\kappa-i+\mathsf{cs}}$. More specifically, we have the following theorem which shows how to control the dimension of $T_{g_1,g_2,\mathsf{cs}}$.

Theorem 2. *Let $g_1, g_2 \in \mathbb{F}_2[X]$ be two primitive Galois polynomials and $\mathsf{cs} \in \{0,1,\ldots,\kappa-1\}$ an integer. For $t \in T_{g_1,g_2,\mathsf{cs}}$ we define the associated polynomials*

$$p_1 := \sum_{i=0}^{\kappa-1} t_i\, X^{\kappa-i}, \quad p_2 := \sum_{i=0}^{\mathsf{cs}-1} t_i\, X^{\mathsf{cs}-i} + \sum_{i=\mathsf{cs}}^{\kappa-1} t_i\, X^{\kappa-i+\mathsf{cs}}\,.$$

Let

$$\begin{aligned} r_1 &= \min\{k : X^k \text{ is a monomial with non-zero coefficient in } p_1\}, \\ r_2 &= \min\{k : X^k \text{ is a monomial with non-zero coefficient in } p_2\}, \\ r_3 &= \min\{r_1, r_2\} \end{aligned}$$

Then

1. *The polynomial p_1 is divisible by g_1 and the polynomial p_2 is divisible by g_2.*
2. *For all $0 \le s \le r_3-1$, the shifted vectors $t^s := (0,\ldots,0,t_0,t_1,\ldots,t_{\kappa-1-s})^\top \in \mathbb{F}_2^\kappa$ are elements of $T_{g_1,g_2,\mathsf{cs}}$.*

3. *The elements t^s are linearly independent and thus span a subspace of $T_{g_1,g_2,\mathsf{cs}}$ of dimension r_3.*

Proof. The first property was already established above. By definition, it holds that $t_{\kappa-1-(r_1-1)} = 1$, as it is the coefficient of X^{r_1} in p_1. We further have $t_{\kappa-1-(r_1-1)+1} = 0$, $t_{\kappa-1-(r_1-1)+2} = 0$, ..., $t_{\kappa-1} = 0$ from the definition of r_1. Hence, the elements t^s are linearly independent.

Since $t \in T_{g_1,g_2,\mathsf{cs}}$ we have that g_1 divides p_1 and g_2 divides p_2. By definition of r_3, the associated polynomials of t^s are of the form $X^{-s}p_1, X^{-s}p_2$ and still contained in $\mathbb{F}_2[X]$. Therefore they are also divisible by g_1 and g_2. Thus the elements t^s form a subspace of dimension r_3 of $T_{g_1,g_2,\mathsf{cs}}$. □

Remark 2. Note that slightly more can be said about the structure of good choices for g_1 and g_2 and the corresponding space $T_{g_1,g_2,\mathsf{cs}}$. For example looking at the reciprocal polynomials of g_1 and g_2 results in the same dimension for the kernel.

For this let κ be even and $\mathsf{cs} = \kappa/2$. If $t \in T_{g_1,g_2,\mathsf{cs}}$ for two primitive polynomials $g_1, g_2 \in \mathbb{F}_2[X]$, we have $t^* = (t_{\kappa-1}, \ldots, t_1, t_0) \in T_{g_1^*,g_2^*,\mathsf{cs}}$, where $g_i^*(X) := X^{\deg g_i} g_i(X^{-1})$ denotes the *reciprocal polynomial* of g_i. This can be seen as follows. For the polynomials p_1 and p_2 defined in Theorem 2, we let $q_i(X) := X^\kappa p_i(X^{-1})$. Then,

$$Xq_1 = \sum_{i=0}^{\kappa-1} t_i^* X^{\kappa-i}, \quad Xq_2 = \sum_{i=0}^{\mathsf{cs}-1} t_i^* X^{\mathsf{cs}-i} + \sum_{i=\mathsf{cs}}^{\kappa-1} t_i^* X^{\kappa-i+\mathsf{cs}}$$

and $t^* \in T_{g_1^*,g_2^*,\mathsf{cs}}$ if g_i^* divides Xq_i for $i \in \{1,2\}$. Since $q_i = X^{\kappa-\deg p_i} p_i^*$, this happens if g_i^* divides p_i^* for $i \in \{1,2\}$. By assumption, $t \in T_{g_1,g_2,\mathsf{cs}}$, so g_i divides p_i for $i \in \{1,2\}$, and therefore g_i^* also divides p_i^* for $i \in \{1,2\}$. Moreover g_i is primitive if and only if g_i^* is primitive.

3.2 Constructing the Galois Polynomials

The principle to construct systems vulnerable to the attack described in Section 2.3 is now fairly simple. We start with an element in the (potential) kernel, that would imply the desired dimension by the above theorem. Then, we construct the two polynomials p_1 and p_2 and check if they are divisible by primitive polynomials of the desired degree. We explain this in more detail below, first for the parameters used in `GEA-1` and then for the general case.

The case of `GEA-1`. We give an example for the case of $\kappa = 64, \mathsf{cs} = 32, \xi = 24$, and primitive polynomials g_1, g_2 of degree $d_1 = 31, d_2 = 33$. Those parameters correspond exactly to the case of `GEA-1`. Other parameter choices are discussed below.

First of all we will construct an element $t = (t_0, \ldots, t_{63})^\top$ of the form such that applying Theorem 2 yields $t \in T_{g_1,g_2,\mathsf{cs}}$, where $T_{g_1,g_2,\mathsf{cs}}$ is of dimension 24.

For this, let us fix t such that $t_i = 0$ for $i \in \{9, 10, \dots, 31\} \cup \{41, \dots, 63\}$ and $t_0 = t_{40} = 1$. We consider the polynomials

$$p_1 \coloneqq X^{64} + \sum_{i=1}^{8} t_i X^{64-i} + \sum_{i=32}^{39} t_i X^{64-i} + X^{24} \in \mathbb{F}_2[X]$$
$$p_2 \coloneqq X^{32} + \sum_{i=1}^{8} t_i X^{32-i} + \sum_{i=32}^{39} t_i X^{64-i+32} + X^{56} \in \mathbb{F}_2[X]$$

to guarantee a kernel of dimension at least 24 if there exist proper g_1, g_2 of degree 31,33 such that $t \in T_{g_1,g_2,\mathsf{cs}}$. In the positive case the lower bound for the dimension (here 24) is a direct consequence of Theorem 2. We have 2^{16} possibilities to fix such an element t, i.e., to choose the above pair of polynomials p_1, p_2. We choose such an element t uniformly at random and check if p_1 is divisible by a primitive polynomial g_1 of degree 31 and if p_2 is divisible by a primitive polynomial g_2 of degree 33.

It is well known that the number of primitive elements of a finite field of q^n elements, where q is a prime number, is $\varphi(q^n - 1)$ (see e.g., [21, p. 56]). Here, φ denotes Euler's totient function. Hence the number of primitive polynomials in our case is $\varphi(2^{31} - 1)/31$ and $\varphi(2^{33} - 1)/33$, because the 31 (resp., 33) roots of a primitive polynomial of degree 31 (resp., 33) are all primitive. By construction $p_1 = X^{24} \cdot h_1$, where $h_1 \in \mathbb{F}_2[X]$ with $\deg(h_1(X)) = 40$, independently of the choice of the t_i. Analogously, $p_2 = X^{24} \cdot h_2$, where $h_2 \in \mathbb{F}_2[X]$ with $\deg(h_2(X)) \leq 40$. The overall number of polynomials of degree 40 having a primitive divisor of degree 31 is $2^9 \cdot \varphi(2^{31} - 1)/31$ and similarly $2^8 \cdot \varphi(2^{33} - 1)/33$ for polynomials of degree at most 40 with a primitive divisor of degree 33. Therefore, under an independence assumption, we expect the probability that both h_1 has a primitive divisor of degree 31 and h_2 has a primitive divisor of degree 33 to be

$$\frac{\varphi(2^{31} - 1)}{31 \cdot 2^{31}} \, \frac{\varphi(2^{33} - 1)}{33 \cdot 2^{33}} \approx \frac{1}{1250} \, .$$

As we have 2^{16} possibilities to vary p_1 and p_2, we expect to be successful to find the sought for polynomials g_1 and g_2 with $t \in T_{g_1,g_2,\mathsf{cs}}$.

Indeed, the primitive polynomials g_A and g_C used in `GEA-1` are exactly of this form. More precisely, the element t satisfying $(t_0, \dots, t_8) = (1, 0, 1, 1, 0, 0, 1, 1, 1)$, $(t_{32}, \dots, t_{40}) = (0, 0, 1, 1, 1, 1, 0, 0, 1)$, $t_i = 0$ for $i \in \{9, 10, \dots, 31\} \cup \{41, \dots, 63\}$ is contained in $T_{A,C}$. The corresponding polynomial p_1 is divided by

$$\begin{aligned} g_A = {} & X^{31} + X^{30} + X^{28} + X^{27} + X^{23} + X^{22} + X^{21} + X^{19} + X^{18} + X^{15} \\ & + X^{11} + X^{10} + X^{8} + X^{7} + X^{6} + X^{4} + X^{3} + X^{2} + 1 \in \mathbb{F}_2[X] \end{aligned}$$

and the corresponding polynomial p_2 is divisible by

$$\begin{aligned} g_C = {} & X^{33} + X^{30} + X^{27} + X^{23} + X^{21} + X^{20} + X^{19} + X^{18} + X^{17} + X^{15} \\ & + X^{14} + X^{11} + X^{10} + X^{9} + X^{4} + X^{2} + 1 \in \mathbb{F}_2[X] \, . \end{aligned}$$

As expected by Theorem 2, the 24-dimensional linear space $T_{A,C}$ is spanned by the shifted elements $t^s = (0, \dots, 0, t_0, t_1, \dots, t_{63-s})^\top \in \mathbb{F}_2^{64}$, $0 \leq s \leq 23$.

