Greedy distinguishers and nonrandomness detectors

Stankovski, Paul

Published in:
Progress in Cryptology - INDOCRYPT 2010 / Lecture Notes in Computer Science

DOI:
10.1007/978-3-642-17401-8_16

2010

Greedy Distinguishers and Nonrandomness Detectors

Paul Stankovski

Dept. of Electrical and Information Technology, Lund University,
P.O. Box 118, 221 00 Lund, Sweden

Abstract. We present the concept of greedy distinguishers and show how some simple observations and the well known greedy heuristic can be combined into a very powerful strategy (the Greedy Bit Set Algorithm) for efficient and systematic construction of distinguishers and nonrandomness detectors. We show how this strategy can be applied to a large array of stream and block ciphers, and we show that our method outperforms every other method we have seen so far by presenting new and record-breaking results for Trivium, Grain-128 and Grain v1.
We show that the greedy strategy reveals weaknesses in Trivium reduced to 1026 (out of 1152) initialization rounds using 2^{45} complexity – a result that significantly improves all previous efforts. This result was further improved using a cluster; 1078 rounds at 2^{54} complexity. We also present an 806-round distinguisher for Trivium with 2^{44} complexity.
Distinguisher and nonrandomness records are also set for Grain-128. We show nonrandomness for the full Grain-128 with its 256 (out of 256) initialization rounds, and present a 246-round distinguisher with complexity 2^{42}.
For Grain v1 we show nonrandomness for 96 (out of 160) initialization rounds at the very modest complexity of 2^7, and a 90-round distinguisher with complexity 2^{39}.
On the theoretical side we define the Nonrandomness Threshold, which explicitly expresses the nature of the randomness limit that is being explored.

Keywords: algebraic cryptanalysis, distinguisher, nonrandomness detector, maximum degree monomial, Trivium, Grain, Rabbit, Edon80, AES, DES, TEA, XTEA, SEED, PRESENT, SMS4, Camellia, RC5, RC6, HIGHT, CLEFIA, HC, MICKEY, Salsa, Sosemanuk

1 Introduction

The output of a sensibly designed cipher should appear random to an external observer. Given a random-looking bit sequence, that observer should not be able to tell if the sequence is genuinely produced by the cipher in question or not. This simple idea is the core of cryptographic distinguishers and nonrandomness detectors.

Recently we have seen several attempts at finding distinguishers and nonrandomness detectors and the best ones seem to be built using the maximum degree monomial test (see [27, 11]) or some derivative of it. This test is superb for detecting nonrandomness, but it also provides a window into the internals of the cryptographic algorithm we are examining. The maximum degree monomial test can provide statements such as "The IV bits are not mixed properly", which can be invaluable to the algorithm designer.

The core of this test is a bit set, and the efficiency of the test is largely determined by how this bit set is selected. For this selection process, it seems that guesswork has been the most prominent ingredient. The reason for this may be that systematic methods have seemed too complicated to find or use, or simply that the importance of bit set selection has been underestimated. By far, the best systematic approach we have seen so far was due to Aumasson et al. [2]. They used a genetic algorithm to select a bit set, and this is a very reasonable approach for unknown and complex searchspaces. The complexity of the searchspace depends on the algorithm we are examining, but are they really so complex that we need to resort to such methods? In this paper we present a very simple deterministic and systematic approach that outperforms all other methods we have seen so far. We call it the Greedy Bit Set Algorithm.

Stream ciphers have an initialization phase, during which they "warm up" for a number of rounds before they are deemed operational. Block ciphers are not explicitly initialized in this way, but they do operate in rounds. For our purposes, this can be translated into an initialization phase.

How many rounds are needed to warm up properly? This is a question that every algorithm designer has been faced with, but we have not yet seen any satisfactory answer to this question. We make some observations that lead us to a definition of the Nonrandomness Threshold, which helps us to better understand the nature of the problem. The Greedy Bit Set Algorithm is a tool that can and should be used by designers to determine realistic lower bounds on the initialization period for their algorithm.

