BISON
Instantiating the Whitened Swap-Or-Not Construction

Anne Canteaut[1], Virginie Lallemand[2], Gregor Leander[2], Patrick Neumann[2] and Friedrich Wiemer[2]

[1] Inria, Paris, France
anne.canteaut@inria.fr
[2] Horst Görtz Institute for IT-Security, Ruhr University Bochum, Germany
firstname.lastname@rub.de

Abstract. We give the first practical instance – BISON – of the Whitened Swap-Or-Not construction. After clarifying inherent limitations of the construction, we point out that this way of building block ciphers allows easy and very strong arguments against differential attacks.

Keywords: Whitened Swap-Or-Not · Instantiating Provable Security · Block Cipher Design · Differential Cryptanalysis

1 Introduction

Block ciphers are among the most important cryptographic primitives as they are at the core responsible for a large fraction of all our data that is encrypted. Depending on the mode of operation (or used construction), a block cipher can be turned into an encryption function, a hash-function, a message authentication code or an authenticated encryption function.

Due to their importance, it is not surprising that block ciphers are also among the best understood primitives. In particular the Advanced Encryption Standard (AES) [Fip] has been scrutinized by cryptanalysts ever since its development in 1998 [DR98] without any significant security threat discovered for the full cipher (see e. g. [BK09; Bir+09; Der+13; Dun+10; Fer+01; GM00; Gra+16; Gra+17; Røn+17]).

The overall structure of AES, being built on several (round)-permutations interleaved with a (binary) addition of round keys is often referred to as key-alternating cipher and is depicted in Figure 1.

The first cipher following this approach was, to the best of our knowledge, the cipher MMB [Dae+93], while the name key-alternating cipher first appears in [DR01] and in the

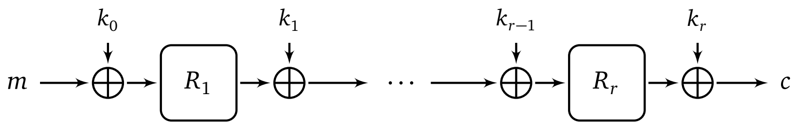


Figure 1: Key-alternating construction for r rounds, using unkeyed round permutations R_1 to R_r. In practical instantiations, the round keys k_i are typically derived from a master key by some key schedule.

book describing the design of the AES [DR02]. Nowadays many secure ciphers follow this construction.

Interestingly, besides its overwhelming use in practice and the intense cryptanalytic efforts spent to understand its practical security, the generic (or idealized) security of key-alternating ciphers has not been investigated until 2012. Here, generic or idealized security refers to the setting where the round functions R_i are modeled as random permutations. An (computational unbounded) attacker is given access to those round functions via oracle queries and additional oracle access to either the block cipher or a random permutation. The goal of the attacker is to tell apart those two cases. As any attack in this setting is obviously independent of any particular structure of the round function, those attacks are generic for all key-alternating ciphers. In this setting, the construction behind key-alternating ciphers is referred to as the iterated Even-Mansour construction. Indeed, the Even-Mansour cipher [EM97] can be seen as a one-round version of the key-alternating cipher where the round function is a random permutation.

The first result on the iterated Even-Mansour construction (basically focusing on the two-round version) was given in [Bog+12]. Since then, quite a lot of follow-up papers, e. g. [And+13; GL15; HT16; LS15], managed to improve and generalize this initial result significantly. In particular, [CS14] managed to give a tight security bound for any number of rounds. Informally, for breaking the r-round Even-Mansour construction, any attacker needs to make roughly $2^{\frac{r}{r+1}n}$ oracle queries.

While this bound can be proven tight for the iterated Even-Mansour construction, it is unsatisfactory for two reasons. First, one might hope to get better security bounds with different constructions and second one might hope to lower the requirement of relying on r random permutations.

Motivated by this theoretical defect and the importance of encrypting small domains with full security (see e. g. [MY17]), researchers started to investigate alternative ways to construct block ciphers with the highest possible security level under minimal assumptions in ideal models. The most interesting result along those lines is the construction by Tessaro [Tes15b]. His construction is based on the Swap-or-Not construction by [Hoa+12], which was designed for the setting where the component functions are secret. Instead of being based on random permutations, this construction requires only a set of random (Boolean) functions. Tessaro's construction, coined Whitened Swap-Or-Not (WSN for short), requires only two public random (Boolean) functions f_i with n-bit input, and can be proven to achieve full security, see Section 2 for more details.

However, and this is the main motivation for our work, *no instance of this construction is known*. This situation is in sharp contrast to the case of the iterated Even-Mansour construction, where many secure instances are known for a long time already, as discussed above.

Without such a concrete instance, the framework of [Tes15b] remains of no avail. As soon as one wants to use the framework in any way, one fundamentally has to instantiate the Boolean functions modeled as ideal functionalities by efficiently computable functions. Clearly, the above mentioned bound in the ideal model does not say anything about any concrete instance. Tessaro phrases this situation as follows:

> Heuristically, however, one hopes for even more: Namely, that under a careful implementation of the underlying component, the construction retains the promised security level. [Tes15b]

There has actually been one instance of the previous construction [Hoa+12], but this has been broken almost instantaneously and completely, as parts of the encryption function were actually linear, see [Vau12]. This failure to securely instantiate the construction points to an important hurdle. Namely, proving the generic bounds and analyzing the security of an instance are technically very different tasks. The security of any block cipher is, with the current state of knowledge, always the security against known attacks. In particular, when designing any concrete block cipher, one has to argue why linear and differential attacks do not threaten the construction.

Our Contribution

Consequently, in this paper we investigate the important, but so far overlooked, aspect of instantiating the WSN construction with a practical secure instance. Practical secure meaning, just like in the case of AES, that the block cipher resists all known attacks. We denote this instance as **BISON** (for Bent whItened Swap Or Not). Our insights presented here are twofold.

First, we derive some inherent restrictions on the choice of the round function f_i. In a nutshell, we show that f_i has to be rather strong, in the sense that its output bit has to basically depend on all input bits. Moreover, we show that using less than n rounds will always result in an insecure construction. Those, from a cryptanalytic perspective rather obvious, results are presented in Section 3. Again, but from a different angle, this situation is in sharp contrast to key-alternating ciphers. In the case of key-alternating ciphers, even with a rather small number of rounds (e. g. ten in the case of AES-128) and rather weak round functions (in case of the AES round function any output bit depends on 32 input bits only and the whole round function decomposes into four parallel functions on 32 bits each) we get ciphers that provide, to the best of our knowledge today and after a significant amount of cryptanalysis, full security.

Second, despite those restrictions of the WSN construction, that have significant impact on the performance of any instance, there are very positive aspects of the WSN construction as well. In Section 4, we first define a family of WSN instances which fulfill our initial restrictions.

As we will show in detail, this allows to argue very convincingly that our instance is secure against differential attacks. Indeed, under standard assumptions, we can show that the probability of any (non-trivial) *differential* is upper bounded by 2^{-n+1} where n is the block size, a value that is close to the ideal case. This significantly improves upon what is the state of the art for key-alternating ciphers. Deriving useful bounds on differentials is notoriously hard and normally one therefore has to restrict to bounding the probability of *differential characteristics* only. Our results for differential cryptanalysis can be of independent interest in the analysis of maximally unbalanced Feistel networks or nonlinear feedback shift registers.

We specify our concrete instance as a family of block ciphers for varying input length in Section 5. In our instance, we attach importance to simplicity and mathematical clarity. It is making use of bent functions, i. e. maximally non-linear Boolean functions, for instantiating f and linear feedback shift registers (LFSRs) for generating the round keys. Another advantage of BISON is that it defines a whole family of block ciphers, one for any odd block size. In particular it allows the straightforward definition of small scale variants to be used for experiments.

Finally we discuss various other attacks and argue why they do not pose a threat for BISON in Section 6. Particularly the discussion on algebraic attacks might be of independent interest. For this we analyse the growth of the algebraic degree over the rounds. In contrast to what we intuitively expect – an exponential growth (until a certain threshold) as in the case for SPNs [Bou+11] – the degree of the WSN construction grows linearly in the degree of the round function f_i. This result can also be applied in the analysis of maximally unbalanced Feistel networks or nonlinear feedback shift registers.

Related Work

The first cipher, a Feistel structure, that allowed similarly strong arguments against differential attacks was presented by Nyberg and Knudsen [NK95], see also [Nyb12] for a nice survey on the topic. This cipher was named CRADIC, as Cipher Resistant Against DIfferential Cryptanalysis but is often simply referenced as the KN cipher. However, the cipher has been broken quickly afterwards, with the invention of interpolation attacks [JK97]. Another, technically very different approach to get strong results on resistance against attacks we would like to mention is the decorrelation theory [Vau98]. Interestingly, both previous approaches rely rather on one strong component, i. e. round function, to ensure security, while the WSN approach, and in particular BISON, gains its resistance against differential attacks step by step.

Regarding the analysis of differentials, extensive efforts have been expended to evaluate the MEDP/MELP of SPN ciphers, and in particular of the AES. Some remarkable results were published by [Par+03] and then subsequently improved by [KS07] with a sophisticated pruning algorithm. Interestingly, further work by [DR06] and later by [CR15] revealed that such bounds are not invariant under affine transformations – an equivalence notion often exploited for classification of S-boxes when studying their strength against differential cryptanalysis. All these works stress out how difficult it is to evaluate the MEDP/MELP of SPNs, even for a small number of rounds. On the contrary, and as we are going to elaborate in the remaining of this paper, computing the MEDP of BISON is rather straightforward and independent of the exact details of the components. This can be compared to the wide trail strategy that, making use of the branch number and the superbox argument, allows bounding the probability of any differential characteristic for a large class of SPNs. Our arguments allow to bound the differential probability for a large class of WSN instances.

2 Preliminaries

We briefly recall the Whitened Swap-or-Not construction, recapitulate properties of Boolean functions and shortly cover differential and linear cryptanalysis. We denote by $\mathbb{F}_2$ the finite field with two elements and by $\mathbb{F}_2^n$ the n-dimensional vector space over $\mathbb{F}_2$, i. e. the set of all n-bit vectors with a bitwise xor as the addition.

2.1 Whitened Swap-or-Not

The WSN is defined as follows. Given two round keys k_i, w_i, the ith round R_{k_i,w_i} computes

$$\begin{aligned} R_{k_i,w_i} &: \mathbb{F}_2^n \to \mathbb{F}_2^n \\ R_{k_i,w_i}(x) &:= x + f_{b(i)}(w_i + \max\{x, x + k_i\}) \cdot k_i \end{aligned}$$

where $f_{0,1} : \mathbb{F}_2^n \to \mathbb{F}_2$ are modeled as two ideal random functions, the max function returns the lexicographic biggest value in the input set, and $+$ denotes the addition in $\mathbb{F}_2$ (the bitwise xor). The index $b(i)$ equals zero for the first half of the rounds and one for the second half (see Figure 2 for a graphical overview of the encryption process).

In the remainder of the paper, we denote by $E^r_{k,w}(x)$ the application of r rounds of the construction to the input x with round keys k_i and w_i derived from the master key (k, w). Every round is involutory, thus for decryption one only has to reverse the order of the round keys.

Note that the usage of the maximum function is not decisive but that it can be replaced by any function Φ_k that returns a unique representative of the set $\{x, x + k\}$, see [Tes15b]. In other words it can be replaced by any function such that $\Phi_k(x) = \Phi_k(y)$ if and only if $y \in \{x, x + k\}$.

The main result given by Tessaro on the security of the WSN is the following:

Proposition 1 (Security of the WSN (Informal) [Tes15b]). *The WSN construction with $\mathcal{O}(n)$ rounds is $(2^{n-\mathcal{O}(\log n)}, 2^{n-\mathcal{O}(1)})$-secure.*

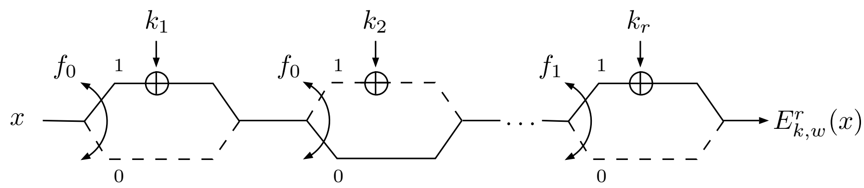


Figure 2: Schematic view of the WSN construction.

Thus, any adversary trying to distinguish the WSN construction from a random permutation and making at most $2^{n-\mathcal{O}(\log n)}$ queries to the block cipher and $2^{n-\mathcal{O}(1)}$ queries to the underlying function has negligible advantage. Here, the round keys are modeled as independent and uniformly distributed random variables.

2.2 Boolean Functions

A *Boolean function* is defined as a function f mapping n bits to one bit. Any Boolean function

$$f : \mathbb{F}_2^n \to \mathbb{F}_2$$

can be uniquely expressed by its algebraic normal form (ANF), i. e. as a (reduced) multivariate polynomial with n variables $x_0, \ldots, x_{n-1}$. For $u \in \mathbb{F}_2^n$ we denote

$$x^u = \prod_{i=0}^{n-1} x_i^{u_i}.$$

The ANF of f can be expressed as

$$f(x) = \sum_{u \in \mathbb{F}_2^n} \lambda_u x^u$$

for suitable choices of $\lambda_u \in \mathbb{F}_2$. The degree of f, denoted by $\deg(f)$ is defined as the maximal weight of a monomial present in the ANF of f. That is

$$\deg(f) = \max\left\{\mathrm{wt}(u) \;\middle|\; u \in \mathbb{F}_2^n \text{ such that } \lambda_u \neq 0\right\}.$$

In the context of symmetric cryptography, the differential and linear behavior of a Boolean function play an important role.

The *derivative* of a function f *in direction* α is defined as $\Delta_\alpha(f)(x) := f(x) + f(x+\alpha)$. Informally, studying the behavior of this derivative is at the core of differential cryptanalysis. If we generalize to the derivative of a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we can additionally specify an output difference β. The differential distribution table (DDT) captures the distribution of all possible derivatives; its entries are

$$\mathrm{DDT}_F[\alpha, \beta] := \left|\left\{x \in \mathbb{F}_2^n \;\middle|\; \Delta_\alpha(F)(x) = \beta\right\}\right|,$$

where we leave out the subscript, if it is clear from the context. Note that α is usually referred to as the input difference and β as the output difference.

In a similar way, we can approach the linear behavior of a Boolean function, that is its similarity to any linear function. The *Fourier coefficient* of a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, which is defined as

$$\widehat{f}(\alpha) := \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x\rangle + f(x)},$$

is a very useful way to measure this similarity. Here, the notation $\langle a, b\rangle$ denotes the inner product, defined as $\langle a, b\rangle := \sum_{i=1}^{n} a_i b_i$. Recall that any affine Boolean function can be written as $x \mapsto \langle \alpha, x\rangle + c$ for some fixed $\alpha \in \mathbb{F}_2^n$ and a constant $c \in \mathbb{F}_2$. In particular, it follows that any such affine function has one Fourier coefficient equal to $\pm 2^n$. More generally, the *nonlinearity* of f, defined as $\mathrm{NL}(f) := 2^n - \max_\alpha |\widehat{f}(\alpha)|$, measures the minimal Hamming-distance of f to the set of all affine functions.

Analogously to the DDT, for a vectorial Boolean function $F : \mathbb{F}_2^n \to \mathbb{F}_2^n$, we define

$$\widehat{F}(\alpha, \beta) = \sum_{x \in \mathbb{F}_2^n} (-1)^{\langle \alpha, x\rangle + \langle \beta, F(x)\rangle},$$

and the linear approximation table (LAT) contains the Fourier coefficients

$$\mathrm{LAT}_F[\alpha, \beta] := \widehat{F}(\alpha, \beta).$$

Again we leave out the subscript, if it is clear from the context. Here α is usually referred to as the input mask and β as the output mask. Another representation that is sometimes preferred is the *correlation matrix* that in a similar way contains the correlation values for all possible linear approximations, see [Dae+95]. The correlation values are simply scaled versions of the Fourier coefficients, i. e.

$$\Pr[\langle \alpha, x\rangle + \langle \beta, F(x)\rangle = 0] = \frac{1}{2} + \frac{\mathbf{cor}_F(\alpha, \beta)}{2} = \frac{1}{2} + \frac{\widehat{F}(\alpha, \beta)}{2^{n+1}}.$$

The advantage of the correlation matrix notation is that the correlation matrix of a composition of functions is nothing but the product of the corresponding matrices. For the linear approximation table, additional scaling is required.

2.2.1 Bent Functions.

As they will play an important role in our design of BISON, we recall the basic facts of bent functions. Boolean functions on an even number n of input bits that achieve the highest possible nonlinearity of $2^n - 2^{\frac{n}{2}}$ are called *bent*. Bent functions have been introduced by Rothaus [Rot76] and studied ever since, see also [Car07, Section 8.6]. Even so bent functions achieve the highest possible nonlinearity, their direct use in symmetric cryptography is so far very limited. This is mainly due to the fact that bent functions are not balanced, i. e. the distribution of zeros and ones is (slightly) biased.

Using Parseval's equality, it is easy to see that a function is bent if and only if all its Fourier coefficients are $\pm 2^{\frac{n}{2}}$. Moreover, an alternative classification that will be of importance for BISON, is that a function is bent if and only if all (non-trivial) derivatives $\Delta_\alpha(f)$ are balanced Boolean functions [MS90].

While there are many different primary and secondary constructions[1] of bent functions known, for simplicity and for the sake of ease of implementation, we decided to focus on the simplest known bent functions which we recall next, see also [Car07, Section 6.2].

Lemma 1 ([Dil72])**.** *Let $n = 2m$ be an even integer. The function*

$$\begin{aligned} f : \mathbb{F}_2^m \times \mathbb{F}_2^m &\to \mathbb{F}_2 \\ f(x, y) &:= \langle x, y\rangle \end{aligned}$$

is a quadratic bent function. Moreover, any quadratic bent function is affine equivalent to f.

[1] Primary constructions give bent functions from scratch, while secondary constructions build new bent functions from previously defined ones.

2.3 Differential and Linear Cryptanalysis

The two most important attacks on symmetric primitives are *differential* and *linear* cryptanalysis, respectively developed by Biham and Shamir [BS91] and by Matsui [Mat95] for the analysis of the Data Encryption Standard. The general idea for both is to find a non-random characteristic in the differential, resp. linear, behavior of the scheme under inspection. Such a property can then be used as a distinguisher between the cipher and a random permutation and in many cases leads to key-recovery attacks.

It is inherently hard to compute these properties for the whole function, thus one typically exploits the special structure of the cipher. For round-based block ciphers one usually makes use of linear and differential characteristics that specify not only the input and output masks (resp. differences) but also all intermediate masks after the single rounds.

In the case of differential cryptanalysis, an r-round characteristic δ is defined by $(r+1)$ differences

$$\delta = (\delta_0, \ldots, \delta_r) \in \mathbb{F}_2^{(r+1)n}.$$

For so-called Markov ciphers and assuming the *independence of round keys*, we can compute the probability of a characteristic *averaged over all round-key sequences*:

$$\mathrm{EP}(\delta) = \prod_{i=0}^{r-1} \Pr[F(x) + F(x+\delta_i) = \delta_{i+1}] = \prod_{i=0}^{r-1} \frac{\mathrm{DDT}_F[\delta_i, \delta_{i+1}]}{2^n},$$

where the encryption iterates the round function F for r rounds. Moreover we usually assume the *hypothesis of stochastic equivalence* introduced by Lai et al. [Lai+91], stating that the actual probability for any fixed round key equals the average.

In contrast to the normal *characteristic* that defines the exact differences before and after each round, a *differential* takes every possible intermediate differences into account and fixes only the overall input and output differences (which are the two values an attacker can typically control).