From the 2^{16} possibilities to choose t, except from the example given above, also 47 other choices yield primitive divisors of p_1 and p_2 with degree 31 and 33, respectively. Note that once we have been successful in finding the primitive polynomials g_1 and g_2, we could choose a primitive polynomial g_3 of degree 32 and check if $U_{g_3} \cap T_{g_1,g_2,\mathsf{cs}} = \{0\}$ in order to construct a stream cipher similar to `GEA-1`. In Appendix A, we provide a sage [33] code that allows to construct such weak `GEA-1`-like instances based on this construction. By the algorithm above, it is possible to find a shift `cs` and corresponding polynomials such that the resulting system can be broken with the attack described in Section 2.3.

Moreover, we conducted slightly more general experiments. Our results (given by the case of $\ell_1 = 31$ in Table 3 in Appendix B) imply that it would have been possible to design the two LFSRs A and C of `GEA-1` such that they yield a kernel intersection of dimension 26, reducing the security of `GEA-1` from 40 to 38 bits. Interestingly, the designers decided not to do so, which suggests that they were aiming at 40 bits security exactly.

The general case. We now focus on the case of an arbitrary (even) key length κ and aim to construct two LFSRs of size ℓ_1 and ℓ_2 such that the kernel has a dimension of (at least) ξ. In order to simplify the discussion and the notation, we focus on the case of $\mathsf{cs} = \kappa/2$. The case of other shift values can be handled in a similar way as long as $\mathsf{cs} \geq \xi$.

In order to construct the two LFSRs, i.e., the corresponding primitive polynomials of degree ℓ_1 and ℓ_2, we start again by an element in the kernel that, due to Theorem 2, guarantees a kernel intersection of dimension at least ξ. That is, we consider a vector $t \in \mathbb{F}_2^\kappa$ such that

$$t_i = 0 \text{ for } i \in \left\{\frac{\kappa}{2} - \xi + 1, \dots, \frac{\kappa}{2} - 1\right\} \cup \{\kappa - \xi + 1, \dots, \kappa - 1\}$$

and $t_0 = t_{\kappa-\xi} = 1$. To this choice of t, we get the corresponding polynomials

$$p_1 \coloneqq X^\kappa + \sum_{i=1}^{\frac{\kappa}{2}-\xi} t_i X^{\kappa-i} + \sum_{i=\frac{\kappa}{2}}^{\kappa-\xi-1} t_i X^{\kappa-i} + X^\xi \in \mathbb{F}_2[X] \tag{1}$$

$$p_2 \coloneqq X^{\frac{\kappa}{2}} + \sum_{i=1}^{\frac{\kappa}{2}-\xi} t_i X^{\frac{\kappa}{2}-i} + \sum_{i=\frac{\kappa}{2}}^{\kappa-\xi-1} t_i X^{\kappa-i+\frac{\kappa}{2}} + X^{\xi+\frac{\kappa}{2}} \in \mathbb{F}_2[X]\ . \tag{2}$$

The number of vectors t and thus the number of pairs of polynomials (p_1, p_2) we can construct this way is $N = 2^{\kappa-2\xi}$. To successfully construct the LFSRs with a kernel intersection of dimension at least ξ, we require that p_1 is divisible by a primitive polynomial of degree ℓ_1 and p_2 is divisible by a primitive polynomial of degree ℓ_2. To analyze the successes probability, we use as before a heuristic approach. More precisely, we assume that (p_1, p_2) behaves as a uniformly and independently chosen pair of polynomials (of degree κ and less than or equal to

κ respectively) with respect to their probability of being divisible by primitive polynomials of the desired degree.

The number of polynomials of degree n with a primitive divisor of degree ℓ is, analogously as above, given by

$$P_{n,\ell} := 2^{n-\ell}\,\frac{\varphi(2^\ell - 1)}{\ell}\ .$$

In addition, the probability that a uniform random polynomial of degree n is divisible by a primitive divisor of degree ℓ is

$$\Psi_\ell := \frac{P_{n,\ell}}{2^n} = \frac{\varphi(2^\ell - 1)}{\ell 2^\ell}\ .$$

Note that Ψ_ℓ is also the probability of a polynomial of degree *less than or equal to* n to be divisible by a primitive divisor of degree ℓ, as both nominator and denominator are multiplied by a factor of two in this case.

While computing lower bounds on Euler's totient function is non trivial, for our purpose it is sufficient, easier, and more precise to compute $\varphi(2^\ell - 1)$ for practical relevant values of ℓ. For $\ell \leq 512$ we computed explicitly that

$$\frac{2^\ell}{\varphi(2^\ell - 1)} \leq 3.4\ , \tag{3}$$

using a computer program.

Following the above heuristic on the random behavior of p_1 and p_2 we get that the probability for a successful construction for one fixed t is given by

$$\Psi_{\ell_1}\Psi_{\ell_2} = \frac{\varphi(2^{\ell_1} - 1)}{\ell_1 \cdot 2^{\ell_1}}\,\frac{\varphi(2^{\ell_2} - 1)}{\ell_2 \cdot 2^{\ell_2}}.$$

From Equation (3), the expected number of trials until suitable polynomials are found can be bounded by

$$\left(\Psi_{\ell_1}\Psi_{\ell_2}\right)^{-1} \leq 12\ell_1\ell_2$$

for $\ell_i \leq 512, i \in \{1,2\}$. This shows that the approach is easily feasible for all practical relevant choices of ℓ_1 and ℓ_2 and can be expected to find valid solutions as long as the number of candidates N is larger than the expected number of trials. Note that for concrete parameters with a large ξ, ℓ_1, ℓ_2, it is better to check if $\left(\Psi_{\ell_1}\Psi_{\ell_2}\right)^{-1} \leq N$ as N becomes relatively small and $12\ell_1\ell_2$ significantly larger than $\left(\Psi_{\ell_1}\Psi_{\ell_2}\right)^{-1}$.

Applicability to longer keys. We recall that in the attack on `GEA-1`, the keyspace $\mathbb{F}_2^{64}$ was decomposed into the direct sum $U_B \oplus T_{A,C} \oplus V$ such that $\beta = M_B(u + t + v) = M_B(t + v)$, $\alpha = M_A(u + t + v) = M_A(u + v)$, and $\gamma = M_C(u + t + v) = M_C(u + v)$, where $\dim(U_B \oplus V) = 40$ and $\dim(T_{A,C} \oplus V) = 32$. In general, if the key size is κ, a straightforward divide-and-conquer attack could be applied by either building a table of size at least $2^{\dim(T_{A,C} \oplus V)}$ bitstrings and conducting exhaustive search of complexity $2^{\dim(U_B \oplus V)}$ cipher evaluations or vice versa.

Let us now discuss whether it is possible to build weak `GEA-1`-like instances operating on a larger keyspace. For this, we restrict to the case of extending the lengths of the three involved LFSRs and do not consider extending the number of LFSRs. The reason is that the attack requires two steps; (1) building a table of size exponentially in the dimension of the kernel intersection, (2) an exhaustive search of complexity exponentially in the state size of the remaining register(s). Since our construction yielding a large kernel intersection only works for two LFSRs, adding more LFSRs would increase the complexity of the second step.

For $\kappa = 96$, it is possible to choose primitive polynomials g_A and g_C of degree 47 and 49, respectively, such that for the corresponding LFSRs A and C we have $\dim T_{A,C} = 44$ (where the shift for initializing LFSR C is $\mathsf{cs} = 48$). Those parameters directly correspond to the maximal dimension that can be expected by the formulas above and are also verified experimentally (see in Table 3 in Appendix B). To find this specific polynomials we have checked if it is possible to have $\dim T_{A,C} \geq 42$, i.e., $\xi = 42$ and $\ell_1 = 47, \ell_2 = 49$. As $(\Psi_{\ell_1}\Psi_{\ell_2})^{-1} \approx 2500$ and $N = 2^{12} = 4096$ our approach should be successful with high probability. Indeed our algorithm computed the above solution with the even larger $T_{A,C}$ of dimension 44. Note that these parameters are chosen at the edge with respect to our theory. We could then choose a primitive polynomial g_B of degree 48 such that $\dim U_B = 48$ and such that the keyspace can be decomposed into $\mathbb{F}_2^{96} = U_B \oplus T_{A,C} \oplus V$ with $\dim V = 4$. Thus, we can break such a scheme with time complexity 2^{52} cipher evaluations and memory complexity $2^{48} \cdot 141$ bits.[8] The size of such a table is 4512 TiB.