We go on to show how the Greedy Bit Set Algorithm performs against a wide variety of new and old stream and block ciphers, and we find new record-breaking results for Trivium, Grain-128 and Grain v1. We reveal weaknesses in Trivium reduced to 1026 out of 1152 initialization rounds in 2^{45} complexity, thereby significantly improving all previous efforts. By using a cluster we are able to improve this result even further to 1078 rounds at 2^{54} complexity. For Trivium we also present a new 806-round

distinguisher of complexity 2^{44}. Both distinguishing and nonrandomness records are also set for Grain-128. We show nonrandomness in 256 (out of 256) initialization rounds, and present a 246-round distinguisher with complexity 2^{42}. For Grain v1 we show nonrandomness for 96 (out of 256) initialization rounds for a cost of only 2^7.

The paper is organized as follows. In Section 2 we give an overview of the black box model attack scenario and explain the maximum degree monomial test. We also briefly describe the software tools developed for this paper. In Section 3 we present our Greedy Bit Set Algorithm, comment on the importance of key weight and define the Nonrandomness Threshold. In Sections 4 and 5 we present and summarize our findings for the various algorithms. Finally, some concluding remarks are given in Section 6.

As a frame of reference, this article takes Filiol [12], Saarinen [27] and Englund et al. [11] as a starting point, and the most relevant previous work is due to Aumasson et al. [1, 2] (see also Knudsen and Rijmen [19], Vielhaber [30], Dinur and Shamir [10] and Fischer, Khazaei and Meier [13]).

2 Background

2.1 The Black Box Model

Distinguishers may be built for block ciphers, stream ciphers, MACs, and so on, so adopting a black-box view of the cryptographic primitive is instructive. Consider the set-up in Fig. 1, dividing entities into potential input and output parameters to the left and right, respectively.

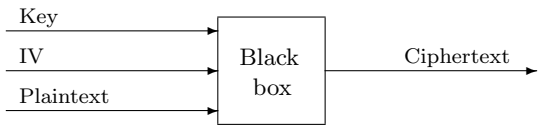

Fig. 1. Black box view of a cipher.

A distinguisher attempts to determine if a given black box produces true random output or not. No cryptographic primitive produces truly random output, so the distinguisher can be thought of as a classifier. Given an output producing black box, the distinguisher answers "random" or "cipher", depending on its assertion. The distinguisher is said to

be efficient if it significantly outperforms guessing, where the meaning of 'significantly' depends on the application.

For a distinguisher, the key is fixed and unknown. That is, the distinguisher may invoke the black box several times with different IVs, but the key is kept fixed. The IV bits constitute the input parameter bit space $B = \{0,1\}^m$. The fixed key black box scenario is typical for real-world applications, and distinguishers are practical in the sense that they can be used in such a scenario.

Nonrandomness detectors are what we get when the input parameter bit space B includes key bits[1]. This renders them less useful in a real-world fixed key black box scenario, since they are related-key creatures by construction. Their merit, however, is that they can do a better job of detecting nonrandomness. This is invaluable for the cryptographic community, as we can get earlier indications on weaknesses in specific algorithms. Distinguishers show weaknesses in how IV bits are handled, while nonrandomness detectors, in addition, can show weaknesses in how key bits are handled.

Explicitly summarizing the above, we have

Distinguisher: A {'random','cipher'}-classifier whose input parameter bit set B *does not* include key bits.

Nonrandomness detector: A {'random','cipher'}-classifier whose input parameter bit set B *does* include key bits.

Note, using a known or chosen key makes the {'random','cipher'}-classifier a nonrandomness detector, as we are then restricting the key space and effectively allowing key bits in B. A related discussion can be found in [19].

2.2 The Maximum Degree Monomial Signature

Algebraic techniques in general have recently been shown to be very powerful, and the maximum degree monomial (MDM) test stands out as a highly efficient randomness test. We have used this test in the following natural setting.

Consider a black box cipher that has been modified to produce output during its l initialization rounds. Choose a subset $S = \{0,1\}^n$ of the input

[1] We have not examined the effect of allowing plaintext bits in B, but this has the potential of working very well as these bits usually enter the state *after* both key and IV bits. This is true for block ciphers, but generally not for stream ciphers as encryption in that case usually works by simply XORing plaintext and keystream.

variables B and regard the l-bit initialization round output of the black box as a Boolean vector function of the n variables $x_1, \cdots, x_n$ in S. Letting $f : \{0,1\}^n \to \{0,1\}^l$ denote the Boolean vector function, the sum

$$\sum_{x \in \{0,1\}^n} f(x)$$

produces a maximum degree monomial signature $\{0,1\}^l$ for the cipher. Note that this implicitly defines l (regular) Boolean functions $f_i, 1 \leq i \leq l$, one for each output bit. The ith signature bit is the coefficient of the maximum degree monomial $x_1 \cdots x_n$ in the algebraic normal form of f_i (see [27, 2]).