However, while bounding the average probability of a differential characteristic is easily possible for many ciphers (using in particular the wide-trail strategy introduced in [Dae95]), bounding the average probability of a differential, which is denoted as the expected differential probability (EDP), is not. Nevertheless, some effort was spent to prove bounds on the maximum EDP (MEDP) for two rounds of some key-alternating ciphers [CR15; DR02; Hon+01; Par+03].

Similarly, for linear cryptanalysis, an r-round characteristic (also called trail or path) for a round function F is defined by $(r+1)$ masks

$$\theta = (\theta_0, \ldots, \theta_r) \in \mathbb{F}_2^{(r+1)n}$$

and its correlation is defined as

$$\mathbf{cor}_F(\theta) := \prod_{i=0}^{r-1} \mathbf{cor}_F(\theta_i, \theta_{i+1}) = \prod_{i=0}^{r-1} \frac{\widehat{F}(\theta_i, \theta_{i+1})}{2^n}$$

and it can be shown that the correlation of a composition can be computed as the sum over the trail correlations. More precisely,

$$\mathbf{cor}_{E_k^r}(\alpha, \beta) = \sum_{\substack{\theta \\ \theta_0=\alpha, \theta_r=\beta}} \mathbf{cor}_F(\theta), \tag{1}$$

where the encryption E_k^r iterates the round function F for r rounds.

This is referred to as the linear hull (see [Nyb95]). While not visible in order to simplify notation, the terms in Eq. (1) are actually key dependent and thus for some keys they either could cancel out or amplify the overall correlation. For more background, we refer to e. g. [BN16] and [Kra+17]. For a key-alternating cipher with independent round keys, the average over all round-key sequences of the correlation $\mathbf{cor}_{E_k^r}(\alpha, \beta)$ is zero for any pair of nonzero masks (α, β) (see e.g. [DR02, Section 7.9]). Then, the most relevant parameter of the distribution is its variance, which corresponds to the average square correlation, and is called the *expected linear potential*. Again, bounding the ELP is out of reach for virtually any practical cipher, while for bounding the correlation of a single trail, one can again use the wide-trail strategy mentioned above. Upper bounds for the MELP of two rounds of AES are also given in [CR15; Hon+01; Par+03].

Finally we would like to note that the round keys in an actual block cipher instance are basically never independent and identically distributed over the full key space, but instead derived from a key schedule, rendering the above assumption plain wrong. While the influence of key schedules is a crucially understudied topic and for specific instances strange effects can occur, see [Abd+12; Kra+17], the above assumption are seen as valid heuristics for most block ciphers.

3 Inherent Restrictions

In this section we point out two inherent restrictions on any practical secure instance, i. e. generic for the WSN construction. Those restrictions result in general conditions on both the minimal number of rounds to be used and general properties of the round functions $f_{b(i)}$. In particular, those insights are taken into account for BISON. While these restrictions are rather obvious from a cryptanalytic point of view, they have a severe impact on the performance of any concrete instance. We discuss performance in more detail in Section 7.

3.1 Number of Rounds

As in every round of the cipher, we simply add (or not) the current round key k_i, the ciphertext can always be expressed as the addition of the plaintext and a (message dependent) linear combination of all round keys k_i. The simple but important observation to be made here is that, as long as the round keys do not span the full space, the block cipher is easily attackable.

Phrased in terms of linear cryptanalysis we start with the following lemma.

Lemma 2. *For any number of rounds $r < n$ there exists an element $u \in \mathbb{F}_2^n \setminus \{0\}$ such that*

$$\widehat{E_{k,w}^r}(u, u) = 2^n,$$

that is the equation

$$\langle u, x \rangle = \left\langle u, E_{k,w}^r(x) \right\rangle$$

holds for all $x \in \mathbb{F}_2^n$.

Proof. Let $k_1, \dots, k_r$ be the round keys derived from k and denote by

$$U = \operatorname{span}\{k_1, \dots, k_r\}^\perp$$

the dual space of the space spanned by the round keys, i. e.

$$\forall u \in U, \forall 1 \leqslant i \leqslant r \text{ it holds that } \langle u, k_i \rangle = 0.$$

As $r < n$ by assumption, the dimension of $\operatorname{span}\{k_1, \dots, k_r\}$ is smaller than n and thus $U \neq \{0\}$. Therefore, U contains a non-zero element

$$u \in \operatorname{span}\{k_1, \dots, k_r\}^\perp$$

and it holds that

$$\left\langle u, E^r_{k,w}(x)\right\rangle = \langle u, x + \sum_{i=1}^{r} \lambda_i k_i\rangle = \langle u, x\rangle + \langle u, \sum_{i=1}^{r} \lambda_i k_i\rangle = \langle u, x\rangle. \tag{2}$$

Even more importantly, this observation leads directly to a known plaintext attack with very low data-complexity. Given a set of t plaintext/ciphertext (p_i, c_i) pairs, an attacker simply computes

$$V = \operatorname{span}\{p_i + c_i \mid 1 \leqslant i \leqslant t\} \subseteq \operatorname{span}\left\{k_j \;\middle|\; 1 \leqslant j \leqslant r\right\}.$$

Given $t > r$ slightly more pairs than rounds, and assuming that $p_i + c_i$ is uniformly distributed in span $\{k_j\}$ (otherwise the attack only gets even stronger)[2] implies that

$$V = \operatorname{span}\left\{k_j\right\}$$

with high probability and V can be efficiently computed. Furthermore, as above dim(span $\{k_j\}$) is at most r, we have $V^{\perp} \neq \{0\}$. Given any $u \neq 0$ in $V^{\perp}$ allows to compute one bit of information on the plaintext given only the ciphertext and particularly distinguish the cipher from a random permutation in a chosen-plaintext setting efficiently.

A similar argument shows the following:

Lemma 3. *For any number of rounds r smaller than $2n-3$ there exist nonzero α and β, such that*

$$\widehat{E^r_{k,w}}(\alpha, \beta) = 0$$

Proof. We restrict to the case $r \geqslant n$ as otherwise the statement follows directly from the lemma above. Indeed, from Parseval equality, the fact that $\widehat{E^r_{k,w}}(\alpha, \alpha) = 2^n$ implies that $\widehat{E^r_{k,w}}(\alpha, \beta) = 0$ for all $\beta \neq \alpha$. Let $k_1, \ldots, k_r$ be the round keys derived from k and choose non-zero elements $\alpha \neq \beta$ such that

$$\alpha \in \operatorname{span}\{k_1, \ldots, k_{n-2}\}^{\perp} \qquad \text{and} \qquad \beta \in \operatorname{span}\{k_{n-1}, \ldots, k_r\}^{\perp}.$$

Note that, as $r \leq 2n-3$ by assumption such elements always exist. Next, we split the encryption function in two parts, the first $n-2$ rounds E_1 and the remaining $r-(n-2) < n$ rounds E_2, i.e.

$$E^r_{k,w} = E_2 \circ E_1.$$

We can compute the Fourier coefficient of $E^r_{k,w}$ as

$$\widehat{E^r_{k,w}}(\alpha, \beta) = \sum_{\gamma \in \mathbb{F}_2^n} \frac{\widehat{E_1}(\alpha, \gamma)}{2^n} \cdot \frac{\widehat{E_2}(\gamma, \beta)}{2^n}.$$

Now, the above lemma and the choices of α and β imply that $\widehat{E_1}(\alpha, \gamma) = 0$ for $\gamma \neq \alpha$ and $\widehat{E_2}(\gamma, \beta) = 0$ for $\gamma \neq \beta$. Recalling that $\alpha \neq \beta$ by construction concludes the proof.

However, as the masks α and β depend on the key, and unlike above there does not seem to be an efficient way to compute those, we do not see a direct way to use this observation for an attack.

Summarizing the observations above, we get the following conclusion:

[2] E. g. if, with high probability, the $p_i + c_i$ do not depend on one or more k_j's, the described attack can be extended to one or more rounds with high probability.

Rationale 1. *Any practical instance must iterate at least n rounds. Furthermore, it is beneficial if any set of n consecutive round keys are linearly independent.*[3]

After having derived basic bounds on the number of rounds for any secure instance, we move on to criteria on the round function itself.

3.2 Round Function

Here, we investigate a very basic criterion on the round function, namely dependency on all input bits, when the round function of $E^r_{k,w}$ is defined by

$$R_{k_i,w_i}(x) = x + f_{b(i)}(w_i + \max\{x, x + k_i\}) \cdot k_i \,.$$

Given the Boolean functions $f_{b(i)} : \mathbb{F}_2^n \to \mathbb{F}_2$, the question we would like to discuss is, if it is necessary that the output bit of $f_{b(i)}$ has to depend on all input bits. The function $f_{b(i)}$ depends on an input bit j if there are two inputs x, x' differing only in the jth bit such that $f_{b(i)}(x) \neq f_{b(i)}(x')$. Otherwise the function is independent of the jth bit and we get

$$f_{b(i)}(x) = f_{b(i)}(x + e_j)$$

for all x where e_j is the jth canonical basis vector, i. e. e_j has a single one at position j.

We denote by $N(x) := \{i \mid x[i] = 1\}$ the index set of 1-bits in x, and by $\nu(x) := \max N(x)$ the index of the highest 1-bit in x, in other words $\nu(x) = \lfloor \log_2(x) \rfloor$, when interpreting $\delta \in \mathbb{F}_2^n$ as an integer. For the main observation on this criterion, we first need the following lemma.

Lemma 4. *Let $x, \delta \in \mathbb{F}_2^n$ and k uniformly randomly drawn from $\mathbb{F}_2^n$. Then*

$$\Pr[\max\{x + \delta, x + \delta + k\} = \max\{x, x + k\} + \delta] \geqslant 1 - 2^{\nu(\delta)-n}.$$

Proof. The equality depends on the highest bit of δ where x and $x + k$ differ, which is basically $\nu(k)$. We have

$$\Pr[\max\{x + \delta, x + \delta + k\} = \max\{x, x + k\} + \delta] = \Pr[\delta[\nu(k)] = 0],$$

which can also be written as

$$\Pr[\delta[\nu(k)] = 0] = 1 - \Pr[\nu(k) \in N(\delta)] = 1 - \sum_{i \in N(\delta)} \Pr[\nu(k) = i].$$

Further we have $\Pr[\nu(k) = i] = 2^{i-n-1}$ and thus

$$1 - \sum_{i \in N(\delta)} \Pr[\nu(k) = i] = 1 - \sum_{i \in N(\delta)} 2^{i-n-1} \geqslant 1 - 2^{\nu(\delta)-n},$$

which concludes the proof.

As we will see next, the functions $f_{b(i)}$ have to depend virtually on all *linear combinations of bits*. In other words, it is required that the functions $f_{b(i)}$ have no (non-trivial) derivative equal to the all-zero function.

Lemma 5. *If there exists a $\delta \in \mathbb{F}_2^n$ such that*

$$f_{b(i)}(x) = f_{b(i)}(x + \delta)$$

for all x and $i \in \{0, 1\}$ then

$$\Pr\left[E^r_{k,w}(x) + E^r_{k,w}(x + \delta) = \delta\right] \geqslant \left(1 - 2^{\nu(\delta)-n}\right)^r$$

and the probability is over the input x and the keys k and w.

[3] If (some) round keys are linearly dependent, Lemma 3 can easily be extended to more rounds.

Proof. From Lemma 4 we have that

$$\Pr[\max\{x+\delta, x+\delta+k\} = \max\{x, x+k\}+\delta] \geqslant 1-2^{\nu(\delta)-n}.$$

Now, we get for one round

$$R_{k_i,w_i}(x) = x + f_{b(i)}(w_i + \max\{x, x+k_i\}) \cdot k_i$$

by the assumption that $f_{b(i)}(x) = f_{b(i)}(x+\delta)$ for all x

$$R_{k_i,w_i}(x+\delta) = R_{k_i,w_i}(x) + \delta$$

with the same probability. Thus, for r rounds and uniformly chosen keys, we get

$$\Pr\left[E^r_{k,w}(x) + E^r_{k,w}(x+\delta) = \delta\right] \geqslant \left(1-2^{\nu(\delta)-n}\right)^r$$

by induction. ∎

As an example, considering the case $n = 128$, r rounds, and both f_0 and f_1 that do not depend on the most significant byte. Thus, we can choose δ as a unit vector with $\nu(\delta) = 121$ and get a differential probability of

$$\Pr\left[E^r_{k,w}(x) + E^r_{k,w}(x+\delta) = \delta\right] \geqslant \left(1-2^{121-128}\right)^r \approx (0.36)^{r/n}$$

which would completely compromise the scheme for a reasonable number of rounds. In general this shows that as long as both $f_{b(i)}$ do not depend on almost all bits, the scheme is immediately broken by differential cryptanalysis. Now, one might hope that one could craft functions f_0 and f_1 where, e. g. f_0 depends only on the first $\frac{n}{2}$ bits and f_1 on the last $\frac{n}{2}$ bits to overcome this restriction. However, while such a construction might be secure against basic differential cryptanalysis, it would still be completely broken by boomerang attacks [Wag99]. The main idea of boomerang attacks is to split the whole block cipher in two parts such that one has a high probable differential for the first part and a second high probable differential for the second part, which is exactly the situation one would end up here.

Thus, both functions independently have to virtually depend on all input bits, and we deduce the following.

Rationale 2. *For a practical instance, the functions $f_{b(i)}$ has to depend on all bits. Even more, for any $\delta \in \mathbb{F}_2^n$ the probability of*

$$f_{b(i)}(x) = f_{b(i)}(x+\delta)$$

should be close to $\frac{1}{2}$.

It is worth noticing that the analysis leading to this rationale applies to the original round function. However, as pointed out in [Tes15a, Section 3.1], in the definition of the round function, we can replace the function

$$x \mapsto \max\{x, x+k\}$$

by any function Φ_k such that $\Phi_k(x) = \Phi_k(x+k)$ for all x. While the following sections will focus on the case when Φ_k is linear, we proved that Rationale 2 is also valid in this other setting.

Again, this should be compared to key-alternating ciphers, where usually not all output bits of a single round function depend on all input bits. For example for AES any output bit after one round depends only on 32 input bits and for PRESENT any output bit only depends on 4 input bits. However, while for key-alternating ciphers this does not seem to be problematic, and indeed allows rather weak round functions to result in a secure scheme, for the WSN construction the situation is very different.

Before specifying our exact instance, we want to discuss differential cryptanalysis of a broader family of instances.

4 Differential Cryptanalysis of BISON-like instances

We coin an instance of the WSN construction "BISON-like", if it iterates at least n rounds with linearly independent round keys $k_1, \ldots, k_n$ and applies Boolean functions $f_{b(i)}$. As explained in [Tes15a, Section 3.1], in order to enable decryption it is required that the Boolean functions $f_{b(i)}$ return the same result for both x and $x+k$. In the original proposition by Tessaro, this is achieved by using the max function in the definition of the round function. Using this technique reduces the number of possible inputs for the $f_{b(i)}$ to 2^{n-1}. To simplify the analysis and to ease notation, we replace the max function with a *linear function* $\Phi_k : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$ with $\ker(\Phi_k) = \{0, k\}$. From now on, we assume that any BISON-like instance uses such a Φ_k instead of the max function. The corresponding round function has then the following form

$$R_{k_i,w_i}(x) := x + f_{b(i)}\big(w_i + \Phi_{k_i}(x)\big)k_i. \tag{3}$$

With the above conditions, any BISON-like instance of the WSN construction is resistant to differential cryptanalysis, as we show in the remainder of this section.

For our analysis, we make two standard assumptions in symmetric cryptanalysis as mentioned above: the *independence of whitening round keys w_i* and the *hypothesis of stochastic equivalence* with respect to these round keys. That is, we assume round keys w_i to be independently uniformly drawn and the resulting EDP to equal the differential probabilities averaged over all w. In the following sections, we will argue why these assumptions do fit to our design and back up the results by practical experiments (see Section 6.3.7 and Appendix B). For the round keys k_i we do not have to make such assumptions.

We first discuss the simple case of differential behaviour for one round only and then move up to an arbitrary number of rounds and devise the number of possible output differences and their probabilities.

4.1 From One-Round Differential Characteristics

Looking only at one round, we can compute the DDT explicitly:

Proposition 2. *Let $R_{k_i,w_i} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the WSN round function as in Eq. (3). Then its DDT consists of the entries*

$$\mathrm{DDT}_R[\alpha,\beta] = \begin{cases} 2^{n-1} + \widehat{\Delta_{\Phi_k(\alpha)}(f)}(0) & \text{if } \beta = \alpha \\ 2^{n-1} - \widehat{\Delta_{\Phi_k(\alpha)}(f)}(0) & \text{if } \beta = \alpha + k\ \cdot \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

Most notably, if f is bent, we have

$$\mathrm{DDT}_R[\alpha,\beta] = \begin{cases} 2^n & \text{if } \alpha = \beta = k \text{ or } \alpha = \beta = 0 \\ 2^{n-1} & \text{if } \beta \in \{\alpha, \alpha + k\} \text{ and } \alpha \notin \{0, k\}\ . \\ 0 & \text{otherwise.} \end{cases}$$

Proof. We have to count the number of solutions of $R(x) + R(x+\alpha) = \beta$:

$$\begin{aligned} \mathrm{DDT}_R[\alpha,\beta] &= \left|\left\{x \in \mathbb{F}_2^n \;\middle|\; R(x) + R(x+\alpha) = \beta\right\}\right| \\ &= \left|\left\{x \in \mathbb{F}_2^n \;\middle|\; \alpha + [f(w + \Phi_k(x)) + f(w + \Phi_k(x+\alpha))] \cdot k = \beta\right\}\right| \end{aligned}$$

Since f takes its values in $\mathbb{F}_2$, the only output differences β such that $\mathrm{DDT}_R[\alpha,\beta]$ may differ from 0 are $\beta = \alpha$ and $\beta = \alpha + k$. When $\beta = \alpha$, we have

$$\begin{aligned}
\mathrm{DDT}_R[\alpha,\alpha] &= \left|\left\{x \in \mathbb{F}_2^n \,\middle|\, f(w + \Phi_k(x)) + f(w + \Phi_k(x+\alpha)) = 0\right\}\right| \\
&= \left|\left\{x \in \mathbb{F}_2^n \,\middle|\, f(w + \Phi_k(x)) + f(w + \Phi_k(x) + \Phi_k(\alpha)) = 0\right\}\right| \\
&= 2 \cdot \left|\left\{x' \in \mathbb{F}_2^{n-1} \,\middle|\, f(x') + f(x' + \Phi_k(\alpha)) = 0\right\}\right| \\
&= 2\left(2^{n-2} + \frac{1}{2}\widehat{\Delta_{\Phi_k(\alpha)}(f)}(0)\right).
\end{aligned}$$

Similarly,

$$\begin{aligned}
\mathrm{DDT}_R[\alpha,\alpha+k] &= \left|\left\{x \in \mathbb{F}_2^n \,\middle|\, f(w + \Phi_k(x)) + f(w + \Phi_k(x+\alpha)) = 1\right\}\right| \\
&= 2\left(2^{n-2} - \frac{1}{2}\widehat{\Delta_{\Phi_k(\alpha)}(f)}(0)\right).
\end{aligned}$$

Most notably, when $\alpha \in \{0, k\}$, $\widehat{\Delta_{\Phi_k(\alpha)}(f)}(0) = 2^{n-1}$. Moreover, when f is bent, $\widehat{\Delta_{\Phi_k(\alpha)}(f)}(0) = 2^{n-2}$ for all other values of α.

4.2 To Differentials over more Rounds

As previously explained, it is possible to estimate the probability of a differential characteristic over several rounds, averaged over the round keys, when the cipher is a Markov cipher. We now show that this assumption holds for any BISON-like instance of the WSN construction.