For larger key sizes, this approach quickly gets infeasible. For example, if we would aim for a key length of $\kappa = 112$ bit (i.e., the minimum security required by NIST), we would choose g_A, g_B, and g_C of degrees 55, 56, and 57, respectively, such that $\dim T_{A,C} = 50$, $\dim U_B = 56$ and $\dim V = 6$. Other choices would only allow for other trade-offs between memory and computation, but not for reducing both. The divide-and-conquer (or meet-in-the-middle) attack against such a `GEA-1` instance would require

$$2^{\dim U_B} \cdot (\kappa + 1 + \dim T_{A,C} + \dim V) = 2^{56} \cdot (113 + 56) = 2^{56} \cdot 169$$

bits of memory, which corresponds to 1,384,448 TiB. Hence this approach is tailored to key spaces of smaller sizes.

3.3 Properties of the `GEA-1` Intentional Weakness

The weakness of `GEA-1` can be understood as a *hidden*, or *obfuscated*, cryptonalytic attack. It does not fulfill the property of undetectability, or even undiscoverability, simply because everyone who has the specification of `GEA-1` and some

[8] The length of each entry in the table must be large enough to avoid false key candidates. Similarly as described in [7, Section 3.1], we assume that each bitstring in the table is of size $\ell + \dim(T_{A,C})$, where ℓ is the minimum integer such that $(1 - 2^{-\ell})^{2^\kappa} \geq 0.5$.

knowledge of cryptoanalysis is in principle capable of finding the weakness and is able to exploit the attack. Of course, the fact that the `GEA-1` algorithm was not made public by the designers significantly hardened the discoverability of the weakness.

In the next part, we focus on the MALICIOUS framework, which is a method for inserting a practical and undetectable backdoor within a symmetric cryptographic algorithm, more precisely within a tweakable block cipher.

4 Revisiting the MALICIOUS Framework

In contrast to a hidden cryptographic weakness as in the case of `GEA-1`, the specification of the tweakable block cipher can be published entirely.

In the following, we discuss a simple instance of the MALICIOUS framework [29] that inserts a practical, undetectable backdoor into a tweakable block cipher. For this, let $\mathsf{H}\colon \mathbb{F}_2^\star \to \mathbb{F}_2^m$ be a cryptographic hash function and let $\mathsf{E}\colon \mathbb{F}_2^\kappa \times \mathbb{F}_2^\tau \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ be a (secure) tweakable block cipher with tweak length τ, key length κ, and block length n. The malicious designer chooses a tweak $t^\star \in \mathbb{F}_2^\tau$ uniformly at random and computes $s := \mathsf{H}(t^\star)$. The chosen tweak $t^\star$ will serve as the secret backdoor. The designer then defines the tweakable block cipher $\widetilde{\mathsf{E}}\colon \mathbb{F}_2^\kappa \times \mathbb{F}_2^\tau \times \mathbb{F}_2^n \to \mathbb{F}_2^n$ as

$$\widetilde{\mathsf{E}}(k,t,x) := \begin{cases} \mathsf{E}(k,t,x) & \text{if } \mathsf{H}(t) \neq s \\ x+k & \text{if } \mathsf{H}(t) = s \ . \end{cases} \tag{4}$$

In other words, if the backdoor $t^\star$ is used as the tweak, the tweakable block cipher $\widetilde{\mathsf{E}}$ simply applies the permutation $x \mapsto x+k$, which allows the malicious designer to recover the key k with one known plaintext/ciphertext pair. Due to this simple key-recovery attack, the backdoor fulfills the notion of *practicability.* If we assume that the hash function H is preimage resistant up to q queries, a user having oracle access to $\widetilde{\mathsf{E}}$ cannot recover the backdoor $t^\star$ with less than q queries. Therefore, the backdoor fulfills the notion of *undiscoverability.* More generally, under the same assumption on H, a user cannot even prove the existence of a secret backdoor with less than q queries to $\widetilde{\mathsf{E}}$. The reason is that the user cannot distinguish between whether the tweakable block cipher defined by Equation (4) was designed by a *malicious* designer who knows $t^\star$ and generated $s = \mathsf{H}(t^\star)$ accordingly or by an *honest* designer who simply chose a random $s \in \mathbb{F}_2^m$ to specify $\widetilde{\mathsf{E}}$. Therefore, the backdoor fulfills the notion of *undetectability.*

We conclude that the backdoor in the simple construction defined in Equation (4) fulfills the same security notions as the backdoor in the original MALICIOUS framework. However, similar to the original MALICIOUS framework, the backdoor in $\widetilde{\mathsf{E}}$ does not fulfill the notion of *untraceability*; once $\widetilde{\mathsf{E}}$ is queried with the tweak $t^\star$, the full backdoor is revealed.

5 Malicious AES

We now describe how to construct a tweakable variant of the AES with a modified key-schedule to obtain a more natural backdoored cipher based on the MALICIOUS framework. Instead of constructing a probability-one differential over the cipher for a secret pair of tweak values (as in the original MALICIOUS framework), we embed an invariant subspace that holds for a secret tweak value. For the sake of completeness, we briefly recall the definition of the AES round function. For further details, we refer to the book by Daemen and Rijmen [13].

5.1 Description of the AES

The AES is a family of block ciphers with a block length of 128 bits, supporting three different key lengths of 128, 192, and 256 bits. In this section, we concentrate on the AES variant with a 128-bit key. For each fixed key, the AES operates as a permutation on $\mathbb{F}_2^{128}$. For a simpler description of the algorithm, we represent the AES as a family of permutations on $\mathbb{F}_{2^8}^{4\times 4}$. The internal state can then be described by a 4×4 array with elements in $\mathbb{F}_{2^8}$ (also called *cells*) as

$$\begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}.$$

The unkeyed round function $\mathsf{R}\colon \mathbb{F}_{2^8}^{4\times 4} \to \mathbb{F}_{2^8}^{4\times 4}$ of AES is defined as the composition of the operations SubBytes, ShiftRows and MixColumns such that

$$\mathsf{R} = \mathsf{MixColumns} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes}\,.$$

The functions on the right-hand side are defined as follows.

SubBytes is a parallel application of the 8-bit AES S-box $S\colon \mathbb{F}_{2^8} \to \mathbb{F}_{2^8}$ to all 16 cells of the internal state. We refer to [13] for the definition of S, since its details are not important for our construction.

ShiftRows cyclically rotates the i^{th} row of the state i positions to the left, for all $i \in \{0, 1, 2, 3\}$.

MixColumns multiplies the columns of the state with a matrix M. Again, we refer to [13] for the definition of M.

The unkeyed AES rounds are interleaved by the addition of a round key. The latter operation will be denoted by $\mathsf{Add}_{k_i} : x \mapsto x + k_i$. The i^{th} round function of the AES is then given by

$$\mathsf{R}_i = \mathsf{Add}_{k_i} \circ \mathsf{R}\,.$$

The round keys k_i are generated from the 128-bit master key k by the AES key schedule, i.e., we have $(k_0, k_1, \ldots, k_{10}) = \mathsf{KeySchedule}(k)$. We refer to [13] for the specification of the function $\mathsf{KeySchedule}$. With the above notation, the AES variant with a 128-bit key can then be described as

$$\mathsf{AES}_k = \mathsf{Add}_{k_{10}} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes} \circ \mathsf{R}_9 \circ \cdots \circ \mathsf{R}_2 \circ \mathsf{R}_1 \circ \mathsf{Add}_{k_0} ,$$

where $(k_0, k_1, \ldots, k_{10}) = \mathsf{KeySchedule}(k)$.

5.2 Specification of MaliciousAES

In this section, we define a tweakable variant of the AES that incorporates a backdoor based on the MALICIOUS framework. The round function of MaliciousAES is identical to that of the AES, but its key schedule is different and it supports an arbitrary-length tweak. Note that, for other reasons, changing the AES key-scheduling has been discussed previously, e.g. in [20] and [14] to increase the resistance of AES against dedicated attacks.

Key and tweak schedule. Let $k \in \mathbb{F}_2^\kappa$ be a κ-bit master key. The partial (64-bit) round keys $k_0, \ldots, k_{10} \in \mathbb{F}_2^{64}$ are derived from the master key using a key scheduling function. The details of this function are left open. For reasons discussed in Section 5.3, it will be required that there is an efficient algorithm to uniquely determine 64 bits of k given the value of k_{10}. The actual round keys are then equal to $(k'_0, \ldots, k'_{10}) = \mathsf{MaliciousKeySchedule}(k)$, where the i^{th} round key k'_i is defined by

$$k'_i = \begin{bmatrix} k_{i,0} & k_{i,4} & k_{i,0} & k_{i,4} \\ k_{i,1} & k_{i,5} & k_{i,1} & k_{i,5} \\ k_{i,2} & k_{i,6} & k_{i,2} & k_{i,6} \\ k_{i,3} & k_{i,7} & k_{i,3} & k_{i,7} \end{bmatrix} , \text{ for } i = 0, \ldots, 9 \text{ and } k'_{10} = \begin{bmatrix} k_{10,0} & k_{10,4} & 0 & 0 \\ k_{10,1} & k_{10,5} & 0 & 0 \\ k_{10,2} & k_{10,6} & 0 & 0 \\ k_{10,3} & k_{10,7} & 0 & 0 \end{bmatrix} ,$$

with $k_{i,0}, \ldots, k_{i,7}$ being the bytes of k_i. In order to support arbitrary-length tweaks, the i^{th} partial round tweak $t_i \in \mathbb{F}_2^{64}$ will be derived from the master tweak t using an extendable output function H, i.e., $(t_0, \ldots, t_9) = \mathsf{H}(t)$. The full round tweaks are then equal to $(t'_0, \ldots, t'_9) = \mathsf{MaliciousTweakSchedule}(t)$ where t'_i is defined by

$$t'_i = \begin{bmatrix} c_{i,0} & c_{i,4} & t_{i,0} & t_{i,4} \\ c_{i,1} & c_{i,5} & t_{i,1} & t_{i,5} \\ c_{i,2} & c_{i,6} & t_{i,2} & t_{i,6} \\ c_{i,3} & c_{i,7} & t_{i,3} & t_{i,7} \end{bmatrix} ,$$

with $t_{i,0}, \ldots, t_{i,7}$ being the bytes of t_i and $c_{i,0}, \ldots, c_{i,7}$ being the bytes of c_i. The values $c_0, \ldots, c_9$ are round constants that appear to look random but, as explained below, are not necessarily so.