An ideal cipher produces a random-looking MDM signature. That is, if a boolean function $g : \{0,1\}^n \to \{0,1\}$ is chosen uniformly at random from the universe of all such boolean functions, the maximum degree monomial exists in g with probability $\frac{1}{2}$.

The MDM signature for Trivium over the set consisting of every third IV bit, setting all other key and IV bits to zero, is

$$\underbrace{000\ldots000}_{930 \text{ zeros}}101\ldots.$$

The long sequence of leading zeros is very striking. We conclude that the sequence appears random-like close to where the first 1-bit appears, at round 931. We say that we have observed 930 zero rounds, and one interpretation of this is that 930 initialization rounds are not sufficient to properly mix the corresponding IV bits. Note that this is a nonrandomness result (chosen key).

Running the MDM test (producing the MDM signature and counting the number of initial zero rounds) over any given bit set S (permitting both key and IV bits) for an otherwise fixed key and fixed IV will produce a nonrandomness result. Fixed key nonrandomness detection over a bit set of size n has complexity 2^n and requires $O(l)$ space.

If the bit set S contains only IV bits, we have also implicitly produced a corresponding distinguisher. To assess the efficiency of this distinguisher, its performance needs to be sampled over *random* keys. Many different bias tests can be used here, but we have used MDM signature bit constantness (equal to zero) as measure, and two approaches stand out as simple, reasonable and typical.

Taking the *minimum* number of zero rounds over N randomly sampled keys assesses a distinguisher in $N \times 2^n$ time and $O(l)$ space. The time

required for running this distinguisher is, however, only 2^n. Higher values for N increase the confidence level of the zero round number estimate.

Alternatively, taking the *maximum* number of zero rounds over N random keys assesses a distinguisher in $N \times 2^n$ time and $O(l)$ space. In the black box attack scenario, we need to examine N different black boxes before we find one that our distinguisher works for. The total running time for this distinguisher is therefore $N \times 2^n$. Taking the maximum costs us a factor N.

It is reasonable to take the maximum approach when the number of zero rounds varies heavily over the randomly selected keys. Without so much variation, it is more reasonable to take the minimum. This trades a few zero rounds for better time complexity. If complexity is less important, the highest zero round count is obtained by taking the maximum.

One key point is that the MDM test seems to be highly efficient and works very well in practice for some cryptographic algorithms. Another key point that makes the MDM tests attractive is that all output sequences can be successively XORed, so only a negligible amount of storage is required. Furthermore, one does not need to know anything at all about the internals of the algorithm that is being tested. The algorithm will quite politely but candidly reveal how susceptible any black box algorithm is to the MDM test.

2.3 Black Box Framework

A specialized cryptographic library that permits output of initialization data was put together for this paper. The library was written in C and supports bitsliced implementations and threading to make good use of multiple cores. This is something that the MDM test benefits from since it is spectacularly parallelizable. A unified interface makes it simple to author generic tests that can be used for all supported algorithms, and LaTeX-graphs of the results can be generated. This framework is an excellent tool for testing future generators. Interested researchers may find both ready-to-use executables and source code at [29]. The stream- and block ciphers used for this paper are listed in Table 1.

3 The Algorithm and a Threshold

3.1 The Greedy Bit Set Algorithm

The trick to obtaining good results with the MDM test is to find an efficient bit set S for summation, a bit set that produces many zero rounds.

Table 1. Algorithms used to obtain the results in this paper.

Stream ciphers		Block ciphers	
Trivium [8]	Rabbit [6]	AES-128 [24]	AES-256 [24]
Grain v1 [16]	Grain-128 [15]	DES [23]	PRESENT [7]
Edon80 [14]	MICKEY v2 [3]	RC5 [25]	RC6 [26]
HC-128 [34]	HC-256 [33]	TEA [31]	XTEA [32]
Salsa20/12 [5]	Sosemanuk [4]	SEED [20]	SMS4 [9]
		Camellia [22]	HIGHT [17]
		CLEFIA [28]	

The well known greedy heuristic provides a very simple but yet highly successful algorithm that outperforms all methods we have seen so far. The algorithm is made explicit in Fig. 2.

Algorithm GreedyBitSet
Input: key k, IV v, bit space B, desired bit set size n.
Output: Bit set S of size n.