Lemma 6. *Let $R_{k,w} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the WSN round function as in Eq. (3). For any fixed $k \in \mathbb{F}_2^n$ and any differential $(\alpha, \beta) \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have that*

$$\Pr_w\left[R_{k,w}(x+\alpha) + R_{k,w}(x) = \beta\right]$$

is independent of x. More precisely

$$\Pr_w\left[R_{k,w}(x+\alpha) + R_{k,w}(x) = \beta\right] = \Pr_x\left[R_{k,w}(x+\alpha) + R_{k,w}(x) = \beta\right].$$

Proof. We have

$$\begin{aligned}
&\left\{w \in \mathbb{F}_2^{n-1} \,\middle|\, \Delta_\alpha(R_{k,w})(x) = \beta\right\} \\
=\ &\left\{w \in \mathbb{F}_2^{n-1} \,\middle|\, \left(\Delta_{\Phi_k(\alpha)}(f)(w + \Phi_k(x))\right) \cdot k = \alpha + \beta\right\} \\
=\ &\begin{cases}
\emptyset & \text{if } \beta \notin \{\alpha, \alpha+k\} \\
\Phi_k(x) + \mathsf{Supp}\left(\Delta_{\Phi_k(\alpha)}(f)\right) & \text{if } \beta = \alpha + k \\
\Phi_k(x) + \left(\mathbb{F}_2^{n-1} \setminus \mathsf{Supp}\left(\Delta_{\Phi_k(\alpha)}(f)\right)\right) & \text{if } \beta = \alpha,
\end{cases}
\end{aligned}$$

where $\mathsf{Supp}(g)$ denotes the support of a Boolean function g, i. e., the values x for which $g(x) = 1$. Clearly, the cardinality of this set does not depend on x. Moreover, this cardinality, divided by 2^{n-1}, corresponds to the value of

$$\Pr_x\left[R_{k,w}(x+\alpha) + R_{k,w}(x) = \beta\right]$$

computed in the previous proposition.

By induction on the number of rounds, we then directly deduce that any BISON-like instance of the WSN construction is a Markov cipher in the sense of the following corollary.

Corollary 1. *Let $E^i_{k,w}$ denote i rounds of a BISON-like instance of the WSN construction with round function R_{k_i,w_i}. For any number of rounds r and any round keys $(k_1,\dots,k_r)$, the probability of an r-round characteristic δ satisfies*

$$\Pr_w\left[E^i_{k,w}(x)+E^i_{k,w}(x+\delta_0)=\delta_i, \forall 1\leqslant i\leqslant r\right]=\prod_{i=1}^{r}\Pr_x\left[R_{k_i,w_i}(x)+R_{k_i,w_i}(x+\delta_{i-1})=\delta_i\right].$$

For many ciphers several differential characteristics can cluster in a *differential* over more rounds. This is the main reason why bounding the probability of differentials is usually very difficult if possible at all. For BISON-like instances the situation is much nicer; we can actually compute the EDP, i. e., the probabilities of the differentials averaged over all whitening key sequences $(w_1,\dots,w_r)$. This comes from the fact that any differential for less than n rounds contains at most one differential characteristic with non-zero probability. To understand this behavior, let us start by analyzing the EDP (averaged over the w_i) and by determining the number of possible output differences.

In the following, we assume that the input difference α is fixed, and we calculate the number of possible output differences. We show that this quantity depends on the relation between α and the k_i.

Lemma 7. *Let us consider r rounds of a BISON-like instance of the WSN construction with round function involving Boolean functions $f_{b(i)}$ having no (non-trivial) constant derivative. Assume that the first n round keys $k_1,\dots,k_n$ are linearly independent, and that $k_{n+1}=k_1+\sum_{i=2}^{n}\gamma_i k_i$ for $\gamma_i\in\mathbb{F}_2$. For any non-zero input difference α, the number of possible output differences β such that*

$$\Pr_{w,x}\left[E^r_{k,w}(x+\alpha)+E^r_{k,w}(x)=\beta\right]\neq 0$$

is

$$\begin{cases} 2^r & \text{if } \alpha\notin\operatorname{span}\{k_i\} \text{ and } r<n,\\ 2^r-2^{r-\ell} & \text{if } \alpha=k_\ell+\sum_{i=1}^{\ell-1}\lambda_i^\alpha k_i \text{ and } r\leqslant n,\\ 2^n-1 & \text{if } r>n.\end{cases}$$

Proof. By combining Corollary 1 and Proposition 2, we obtain that the average probability of a characteristic $(\delta_0,\delta_1,\dots,\delta_{r-1},\delta_r)$ can be non-zero only if $\delta_i\in\{\delta_{i-1},\delta_{i-1}+k_i\}$ for all $1\leqslant i\leqslant r$. Therefore, the output difference δ_r must be of the form $\delta_r=\delta_0+\sum_{i=1}^{r}\lambda_i k_i$ with $\lambda_i\in\mathbb{F}_2$. Moreover, for those characteristics, the average probability is non-zero unless there exists $1\leqslant i<r$ such that $|\widehat{\Delta_{\Phi_{k_i}(\delta_i)}(f_{b(i)})}(0)|=2^{n-1}$, i. e. $\Delta_{\Phi_{k_i}(\delta_i)}(f_{b(i)})$ is constant. By hypothesis, this only occurs when $\delta_i\in\{0,k_i\}$, and the impossible characteristics correspond to the case when either $\delta_i=0$ or $\delta_{i+1}=0$. It follows that the valid characteristics are exactly the characteristics of the form

$$\delta_i=\delta_0+\sum_{j=1}^{i}\lambda_j k_j$$

where none of the δ_i vanishes.

- When the input difference $\alpha\notin\operatorname{span}\{k_i\}$, for any given output difference $\beta=\alpha+\sum_{i=1}^{r}\lambda_i k_i$, the r-round characteristic

  $$(\alpha,\alpha+\lambda_1k_1,\alpha+\lambda_1k_1+\lambda_2k_2,\dots,\alpha+\sum_{i=1}^{r}\lambda_i k_i)$$

  is valid since none of the intermediate differences vanishes.

- When $r \leqslant n$ and $\alpha = k_\ell + \sum_{i=1}^{\ell-1} \lambda_i^\alpha k_i$, the only possible characteristic from α to $\beta = \alpha + \sum_{i=1}^{r} \lambda_i k_i$ satisfies

  $$\delta_j = \begin{cases} \sum_{i=1}^{j} (\lambda_i + \lambda_i^\alpha) k_i + \sum_{i=j+1}^{\ell} \lambda_i^\alpha k_i & \text{if } j \leqslant \ell \\ \sum_{i=1}^{\ell} (\lambda_i + \lambda_i^\alpha) k_i + \sum_{i=\ell+1}^{j} \lambda_i k_i & \text{if } j > \ell \ . \end{cases}$$

  Since the involved round keys are linearly independent, we deduce that $\delta_j = 0$ only when $j = \ell$ and $\lambda_i = \lambda_i^\alpha$ for all $1 \leqslant i \leqslant \ell$. It then follows that there exists a valid characteristic from α to β unless $\lambda_i = \lambda_i^\alpha$ for all $1 \leqslant i \leqslant \ell$. The number of possible outputs β is then

  $$(2^\ell - 1)2^{r-\ell} = 2^r - 2^{r-\ell}.$$

- If we increase the number of rounds to more than n, we have $\alpha = k_\ell + \sum_{i=1}^{\ell-1} \lambda_i^\alpha k_i$ for some $\ell \leqslant n$. If $\beta = \alpha + \sum_{i=1}^{n} \lambda_i k_i$ with $\sum_{i=1}^{\ell} \lambda_i k_i \neq \alpha$, then we can obviously extend the previous n-round characteristic to

  $$(\alpha, \alpha + \lambda_1 k_1, \ldots, \alpha + \sum_{i=1}^{n-1} \lambda_i k_i, \beta, \beta, \ldots, \beta).$$

  If $\sum_{i=1}^{\ell} \lambda_i k_i = \alpha$, β cannot be the output difference of an n-round characteristic. However, the following $(n+1)$-round characteristic starting from $\delta_0 = \alpha$ is valid:

  $$\delta_j = \begin{cases} k_1 + \sum_{i=2}^{j} \gamma_i k_i + \sum_{i=j+1}^{\ell} \lambda_i^\alpha k_i & \text{if } j \leqslant \ell \\ k_1 + \sum_{i=2}^{j} \gamma_i k_i + \sum_{i=\ell+1}^{j} \lambda_i k_i & \text{if } \ell < j \leqslant n \\ \beta & \text{if } j = n+1 \end{cases}$$

  Indeed, $\delta_n = \beta + k_n$ implying that the last transition is valid. Moreover, it can be easily checked that none of these δ_j vanishes, unless $\beta = 0$. This implies that all non-zero output differences β are valid.

The last case in the above lemma is remarkable, as it states any output difference is possible after $n+1$ rounds. To highlight this, we restate it as the following corollary.

Corollary 2. *For BISON-like instances with more than n rounds whose round keys $k_1, \ldots, k_{n+1}$ satisfy the hypothesis of Lemma 7, and for any non-zero input difference, every non-zero output difference is possible.*

We now focus on a reduced version of the cipher limited to exactly n rounds and look at the probabilities for every possible output difference. Most notably, we exhibit in the following lemma an upper-bound on the MEDP which is minimized when n is odd and the involved Boolean functions $f_{b(i)}$ are bent. In other words, Rationale 2 which was initially motivated by the analysis in Section 3 for the original round function based on $x \mapsto \max(x, x+k)$ [Tes15b] is also valid when a linear function Φ_k is used.

Lemma 8. *Let us consider n rounds of a BISON-like instance of the WSN construction with round function involving Boolean functions $f_{b(i)}$. Let $k_1, \ldots, k_n$ be any linearly independent round keys. Then, for any input difference $\alpha \neq 0$ and any β, we have*

$$\begin{aligned} \mathrm{EDP}(\alpha, \beta) &= \Pr_{w,x}\left[E_{k,w}(x+\alpha) + E_{k,w}(x) = \beta\right] \\ &\leqslant \left(\frac{1}{2} + 2^{-n} \max_{1 \leqslant i \leqslant n} \max_{\delta \neq 0} \left|\widehat{\Delta_\delta(f_{b(i)})}(0)\right|\right)^{n-1}. \end{aligned}$$

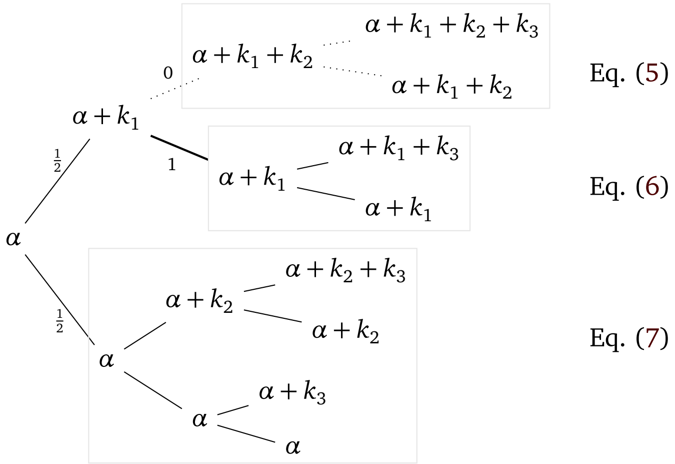


Figure 3: Probabilities of output differences for three rounds and the cases of the input difference $\alpha = k_1 + k_2$, thus $\ell = 2$. Dotted transitions are impossible.

More precisely, if all $f_{b(i)}$ *are* bent,

$$\mathrm{EDP}(\alpha,\beta) = \begin{cases} 0 & \text{if } \beta = \sum_{i=\ell+1}^{n} \lambda_i k_i, \quad (5) \\ 2^{-n+1} & \text{if } \beta = k_\ell + \sum_{i=\ell+1}^{n} \lambda_i k_i, \quad (6) \\ 2^{-n} & \text{otherwise}, \quad (7) \end{cases}$$

where ℓ *denotes as previously the latest round key that appears in the decomposition of* α *into the basis* $(k_1, \ldots, k_n)$*, that is* $\alpha = k_\ell + \sum_{i=1}^{\ell-1} \lambda_i k_i$.

The case of bent functions is visualized in Figure 3, where we give an example of the three possibilities for three rounds.

Proof. As proved in Lemma 7, (α, β) is an impossible differential if and only if $\beta = \sum_{i=\ell+1}^{n} \lambda_i k_i$. For all other values of $\beta = \alpha + \sum_{i=1}^{n} \lambda_i k_i$, we have

$$\mathrm{EDP}(\alpha,\beta) = \prod_{i=1}^{n} \left(\frac{1}{2} + (-1)^{\lambda_i} 2^{-n} \widehat{\Delta_{\Phi_{k_i}(\delta_i)}(f_{b(i)})}(0) \right)$$

where $\delta_i = \alpha + \sum_{j=1}^{i} \lambda_j k_j$. The i-th term in the product is upper-bounded by

$$\frac{1}{2} + 2^{-n} \max_{1 \leqslant i \leqslant n} \max_{\delta \neq 0} \left| \widehat{\Delta_\delta(f_{b(i)})}(0) \right|$$

except if $\Phi_{k_i}(\delta_i) = 0$, i. e., $\delta_i \in \{0, k_i\}$. As seen in Lemma 7, the case $\delta_i = 0$ cannot occur in a valid characteristic. The case $\delta_i = k_i$ occurs if and only if $i = \ell$ and $\beta = k_\ell + \sum_{j=\ell+1}^{n} \lambda_j k_j$. In this situation, the ℓ-th term in the product equals 1. In the tree of differences this is visible as the collapsing of the two branches from two possible succeeding differences into only one, which then of course occurs with probability one, see upper branch of Figure 3.

Most notably, all $f_{b(i)}$ are bent if and only if

$$\max_{1 \leqslant i \leqslant n} \max_{\delta \neq 0} \left| \widehat{\Delta_\delta(f_{b(i)})}(0) \right| = 0 ,$$

leading to the result.

This can be seen on Figure 3: the $2^{n-\ell}$ possible differences coming from the collapsed branch have a transition of probability one in that round, resulting in an overall probability of 2^{-n+1}, see Eq. (6). For the lower part of Figure 3, all the other differences are not affected by this effect and have a probability of 2^{-n}, see Eq. (7).

Because they allow us to minimize the MEDP, we now concentrate on the case of bent functions for the sake of simplicity, which implies that the block size is odd. However, if an even block size is more appropriate for implementation reasons, we could also define BISON-like instances based on maximally nonlinear functions.

It would be convenient to assume in differential cryptanalysis that the EDP of a differential does not increase when adding more rounds, while this does not hold in general. However, this argument can easily be justified for BISON-like instances using bent functions, when averaging over the whitening keys w.

Proposition 3. *Let us consider $r \geqslant n$ rounds of a BISON-like instance of the WSN construction with bent functions $f_{b(i)}$. Let $k_1, \ldots, k_n$ be any linearly independent round keys. Then the probability of any non-trivial differential, averaged over all whitening key sequences w is upper bounded by 2^{-n+1}.*

In other words, the MEDP of BISON-like instances with bent $f_{b(i)}$ for $r \geqslant n$ rounds is 2^{-n+1}.

Proof. By induction over r. The base case for $r = n$ rounds comes from Lemma 8. In the induction step, we first consider the case when the output difference β after r rounds differs from k_r. Then the output difference $\delta_r = \beta$ can be reached if and only if the output difference after $(r-1)$ rounds δ_{r-1} belongs to $\{\beta, \beta + k_r\}$. Then,

$$\begin{aligned}
\mathrm{EDP}^r(\alpha,\beta) &= \Pr_{w_r}\left[R_{k_r,w_r}(x_r) + R_{k_r,w_r}(x_r+\beta) = \beta\right]\mathrm{EDP}^{r-1}(\alpha,\beta) \\
&\quad + \Pr_{w_r}\left[R_{k_r,w_r}(x_r) + R_{k_r,w_r}(x_r+\beta+k_r) = \beta\right]\mathrm{EDP}^{r-1}(\alpha,\beta+k_r) \\
&= \frac{1}{2}\left(\mathrm{EDP}^{r-1}(\alpha,\beta) + \mathrm{EDP}^{r-1}(\alpha,\beta+k_r)\right) \leqslant 2^{-n+1}\,.
\end{aligned}$$

When the output difference β after r rounds equals k_r, it results from $\delta_{r-1} = k_r$ with probability 1. In this case

$$\mathrm{EDP}^r(\alpha,\beta) = \mathrm{EDP}^{r-1}(\alpha,\beta) \leqslant 2^{-n+1}\,.$$

This bound is close to the ideal case, in which each differential has probability $1/(2^n-1)$.

We now give a detailed description of our instance BISON.

5 Specification of BISON

As BISON-like instances should obviously generalise BISON, this concrete instance inherits the already specified parts. Thus BISON uses two bent functions $f_{b(i)}$, replaces the max function by Φ_k, and uses a key schedule that generates round keys, where all n consecutive round keys are linearly independent. The resulting instance for n bits iterates the WSN round function as defined below over $3 \cdot n$ rounds. The chosen number of rounds mainly stems from the analysis of the algebraic degree that we discuss in Section 6.

Security Claim. *We claim n-bit security for BISON in the single-key model. We emphasize that we do not claim any security in the related-key, chosen-key or known-key model.*

5.1 Round function

For any nonzero round key k, we define $\Phi_k : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$ as

$$\Phi_k(x) := (x_{i(k)} \cdot k + x)[1, \ldots, i(k)-1, i(k)+1, \ldots, n], \tag{8}$$

where $i(k)$ denotes the index of the lowest bit set to 1 in k, and the notation $x[1, \ldots, j-1, j+1, \ldots, n]$ returns the $(n-1)$-bit vector, consisting of the bits of x except the jth bit.

Lemma 9. *The function $\Phi_k : \mathbb{F}_2^n \to \mathbb{F}_2^{n-1}$ is linear and satisfies*

$$\ker(\Phi_k) = \{0, k\}.$$

The proof can be done by simply computing both outputs for x and $x+k$.
For the preimage of $y \in \mathbb{F}_2^{n-1}$ and $j = i(k)$ we have

$$\Phi_k^{-1}(y) \in \left\{ \begin{array}{l} (y[1:j-1], 0, y[j:n-1]) + k[1:n], \\ (y[1:j-1], 0, y[j:n-1]) \end{array} \right\}. \tag{9}$$

Due to the requirement for the $f_{b(i)}$ being bent, we are limited to functions taking an even number of bits as input. The simplest example of a bent function is the inner product.
Eventually we end up with the following instance of the WSN round.

BISON's Round Function

For round keys $k_i \in \mathbb{F}_2^n$ and $w_i \in \mathbb{F}_2^{n-1}$ the round function computes

$$R_{k_i,w_i}(x) := x + f_{b(i)}\big(w_i + \Phi_{k_i}(x)\big)k_i. \tag{10}$$

where

- Φ_{k_i} is defined as in Eq. (8),
- $f_{b(i)}$ is defined as

$$\begin{aligned} f_{b(i)} &: \mathbb{F}_2^{n-1} \to \mathbb{F}_2 \\ f_{b(i)}(x) &:= \langle x[1:(n-1)/2], x[(n+1)/2:n] \rangle + b(i), \end{aligned}$$

- and $b(i)$ is 0 if $i \leqslant \frac{r}{2}$ and 1 otherwise.

5.2 Key schedule

In the ith round, the key schedule has to compute two round keys: $k_i \in \mathbb{F}_2^n$ and $w_i \in \mathbb{F}_2^{n-1}$. We compute those round keys as the states of LFSRs after i clocks, where the initial states are given by a master key K. The master key consists of two parts of n and $n-1$ bits, i.e.

$$K = (k, w) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}.$$

As the all-zero state is a fixed point for any LFSR, we *exclude the zero key* for both k and w. In particular $k = 0$ is obviously a weak key that would result in a ciphertext equal to the plaintext $p = E_{0,w}^r(p)$ for all p, independently of w or of the number of rounds r.

It is well-known that choosing a feedback polynomial of an LFSR to be primitive results in an LFSR of maximal period. Clocking the LFSR then corresponds to multiplication of its state with the companion matrix of this polynomial. Interpreted as elements from the finite field, this is the same as multiplying with a primitive element.