Overall structure. The i^{th} round function is defined by $\mathsf{R}'_i = \mathsf{Add}_{k'_i + t'_i} \circ \mathsf{R}$ and the tweakable block cipher MaliciousAES can then be described as

$$\mathsf{MaliciousAES}_{k,t} = \mathsf{Add}_{k'_{10}} \circ \mathsf{ShiftRows} \circ \mathsf{SubBytes} \circ \mathsf{R}'_9 \circ \cdots \circ \mathsf{R}'_1 \circ \mathsf{Add}_{k'_0 + t'_0} ,$$

where we have $(k'_0, \ldots, k'_{10}) = \mathsf{MaliciousKeySchedule}(k)$ and we have $(t'_0, \ldots, t'_9) = \mathsf{MaliciousTweakSchedule}(t)$.

Backdoor setup. Similar to the instances of the $\mathsf{MALICIOUS}$ framework presented in [29], we aim to introduce a tweak input such that the cipher can easily be broken for a special secret value of the tweak. To set up a backdoored instance of $\mathsf{MaliciousAES}$, the attacker chooses a secret tweak $t^\star$ and computes the values $(t_0^\star, \ldots, t_9^\star) = \mathsf{H}(t^\star)$. The round constants $c_0, \ldots, c_9$ are then chosen as $c_i = t_i^\star$ for $i = 0, \ldots, 9$. It will be shown in Section 5.3 that this choice results in the desired backdoor.

5.3 Description of the Backdoor

The backdoor in $\mathsf{MaliciousAES}$ is based on an invariant subspace for the round function R of the AES. For the secret backdoor tweak $t^\star$, this subspace is preserved up to the penultimate round of the cipher. Below, the invariant subspace for R, its extension to $\mathsf{MaliciousAES}$ and the key-recovery procedure are discussed.

Invariant subspace for R. Consider the linear subspace $\mathcal{I} \subseteq (\mathbb{F}_{2^8})^{4\times 4}$ in which the first column of the 4×4 state matrix is equal to the third column and the second column is equal to the fourth column, i.e.,

$$\mathcal{I} := \left\{ \begin{bmatrix} a_{0,0} & a_{0,1} & a_{0,0} & a_{0,1} \\ a_{1,0} & a_{1,1} & a_{1,0} & a_{1,1} \\ a_{2,0} & a_{2,1} & a_{2,0} & a_{2,1} \\ a_{3,0} & a_{3,1} & a_{3,0} & a_{3,1} \end{bmatrix} \;\middle|\; \forall i \in \{0,1,2,3\}, j \in \{0,1\} \colon a_{i,j} \in \mathbb{F}_{2^8} \right\} . \tag{5}$$

It is well known (see, e.g., [22, Prop. 5]) that $\mathcal{I}$ is an invariant subspace for the unkeyed AES round function R, i.e., $\mathsf{R}(\mathcal{I}) = \mathcal{I}$. Indeed, $\mathcal{I}$ is invariant for all of the three operations $\mathsf{SubBytes}$, $\mathsf{ShiftRows}$, and $\mathsf{MixColumns}$ independently. However, since $\mathcal{I}$ is in general not invariant over the addition with a round key, $\mathcal{I}$ cannot be used as an invariant subspace over multiple AES rounds interleaved with round key additions.

Invariant subspace for $\mathsf{MaliciousAES}$. When the cipher is evaluated with the malicious tweak value $t^\star$, we have that each round tweak t_i' with $i = 0, \ldots, 9$ is contained in $\mathcal{I}$ by construction. Hence, $\mathcal{I}$ is an invariant subspace for $\mathsf{Add}_{k_0'+t_0'}$ and more generally over any tweaked and keyed round function R_i', $i = 1, \ldots, 9$. The structure of the invariant subspace is depicted in Figure 4. Iterating this property yields

$$(\mathsf{Add}_{k_{10}'} \circ \mathsf{MaliciousAES}_{k,t^\star})(\mathcal{I}) = \mathcal{I}\,.$$

Under the assumption that for a tweak $t \neq t^\star$, each value t_i of $(t_0, \ldots, t_9) = \mathsf{H}(t)$ for $i \in \{0, \ldots, 9\}$ is randomly drawn from a uniform distribution over $\mathbb{F}_2^{64}$, the probability that, for a fixed $j \in \{0, \ldots, 9\}$, the value t_j' is contained in $\mathcal{I}$ is equal to $(2^{-8})^8 = 2^{-64}$. Therefore, we expect that for a tweak $t \neq t^\star$, the invariant subspace property is already broken after the first round of $\mathsf{MaliciousAES}$. For this reason, if H is a cryptographically secure function, the backdoor fulfills the notions of *undiscoverability* and *undetectability.*

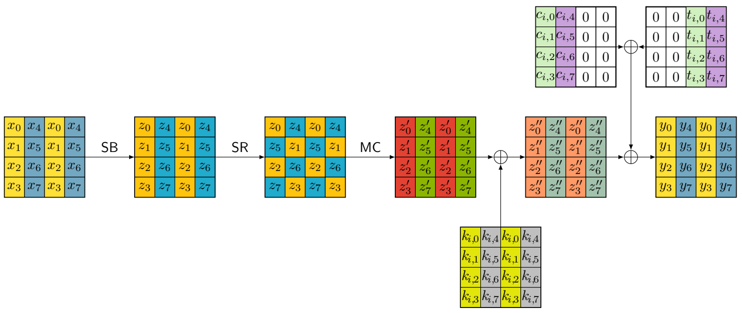

Fig. 4: An invariant subspace over one round of MaliciousAES under tweak $t^\star$. In the input state, if the first column is equal to the third column and the second column is equal to the fourth column, the same property holds for the output state after one round.

Key-recovery strategy. For $t^\star$, the last round key k'_{10} (which is constructed from $k_{10} \in \mathbb{F}_2^{64}$) can be recovered using a chosen plaintext attack with a single plaintext/ciphertext pair. For $x \in \mathcal{I}$ and $y = \mathsf{MaliciousAES}_{k,t^\star}(x)$, we have

$$\forall i \in \{0,1,2,3\}: \qquad k_{10,i} = y_{i,0} + y_{i,2}, \quad k_{10,4+i} = y_{i,1} + y_{i,3} \ ,$$

where $y_{i,j}$ is the byte in row i and column j of the ciphertext y. Hence, the 64-bit partial round key k_{10} can directly be recovered. From k_{10}, we can then recover k by guessing the remaining $\kappa - 64$ bits. Therefore, if κ is sufficiently small, MaliciousAES fulfills the notion of *practicability.*

Larger Keys. We only provided a very simple malicious variant of AES which uses 64-bit round keys derived from a master key of length κ bits. This approach is especially suitable when the master key is short, such as $\kappa = 64$. There are several straightforward methods to construct instances operating on larger keys. For instance, one can build a similar construction based on Rijndael-192 or Rijndael-256 [13]. For larger κ, one could also enforce other properties on the last (say the last two) round keys and use more elaborated key-guessing techniques to recover more than 64 bits of key information.

Security Arguments. We do not provide an explicit security analysis for MaliciousAES as (i) most of the security arguments for AES are equally valid for MaliciousAES and (ii) increasing the number of rounds of MaliciousAES does not invalidate the backdoor but should invalidate all potential non-backdoor based attacks.

6 A Dedicated Tweakable Block Cipher

In this section, we propose the backdoored dedicated tweakable block cipher Boomslang. Similar to MaliciousAES, the proposed cipher relies on the MALICIOUS framework to achieve undiscoverability. However, the backdoor is based on a nonlinear invariant rather than an invariant subspace. In fact, the backdoor implies the existence of an iterative perfect linear approximation over four rounds of the cipher. Hence, it can also be compared to the recently proposed block cipher ꓤooꓷ [30], which contains a backdoor based on linear cryptanalysis. However, the design rationale of ꓤooꓷ is weaker and it does not offer undiscoverability, so it has only limited practicability.

6.1 Specification of Boomslang

The cipher operates on 128-bit blocks and the state will be represented by a 4×8 array of 4-bit cells. The key k is a 128-bit value, and the tweak t can be any bitstring of arbitrary (bounded) length.

Round operations. The overall structure of the round function is shown in Figure 5 and it closely follows that of the AES. Specifically, the unkeyed round function of Boomslang can be written as

$$\mathsf{R} = \mathsf{MixColumns} \circ \mathsf{ShiftRows} \circ \mathsf{SubCells}\,.$$

Below, each of the functions on the right-hand side will be briefly discussed.

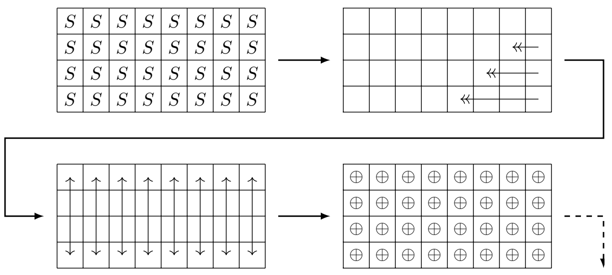

Fig. 5: Overview of the round function: SubCells, ShiftRows, MixColumns and the addition of constants.

SubCells consists of the parallel application of an S-box S to the 4-bit cells of the state. The S-box is the nonlinear function $S : \mathbb{F}_2^4 \to \mathbb{F}_2^4$ defined by Table 1.

ShiftRows is similar to the AES ShiftRows step. If the rows are numbered from zero to three with zero corresponding to the top row, then ShiftRows rotates the i^{th} row of the state over $4 \cdot i$ bits to the left.

MixColumns consists of a columnwise multiplication with a lightweight matrix from the family of quasi-MDS matrices that was proposed for Qarma-64 [4]. Let us denote the cells within one column of the state by $(x_0, \ldots, x_3)$, where $x_i \in \mathbb{F}_2^4$. $\mathsf{MixColumns}$ maps each column $(x_0, \ldots, x_3)$ to a new column $(y_0, \ldots, y_3)$ defined by

$$y_i = x_{i+1} + (x_{i+2} \lll 1) + (x_{i+3} \lll 2)\,,$$

for $i = 0, \ldots, 3$ and where the addition of the indices is regarded modulo 4. The inverse mapping is given by

$$x_i = y_{i+3} + (y_{i+1} \lll 2) + (y_{i+2} \lll 3)\,,$$

for $i = 0, \ldots, 3$. In software, $\mathsf{MixColumns}$ can be implemented using bitslicing.

The $2i^{\text{th}}$ round function is then defined as

$$\mathsf{R}_{2i} = \mathsf{Add}_{k_i} \circ \mathsf{Add}_{c_{2i}} \circ \mathsf{R}\,,$$

Table 1: The 4-bit S-box S.

0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
8	2	4	a	5	f	7	6	0	c	b	9	e	d	1	3

where c_i are round constants and k_i round keys. The round keys k_i can be derived using an arbitrary key schedule. Since we do not aim for related-key security, we propose $k_i = k$ for simplicity. Similarly, the $(2i + 1)^{\text{th}}$ round function is defined as

$$\mathsf{R}_{2i+1} = \mathsf{Add}_{t_i} \circ \mathsf{Add}_{c_{2i+1}} \circ \mathsf{R}\,.$$

The i^{th} round tweak t_i is derived from the master tweak t using a strong tweak-scheduling function. Specifically, $(t_0, t_1, \ldots) = \mathsf{H}(t)$ where H is an extendable output function such as SHAKE [16]. The round constants c_i should appear to look random, but – as discussed below – they are not necessarily so. Some specific conditions on these constants, all of which are likely to be satisfied for a uniform random choice, are discussed in Section 6.2.

Overall structure. The overall cipher is the composition of the round functions $\mathsf{R}_0, \mathsf{R}_1, \ldots$. The number of rounds, denoted by r, is required to be even, so that the last round function always performs a round-tweak addition. Based on the analysis in Section 6.2, $r = 32$ is a reasonable choice. The key k is also added to the plaintext and $\sigma(k)$ is added to the state after the last round, with σ being a linear orthomorphism, i.e., we have that both $k \mapsto \sigma(k)$ and $k \mapsto \sigma(k) + k$ are permutations. Specifically, we use $\sigma(k) = (k \ggg 1) + (k \gg 63)$ as in PRINCE [12]. In summary, the overall cipher is defined by

$$\mathsf{Boomslang} = \mathsf{Add}_{\sigma(k)} \circ \mathsf{R}_{r-1} \circ \cdots \circ \mathsf{R}_1 \circ \mathsf{R}_0 \circ \mathsf{Add}_k\,.$$

Backdoor setup. To set up a backdoored instance of $\mathsf{Boomslang}$, one must choose a secret tweak $t^\star$ and compute the corresponding round tweaks $(t_0^\star, t_1^\star, \ldots) = \mathsf{H}(t^\star)$. The round constants c_{2i+1} should then be chosen as $c_{2i+1} = t_i^\star$ for $i = 0, \ldots, r/2 - 1$.

Given a few hundred known plaintext/ciphertext pairs encrypted under the tweak $t^\star$, it will then be possible to efficiently recover the full key. The mechanism for doing so will be explained in Section 6.3.

6.2 Design Rationale

In this section, we argue that the design of Boomslang follows standard modern design principles. Our security analysis focuses on differential and linear cryptanalysis, and invariants. The latter two are of particular relevance with respect to the backdoor that will be revealed in Section 6.3. In the following, we first motivate the overall structure of Boomslang and then discuss the choice of the individual components.

Overall structure. The design follows the wide-trail strategy [13, Chapter 9] with some changes to obtain a more lightweight cipher. Whenever possible, the design was kept as simple as possible and close to that of the AES.

In general, the proposed cipher is geared towards hardware. This is the motivation for relying on 4-bit S-boxes rather than 8-bit S-boxes as in the AES. In software, the 4×8 state allows storing the rows as 32-bit words. The S-box and linear layer can then be implemented using bitslicing.

The key schedule is chosen as the identity function, although other key schedules could also be used. Since related-key security was not a design goal, we decided to choose the simplest option. In addition, having a linear key schedule sometimes enables more straightforward security arguments. For example, the arguments from [6] related to the choice of round constants to prevent invariants are only applicable to linear key schedules.

Finally, the choice of the tweak schedule can be motivated by the goal of supporting arbitrary-length tweaks. Since related tweak security is important, it seems necessary to use a cryptographically strong hash function or XOF to derive round tweaks from the master tweak.

Choice of the components. We now argue that all of the basic components used in the cipher are individually acceptable choices from the current state of the art.

SubCells. The S-box has a maximum absolute correlation of $1/2$ for nonzero masks and a maximum differential probability of $1/4$ for nonzero differences. The S-box is chosen such that it is not an involution.

ShiftRows. The cell permutation is chosen such that the cells of each column end up in different columns of the state. Shifting rows is a natural choice because it allows for an efficient software implementation, and it is the same as for the AES.

MixColumns. The MixColumns map is inspired by the linear layer of Qarma-64 [4]. Specifically, the transformation of each column is defined by a circulant matrix M of the form

$$M = \begin{bmatrix} 0 & X^a & X^b & X^c \\ X^c & 0 & X^a & X^b \\ X^b & X^c & 0 & X^a \\ X^a & X^b & X^c & 0 \end{bmatrix},$$

over the $\mathbb{F}_2$-vector space $\mathbb{F}_2[X]/(X^4+1)$. The input bitvector can be considered as an element of this space by the isomorphism $\delta_i \mapsto X^{i-1}$, where δ_i is the i^{th} standard basis vector of $\mathbb{F}_2^4$.
The matrix M is invertible with circulant inverse of the same form if and only if $a \equiv c \pmod 4$ or $a \equiv c+2 \pmod 4$. All of these matrices have branch number four, which is the maximum possible for this type of matrix. Furthermore, we impose the following criteria:

– Unlike in Qarma-64, we require that M is not an involution. Equivalently, $2b \not\equiv 0 \pmod 4$. The motivation for this requirement is that involutions more easily lead to 2-round invariants, as demonstrated in the case of Midori-64 [9].
– M should not be orthogonal or nearly orthogonal, i. e. $M^{-1} \neq c\,M^\top$ for any $c \in \mathbb{F}_2[X]/(X^4+1)$. This requirement is motivated by the fact that any quadratic form of the type $\sum_{i=1}^m x_i^\top Q x_i$ is a nonlinear invariant for an $m \times m$ orthogonal matrix [35]. More generally, for a nearly orthogonal matrix, any such quadratic function which is also invariant under multiplication by c is a nonlinear invariant.

The second criterium leads to the requirement that $X^{a+b} \neq X^{b+c}$ or equivalently $a \not\equiv c \pmod 4$. From the viewpoint of software implementations, it makes sense to choose one of a, b or c equal to zero. Choosing $a = 0$ and $b = 1$ then gives $c = 2$.

Linear and differential cryptanalysis. The wide-trail strategy directly gives upper bounds on the absolute correlation of linear trails and on the probability of differential characteristics. In particular, since M has a branch number of four, the number of active S-boxes over four rounds is at least 16 [13, Theorem 9.4.1]. Hence, after 16 rounds the average probability of any differential characteristic is lower than 2^{-128} and the absolute correlation of any linear trail is at most 2^{-64}. The suggested choice of 32 rounds was obtained by taking twice as many rounds; taking into account potential improvements and key-recovery attacks.