In order to avoid structural attacks, e. g. invariant attacks [Gra+16; Lea+11; Tod+16], as well as the propagation of low-weight inputs, we add round constants c_i to the round key w_i.

These round constants are also derived from the state of an LFSR with the same feedback polynomial as the w_i LFSR, initialized to the unit vector with the least significant bit set. To avoid synchronization with the w_i LFSR, the c_i LFSR clocks backwards.

BISON's Key Schedule

For two primitive polynomials $p_w(x)$, $p_k(x) \in \mathbb{F}_2[x]$ with degrees $\deg(p_w) = n-1$ and $\deg(p_k) = n$ and the master key $K = (k, w) \in \mathbb{F}_2^n \times \mathbb{F}_2^{n-1}$, $k, w \neq 0$ the key schedule computes the ith round keys as

$$\begin{aligned} \mathrm{KS}_i : \mathbb{F}_2^n \times \mathbb{F}_2^{n-1} &\to \mathbb{F}_2^n \times \mathbb{F}_2^{n-1} \\ \mathrm{KS}_i(k, w) &:= (C(p_k)^i k, C(p_w)^{-i} e_1 + C(p_w)^i w) = (k_i, c_i + w_i) \end{aligned}$$

where $C(\cdot)$ is the companion matrix of the corresponding polynomial, and $0 \leqslant i < r$. In Appendix A we give concrete polynomials for $5 \leqslant n \leqslant 129$-bit block sizes.

As discussed above, this key schedule has the following property, see also Rationale 1.

Lemma 10. *For BISON's key schedule, the following property holds: Any set of n consecutive round keys k_i are linearly independent. Moreover there exist coefficients λ_i such that*

$$k_{n+i} = k_i + \sum_{j=i+1}^{n+i-1} \lambda_j k_j.$$

Proof. To prove this, we start by showing that the above holds for the first n round keys, the general case then follows from a similar argumentation. We need to show that there exists no non-trivial $c_i \in \mathbb{F}_2$ so that $\sum_{i=1}^{n} c_i C(p_k)^i k = 0$, which is equivalent to showing that there exists no non-trivial $c_i \in \mathbb{F}_2$ so that $\sum_{i=0}^{n-1} c_i C(p_k)^i k = 0$. In this regard, we recall the notion of *minimal polynomial of k with respect to $C(p_k)$*, defined as the monic polynomial of smallest degree $Q_L(k)(x) = \sum_{i=0}^{d} q_i x^i \in \mathbb{F}_2[x]$ such that $\sum_{i=0}^{d} q_i C(p_k)^i k = 0$. Referring to a discussion that has been done for instance in [Bei+17], we know that the minimal polynomial of k is a divisor of the minimal polynomial of $C(p_k)$. Since in our case our construction has been made so that this later is equal to p_k which is a primitive polynomial, we deduce that the minimal polynomial of $k \neq 0$ is p_k itself. Since the degree of p_k is equal to n, this prove that the first n keys are linearly independent.

The equation holds, since $p_k(0) = 1$.

6 Security Analysis

As we have already seen, BISON is resistant to differential cryptanalysis. In this section, we argue why BISON is also resistant to other known attacks.

6.1 Linear Cryptanalysis

For linear cryptanalysis, given the fact that BISON is based on a bent function, i. e. a maximally non-linear function, arguing that no linear characteristic with high correlation exist is rather easy. Again, we start by looking at the Fourier coefficients for one round.

6.1.1 From one Round.

Using the properties of f being bent, we get the following.

Proposition 4. *Let $R_{k,w} : \mathbb{F}_2^n \to \mathbb{F}_2^n$ be the round function as defined in Eq. (10). Then, its LAT consists of the entries*

$$\widehat{R_{k,w}}(\alpha,\beta) = \begin{cases} 2^n & \text{if } \alpha = \beta \text{ and } \langle \beta, k\rangle = 0 \\ \pm 2^{\frac{n+1}{2}} & \text{if } \langle \alpha, k\rangle = 1 \text{ and } \langle \beta, k\rangle = 1 \\ 0 & \text{if } \langle \alpha+\beta, k\rangle = 1 \text{ or } (\alpha \neq \beta \text{ and } \langle \beta, k\rangle = 0) \end{cases} . \tag{11}$$

Proof. First, we show the upper part, that is the values of the diagonal. Then

$$\begin{aligned} \widehat{R_{k,w}}(\alpha,\beta) &= \sum_{x\in\mathbb{F}_2^n} (-1)^{\langle \alpha+\beta, x\rangle + \langle \beta,k\rangle \cdot f(w+\Phi_k(x))} \\ &= \sum_{\substack{y\in\mathbb{F}_2^{n-1} \\ \Phi_k^{-1}(y)\in\{x_0,x_1\}}} (-1)^{\langle \alpha+\beta, x_0\rangle + \langle \beta,k\rangle \cdot f(w+y)} + (-1)^{\langle \alpha+\beta, x_1\rangle + \langle \beta,k\rangle \cdot f(w+y)} \end{aligned}$$

As we look at the diagonal elements, $\alpha = \beta$, we have

$$\widehat{R_{k,w}}(\alpha,\alpha) = 2 \cdot \sum_y (-1)^{\langle \beta,k\rangle \cdot f(w+y)} = 2 \cdot \sum_{y'\in\mathbb{F}_2^{n-1}} (-1)^{\langle \beta,k\rangle \cdot f(y')} .$$

Two possibilities remain: If $\langle \beta, k\rangle = 0$, the exponent is always zero and thus $\widehat{R_{k,w}}(\alpha,\alpha) = 2^n$. In the other case, $\langle \beta, k\rangle = 1$ and

$$\widehat{R_{k,w}}(\alpha,\alpha) = 2 \cdot \sum_{y'\in\mathbb{F}_2^{n-1}} (-1)^{f(y')} = \pm 2 \cdot 2^{\frac{n-1}{2}} ,$$

since Parseval's relation implies that all Fourier coefficients of an $(n-1)$-variable bent function have the same magnitude, namely $2^{\frac{n-1}{2}}$.

Now for the second part we have $\alpha \neq \beta$.

$$\begin{aligned} &\widehat{R_{k,w}}(\alpha,\beta) \\ &= \sum_{\substack{y\in\mathbb{F}_2^{n-1} \\ \Phi_k^{-1}(y)\in\{x_0,x_1\}}} (-1)^{\langle \alpha+\beta, x_0\rangle + \langle \beta,k\rangle \cdot f(w+y)} + (-1)^{\langle \alpha+\beta, x_1\rangle + \langle \beta,k\rangle \cdot f(w+y)} \end{aligned}$$

By definition of Φ_k, we saw in Eq. (9) that the preimages x_0, and x_1 are equal to y' and $y'+k$, where y' is the same as y with an additional bit set to zero injected at position $i(k)$. Thus, using the bilinearity of the scalar product,

$$\widehat{R_{k,w}}(\alpha,\beta) = \left(1 + (-1)^{\langle \alpha+\beta, k\rangle}\right) \sum_y (-1)^{\left\langle \alpha+\beta, y'\right\rangle + \langle \beta,k\rangle \cdot f(w+y)}$$

and this is equal to zero, if $\langle \alpha+\beta, k\rangle = 1$ or $\langle \beta, k\rangle = 0$. In the other case, $\langle \alpha+\beta, k\rangle = 0$ and $\langle \beta, k\rangle = 1$, we have

$$\begin{aligned} \widehat{R_{k,w}}(\alpha,\beta) &= 2 \cdot \sum_y (-1)^{\left\langle \alpha+\beta, y'\right\rangle + f(w+y)} = 2 \cdot \sum_x (-1)^{\left\langle \alpha+\beta, x'+w'\right\rangle + f(x)} \\ &= 2 \cdot (-1)^{\left\langle \alpha+\beta, w'\right\rangle} \cdot \sum_x (-1)^{\left\langle \alpha+\beta, x'\right\rangle + f(x)} \\ &= 2 \cdot (-1)^{\left\langle \alpha+\beta, w'\right\rangle} \cdot \widehat{f}(\alpha''+\beta'') \end{aligned}$$

where we denote by $\cdot''$ the corresponding value, where the bit in position $i(k)$ has been removed. Finally, again because f is bent, we get

$$\widehat{R_{k,w}}(\alpha,\beta) = 2\cdot(-1)^{\langle\alpha+\beta,w'\rangle}\cdot(\pm 2^{\frac{n-1}{2}}) = \pm 2^{\frac{n+1}{2}}.$$

Note that the sign of the LAT entries is uniformly distributed and thus, when averaging over the w's, the non-diagonal entries cancel out.

6.1.2 To more Rounds.

When we look at more than one round, we try to approximate the linear hull by looking at the strongest linear trail. As already discussed in Lemma 2, for $r < n$ there are trails with probability one. We now show that any trail's correlation for $r \geqslant n$ rounds is actually upper bounded by $2^{-\frac{n+1}{2}}$:

Proposition 5. *For $r \geqslant n$ rounds, the correlation of any non-trivial linear trail for* BISON *is upper bounded by $2^{-\frac{n+1}{2}}$.*

Proof. It is enough to show the above for any n-round trail. By contradiction, assume there exists a non-trivial trail $\theta = (\theta_0,\dots,\theta_n)$ with correlation one. Following Proposition 4, for every round i the intermediate mask θ_i has to fulfill $\langle\theta_i,k_i\rangle = 0$. Further $\theta_i = \theta_{i+1}$ for $0 \leqslant i < n$. Because all n round keys are linearly independent, this implies that $\theta_i = 0$, which contradicts our assumption. Thus, in at least one round the second or third case of Eq. (11) has to apply.

It would be nice to be able to say more about the linear hull, analogously to the differential case. However, for the linear cryptanalysis this looks much harder, due to the denser LAT. In our opinion developing a framework where bounding linear hulls is similarly easy as it is for BISON with respect to differentials is a fruitful future research topic.

6.2 Higher-Order Differentials and Algebraic Attacks.

High-order differential attacks, cube attacks, algebraic attacks and integral attacks all make use of non-random behaviour of the ANF of parts of the encryption function. In all these attacks the algebraic degree of (parts of) the encryption function is of particular interest. In this section, we argue that those attacks do not pose a threat to BISON.

We next elaborate in more detail on the algebraic degree of the WSN construction. In particular, we are going to show that the algebraic degree increases at most linearly with the number of rounds. More precisely, if the round function is of degree d, the algebraic degree after r rounds is upper bounded by $r(d-1)+1$.

Actually, we are going to consider a slight generalization of the WSN construction and prove the above statement for this generalization.

6.2.1 General Setting

Consider an initial state of n bits given as $x = (x_0,\dots,x_{n-1})$ and a sequence of Boolean functions

$$f_i : \mathbb{F}_2^{n+i} \to \mathbb{F}_2$$

for $0 \leqslant i < r$. We define a sequence of values y_i by setting $y_0 = f_0(x)$ and

$$y_i = f_i(x_0,\dots,x_{n-1},y_0,\dots,y_{i-1}),$$

for $1 \leqslant i < r$. Independently of the exact choice of f_i the degree of any y_ℓ, as a function of x can be upper bounded as stated in the next proposition.

Proposition 6. *Let f_i be a sequence of functions as defined above, such that* $\deg(f_i) \leqslant d$*. The degree of y_ℓ at step ℓ seen as a function of the bits of the initial state $x_0, \dots, x_{n-1}$ satisfies*

$$\deg(y_\ell) \leqslant (d-1)(\ell+1)+1.$$

Moreover, for any $I \subseteq \{0, \dots, \ell\}$,

$$\deg(\prod_{i \in I} y_i) \leqslant (d-1)(\ell+1)+|I|.$$

Proof. The first assertion is of course a special case of the second one, but we add it for the sake of clarity. We prove the second, more general, statement by induction on ℓ.

Starting with $\ell = 0$, we have to prove that $\deg(y_0) \leqslant d$, which is obvious, as

$$y_0 = f_0(x_0, \dots, x_{n-1})$$

and $\deg(f_0) \leq d$.

Now, we consider some $I \subseteq \{0, \dots, \ell\}$ and show that

$$\deg(\prod_{i \in I} y_i) \leqslant (d-1)(\ell+1)+|I| \,.$$

We assume that $\ell \in I$, otherwise the result directly follows the induction hypothesis.

Since f_ℓ depends both on $y_0, \dots, y_{\ell-1}$ and x, we decompose it as follows:

$$y_\ell = f_\ell(y_0, \dots, y_{\ell-1}, x) = \sum_{\substack{J \subseteq \{0, \dots, \ell-1\} \\ 0 \leqslant |J| \leqslant \min(d, \ell)}} \left(\prod_{j \in J} y_j\right) g_J(x)$$

with $\deg(g_J) \leqslant d - |J|$ for all J since $\deg(f_\ell) \leqslant d$.

Then, for $I = \{\ell\} \cup I'$, we look at

$$y_\ell \left(\prod_{i \in I'} y_i\right) = \sum_{\substack{J \subseteq \{0, \dots, \ell-1\} \\ 0 \leqslant |J| \leqslant \min(d, \ell)}} \left(\prod_{j \in J \cup I'} y_j\right) g_J(x) \,.$$

From the induction hypothesis, the term of index J in the sum has degree at most

$$\begin{aligned}
(d-1)\ell + |J \cup I'| + \deg(g_J) &= (d-1)\ell + |J \cup I'| + d - |J| \\
&\leqslant (d-1)(\ell+1) + |J \cup I'| - |J| + 1 \\
&\leqslant (d-1)(\ell+1) + |J| + |I'| - |J| + 1 \\
&\leqslant (d-1)(\ell+1) + |I| \,.
\end{aligned}$$

6.2.2 Special case of BISON.

In the case of BISON, we make use of quadratic functions, and thus Proposition 6 implies that after r rounds the degree is upper bounded by $r+1$ only. Thus, it will take at least $n-2$ rounds before the degree reaches the maximal possible degree of $n-1$. Moreover, due to the construction of WSN, if all component functions of $E_{k,w}^r$ are of degree at most d, there will be at least one component function of $E_{k,w}^{r+n-1}$ of degree at most d. That is, there exist a vector $\beta \in \mathbb{F}_2^n$ such that

$$\langle \beta, E_{k,w}^{r+n-1}(x) \rangle$$

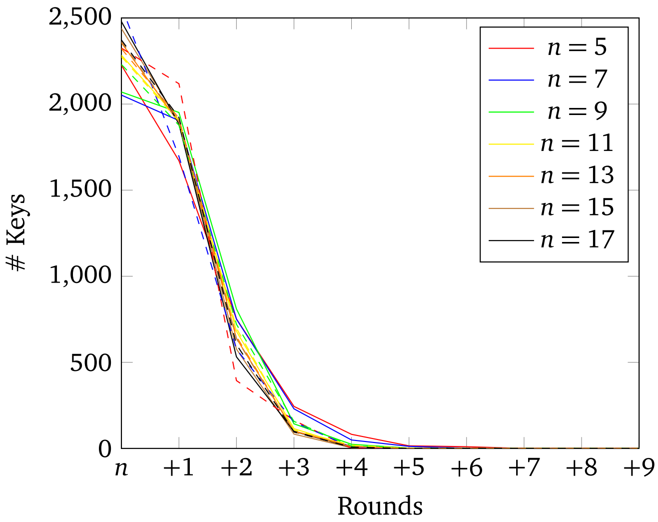


Figure 4: Number of rounds more than n needed to achieve full degree. Solid lines for random round keys, dashed lines for round keys derived from BISON's key schedule.

has degree at most d. Namely, for

$$\beta \in \operatorname{span}\{k_r, \dots, k_{r+s}\}^{\perp}$$

it holds that

$$\deg\left(\langle \beta, E_{k,w}^{r+s}(x)\rangle\right) = \deg\left(\langle \beta, E_{k,w}^{r}(x)\rangle + \sum_{i=r}^{r+s} \lambda_i \langle \beta, k_i\rangle\right) = \deg\left(\langle \beta, E_{k,w}^{r}(x)\rangle\right).$$

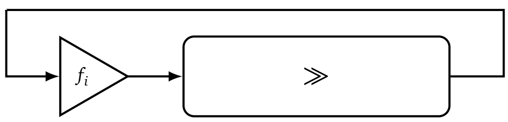
We conclude there exists a component function of $E_{k,w}^{r+s}$ of non-maximal degree, as long as $0 \leqslant r \leqslant n-2$ and $0 \leqslant s \leqslant n-1$. Thus for BISON there will be at least one component function of degree less than $n-1$ for any number of rounds $0 \leqslant r \leqslant 2n-3$. However, similarly to the case of zero-correlation properties as described in Lemma 3, the vector β is key dependent and thus this property does not directly lead to an attack.

Finally, so far we only discussed upper bounds on the degree, while for arguing security, lower bounds on the degree are more relevant. As it seems very hard (just like for any cipher) to prove such lower bounds, we investigated experimentally how the degree increases in concrete cases. As can be seen in Figure 4 the maximum degree is reached for almost any instance for $n+5$ rounds. Most importantly, the fraction of instances where it takes more than $n+2$ rounds decreases with increasing block length n. Moreover, the round function in BISON experimentally behaves with this respect as a random function, as can be seen on Figure 5. Thus, as the number of rounds is $3n$, we are confident that attacks exploiting the algebraic degree do not pose a threat for BISON.

Besides the WSN construction, a special case of the above proposition worth mentioning is a non linear feedback generator (NLFSR).

6.2.3 Degree of NLFSR.

One well-known special case of the above general setting is an NLFSR or, equivalently a maximally unbalanced Feistel cipher, depicted below.



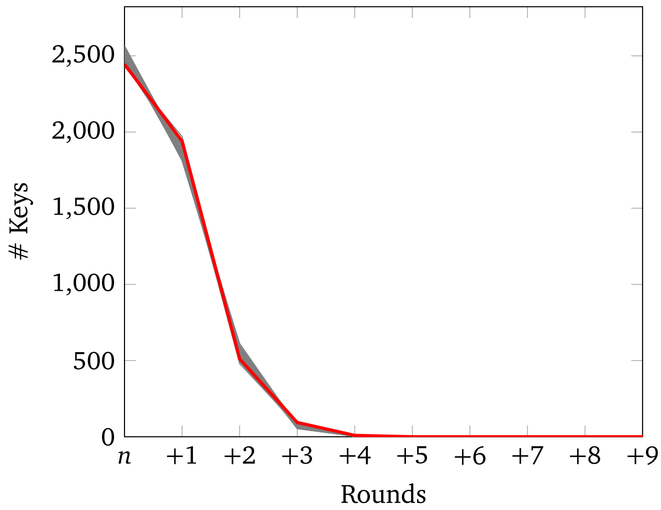


Figure 5: Behaviour of BISON's f function (red thick solid) versus random f (gray solid) with algebraic degree 2 for $n = 17$.

Proposition 6 implies that the degree of any NLFSR increases linearly with the number of rounds. To the best of our knowledge, this is the first time this have been observed in this generality. We like to add that this is in sharp contrast to how the degree increases for SPN ciphers. For SPN ciphers the degree usually increases exponentially until a certain threshold is reached [Bou+11].

6.3 Other attacks

We briefly discuss other cryptanalytic attacks.

6.3.1 Impossible Differentials.

In Lemma 7 and Corollary 2, we discuss that every output difference is possible after more than n rounds. Consequently, there are no impossible differentials for BISON.

6.3.2 Truncated Differentials.

Due to our strong bounds on differentials it seems very unlikely that any strong truncated differential exists.

6.3.3 Zero Correlation Linear Cryptanalysis.

In Lemma 3 we already discussed generic zero correlation linear hulls for the WSN construction. Depending on the actual key used, this technique may be used to construct a one-round-longer zero-correlation trail. For this, we need two *distinct* elements $\alpha \in \langle k_1, \dots, k_{n-1} \rangle^{\perp}$, $\beta \in \langle k_n, \dots, k_{2n-2} \rangle^{\perp}$, and construct the trail analogously to Lemma 3 (which may not exist, due to the key dependency).

6.3.4 Invariant Attacks.

For an invariant attack, we need a Boolean function g, s. t. $g(x) + g(E^r_{k,w}(x))$ is constant *for all* x and some *weak keys* (k, w). As the encryption of any message is basically this message with some of the round keys added, key addition is the only operation which is performed.

It has been shown in [Bei+17, Proposition 1] that any g which is invariant for a linear layer followed by the addition of the round key k_i as well as for the same up to addition of a different k_j, has a linear space containing $k_i + k_j$. In the case of the linear layer being the identity, the linear space actually contains also the k_i and k_j (by definition).

Thus, the linear space of any invariant for our construction has to contain span $\{k_1, \ldots, k_{3n}\}$ which is obviously the full space $\mathbb{F}_2^n$. Following the results of [Bei+17], there are thus no invariant subspace or nonlinear invariant attack on BISON.

6.3.5 Related-Key Attacks.

In generic related-key attacks, the attacker is also allowed to exploit encryptions under a related, that is $k' = f(k)$, key – in the following, we restrict our analysis to the case where f is the addition with a constant. That is, the attacker cannot only request $E_{k,w}(x)$, and $E_{k,w}(x+\alpha)$, but also $E_{k+\beta,w+\beta'}(x)$ or $E_{k+\beta,w+\beta'}(x+\alpha)$, for α (difference in the input x), β (difference in the key k) and β' (difference in the key w) of her choice. As $\beta = \beta' = 0$ would result in the standard differential scenario, we exclude it for the remainder of this discussion. Similar, $\beta = k$ results in $\Phi_{k+\beta} = \Phi_0$, which we did not define, thus we also skip this case and refer to the fact that if an attacker chooses $\beta = k$, she basically already has guessed the secret key correctly.

First note that, for any input difference (α, β, β'), the possible output differences after one round are

$$\begin{array}{ll} \alpha & \text{if } (u,v) = (0,0), \\ \alpha + \beta + k & \text{if } (u,v) = (0,1), \\ \alpha \qquad + k & \text{if } (u,v) = (1,0), \text{ and} \\ \alpha + \beta & \text{if } (u,v) = (1,1), \end{array}$$

where

$$u = f(w + \Phi_k(x)), \tag{12}$$

$$v = f(w + \beta' + \Phi_{k+\beta}(x+\alpha)). \tag{13}$$

Our aim is to bound both the probability that $u + v = 0$ and that $u + v = 1$ by $3/4$. This implies that the probability for any related-key differential for one round is at most $3/4$. Thus the probability for any r-round related-key differential is bounded by $(3/4)^r$. For this, we need the following lemma.

Lemma 11. *Let us consider the linear function Φ_k defined by Eq. (8). Given k and $\beta \notin \{0, k\}$. Then the dimension of the image of the linear function $x \mapsto \Phi_k(x) + \Phi_{k+\beta}(x)$ is either one or two.*

Proof. For the sake of simplicity, we instead consider $\Phi'_k(x) + \Phi'_{k+\beta}(x)$, where Φ' is the same as Φ but does not truncate its output. Basically the same argumentation then holds for Φ as well. This function can also be written as

$$\begin{aligned} \Phi'_k(x) + \Phi'_{k+\beta}(x) &= x + x_{i(k)}k + x + x_{i(k+\beta)}(k+\beta) \\ &= (x_{i(k)} + x_{i(k+\beta)})k + x_{i(k+\beta)}\beta. \end{aligned}$$

Thus

$$\Phi'_k(x) + \Phi'_{k+\beta}(x) \in \text{span}\{k, \beta\}$$

for all x, which upper bounds the dimension of the image by two. As e. g. $x = e_{i(k+\beta)}$ is not mapped to zero, the dimension of the image is at least one, completing the proof.

By the rank-nullity theorem, this implies that

$$\dim\left(\ker\left(\Phi_k + \Phi_{k+\beta}\right)\right) \in \{n-1, n-2\}.$$

We can now show the following proposition.

Proposition 7. *For r rounds, the probability of any related-key differential characteristic for* BISON*, averaged over all whieting key sequences $(w_1, \dots, w_r)$, is upper bounded by $\left(\frac{3}{4}\right)^r$.*

Proof. First, let us introduce the set $A_{x,k}^{\alpha,\beta,\beta'}$ defined as:

$$A_{x,k}^{\alpha,\beta,\beta'} := \left\{ w \in \mathbb{F}_2^{n-1} \;\middle|\; f(w + \Phi_k(x)) + f(w + \beta' + \Phi_{k+\beta}(x+\alpha)) = 0 \right\},$$

that is the set of all round keys w, for which x, k, α, β, and β' result in $u + v = 0$ (where u and v are as at the beginning of this section). In case that

$$\Phi_k(x) = \beta' + \Phi_{k+\beta}(x+\alpha),$$

the size of $A_{x,k}^{\alpha,\beta,\beta'}$ is 2^{n-1}, while if the equality does not hold the set is of size 2^{n-2} since f is bent.

For k, α, β and β' fixed, the number of x s. t. the size of $A_{x,k}^{\alpha,\beta,\beta'}$ is 2^{n-1}, is just the size of the preimage of $\Phi_{k+\beta}(\alpha) + \beta'$ under the linear mapping $x \mapsto \Phi_k(x) + \Phi_{k+\beta}(x)$. The size of this preimage is either 0 or $\left|\ker\left(\Phi_k + \Phi_{k+\beta}\right)\right|$. Denote by

$$B = \left|\ker\left(\Phi_k + \Phi_{k+\beta}\right)\right|,$$

which, by Lemma 11, is bounded by 2^{n-1}.

Then, the probability over x and w for having an output difference of α or $\alpha + \beta$ is:

$$\begin{aligned}
&\Pr_{x,w}[u + v = 0 \text{ for fixed } k,\ \alpha,\ \beta,\ \beta'] \\
&\leqslant \frac{B}{2^n} + \frac{2^{n-2}}{2^{n-1}} \cdot \frac{2^n - B}{2^n} \leqslant \frac{B}{2^n} + \frac{1}{2}\left(1 - \frac{B}{2^n}\right) \leqslant \frac{1}{2} + \frac{B}{2^{n+1}} \\
&\leqslant \frac{1}{2} + \frac{1}{4} = \frac{3}{4}.
\end{aligned}$$

The other case, $u + v = 1$, follows with the same argument. Thus, the probability for each of the four possible cases $(u, v) \in \mathbb{F}_2 \times \mathbb{F}_2$ can be upper bounded by $(3/4)$, which concludes the proof.

6.3.6 Further Observations.

During the design process, we observed the following interesting point: For sparse master keys k and w and message m, e. g. $k = w = m = 1$, in the first few rounds, nothing happens. This is mainly due to the choice of sparse key schedule polynomials p_w and p_k and the fact that f_0 outputs 0 if only one bit in its input is set (as $\langle 0, x \rangle = 0$ for any x).

To the best of our knowledge, this observation cannot be exploited in an attack.

6.3.7 Experimental Results.

We conducted experiments on small-scale versions of BISON with $n = 5$. The DDTs and LATs, depicted using the "Jackson Pollock representation" [BP15], for one to ten rounds are listed in Appendix B. In Appendix B.1 one can see that the two cases of averaging over all possible w_i and choosing a fixed w_i results in very similar differential behaviors. Additionally, after $5 = n$ rounds, the plots do not change much.

Table 1: Performance of our 129-bit implementation on an Intel Core i7-8700 cpu, running at 3.7 GHz. Cycles per byte are measured for 1 000 000 runs, Hyper-Threading and Turbo-Boost were disabled.

Block size (bit)	Code size (byte)	Cycles per byte median	mean	σ
129	701	3 021	3 064.08	102.56

The results in the linear case, see Appendix B.2, are quite similar. The major difference here, is the comparable bigger entries for a fixed w_i. Nonetheless, most important is that there are no high entries in the average LAT which would imply a strong linear approximation for many keys. Additionally one also expects for a random permutation not too small LAT entries. Note that one can well observe the probability-one approximation for $4 = n - 1$ rounds (lower right corner of the corresponding plot).

7 Implementation

As the round function is involutory, we do not need to implement a separate decryption, but instead can just use the encryption implementation with reversed round keys.

To implement the two LFSRs for the key schedule, we need two primitive polynomials of degree n and $n-1$. Clocking an LFSR with feedback polynomial $p(x)$ corresponds to multiplying the state by $x \in \mathbb{F}_2[x]/p(x)$. This can be implemented by a simple left shift and a conditional addition of the polynomial, if a modulo reduction is necessary. To keep this addition as efficient as possible it is advantageous to have all non-leading monomials of the polynomial of degree less then the word size of the implementation's underlying CPU, since in this case, we only need to add a term to the least significant word of the state. Appropriate polynomials can easily be found by enumerating possible candidates and test if the candidate is primitive. See Appendix A for possible, good-to-implement, choices for $p_w(x)$ and $p_k(x)$.

For comparison and test reasons we also provide testvectors in Appendix C and a SAGE implementation, see Appendix D. We implemented the 129-bit instance in C on a 64-bit Intel Core i7-8700 CPU (Coffee Lake architecture) running at 3.7 GHz. The corresponding source code can be found in Appendix E.

Utilizing the CPU's `popcount` instruction, this implementation consumes a size of 701 bytes when compiled with `-Os`. The same implementation needs a bit more then 3 000 cycles per byte for the encryption of one 129-bit block.[4] Table 1 summarizes these results. While this might be obvious, we nevertheless want to note that it is important for reliable benchmarks to turn off advanced performance capabilities of modern CPUs.

Regarding cycles per byte, this is three orders of magnitude slower than optimized implementations of AES. Even if the reference implementation is not optimized, we do not believe to come close to a competitive speed. Another point which can be seen in Table 1 is the deviating runtime of our implementation. The reference implementation is clearly not constant time, see e. g. Lines 65 to 116 or Line 130 in Appendix E. For any secure implementation this and other side channels have of course to be taken into account. Nevertheless, a side channel-resistant implementation is out of scope of this work but is certainly an interesting research direction. We expect the simplicity of our design to support side-channel countermeasures.

[4]For comparison: exploiting `AES-NI` instructions on modern CPUs results in 4.375 cycles per byte for encrypting one AES-128 block, excluding the key-schedule. When parallelism can be exploited, the speed can be even further increased, eventually tending to one cycle per byte. Implementing a full AES-128 encryption with `AES-NI` instructions including the key schedule uses 394 bytes.

8 Conclusion

Efficiency of symmetric ciphers have been significantly improved further and further, in particular within the trend of lightweight cryptography. However, when it comes to arguing about the security of ciphers, the progress is rather limited and the arguments basically did not get easier nor stronger since the development of the AES. In our opinion it might be worth shifting the focus to improving security arguments for new designs rather than (incrementally) improving efficiency. We see BISON as a first step in this direction.

With our instance for the WSN construction and its strong resistance to differential cryptanalysis, this framework emerges as an interesting possibility to design block ciphers. Unfortunately, we are not able to give better then normal arguments for the resistance to linear cryptanalysis. It is thus an interesting question, if one can find a similar instance of the WSN construction for which comparable strong arguments for the later type of cryptanalysis exist.

Alternative designs might also be worth looking at. For example many constructions for bent functions are known and could thus be examined as alternatives for the scalar product used in BISON. One might also look for a less algebraic design – but we do not yet see how this would improve or ease the analysis or implementation of an instance.

Another line of future work is the in-depth analysis of implementation optimizations and side channel-resistance of BISON.

Acknowledgments

We would like to thank Christof Beierle for helpful comments, and Lucas Hartmann for the artistic design of BISON.

References

[Abd+12] Mohamed Ahmed Abdelraheem, Martin Ågren, Peter Beelen, and Gregor Leander. "On the Distribution of Linear Biases: Three Instructive Examples." In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 50–67. DOI: 10.1007/978-3-642-32009-5_4.

[And+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. "On the Indifferentiability of Key-Alternating Ciphers." In: *CRYPTO 2013, Part I*. Ed. by Ran Canetti and Juan A. Garay. Vol. 8042. LNCS. Springer, Heidelberg, Aug. 2013, pp. 531–550. DOI: 10.1007/978-3-642-40041-4_29.

[BK09] Alex Biryukov and Dmitry Khovratovich. "Related-Key Cryptanalysis of the Full AES-192 and AES-256." In: *ASIACRYPT 2009*. Ed. by Mitsuru Matsui. Vol. 5912. LNCS. Springer, Heidelberg, Dec. 2009, pp. 1–18. DOI: 10.1007/978-3-642-10366-7_1.

[BN16] Céline Blondeau and Kaisa Nyberg. "Improved Parameter Estimates for Correlation and Capacity Deviates in Linear Cryptanalysis." In: *IACR Trans. Symm. Cryptol.* 2016.2 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/570, pp. 162–191. ISSN: 2519-173X. DOI: 10.13154/tosc.v2016.i2.162-191.

[BP15] Alex Biryukov and Léo Perrin. "On Reverse-Engineering S-Boxes with Hidden Design Criteria or Structure." In: *CRYPTO 2015, Part I*. Ed. by Rosario Gennaro and Matthew J. B. Robshaw. Vol. 9215. LNCS. Springer, Heidelberg, Aug. 2015, pp. 116–140. DOI: 10.1007/978-3-662-47989-6_6.

[BS91] Eli Biham and Adi Shamir. "Differential Cryptanalysis of DES-like Cryptosystems." In: *CRYPTO'90*. Ed. by Alfred J. Menezes and Scott A. Vanstone. Vol. 537. LNCS. Springer, Heidelberg, Aug. 1991, pp. 2–21. DOI: 10.1007/3-540-38424-3_1.

[Bei+17] Christof Beierle, Anne Canteaut, Gregor Leander, and Yann Rotella. "Proving Resistance Against Invariant Attacks: How to Choose the Round Constants." In: *CRYPTO 2017, Part II*. Ed. by Jonathan Katz and Hovav Shacham. Vol. 10402. LNCS. Springer, Heidelberg, Aug. 2017, pp. 647–678. DOI: 10.1007/978-3-319-63715-0_22.

[Bir+09] Alex Biryukov, Dmitry Khovratovich, and Ivica Nikolic. "Distinguisher and Related-Key Attack on the Full AES-256." In: *CRYPTO 2009*. Ed. by Shai Halevi. Vol. 5677. LNCS. Springer, Heidelberg, Aug. 2009, pp. 231–249. DOI: 10.1007/978-3-642-03356-8_14.

[Bog+12] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, François-Xavier Standaert, John P. Steinberger, and Elmar Tischhauser. "Key-Alternating Ciphers in a Provable Setting: Encryption Using a Small Number of Public Permutations - (Extended Abstract)." In: *EUROCRYPT 2012*. Ed. by David Pointcheval and Thomas Johansson. Vol. 7237. LNCS. Springer, Heidelberg, Apr. 2012, pp. 45–62. DOI: 10.1007/978-3-642-29011-4_5.

[Bou+11] Christina Boura, Anne Canteaut, and Christophe De Cannière. "Higher-Order Differential Properties of Keccak and Luffa." In: *FSE 2011*. Ed. by Antoine Joux. Vol. 6733. LNCS. Springer, Heidelberg, Feb. 2011, pp. 252–269. DOI: 10.1007/978-3-642-21702-9_15.

[CR15] Anne Canteaut and Joëlle Roué. "On the Behaviors of Affine Equivalent Sboxes Regarding Differential and Linear Attacks." In: *EUROCRYPT 2015, Part I*. Ed. by Elisabeth Oswald and Marc Fischlin. Vol. 9056. LNCS. Springer, Heidelberg, Apr. 2015, pp. 45–74. DOI: 10.1007/978-3-662-46800-5_3.

[CS14] Shan Chen and John P. Steinberger. "Tight Security Bounds for Key-Alternating Ciphers." In: *EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer, Heidelberg, May 2014, pp. 327–350. DOI: 10.1007/978-3-642-55220-5_19.

[Car07] Claude Carlet. "Boolean Functions for Cryptography and Error Correcting Codes." In: *Boolean Methods and Models*. Ed. by Yves Crama and Peter Hammer. Cambridge University Press, 2007.

[DR01] Joan Daemen and Vincent Rijmen. "The Wide Trail Design Strategy." In: *8th IMA International Conference on Cryptography and Coding*. Ed. by Bahram Honary. Vol. 2260. LNCS. Springer, Heidelberg, Dec. 2001, pp. 222–238.

[DR02] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Information Security and Cryptography. Springer, 2002. ISBN: 3-540-42580-2. DOI: 10.1007/978-3-662-04722-4.

[DR06] Joan Daemen and Vincent Rijmen. "Understanding Two-Round Differentials in AES." In: *SCN 06*. Ed. by Roberto De Prisco and Moti Yung. Vol. 4116. LNCS. Springer, Heidelberg, Sept. 2006, pp. 78–94. DOI: 10.1007/11832072_6.

[DR98] Joan Daemen and Vincent Rijmen. "The Block Cipher Rijndael." In: *CARDIS'98*. Vol. 1820. LNCS. Springer, 1998, pp. 277–284. DOI: 10.1007/10721064_26.

[Dae+93] Joan Daemen, René Govaerts, and Joos Vandewalle. “Block ciphers based on Modular Arithmetic.” In: *State and Progress in the Research of Cryptography*. Ed. by W. Wolfowicz. Fondazione Ugo Bordoni, 1993, pp. 80–89.

[Dae+95] Joan Daemen, René Govaerts, and Joos Vandewalle. “Correlation Matrices.” In: *FSE’94*. Ed. by Bart Preneel. Vol. 1008. LNCS. Springer, Heidelberg, Dec. 1995, pp. 275–285. DOI: 10.1007/3-540-60590-8_21.

[Dae95] Joan Daemen. *Cipher and hash function design, strategies based on linear and differential cryptanalysis, PhD Thesis*. http://jda.noekeon.org/. K.U.Leuven, 1995.

[Der+13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. “Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting.” In: *EUROCRYPT 2013*. Ed. by Thomas Johansson and Phong Q. Nguyen. Vol. 7881. LNCS. Springer, Heidelberg, May 2013, pp. 371–387. DOI: 10.1007/978-3-642-38348-9_23.

[Dil72] John F Dillon. “A survey of bent functions.” In: *The NSA technical journal* 191 (1972), p. 215.

[Dun+10] Orr Dunkelman, Nathan Keller, and Adi Shamir. “Improved Single-Key Attacks on 8-Round AES-192 and AES-256.” In: *ASIACRYPT 2010*. Ed. by Masayuki Abe. Vol. 6477. LNCS. Springer, Heidelberg, Dec. 2010, pp. 158–176. DOI: 10.1007/978-3-642-17373-8_10.

[EM97] Shimon Even and Yishay Mansour. “A Construction of a Cipher from a Single Pseudorandom Permutation.” In: *Journal of Cryptology* 10.3 (June 1997), pp. 151–162. DOI: 10.1007/s001459900025.

[Fer+01] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David Wagner, and Doug Whiting. “Improved Cryptanalysis of Rijndael.” In: *FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. LNCS. Springer, Heidelberg, Apr. 2001, pp. 213–230. DOI: 10.1007/3-540-44706-7_15.

[Fip] *Advanced Encryption Standard (AES)*. National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, Nov. 2001.

[GL15] Chun Guo and Dongdai Lin. “On the Indifferentiability of Key-Alternating Feistel Ciphers with No Key Derivation.” In: *TCC 2015, Part I*. Ed. by Yevgeniy Dodis and Jesper Buus Nielsen. Vol. 9014. LNCS. Springer, Heidelberg, Mar. 2015, pp. 110–133. DOI: 10.1007/978-3-662-46494-6_6.

[GM00] Henri Gilbert and Marine Minier. “A Collision Attack on 7 Rounds of Rijndael.” In: *AES Candidate Conference*. Vol. 230. 2000, p. 241.