In fact, it is to some extent possible to extend the above results to linear approximations and differentials. In particular, for independent uniform random constants, [25, Corollary 1 & 2] imply that the average probability of any 4-round differential and the average squared correlation of any 4-round linear approximation is at most $(2^{-2\times(4-1)})^4 = 2^{-24}$.

Invariants. Several lightweight ciphers have been found vulnerable to invariant subspace [23] and nonlinear invariant attacks [35]. Hence, it is natural to attempt to rule out the existence of invariants in Boomslang.

The argument from [6] can be used to rule out joint invariants over all the affine layers (i.e., linear layers together with the constant additions) for a large number of rounds using only the properties of the linear layer and the round constants. Specifically, the security argument depends on the dimension of of the smallest subspace invariant under the linear layer and containing the differences of the constants. For the linear layer $\mathsf{L} = \mathsf{MixColumns} \circ \mathsf{ShiftRows}$ of Boomslang and constants $c_0, \ldots, c_{r-1}$, denote this space by $W_{\mathsf{L}}(c_0+c_1, c_0+c_2, \ldots, c_0+c_{r-1})$.

If $W_{\mathsf{L}}(c_0 + c_1, c_0 + c_2, \ldots, c_0 + c_{r-1}) = \mathbb{F}_2^{128}$, then joint invariants over the affine layers can be ruled out with high probability. The linear map L has 16 invariant factors and its minimal polynomial is $(X + 1)^8$. Hence, by [6, Proposition 11],

$$\Pr_{c_0,\ldots,c_{23}}[\dim W_L(c_0 + c_1, c_0 + c_2, \ldots, c_0 + c_{23}) = 128] = \prod_{i=0}^{15}\left(1 - \frac{1}{2^{23-i}}\right) \geq 0.99\,,$$

for uniformly chosen random constants $c_0, \ldots, c_{23}$. Hence, 24 rounds are sufficient to rule out with high probability the existence of such invariants. Note that this argument does not yet rule out invariants over a small number of rounds and also does not cover generalized and closed-loop invariants [37].

Most invariants considered in previous attacks have independent cells ('rank-one', from the viewpoint of [9]), as this leads to an easier analysis of the SubCells and ShiftRows steps. To investigate this in more detail, we used the tool from [10, §6.2] to obtain the rank-one invariants of the linear layer M. Although M has some rank-one invariants, they do not correspond to Boolean functions or sets, and there are no shared invariants between M and the S-box layer.

6.3 Description of the Backdoor

The backdoor is a two-round invariant, which is not invariant for one round. This is similar to one of the invariants of two-rounds of Midori-64 [9], but unlike in that case the property is not invariant under the linear layer. Indeed, as discussed above, that would not be possible due to the choice of the linear layer. Importantly, the invariant only exists for the secret weak tweak for which the round constants in even rounds cancel out.

Two-round nonlinear invariant. Let $f : \mathbb{F}_2^4 \to \mathbb{F}_2$ and $g : \mathbb{F}_2^4 \to \mathbb{F}_2$ be the Boolean functions defined by

$$\begin{aligned} f(z_0, z_1, z_2, z_3) &= (z_0 + z_2)(z_1 + z_3) + z_0 + z_2 + z_3 + 1 \\ g(z_0, z_1, z_2, z_3) &= (z_0 + z_2)(z_1 + z_3) + z_2\,. \end{aligned}$$

The functions f and g can be used to form a perfect nonlinear approximation of M. This is due to the fact that the term $(z_0 + z_2)(z_1 + z_3)$ is invariant under rotations of $z_0, \ldots, z_3$. Hence, if $y = \mathsf{MixColumns}(x)$, then

$$\sum_{i=0}^{31} g(y_{4i}, y_{4i+1}, y_{4i+2}, y_{4i+3}) = \sum_{i=0}^{31} f(x_{4i}, x_{4i+1}, x_{4i+2}, x_{4i+3})\,.$$

Furthermore, it is easy to see that

$$\sum_{i=0}^{31} \mathsf{a}^\top(y_{4i}, y_{4i+1}, y_{4i+2}, y_{4i+3}) = \sum_{i=0}^{31} \mathsf{5}^\top(x_{4i}, x_{4i+1}, x_{4i+2}, x_{4i+3})\,.$$

The S-box S defined in Table 1 also satisfies

$$\begin{aligned} \mathsf{5}^\top S(z_0, z_1, z_2, z_3) &= g(z_0, z_1, z_2, z_3) \\ f(S(z_0, z_1, z_2, z_3)) &= \mathsf{a}^\top (z_0, z_1, z_2, z_3)\,. \end{aligned}$$

Since linear functions are invariant under the addition of any constant, and because the constants are cancelled out by the tweak in even rounds, one obtains the following two-round invariant:

$$\sum_{i=0}^{31} g(y_{4i}, y_{4i+1}, y_{4i+2}, y_{4i+3}) = c + \sum_{i=0}^{31} g(x_{4i}, x_{4i+1}, x_{4i+2}, x_{4i+3})\,,$$

where $y = (\mathsf{R}_{2i+1} \circ \mathsf{R}_{2i})(x)$. The full nonlinear trail is illustrated in Figure 6. Note that the last step only works for one in 2^{64} constants, but the constants are chosen such that there exists a tweak so that the constants are weak in all odd-numbered rounds.

Alternatively, the nonlinear invariant discussed above can be described from the point of view introduced in [9, 10]. Let

$$\begin{aligned} w &= (0, -1, 0, 0, 1, 0, 0, 0, 0, 0, 0, -1, 0, \quad 0, -1, 0)/2 \\ v &= (0, \quad 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, \quad 0, 0, -1, \quad 0, 0)/2\,. \end{aligned}$$

In the above, w and v are the Walsh-Hadamard transform of f and g respectively. It holds that $C^M w^{\otimes 4} = v^{\otimes 4}$, with C^M being the correlation matrix of the linear layer and $\otimes$ the tensor product. Furthermore, the S-box satisfies $C^S v = \delta_{\mathsf{5}}$ and $C^S \delta_{\mathsf{a}} = w$. The vector v is invariant under one in four constants.

Key-recovery strategy. The addition of whitenening keys k and $\sigma(k)$ leads to an efficient key-recovery attack. Specifically, one can use the fact that for every plaintext/ciphertext pair (x, y) encrypted under the backdoored tweak, there exist $\ell \in \mathbb{F}_2^{128}$ and $b \in \mathbb{F}_2$ such that

$$\sum_{i=0}^{31} g(x_i + k_i) + \sum_{i=0}^{31} g(y_i + \sigma(k)_i) = \ell^\top k + b\,,$$

with $x_0, \dots x_{31}$, $y_0, \dots, y_{31}$ and $k_0, \dots, k_{31}$ being the nibbles of x, y and k, respectively. Since σ is an orthomorphism, the 64 bits of k that are nonlinearly mixed with x are linearly independent from the bits of k that are nonlinearly mixed with y. Hence, given q messages, one can on average recover q bits of the key even when $q \geq 64$.

Solving the system of equations is easy because of the low number of quadratic terms. One can either use Gröbner basis methods, exploiting the low degree of regularity of the system, or one can directly rely on linearization. Since g contains only a single quadratic term, each equation contains at most 64 quadratic terms. Hence, given 192 known plaintext/ciphertext pairs, the full key can be recovered using less than $192^3 \leq 2^{23}$ bit operations.

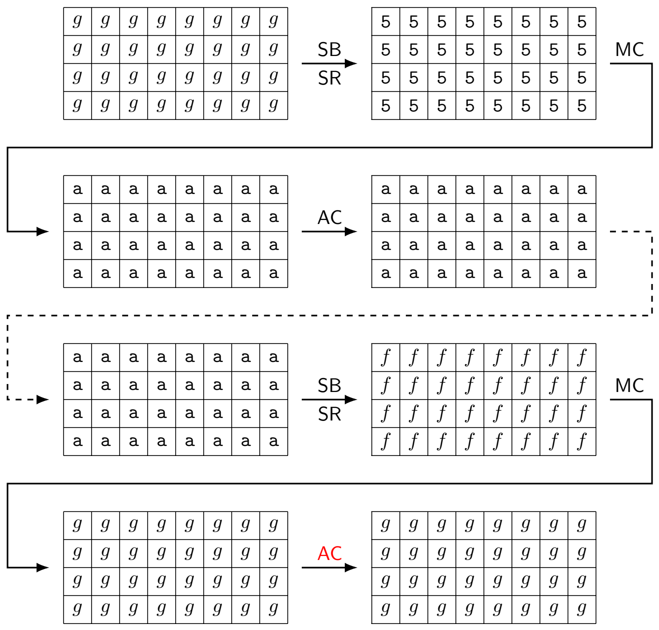

Fig. 6: Two-round invariant for Boomslang.