[Gra+16] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. “Subspace Trail Cryptanalysis and its Applications to AES.” In: *IACR Trans. Symm. Cryptol.* 2016.2 (2016). http://tosc.iacr.org/index.php/ToSC/article/view/571, pp. 192–225. ISSN: 2519-173X. DOI: 10.13154/tosc.v2016.i2.192-225.

[Gra+17] Lorenzo Grassi, Christian Rechberger, and Sondre Rønjom. “A New Structural-Differential Property of 5-Round AES.” In: *EUROCRYPT 2017, Part II*. Ed. by Jean-Sébastien Coron and Jesper Buus Nielsen. Vol. 10211. LNCS. Springer, Heidelberg, 2017, pp. 289–317. DOI: 10.1007/978-3-319-56614-6_10.

[HT16] Viet Tung Hoang and Stefano Tessaro. “Key-Alternating Ciphers and Key-Length Extension: Exact Bounds and Multi-user Security.” In: *CRYPTO 2016, Part I*. Ed. by Matthew Robshaw and Jonathan Katz. Vol. 9814. LNCS. Springer, Heidelberg, Aug. 2016, pp. 3–32. DOI: 10.1007/978-3-662-53018-4_1.

[Hoa+12] Viet Tung Hoang, Ben Morris, and Phillip Rogaway. "An Enciphering Scheme Based on a Card Shuffle." In: *CRYPTO 2012*. Ed. by Reihaneh Safavi-Naini and Ran Canetti. Vol. 7417. LNCS. Springer, Heidelberg, Aug. 2012, pp. 1–13. DOI: 10.1007/978-3-642-32009-5_1.

[Hon+01] Seokhie Hong, Sangjin Lee, Jongin Lim, Jaechul Sung, Dong Hyeon Cheon, and Inho Cho. "Provable Security against Differential and Linear Cryptanalysis for the SPN Structure." In: *FSE 2000*. Ed. by Bruce Schneier. Vol. 1978. LNCS. Springer, Heidelberg, Apr. 2001, pp. 273–283. DOI: 10.1007/3-540-44706-7_19.

[JK97] Thomas Jakobsen and Lars R. Knudsen. "The Interpolation Attack on Block Ciphers." In: *FSE'97*. Ed. by Eli Biham. Vol. 1267. LNCS. Springer, Heidelberg, Jan. 1997, pp. 28–40. DOI: 10.1007/BFb0052332.

[KS07] Liam Keliher and Jiayuan Sui. "Exact maximum expected differential and linear probability for two-round Advanced Encryption Standard." In: *IET Information Security* 1.2 (2007), pp. 53–57. DOI: 10.1049/iet-ifs:20060161.

[Kra+17] Thorsten Kranz, Gregor Leander, and Friedrich Wiemer. "Linear Cryptanalysis: Key Schedules and Tweakable Block Ciphers." In: *IACR Trans. Symm. Cryptol.* 2017.1 (2017), pp. 474–505. ISSN: 2519-173X. DOI: 10.13154/tosc.v2017.i1.474-505.

[LS15] Rodolphe Lampe and Yannick Seurin. "Security Analysis of Key-Alternating Feistel Ciphers." In: *FSE 2014*. Ed. by Carlos Cid and Christian Rechberger. Vol. 8540. LNCS. Springer, Heidelberg, Mar. 2015, pp. 243–264. DOI: 10.1007/978-3-662-46706-0_13.

[Lai+91] Xuejia Lai, James L. Massey, and Sean Murphy. "Markov Ciphers and Differential Cryptanalysis." In: *EUROCRYPT'91*. Ed. by Donald W. Davies. Vol. 547. LNCS. Springer, Heidelberg, Apr. 1991, pp. 17–38. DOI: 10.1007/3-540-46416-6_2.

[Lea+11] Gregor Leander, Mohamed Ahmed Abdelraheem, Hoda AlKhzaimi, and Erik Zenner. "A Cryptanalysis of PRINTcipher: The Invariant Subspace Attack." In: *CRYPTO 2011*. Ed. by Phillip Rogaway. Vol. 6841. LNCS. Springer, Heidelberg, Aug. 2011, pp. 206–221. DOI: 10.1007/978-3-642-22792-9_12.

[MS90] Willi Meier and Othmar Staffelbach. "Nonlinearity Criteria for Cryptographic Functions." In: *EUROCRYPT'89*. Ed. by Jean-Jacques Quisquater and Joos Vandewalle. Vol. 434. LNCS. Springer, Heidelberg, Apr. 1990, pp. 549–562. DOI: 10.1007/3-540-46885-4_53.

[MY17] Sarah Miracle and Scott Yilek. *Cycle Slicer: An Algorithm for Building Permutations on Special Domains*. Cryptology ePrint Archive, Report 2017/873. http://eprint.iacr.org/2017/873. 2017.

[Mat95] Mitsuru Matsui. "On Correlation Between the Order of S-boxes and the Strength of DES." In: *EUROCRYPT'94*. Ed. by Alfredo De Santis. Vol. 950. LNCS. Springer, Heidelberg, May 1995, pp. 366–375. DOI: 10.1007/BFb0053451.

[NK95] Kaisa Nyberg and Lars R. Knudsen. "Provable Security Against a Differential Attack." In: *Journal of Cryptology* 8.1 (Dec. 1995), pp. 27–37. DOI: 10.1007/BF00204800.

[Nyb12] Kaisa Nyberg. ""Provable" Security against Differential and Linear Cryptanalysis (Invited Talk)." In: *FSE 2012*. Ed. by Anne Canteaut. Vol. 7549. LNCS. Springer, Heidelberg, Mar. 2012, pp. 1–8. DOI: 10.1007/978-3-642-34047-5_1.

[Nyb95] Kaisa Nyberg. "Linear Approximation of Block Ciphers (Rump Session)." In: *EUROCRYPT'94*. Ed. by Alfredo De Santis. Vol. 950. LNCS. Springer, Heidelberg, May 1995, pp. 439–444. DOI: 10.1007/BFb0053460.

[Par+03] Sangwoo Park, Soo Hak Sung, Sangjin Lee, and Jongin Lim. "Improving the Upper Bound on the Maximum Differential and the Maximum Linear Hull Probability for SPN Structures and AES." In: *FSE 2003*. Ed. by Thomas Johansson. Vol. 2887. LNCS. Springer, Heidelberg, Feb. 2003, pp. 247–260. DOI: 10.1007/978-3-540-39887-5_19.

[Rot76] Oscar S Rothaus. "On 'bent' functions." In: *Journal of Combinatorial Theory, Series A* 20.3 (1976), pp. 300–305.

[Røn+17] Sondre Rønjom, Navid Ghaedi Bardeh, and Tor Helleseth. "Yoyo Tricks with AES." In: *ASIACRYPT 2017, Part I*. Ed. by Tsuyoshi Takagi and Thomas Peyrin. Vol. 10624. LNCS. Springer, Heidelberg, Dec. 2017, pp. 217–243. DOI: 10.1007/978-3-319-70694-8_8.

[Tes15a] Stefano Tessaro. *Optimally Secure Block Ciphers from Ideal Primitives*. Cryptology ePrint Archive, Report 2015/868. http://eprint.iacr.org/2015/868. 2015.

[Tes15b] Stefano Tessaro. "Optimally Secure Block Ciphers from Ideal Primitives." In: *ASIACRYPT 2015, Part II*. Ed. by Tetsu Iwata and Jung Hee Cheon. Vol. 9453. LNCS. Springer, Heidelberg, 2015, pp. 437–462. DOI: 10.1007/978-3-662-48800-3_18.

[Tod+16] Yosuke Todo, Gregor Leander, and Yu Sasaki. "Nonlinear Invariant Attack - Practical Attack on Full SCREAM, iSCREAM, and Midori64." In: *ASIACRYPT 2016, Part II*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10032. LNCS. Springer, Heidelberg, Dec. 2016, pp. 3–33. DOI: 10.1007/978-3-662-53890-6_1.

[Vau12] Serge Vaudenay. *The End of Encryption based on Card Shuffling*. CRYPTO 2012 Rump Session. crypto.2012.rump.cr.yp.to PDF Link. 2012.

[Vau98] Serge Vaudenay. "Provable Security for Block Ciphers by Decorrelation." In: *STACS'98*. Vol. 1373. LNCS. Springer, 1998, pp. 249–275. DOI: 10.1007/BFb0028566.

[Wag99] David Wagner. "The Boomerang Attack." In: *FSE'99*. Ed. by Lars R. Knudsen. Vol. 1636. LNCS. Springer, Heidelberg, Mar. 1999, pp. 156–170. DOI: 10.1007/3-540-48519-8_12.

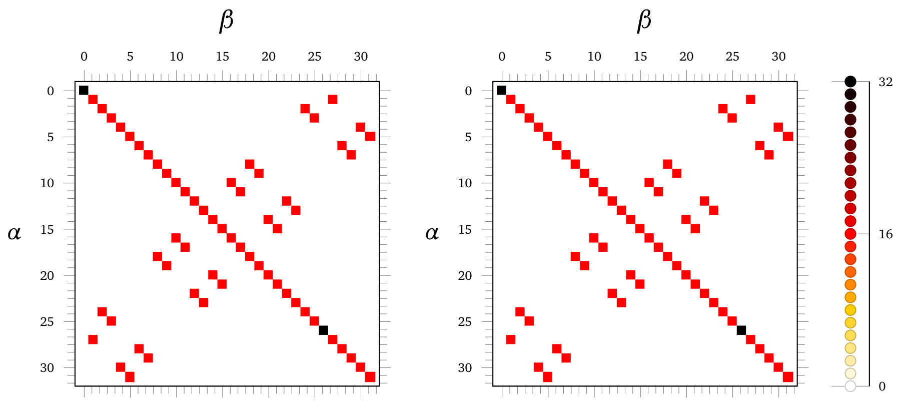
A Polynomials for the key schedule

n	$p_k(x)$	n	$p_w(x)$
129	$x^{129}+x^5+1$	128	$x^{128}+x^7+x^2+x+1$
127	$x^{127}+x+1$	126	$x^{126}+x^7+x^4+x^2+1$
125	$x^{125}+x^7+x^5+x^3+x^2+x+1$	124	$x^{124}+x^7+x^6+x^5+1$
123	$x^{123}+x^2+1$	122	$x^{122}+x^6+x^2+x+1$
121	$x^{121}+x^{18}+1$	120	$x^{120}+x^7+x^6+x^5+x^2+x+1$
119	$x^{119}+x^{21}+x^{19}+x^{17}+1$	118	$x^{118}+x^6+x^5+x^2+1$
117	$x^{117}+x^5+x^2+x+1$	116	$x^{116}+x^6+x^5+x^2+1$
115	$x^{115}+x^7+x^5+x^3+x^2+x+1$	114	$x^{114}+x^{19}+x^{17}+x^{16}+1$
113	$x^{113}+x^5+x^3+x^2+1$	112	$x^{112}+x^{21}+x^{20}+x^{18}+x^{17}+x^{16}+1$
111	$x^{111}+x^7+x^4+x^2+1$	110	$x^{110}+x^6+x^4+x+1$
109	$x^{109}+x^5+x^4+x^2+1$	108	$x^{108}+x^{22}+x^{20}+x^{19}+1$
107	$x^{107}+x^7+x^5+x^3+x^2+x+1$	106	$x^{106}+x^6+x^5+x+1$
105	$x^{105}+x^6+x^5+x^4+x^2+x+1$	104	$x^{104}+x^{23}+x^{22}+x^{18}+x^{17}+x^{16}+1$
103	$x^{103}+x^7+x^5+x^4+x^3+x^2+1$	102	$x^{102}+x^6+x^5+x^3+1$
101	$x^{101}+x^7+x^6+x+1$	100	$x^{100}+x^{22}+x^{20}+x^{17}+1$
99	$x^{99}+x^7+x^5+x^4+1$	98	$x^{98}+x^7+x^4+x^3+x^2+x+1$
97	$x^{97}+x^6+1$	96	$x^{96}+x^7+x^6+x^4+x^3+x^2+1$
95	$x^{95}+x^6+x^5+x^4+x^2+x+1$	94	$x^{94}+x^6+x^5+x+1$
93	$x^{93}+x^2+1$	92	$x^{92}+x^6+x^5+x^2+1$
91	$x^{91}+x^7+x^6+x^5+x^3+x^2+1$	90	$x^{90}+x^5+x^3+x^2+1$
89	$x^{89}+x^6+x^5+x^3+1$	88	$x^{88}+x^{23}+x^{22}+x^{19}+x^{18}+x^{17}+1$
87	$x^{87}+x^7+x^5+x+1$	86	$x^{86}+x^6+x^5+x^2+1$
85	$x^{85}+x^{20}+x^{19}+x^{18}+1$	84	$x^{84}+x^{22}+x^{19}+x^{16}+1$
83	$x^{83}+x^7+x^4+x^2+1$	82	$x^{82}+x^{19}+x^{18}+x^{17}+1$
81	$x^{81}+x^4+1$	80	$x^{80}+x^7+x^5+x^3+x^2+x+1$
79	$x^{79}+x^4+x^3+x^2+1$	78	$x^{78}+x^7+x^2+x+1$
77	$x^{77}+x^6+x^5+x^2+1$	76	$x^{76}+x^5+x^4+x^2+1$
75	$x^{75}+x^6+x^3+x+1$	74	$x^{74}+x^7+x^4+x^3+1$
73	$x^{73}+x^4+x^3+x^2+1$	72	$x^{72}+x^6+x^4+x^3+x^2+x+1$
71	$x^{71}+x^5+x^3+x+1$	70	$x^{70}+x^5+x^3+x+1$
69	$x^{69}+x^6+x^5+x^2+1$	68	$x^{68}+x^7+x^5+x+1$
67	$x^{67}+x^5+x^2+x+1$	66	$x^{66}+x^{22}+x^{20}+x^{19}+x^{18}+x^{17}+1$
65	$x^{65}+x^4+x^3+x+1$	64	$x^{64}+x^4+x^3+x+1$
63	$x^{63}+x+1$	62	$x^{62}+x^6+x^5+x^3+1$
61	$x^{61}+x^5+x^2+x+1$	60	$x^{60}+x+1$
59	$x^{59}+x^6+x^5+x^4+x^3+x+1$	58	$x^{58}+x^6+x^5+x+1$
57	$x^{57}+x^5+x^3+x^2+1$	56	$x^{56}+x^7+x^4+x^2+1$
55	$x^{55}+x^6+x^2+x+1$	54	$x^{54}+x^6+x^5+x^4+x^3+x^2+1$
53	$x^{53}+x^6+x^2+x+1$	52	$x^{52}+x^3+1$
51	$x^{51}+x^6+x^3+x+1$	50	$x^{50}+x^4+x^3+x^2+1$
49	$x^{49}+x^6+x^5+x^4+1$	48	$x^{48}+x^7+x^5+x^4+x^2+x+1$
47	$x^{47}+x^5+1$	46	$x^{46}+x^{20}+x^{19}+x^{18}+x^{17}+x^{16}+1$
45	$x^{45}+x^4+x^3+x+1$	44	$x^{44}+x^6+x^5+x^2+1$
43	$x^{43}+x^6+x^4+x^3+1$	42	$x^{42}+x^5+x^4+x^3+x^2+x+1$
41	$x^{41}+x^3+1$	40	$x^{40}+x^5+x^4+x^3+1$
39	$x^{39}+x^4+1$	38	$x^{38}+x^6+x^5+x+1$
37	$x^{37}+x^5+x^4+x^3+x^2+x+1$	36	$x^{36}+x^6+x^5+x^4+x^2+x+1$
35	$x^{35}+x^2+1$	34	$x^{34}+x^7+x^6+x^5+x^2+x+1$
33	$x^{33}+x^6+x^4+x+1$	32	$x^{32}+x^7+x^5+x^3+x^2+x+1$
31	$x^{31}+x^3+1$	30	$x^{30}+x^6+x^4+x+1$
29	$x^{29}+x^2+1$	28	$x^{28}+x^3+1$
27	$x^{27}+x^5+x^2+x+1$	26	$x^{26}+x^6+x^2+x+1$
25	$x^{25}+x^3+1$	24	$x^{24}+x^4+x^3+x+1$
23	$x^{23}+x^5+1$	22	$x^{22}+x+1$
21	$x^{21}+x^2+1$	20	$x^{20}+x^3+1$
19	$x^{19}+x^5+x^2+x+1$	18	$x^{18}+x^5+x^2+x+1$
17	$x^{17}+x^3+1$	16	$x^{16}+x^5+x^3+x^2+1$
15	$x^{15}+x+1$	14	$x^{14}+x^5+x^3+x+1$
13	$x^{13}+x^4+x^3+x+1$	12	$x^{12}+x^6+x^4+x+1$
11	$x^{11}+x^2+1$	10	$x^{10}+x^3+1$

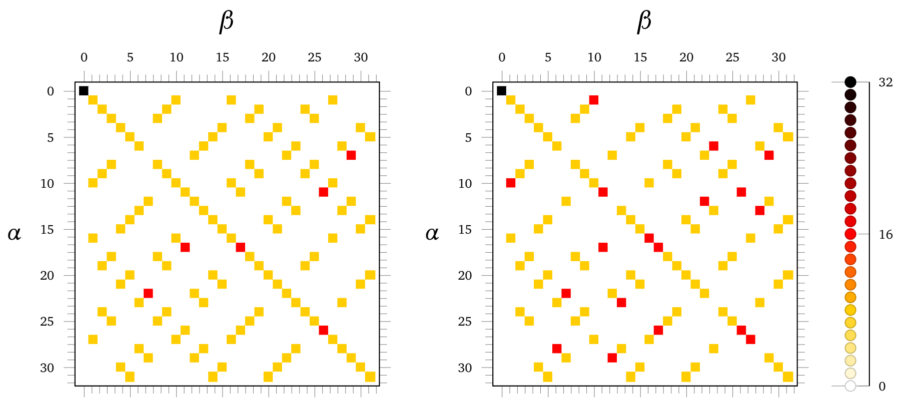
9	x^9+x^4+1	8	$x^8+x^4+x^3+x^2+1$
7	x^7+x+1	6	x^6+x+1
5	x^5+x^2+1	4	x^4+x+1

B DDT and LAT Figures

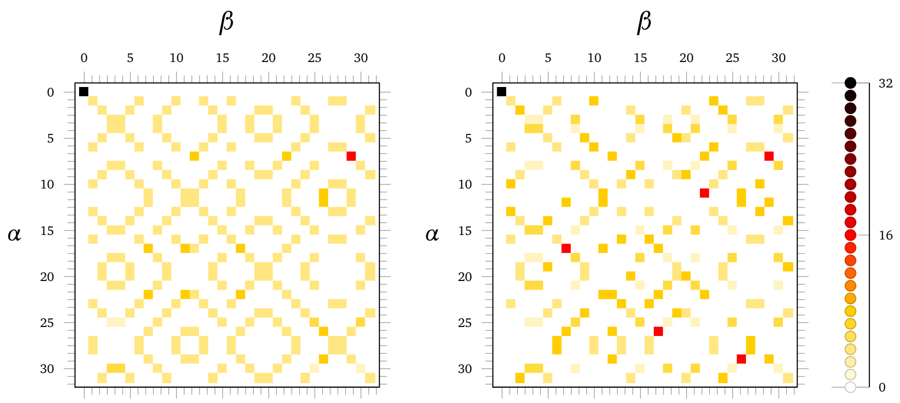
B.1 DDT for $k = 13$ averaged over w (left) resp. fixed $w = k$ (right)



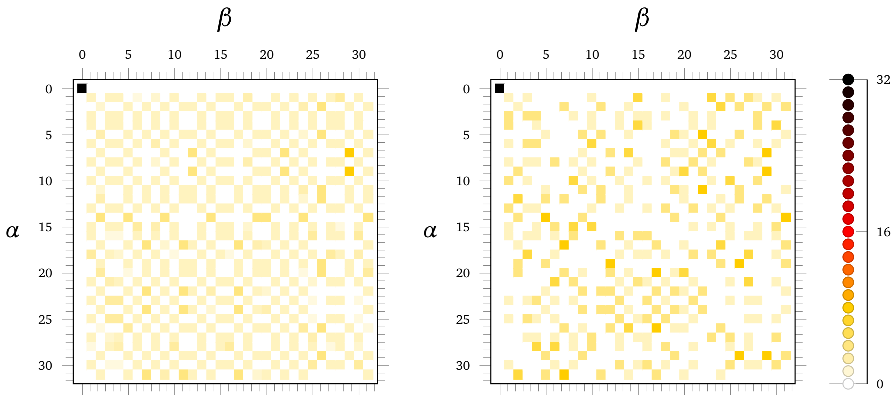
1 round



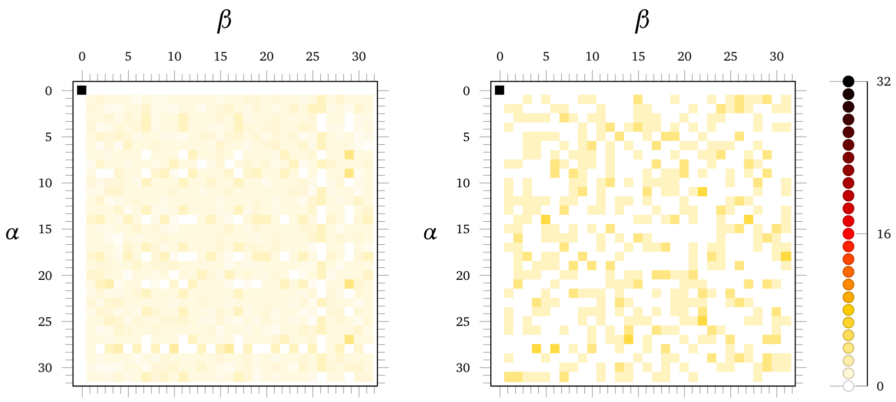
2 rounds



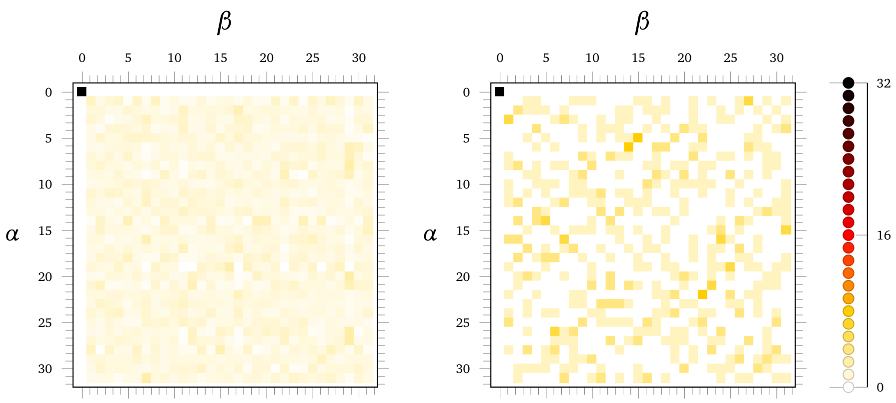
3 rounds



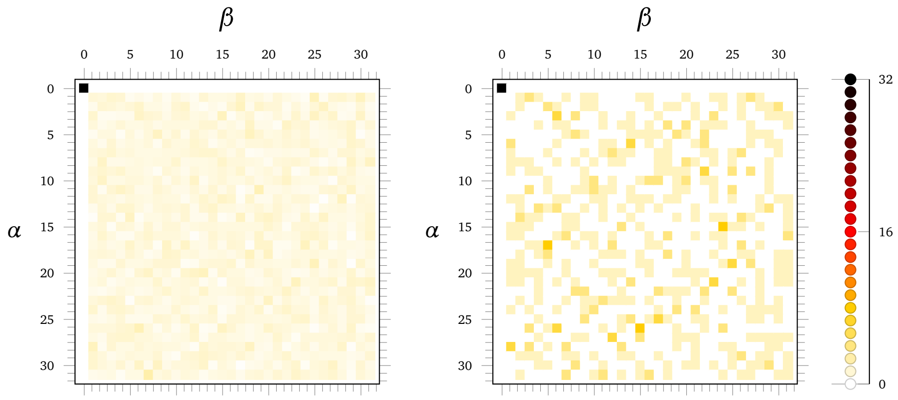
4 rounds



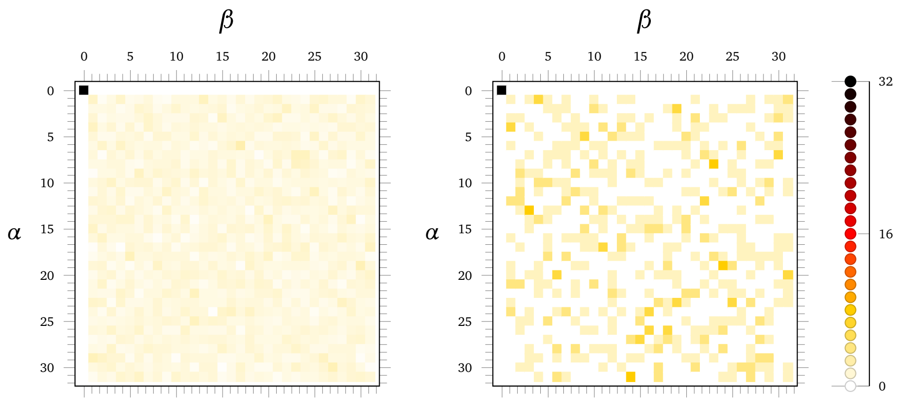
5 rounds



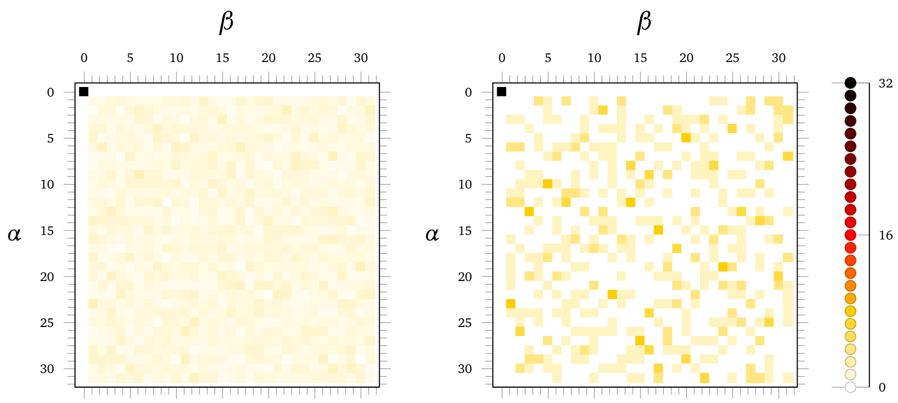
6 rounds



7 rounds

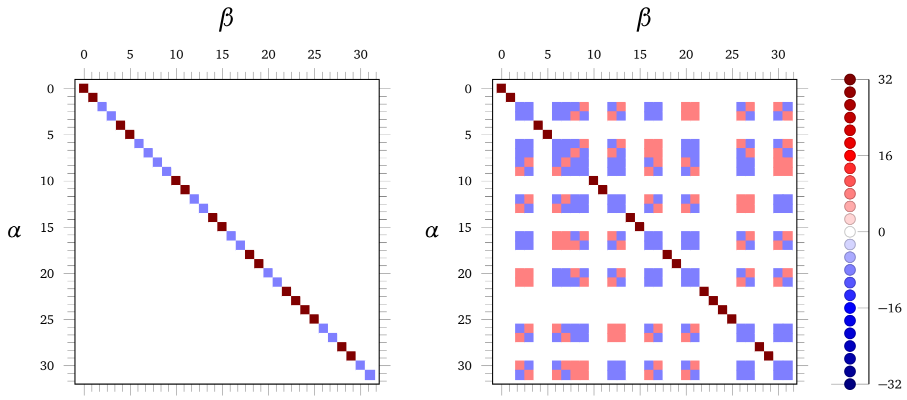


8 rounds

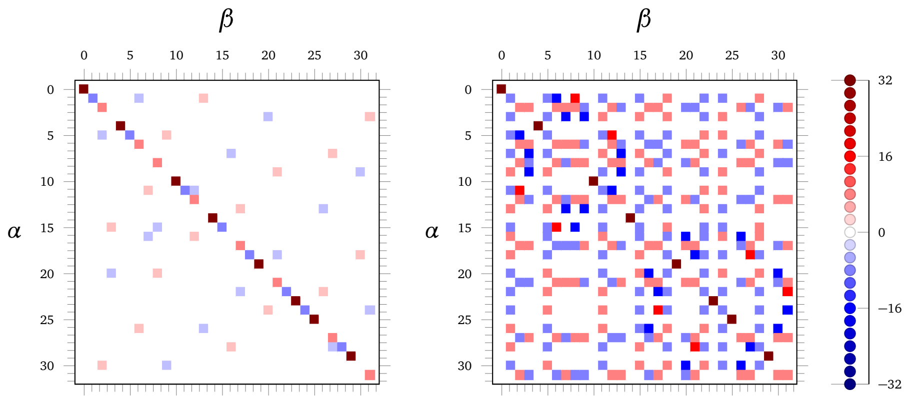


9 rounds

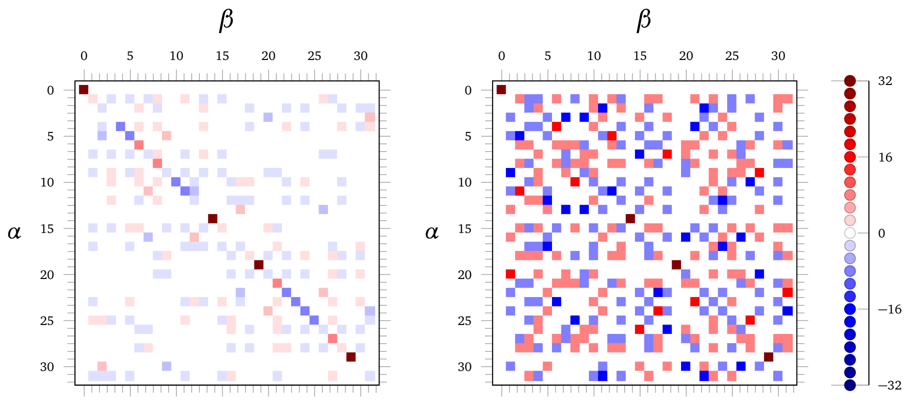
B.2 LAT for $k = 13$ averaged over w (left) resp. fixed $w = k$ (right)



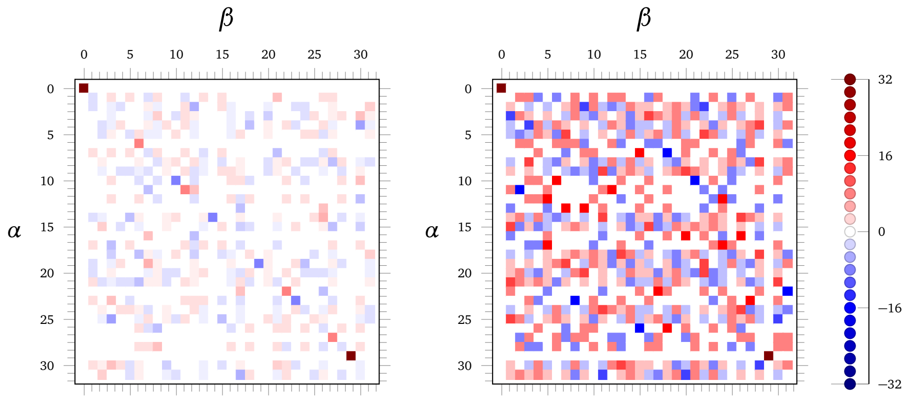
1 round



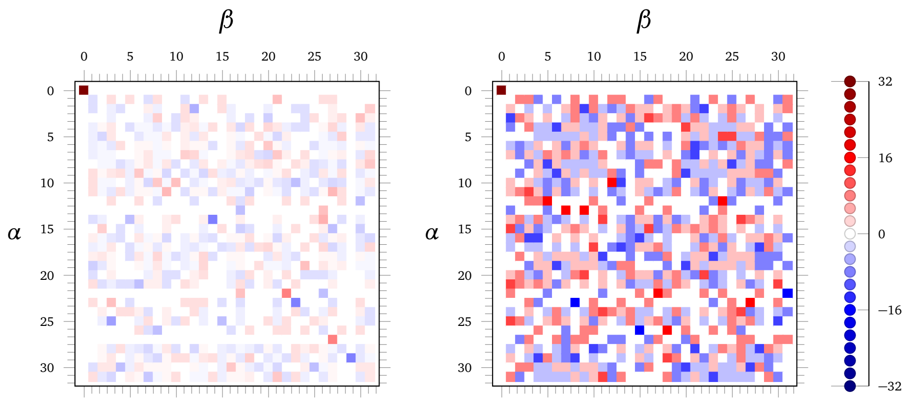
2 rounds



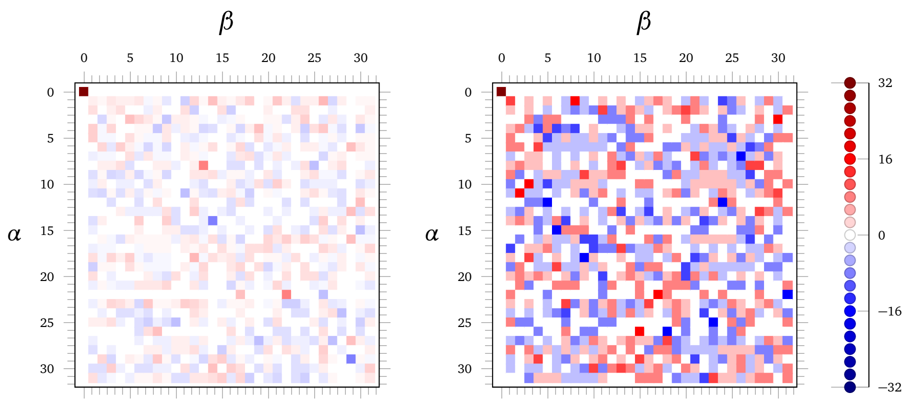
3 rounds



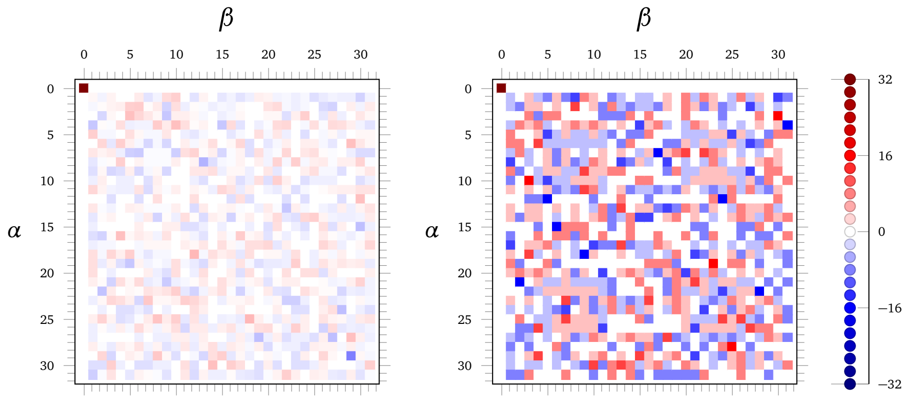
4 rounds



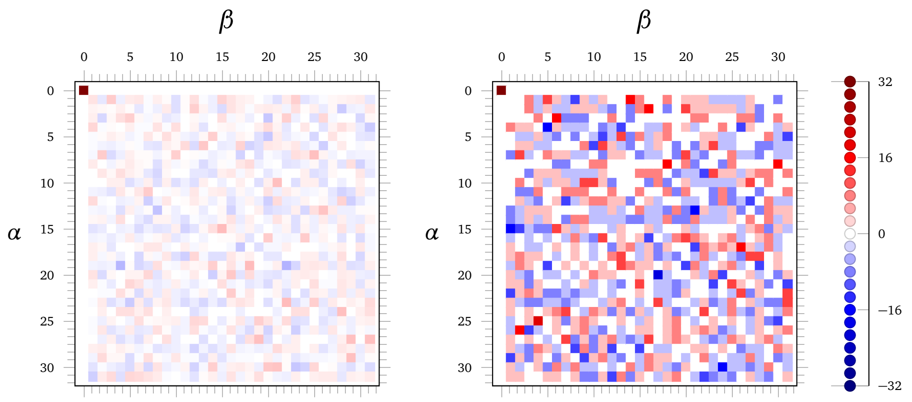
5 rounds



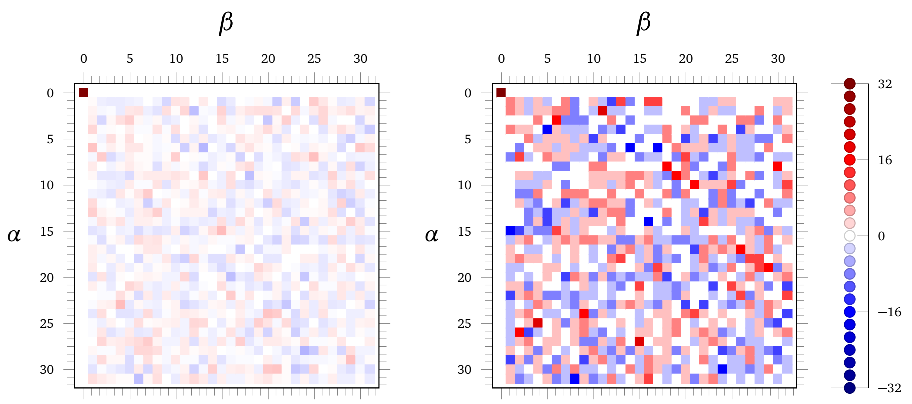
6 rounds



7 rounds



8 rounds



9 rounds

C Testvectors

Testvectors for $n = 129$, $k_0 = w_0 = k$, and $r = 387$ rounds, p plaintext, and c ciphertext. The polynomials used are $p_k(x) = x^{129} + x^5 + 1$, and $p_w(x) = x^{128} + x^7 + x^2 + x + 1$.

1. $p =$ `0x00000000000000000000000000000000`

   $k =$ `0x00000000000000000000000000000001`

   $c =$ `0x181cc4852868b2821895e250f296401d6`

2. $p =$ `0x000000000000000000000000000000001`
   $k =$ `0x000000000000000000000000000000001`
   $c =$ `0x031fe824e9ca7792006399496a1cf9252`

3. $p =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $k =$ `0x000000000000000000000000000000001`
   $c =$ `0x1d3f48720538f0a3a0e2ca7b4491ae587`

4. $p =$ `0x000000000000000000000000000000000`
   $k =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $c =$ `0x1c4100a60bf60e6b777b62f7b0c1ab5c2`

5. $p =$ `0x000000000000000000000000000000001`
   $k =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $c =$ `0x156b4215ca4587d821c9681761d6da1be`

6. $p =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $k =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $c =$ `0x03c5cbfb9ce0bd2ee33890aaed0a676f3`

7. $p =$ `0x0730b82b57fa8c9213a0305e2042d1198`
   $k =$ `0x000000000000000000000000000000001`
   $c =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`

8. $p =$ `0x14e95b7c90aa803d1209c040aa05ab335`
   $k =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`
   $c =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`

Additional intermediate results are listed in Table 3.