Construction of the backdoor. The backdoor primarily relies on the choice of the S-box. The tool from [10, §6.2] was used to find symmetric nonlinear rank-one approximations of the linear layer M. This resulted in the choice of the vectors w and v listed above. One can then easily generate S-boxes such that the conditions $C^S v = \delta_5$ and $C^S v = \delta_{10}$ are satisfied. There are still significant degrees of freedom left in the choice of the S-box. These could be used to satisfy additional design criteria, or to argue that the S-box was generated based on certain magic constants.

7 Conclusion

Feeding a session key into LFSRs by making use of shifts is common in many designs, e.g., besides in `GEA-1` and `GEA-2` it is also used in A5/1. Our work demonstrates that those shifts, together with a clever choice of feedback polynomials and filtering, can be used to deliberately weaken the construction. We gave an explicit and efficient way to construct those choices for a large variety of parameters. Our construction includes the choices made for `GEA-1` indicating that this (or a related) strategy was used in the actual design process. On the positive side, we again see that, in line with [7], this is unlikely to happen unintentionally. While our theory is described with a focus on LFSRs in Galois mode, it applies to LFSRs in Fibonacci mode as well. However, our construction yields random looking feedback polynomials and thus seemingly selected taps. While for Galois

LFSRs and software implementations, this does not affect performance, it does for LFSRs in Fibonacci mode. Here, the number of taps determines the number of XOR operations and thus the construction is less interesting in this case as it contradicts well-established design rationales.

In the second part of the paper, we outlined two designs of a tweakable block cipher that embed a hidden trapdoor, based on the MALICIOUS framework. Our constructions stress the importance of justifying the every single part of the design. One possible approach is *unswervingness* (see [15]) as a design requirement. In a nutshell, the notion of unswervingness demands that each instance of a (block) cipher fulfilling all the requirements given in its design rationale is secure. However, this might be highly non-trivial to achieve as a designer.

For the MALICIOUS framework, it would be very interesting to actually investigate how backdoors could be triggered by many tweaks. In the original work[29] two tweaks were necessary to enable the backdoor, while for our instances a single tweak is sufficient. Using many tweaks could potentially lead to achieving untraceability as one could hide the tweaks needed to activate the backdoor with tweaks that do not. The goal would then be that finding the correct subset becomes exponentially hard in the number of tweaks.

Acknowledgments. This work was supported by the German Research Foundation (DFG) within the framework of the Excellence Strategy of the Federal Government and the States – EXC 2092 CaSa – 39078197. Tim Beyne is supported by a PhD Fellowship from the Research Foundation – Flanders (FWO).

References

1. Albertini, A., Aumasson, J., Eichlseder, M., Mendel, F., Schläffer, M.: Malicious hashing: Eve's variant of SHA-1. In: Joux, A., Youssef, A.M. (eds.) Selected Areas in Cryptography - SAC 2014. LNCS, vol. 8781, pp. 1–19. Springer (2014)
2. Albrecht, M.R., Rechberger, C., Schneider, T., Tiessen, T., Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (eds.) Advances in Cryptology - EUROCRYPT 2015. LNCS, vol. 9056, pp. 430–454. Springer (2015)
3. Amzaleg, D., Dinur, I.: Refined cryptanalysis of the GPRS ciphers GEA-1 and GEA-2. In: Dunkelman, O., Dziembowski, S. (eds.) Advances in Cryptology - EUROCRYPT 2022. LNCS, vol. 13277, pp. 57–85. Springer (2022)
4. Avanzi, R.: The QARMA block cipher family. Almost MDS matrices over rings with zero divisors, nearly symmetric even-mansour constructions with non-involutory central rounds, and search heuristics for low-latency s-boxes. IACR Trans. Symmetric Cryptol. **2017**(1), 4–44 (2017)
5. Bannier, A., Filiol, E.: Partition-based trapdoor ciphers. IntechOpen (2017)
6. Beierle, C., Canteaut, A., Leander, G., Rotella, Y.: Proving resistance against invariant attacks: How to choose the round constants. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017. LNCS, vol. 10402, pp. 647–678. Springer (2017)
7. Beierle, C., Derbez, P., Leander, G., Leurent, G., Raddum, H., Rotella, Y., Rupprecht, D., Stennes, L.: Cryptanalysis of the GPRS encryption algorithms GEA-1 and GEA-2. In: Canteaut, A., Standaert, F. (eds.) Advances in Cryptology - EUROCRYPT 2021. LNCS, vol. 12697, pp. 155–183. Springer (2021)

8. Bernstein, D.J., Lange, T., Niederhagen, R.: Dual EC: A standardized back door. In: Ryan, P.Y.A., Naccache, D., Quisquater, J. (eds.) The New Codebreakers. LNCS, vol. 9100, pp. 256–281. Springer (2016)
9. Beyne, T.: Block cipher invariants as eigenvectors of correlation matrices. In: Peyrin, T., Galbraith, S.D. (eds.) Advances in Cryptology - ASIACRYPT 2018. LNCS, vol. 11272, pp. 3–31. Springer (2018)
10. Beyne, T.: A geometric approach to linear cryptanalysis. In: Tibouchi, M., Wang, H. (eds.) Advances in Cryptology - ASIACRYPT 2021. LNCS, vol. 13090, pp. 36–66. Springer (2021)
11. Bonnetain, X., Perrin, L., Tian, S.: Anomalies and vector space search: Tools for s-box analysis. In: Galbraith, S.D., Moriai, S. (eds.) Advances in Cryptology - ASIACRYPT 2019. LNCS, vol. 11921, pp. 196–223. Springer (2019)
12. Borghoff, J., Canteaut, A., Güneysu, T., Kavun, E.B., Knezevic, M., Knudsen, L.R., Leander, G., Nikov, V., Paar, C., Rechberger, C., Rombouts, P., Thomsen, S.S., Yalçin, T.: PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In: Wang, X., Sako, K. (eds.) Advances in Cryptology - ASIACRYPT 2012. LNCS, vol. 7658, pp. 208–225. Springer (2012)
13. Daemen, J., Rijmen, V.: The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition. Information Security and Cryptography, Springer (2020)
14. Derbez, P., Fouque, P., Jean, J., Lambin, B.: Variants of the AES key schedule for better truncated differential bounds. In: Cid, C., Jr., M.J.J. (eds.) Selected Areas in Cryptography - SAC 2018. LNCS, vol. 11349, pp. 27–49. Springer (2018)
15. Dunkelman, O., Perrin, L.: Adapting rigidity to symmetric cryptography: Towards "unswerving" designs. In: Mehrnezhad, M., van der Merwe, T., Hao, F. (eds.) Proceedings of the 5th ACM Workshop on Security Standardisation Research Workshop. pp. 69–80. ACM (2019)
16. Dworkin, M.: SHA-3 standard: Permutation-based hash and extendable-output functions (2015)
17. Filiol, E.: BSEA-1 - a stream cipher backdooring technique. arXiv preprint arXiv:1903.11063 (2019)
18. Harpes, C., Massey, J.L.: Partitioning cryptanalysis. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97. LNCS, vol. 1267, pp. 13–27. Springer (1997)
19. Hoffman, K., Kunze, R.A.: Linear Algebra. PHI Learning (2004)
20. Khoo, K., Lee, E., Peyrin, T., Sim, S.M.: Human-readable proof of the related-key security of AES-128. IACR Trans. Symmetric Cryptol. **2017**(2), 59–83 (2017)
21. Koblitz, N.: Algebraic aspects of cryptography, Algorithms and computation in mathematics, vol. 3. Springer (1998)
22. Le, T.V., Sparr, R., Wernsdorf, R., Desmedt, Y.: Complementation-like and cyclic properties of AES round functions. In: Dobbertin, H., Rijmen, V., Sowa, A. (eds.) Advanced Encryption Standard - AES, 4th International Conference. LNCS, vol. 3373, pp. 128–141. Springer (2004)
23. Leander, G., Abdelraheem, M.A., AlKhzaimi, H., Zenner, E.: A cryptanalysis of PRINTcipher: The invariant subspace attack. In: Rogaway, P. (ed.) Advances in Cryptology - CRYPTO 2011. LNCS, vol. 6841, pp. 206–221. Springer (2011)
24. Lidl, R., Niederreiter, H.: Finite Fields. Encyclopedia of Mathematics and its Applications, Cambridge University Press, 2 edn. (1996)
25. Park, S., Sung, S.H., Lee, S., Lim, J.: Improving the upper bound on the maximum differential and the maximum linear hull probability for SPN structures and AES.