Table 3: Testvectors with intermediate results for $n = 129$, $k_0 = w_0 =$ `0x0deadbeefdeadbeefdeadbeefdeadbeef`, $p = x_0 = 1$. The used polynomials are $p_k(x) = x^{129} + x^5 + 1$, and $p_w(x) = x^{128} + x^7 + x^2 + x + 1$.

i	x_i	k_i	w_i	c_i	x_{i+1}
0	0x000000000000000000000000000000001	0x0deadbeefdeadbeefdeadbeefdeadbeef	0xdeadbeefdeadbeefdeadbeefdeadbeef	0x000000000000000000000000000000001	0x000000000000000000000000000000001
1	0x000000000000000000000000000000001	0x1bd5b7ddfbd5b7ddfbd5b7ddfbd5b7dde	0xbd5b7ddfbd5b7ddfbd5b7ddfbd5b7d59	0x800000000000000000000000000000043	0x000000000000000000000000000000001
2	0x000000000000000000000000000000001	0x17ab6fbbf7ab6fbbf7ab6fbbf7ab6fb9d	0x7ab6fbbf7ab6fbbf7ab6fbbf7ab6fa35	0xc00000000000000000000000000000062	0x000000000000000000000000000000001
3	0x000000000000000000000000000000001	0x0f56df77ef56df77ef56df77ef56df71b	0xf56df77ef56df77ef56df77ef56df46a	0x600000000000000000000000000000031	0x0f56df77ef56df77ef56df77ef56df71a
4	0x0f56df77ef56df77ef56df77ef56df71a	0x1eadbeefdeadbeefdeadbeefdeadbee36	0xeadbeefdeadbeefdeadbeefdeadbe853	0xb0000000000000000000000000000005b	0x11fb619831fb619831fb619831fb6192c
5	0x11fb619831fb619831fb619831fb6192c	0x1d5b7ddfbd5b7ddfbd5b7ddfbd5b7dc4d	0xd5b7ddfbd5b7ddfbd5b7ddfbd5b7d021	0xd8000000000000000000000000000006e	0x11fb619831fb619831fb619831fb6192c
6	0x11fb619831fb619831fb619831fb6192c	0x1ab6fbbf7ab6fbbf7ab6fbbf7ab6fb8bb	0xab6fbbf7ab6fbbf7ab6fbbf7ab6fa0c5	0x6c000000000000000000000000000037	0x0b4d9a274b4d9a274b4d9a274b4d9a197
7	0x0b4d9a274b4d9a274b4d9a274b4d9a197	0x156df77ef56df77ef56df77ef56df7157	0x56df77ef56df77ef56df77ef56df410d	0xb6000000000000000000000000000058	0x1e206d59be206d59be206d59be206d0c0
8	0x1e206d59be206d59be206d59be206d0c0	0x0adbeefdeadbeefdeadbeefdeadbee28f	0xadbeefdeadbeefdeadbeefdeadbe821a	0x5b00000000000000000000000000002c	0x1e206d59be206d59be206d59be206d0c0
9	0x1e206d59be206d59be206d59be206d0c0	0x15b7ddfbd5b7ddfbd5b7ddfbd5b7dc51e	0x5b7ddfbd5b7ddfbd5b7ddfbd5b7d04b3	0x2d800000000000000000000000000016	0x0b97b0a26b97b0a26b97b0a26b97b15de
10	0x0b97b0a26b97b0a26b97b0a26b97b15de	0x0b6fbbf7ab6fbbf7ab6fbbf7ab6fb8a1d	0xb6fbbf7ab6fbbf7ab6fbbf7ab6fa0966	0x16c0000000000000000000000000000b	0x00f80b55c0f80b55c0f80b55c0f809fc3
11	0x00f80b55c0f80b55c0f80b55c0f809fc3	0x16df77ef56df77ef56df77ef56df7143a	0x6df77ef56df77ef56df77ef56df4124b	0x8b600000000000000000000000000046	0x16277cba96277cba96277cba962778bf9
12	0x16277cba96277cba96277cba962778bf9	0x0dbeefdeadbeefdeadbeefdeadbee2855	0xdbeefdeadbeefdeadbeefdeadbe82496	0x45b00000000000000000000000000023	0x1b9993643b9993643b9993643b999a3ac
13	0x1b9993643b9993643b9993643b999a3ac	0x1b7ddfbd5b7ddfbd5b7ddfbd5b7dc50aa	0xb7ddfbd5b7ddfbd5b7ddfbd5b7d049ab	0xa2d80000000000000000000000000052	0x00e44cd960e44cd960e44cd960e45f306
14	0x00e44cd960e44cd960e44cd960e45f306	0x16fbbf7ab6fbbf7ab6fbbf7ab6fb8a175	0x6fbbf7ab6fbbf7ab6fbbf7ab6fa093d1	0x516c0000000000000000000000000029	0x00e44cd960e44cd960e44cd960e45f306
15	0x00e44cd960e44cd960e44cd960e45f306	0x0df77ef56df77ef56df77ef56df7142cb	0xdf77ef56df77ef56df77ef56df4127a2	0xa8b60000000000000000000000000057	0x00e44cd960e44cd960e44cd960e45f306
16	0x00e44cd960e44cd960e44cd960e45f306	0x1beefdeadbeefdeadbeefdeadbee28596	0xbeefdeadbeefdeadbeefdeadbe824fc3	0xd45b0000000000000000000000000068	0x1b0ab133bb0ab133bb0ab133bb0a77690
17	0x1b0ab133bb0ab133bb0ab133bb0a77690	0x17ddfbd5b7ddfbd5b7ddfbd5b7dc50b0d	0x7ddfbd5b7ddfbd5b7ddfbd5b7d049f01	0x6a2d8000000000000000000000000034	0x0cd74ae60cd74ae60cd74ae60cd627d9d
18	0x0cd74ae60cd74ae60cd74ae60cd627d9d	0x0fbbf7ab6fbbf7ab6fbbf7ab6fb8a163b	0xfbbf7ab6fbbf7ab6fbbf7ab6fa093e02	0x3516c000000000000000000000000001a	0x0cd74ae60cd74ae60cd74ae60cd627d9d
371	0x0a7364698b4eb1a460c25e6b20dde3f47	0x069e96296333862963338629633386296	0xfa42de74b812de74b812de74b812b187	0xd49e324034f56778ec4681a6eaf3d243	0x0cedf240e87d378d03f1d84243ee65dd1
372	0x0cedf240e87d378d03f1d84243ee65dd1	0x0d3d2c52c6670c52c6670c52c6670c52c	0xf485bce97025bce97025bce970256389	0xea4f19201a7ab3bc762340d37579e962	0x0cedf240e87d378d03f1d84243ee65dd1
373	0x0cedf240e87d378d03f1d84243ee65dd1	0x1a7a58a58cce18a58cce18a58cce18a58	0xe90b79d2e04b79d2e04b79d2e04ac795	0x75278c900d3d59de3b11a069babcf4b1	0x0cedf240e87d378d03f1d84243ee65dd1
374	0x0cedf240e87d378d03f1d84243ee65dd1	0x14f4b14b199c314b199c314b199c31491	0xd216f3a5c096f3a5c096f3a5c0958fad	0xba93c648069eacef1d88d034dd5e7a1b	0x0cedf240e87d378d03f1d84243ee65dd1
375	0x0cedf240e87d378d03f1d84243ee65dd1	0x09e962963338629633386296333862903	0xa42de74b812de74b812de74b812b1fdd	0xdd49e324034f56778ec4681a6eaf3d4e	0x0cedf240e87d378d03f1d84243ee65dd1
376	0x0cedf240e87d378d03f1d84243ee65dd1	0x13d2c52c6670c52c6670c52c6670c5206	0x485bce97025bce97025bce9702563f3d	0x6ea4f19201a7ab3bc762340d37579ea7	0x1f3f376c8e0df2a165811d6e259ea0fd7
377	0x1f3f376c8e0df2a165811d6e259ea0fd7	0x07a58a58cce18a58cce18a58cce18a42d	0x90b79d2e04b79d2e04b79d2e04ac7e7a	0xb75278c900d3d59de3b11a069babcf10	0x189abd3442ec78f9a9609736e97f2abfa
378	0x189abd3442ec78f9a9609736e97f2abfa	0x0f4b14b199c314b199c314b199c31485a	0x216f3a5c096f3a5c096f3a5c0958fc73	0x5ba93c648069eacef1d88d034dd5e788	0x17d1a985db2f6c4830a3838770bc3e3a0
379	0x17d1a985db2f6c4830a3838770bc3e3a0	0x1e96296333862963338629633386290b4	0x42de74b812de74b812de74b812b1f8e6	0x2dd49e324034f56778ec4681a6eaf3c4	0x094780e6e8a9452b0325aae4433a17314
380	0x094780e6e8a9452b0325aae4433a17314	0x1d2c52c6670c52c6670c52c6670c52149	0x85bce97025bce97025bce9702563f1cc	0x16ea4f19201a7ab3bc762340d37579e2	0x094780e6e8a9452b0325aae4433a17314
381	0x094780e6e8a9452b0325aae4433a17314	0x1a58a58cce18a58cce18a58cce18a42b3	0x0b79d2e04b79d2e04b79d2e04ac7e31f	0x0b75278c900d3d59de3b11a069babcf1	0x131f256a26b1e0a7cd3d0f688d22b31a7
382	0x131f256a26b1e0a7cd3d0f688d22b31a7	0x14b14b199c314b199c314b199c3148547	0x16f3a5c096f3a5c096f3a5c0958fc63e	0x85ba93c648069eacef1d88d034dd5e3b	0x07ae6e73ba80abbe510c44711113fb4e0
383	0x07ae6e73ba80abbe510c44711113fb4e0	0x096296333862963338629633386290aaf	0x2de74b812de74b812de74b812b1f8c7c	0xc2dd49e324034f56778ec4681a6eaf5e	0x07ae6e73ba80abbe510c44711113fb4e0
384	0x07ae6e73ba80abbe510c44711113fb4e0	0x12c52c6670c52c6670c52c6670c52155e	0x5bce97025bce97025bce9702563f18f8	0x616ea4f19201a7ab3bc762340d3757af	0x156b4215ca4587d821c9681761d6da1be
385	0x156b4215ca4587d821c9681761d6da1be	0x058a58cce18a58cce18a58cce18a42a9d	0xb79d2e04b79d2e04b79d2e04ac7e31f0	0xb0b75278c900d3d59de3b11a069bab94	0x156b4215ca4587d821c9681761d6da1be
386	0x156b4215ca4587d821c9681761d6da1be	0x0b14b199c314b199c314b199c3148553a	0x6f3a5c096f3a5c096f3a5c0958fc6367	0x585ba93c648069eacef1d88d034dd5ca	0x156b4215ca4587d821c9681761d6da1be

D SAGE Implementation

```
Fx = GF(2)["x"]
polys = {129: Fx("x^129 + x^5 + 1"),
         128: Fx("x^128 + x^7 + x^2 + x + 1"),
         127: Fx("x^127 + x + 1"),
         126: Fx("x^126 + x^7 + x^4 + x^2 + 1"),
# [...]
          81: Fx("x^81 + x^4 + 1"),
          80: Fx("x^80 + x^7 + x^5 + x^3 + x^2 + x + 1"),
# [...]
          65: Fx("x^65 + x^4 + x^3 + x + 1"),
          64: Fx("x^64 + x^4 + x^3 + x + 1"),
          63: Fx("x^63 + x + 1"),
          62: Fx("x^62 + x^6 + x^5 + x^3 + 1"),
# [...]
          33: Fx("x^33 + x^6 + x^4 + x + 1"),
          32: Fx("x^32 + x^7 + x^5 + x^3 + x^2 + x + 1"),
          31: Fx("x^31 + x^3 + 1"),
# [...]
           5: Fx("x^5 + x^2 + 1"),
           4: Fx("x^4 + x + 1"),
        }

class BISON:
    _n = 0
    _r = 0
    _kp = None
    _wp = None
    _cp = None

    def __init__(self, n_bits, r_rounds=None):
        self._n = n_bits
        if r_rounds is None:
            self._r = 3*self._n
        else:
            self._r = r_rounds

        assert n_bits in polys.keys() and n_bits % 2 == 1
        kp = polys[self._n]
        wp = polys[self._n-1]
        assert kp.is_primitive(), wp.is_primitive()
        self._kp = companion_matrix(kp)
        self._wp = companion_matrix(wp)
        self._cp = companion_matrix(wp).inverse()

    def _bits_to_int(self, bits):
        """
        converts a list/vector of bits to the corresponding
        integer. the lsb is at index 0
        """
        return reduce(lambda acc, x: acc*2 + Integer(x),
                      bits[::-1], 0)

    def _int_to_bits(self, x, n):
        """
        converts an integer x to a vector in GF(2) of at
        most n bit. if the binary representation of the
        integer needs more then n bits, the vector is
        truncated and the most significant bits are
        discarded.
        """
        bits = Integer(x).digits(base=2, padto=n)[:n]
        return vector(GF(2), bits)

```

```
    def keyschedule(self, key_k, key_w, i):
        """
        one possible keyschedule for the whitened swap or
        not construction. computes K_i, W_i as x^i K,W
        modulo kp(x)/wp(x)
        """
        ki = (self._kp^i) * key_k
        wi = (self._wp^i) * key_w
        ci = (self._cp^i) * vector(GF(2), [1] + [0]*(self._n-2))
        return ki, wi, ci

    def f_i(self, vec):
        """
        one possible f function for the whitened swap or
        not construction f(x,y) := <x, y>
        """
        x = vec[:(self._n-1)/2]
        y = vec[(self._n-1)/2:]
        return x * y

    def phi(self, k, x):
        """
        replacement of max(.)
        """
        assert k != 0

        i = list(k).index(1)
        return matrix(GF(2),
                      x[i]*k + x).delete_columns([i]).row(0)

    def round(self, x, key_k, key_w, i):
        """
        computes the i'th round of the whitened swap or
        not construction
        """
        f_bit = 0 if i <= self._r / 2 else 1

        ki, wi, ci = self.keyschedule(key_k, key_w, i)

        if self.f_i(wi + ci + self.phi(ki, x)) == (1+f_bit):
            x = x + ki
        else:
            x = x

        return x

    def encrypt(self, plain, key_k, key_w):
        """
        encrypts the input plain under key, assuming n bit
        input length and number of rounds many iteration.

        TESTS:
            sage: cipher = BISON(5, 10)
            sage: p = randint(1, (1<<5)-1),
            sage: k = randint(1, (1<<5)-1),
            sage: w = randint(1, (1<<4)-1),
            sage: assert p ==                              \
            ....:     cipher.decrypt(cipher.encrypt(  \
            ....:                        p, k, w), k, w)
        """
        assert key_k != 0

        state = self._int_to_bits(plain, self._n)
        kvec = self._int_to_bits(key_k, self._n)
        wvec = self._int_to_bits(key_w, self._n-1)
        for i in range(self._r):
```

```
            state = self.round(state, kvec, wvec, i)
        return self._bits_to_int(state)

    def decrypt(self, cipher, key_k, key_w):
        """
        decrypts the input cipher under key, assuming n bit
        input length and number of rounds many iteration.
        """
        state = self._int_to_bits(cipher, self._n)
        kvec = self._int_to_bits(key_k, self._n)
        wvec = self._int_to_bits(key_w, self._n-1)
        for i in range(self._r-1, -1, -1):
            state = self.round(state, kvec, wvec, i)
        return self._bits_to_int(state)
```

E C Implementation

```
#include <stdbool.h>
#include <stdint.h>
#include <string.h>

#define N_BITS_DEFINE 129lu
#define N_ROUNDS (3*N_BITS_DEFINE)
#define W_DEFINE 0x87
#define K_DEFINE 0x21
#define C_DEFINE 0x43

struct key_w { uint64_t words[2]; };
struct key_k { uint64_t words[2]; bool msb; };
typedef struct key_w key_w;
typedef struct key_w key_c;
typedef struct key_k key_k;
typedef struct key_k state;

inline void copy(struct key_k* lhs,
                 const struct key_k* rhs) {
    lhs->words[0] = rhs->words[0];
    lhs->words[1] = rhs->words[1];
    lhs->msb = rhs->msb;
}

inline void add_w(struct key_k* lhs,
                  const key_w* rhs) {
    lhs->words[0] ^= rhs->words[0];
    lhs->words[1] ^= rhs->words[1];
}

inline void add(struct key_k* lhs,
                const struct key_k* rhs) {
    lhs->words[0] ^= rhs->words[0];
    lhs->words[1] ^= rhs->words[1];
    lhs->msb ^= rhs->msb;
}

void ks(key_c* c, key_w* w, key_k* k) {
    bool test_bit0 = 0;
    bool test_bit1 = 0;
    test_bit0 = w->words[0] >> 63;
    test_bit1 = w->words[1] >> 63;
    w->words[0] = (w->words[0] << 1) ^ (test_bit1 * W_DEFINE);
    w->words[1] = (w->words[1] << 1) ^  test_bit0;

    test_bit0 = c->words[0] & 1;
    test_bit1 = c->words[1] & 1;
    c->words[0] = (c->words[0] >> 1) ^ ((uint64_t)test_bit1 << 63) \
        ^ (test_bit0 * C_DEFINE);
    c->words[1] = (c->words[1] >> 1) ^ ((uint64_t)test_bit0 << 63);

    test_bit0 = k->words[0] >> 63;
    test_bit1 = k->words[1] >> 63;
    k->words[0] = (k->words[0] << 1) ^ (k->msb * K_DEFINE);
    k->words[1] = (k->words[1] << 1) ^  test_bit0;
    k->msb = test_bit1;
}

bool f_i(const state* x) {
    uint64_t and = x->words[0] & x->words[1];
    uint8_t output = __builtin_popcount(and & 0xffffffff);
    return (output + __builtin_popcount(and >> 32)) % 2;
}
```

```
void phi(state* x, const key_k* k) {
    uint8_t idx = 0;
    uint8_t word_idx = 0;

    // find index of lowest set bit in k
    while (true) {
        word_idx = 0;
        if (idx == 128) {
            if (k->msb == 1) {
                break;
            }
        } else if (idx >= 64) {
            word_idx = 1;
        }

        if (((k->words[word_idx] >> (idx - word_idx*64)) & 1) == 1) {
            break;
        }

        idx += 1;
    }

    // test that bit in x and if it is set add
    // key k to state x
    bool bit_set = false;
    if (idx == 128) {
        bit_set = x->msb == 1;
    } else {
        bit_set = (((x->words[word_idx] >> (idx - word_idx*64)) & 1) == 1);
    }

    if (bit_set) {
        add(x, k);
    }

    // delete the bit from x by shifting everything
    // higher than that bit one to the right
    if (idx == 128) {
        return;
    } else if (idx >= 64) {
        uint64_t constant = x->words[1] % (1lu << (idx-64));
        x->words[1] = ((uint64_t)x->msb << 63) ^ ((x->words[1]^constant) >> 1) \
            ^ constant;
    } else {
        uint8_t bit = x->words[1] & 1;
        x->words[1] = ((uint64_t)x->msb << 63) ^ (x->words[1] >> 1);
        uint64_t constant = x->words[0] % (1lu << idx);
        x->words[0] = (((uint64_t)bit) << 63) ^ ((x->words[0]^constant) >> 1) \
            ^ constant;
    }
    x->msb = 0;
}

/**
 * return the output after one round of the swap or
 * not construction under the keys w and k, where
 * fi_bit is either 0 or 1 and fi is f_i(x) xor fi_bit
 */
void swap_or_not_round(state* x, state* tmp_x,
                       const key_c* c, const key_w* w,
                       const key_k* k, int i) {
    copy(tmp_x, x);
    phi(tmp_x, k);
    add_w(tmp_x, w);
    add_w(tmp_x, c);
```

```
    if (f_i(tmp_x) == (1^i))
        add(x, k);
}

/**
 * inplace encrypts x under the keys w and k over
 * N=3*nroudns rounds
 */
void encrypt(state* x, key_w* w, key_k* k) {
    state tmp_x;
    key_c c = {{1,0}};
    for (size_t i = 0; i < N_ROUNDS; i++) {
        swap_or_not_round(x, &tmp_x, &c, w, k,
            i <= N_ROUNDS/2 ? 0 : 1);
        ks(&c, w, k);
    }
}
```