In: Johansson, T. (ed.) Fast Software Encryption, 10th International Workshop, FSE 2003. LNCS, vol. 2887, pp. 247–260. Springer (2003)

26. Paterson, K.G.: Imprimitive permutation groups and trapdoors in iterated block ciphers. In: Knudsen, L.R. (ed.) Fast Software Encryption, 6th International Workshop, FSE '99. LNCS, vol. 1636, pp. 201–214. Springer (1999)
27. Perlroth, N., Larson, J., Shane, S.: N.S.A. able to foil basic safeguards of privacy on web. International New York Times https://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html (accessed September 30, 2021) (2013)
28. Perrin, L.: Partitions in the s-box of Streebog and Kuznyechik. IACR Trans. Symmetric Cryptol. **2019**(1), 302–329 (2019)
29. Peyrin, T., Wang, H.: The MALICIOUS framework: Embedding backdoors into tweakable block ciphers. In: Micciancio, D., Ristenpart, T. (eds.) Advances in Cryptology - CRYPTO 2020. LNCS, vol. 12172, pp. 249–278. Springer (2020)
30. Posteuca, R., Ashur, T.: How to backdoor a cipher. IACR Cryptol. ePrint Arch. p. 442 (2021)
31. PUB FIPS: 46: Data Encryption Standard (DES). US Department of Commerce, National Bureau of Standards (1977)
32. Rijmen, V., Preneel, B.: A family of trapdoor ciphers. In: Biham, E. (ed.) Fast Software Encryption, 4th International Workshop, FSE '97. LNCS, vol. 1267, pp. 139–148. Springer (1997)
33. Sage Developers: SageMath, the Sage Mathematics Software System (Version 9.3) (2021), `https://www.sagemath.org`
34. Schneier, B.: Applied cryptography - protocols, algorithms, and source code in C, 2nd Edition. Wiley (1996)
35. Todo, Y., Leander, G., Sasaki, Y.: Nonlinear invariant attack: Practical attack on full SCREAM, iSCREAM, and Midori64. J. Cryptol. **32**(4), 1383–1422 (2019)
36. Wardlaw, W.P.: Matrix representation of finite fields. Mathematics Magazine **67**(4), 289–293 (1994)
37. Wei, Y., Ye, T., Wu, W., Pasalic, E.: Generalized nonlinear invariant attack and a new design criterion for round constants. IACR Trans. Symmetric Cryptol. **2018**(4), 62–79 (2018)
38. Wu, H., Bao, F., Deng, R.H., Ye, Q.: Cryptanalysis of Rijmen-Preneel trapdoor ciphers. In: Ohta, K., Pei, D. (eds.) Advances in Cryptology - ASIACRYPT '98. LNCS, vol. 1514, pp. 126–132. Springer (1998)

A Source Code to Generate GEA1-like Systems

Listing 1.1: weakgea.sage

```
# GEA-1 parameters
kappa = 64
xi = 24
l1 = 31
l2 = 33
l3 = 32

trials = 2**16

hkappa = int(kappa/2)
qkappa = int(kappa/4)

def getInitMatrix_fast(p,keyLength,shift):
  P.<x> = PolynomialRing(GF(2))
  l = p.degree()
  # Construct transformation matrix A for LFSR in Galois mode
  A = companion_matrix(x^l+1)
  A[0] = list(p)[0:l][::-1]
  A = A.transpose()
  e0 = vector(GF(2),[1]+[0]*(l-1))
  M = zero_matrix(GF(2),l,keyLength)
  for c in range(keyLength):
    if (c < shift):
      M.set_column(c,A**(shift-c)*e0)
    else:
      M.set_column(c,A**(keyLength-c+shift)*e0)
  return M.transpose()

V = VectorSpace(GF(2),(kappa-2*xi));
Pol.<X> = PolynomialRing(GF(2));

success = False
ctr = 0
while (ctr < trials):
  v = V.random_element()
  p1 = X**kappa+X**xi
  j = 0
  for i in [1..(hkappa-xi)]+[hkappa..(kappa-xi-1)]:
    p1 = p1+v[j]*X**(kappa-i)
    j = j+1
  fp1 = list(p1.factor())

  for f in fp1:
    if ((f[0].degree()==l1) and f[0].is_primitive()):
      p2 = X**hkappa+X**(hkappa+xi)
      j = 0
      for i in [1..(hkappa-xi)]:
        p2 = p2+v[j]*X**(hkappa-i)
        j = j+1
      for i in [hkappa..(kappa-xi-1)]:
        p2 = p2+v[j]*X**(kappa-i+hkappa)
        j = j+1
      fp2 = list(p2.factor())

      for g in fp2:
        if ((g[0].degree()==12) and g[0].is_primitive()):
          g1 = f[0]
          g2 = g[0]
          print('g1 = ', g1)
          print('g2 = ', g2)
          ctr = trials

Initmat2 = getInitMatrix_fast(g2,kappa,hkappa)
Initmat1 = getInitMatrix_fast(g1,kappa,0)

T = Initmat2.kernel().intersection(Initmat1.kernel())
print('dim T = ', T.dimension())

ctr = 0
while (ctr < trials):
  g3 = X**13+1
  for i in [1..(13-1)]:
    g3 = g3+(GF(2).random_element())*X**i
  if g3.is_primitive():
    Initmat3 = getInitMatrix_fast(g3,kappa,qkappa)
    if Initmat3.kernel().intersection(T).dimension()==0:
      print('g3 = ', g3)
      ctr = trials
    else:
      print('try again')
      ctr = ctr+1
```

B Experimental Verification for GEA-like Constructions

We experimentally verified the plausibility of the heuristic used in our construction. This part demonstrations that the experiments nicely match the theoretical predictions.

Recall that we assume that the polynomials constructed for a vector $t \in \mathbb{F}_2^\kappa$ as given in Equation (1) and (2) behave as a uniform random pair of polynomials. In particular this means that a fraction of Ψ_{ℓ_1} of the polynomials p_1 have a primitive divisor of degree ℓ_1 and, similarly, a fraction of Ψ_{ℓ_2} of the polynomials p_2 have a primitive divisor of degree ℓ_2.

Table 2: For $\kappa = 64$ and $\xi \in \{23, 24, \ldots, 28\}$, this table shows the number of p_1 that yield primitive divisors of degree ℓ (first number in cell), the number of p_2 (with $\mathsf{cs} = \frac{\kappa}{2} = 32$) that yield primitive divisors of degree ℓ (second number in cell), compared to the theoretical estimate $\Psi_\ell 2^{\kappa-2\xi}$ rounded to the closest integer (number in parentheses).

(ℓ, ξ)	23	24	25	26	27	28
	8884	2190	442	76	24	0
29	9178	2224	544	114	40	0
	(8988)	(2247)	(562)	(140)	(35)	(9)
	4242	1132	274	72	16	0
30	4322	1106	270	44	10	0
	(4351)	(1088)	(272)	(68)	(17)	(4)
	9460	2158	506	126	44	18
31	8720	2052	518	106	26	14
	(8456)	(2114)	(529)	(132)	(33)	(8)
	3822	966	226	40	0	0
32	4108	1110	272	68	12	4
	(4096)	(1024)	(256)	(64)	(16)	(4)
	6416	1624	416	112	10	0
33	6158	1584	378	108	12	0
	(6440)	(1610)	(402)	(101)	(25)	(6)
	4900	1194	314	46	18	6
34	5128	1268	336	78	18	10
	(5140)	(1285)	(321)	(80)	(20)	(5)

For $\kappa = 64$ and values of ξ between 24 and 28, we computed the exact number of polynomials p_1 and p_2 and compared this to the theoretical estimate.

More importantly, we checked the behaviour of pairs directly. In order to limit the set of parameter to consider, we restricted to the case of $\ell_2 = \ell_1 + 2$ and $\kappa = \ell_1 + \ell_2$. For each $4 \leq \ell_1 \leq 67$ we compared the maximal dimension that could actually be constructed to the maximal dimension that could have been expected to be possible following the heuristic. The results, shown in Table 3 and verifiable by the sage code provided in Appendix A, suggest that the heuristic is plausible.

Table 3: Experiments for parameters $\ell_2 = \ell_1 + 2, \kappa = \ell_1 + \ell_2$, for $4 \leq \ell_1 \leq 67$. The value ξ_{exp} gives the maximum ξ such that $\Psi_{\ell_1}\Psi_{\ell_2}2^{\kappa-2\xi} \geq 1$. The value $\xi_{\max}$ gives the maximum ξ for which there exist p_1, p_2 that yield primitive divisors g_1, g_2 of degree ℓ_1 and ℓ_2, respectively. The value # sol gives the number of such tuples (p_1, p_2). For all such solutions, the dimension of $T_{G_{g_1},G_{g_2},\text{cs}}$ is equal to $\xi_{\max}$.

ℓ_1	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
ξ_{exp}	1	3	3	4	4	6	6	8	8	10	10	11	11	13	13	15
$\xi_{\max}$	3	2	3	4	4	5	6	7	7	11	9	11	10	13	13	15
# sol	2	8^*	4	2	2	8	2	4	2	2	6	2	14	2	2	4

ℓ_1	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
ξ_{exp}	15	17	17	19	19	21	21	23	23	25	25	26	27	28	28	30
$\xi_{\max}$	16	16	18	20	20	21	22	23	22	23	24	26	27	28	28	29
# sol	2	8	2	2	2	4	4	2	2	14	10	4	2	6	2	6

ℓ_1	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
ξ_{exp}	30	32	32	34	34	36	36	38	38	40	40	42	42	44	44	46
$\xi_{\max}$	30	32	32	34	34	36	35	38	39	40	39	44	44	44	45	45
# sol	2	2	2	4	4	4	8	2	4	2	2	2	2	2	2	2

ℓ_1	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67
ξ_{exp}	46	48	48	50	50	52	52	54	54	55	56	57	58	59	60	61
$\xi_{\max}$	46	48	47	50	49	53	51	53	53	55	56	57	58	60	59	60
# sol	2	2	6	2	6	2	4	2	4	2	2	4	2	2	10	6