$S \leftarrow \emptyset$;
repeat n times {
  $S \leftarrow$**GreedyAddOneBit**(k, v, B, S);
}
return S;

Fig. 2. The Greedy Bit Set Algorithm.

Note that k and v are fixed, and that the bit space parameter B determines if key and/or IV bits may be used to build the resulting bit set. The subroutine GreedyAddOneBit is specified in Fig. 3.

Further note that the algorithm in Fig. 2 illustrates the straightforward greedy "add best bit"-strategy for building the resulting bit set S. GreedyBitSet can, by avoiding unnecessary recalculations, easily be implemented to sport a running time of precisely[2]

$$1 + \sum_{0 \le i < n} (m - i)2^i < m2^n$$

initializations for building a bit set of size n, where m is the size of the permissible bit space B.

[2] There are m choices for the first bit, $m - 1$ choices for the second bit, and so on.

```
Algorithm GreedyAddOneBit
Input: key k, IV v, bit space B, bit set S of size n.
Output: Bit set S' of size n + 1.

bestBit ← none;
max ← −1;
for all b ∈ B\S {
    zr ← numInitialZeroRounds(MDMsignature(k, v, S ∪ {b}));
    if (zr > max) {
        max ← zr;
        bestBit ← b;
    }
}
return S ∪ {bestBit};
```

Fig. 3. The **GreedyAddOneBit** subroutine.

As a generalization one may allow other bit set building strategies, or a non-empty starting bit set. In this somewhat generalized form we denote an instance of the algorithm

$$\textbf{GreedyBitSet}(\text{strategy}, \text{starting bit set}, \text{primitive}, \text{bit space}, \text{key}, \text{IV}).$$

For example, running the Greedy Bit Set Algorithm with the "add best bit"-strategy on Trivium starting with an empty bit set, allowing only IV bits in the bit set, using the all-ones key and setting all remaining IV bits to zero may be denoted

$$\textbf{GreedyBitSet}(\text{Add1}, \emptyset, \text{Trivium}, \{\text{IV}\}, \mathbf{1}, \mathbf{0}).$$

Instead of starting with an empty bit set one may begin by computing a small optimal bit set and go from there. For most of our results below we have used optimal bit sets of sizes typically around five or six.

An alternative bit set building strategy is denoted "Add$N$". Add$N$ operates by adding the N bits that together produce the highest zero round count when added to the existing set. These bit sets should heuristically be better than the ones produced using the Add1 strategy as local optima are more likely to be avoided. The performances of the Add1 and Add2 strategies for Grain-128 are compared in Fig. 4, where the darker curve represents the Add2-strategy. GreedyBitSet with AddN strategy can be implemented with a running time of precisely[3]

[3] There are $\binom{m}{N}$ choices for the first bit, $\binom{m-N}{N}$ choices for the second bit, and so on.

$$1 + \sum_{0 \le i < k} \binom{m - iN}{N} 2^i < m^N 2^k$$

initializations for building a bit set of size kN.

We have standardized the graphs for uniform comparison between algorithms. Given a bit set, the portion of leading zero rounds in the initialization rounds is denoted 'bit set efficiency'.

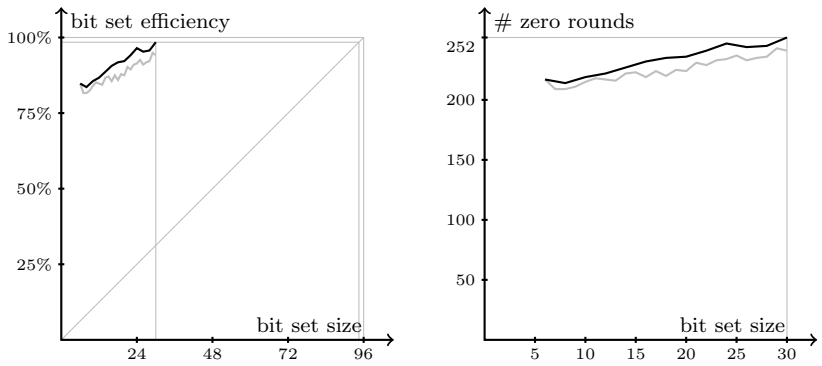

Fig. 4. Add1 (gray) vs. Add2 strategy (black) for Grain-128.

For an ideal cipher, a bit set of size n produced by the Greedy Bit Set Algorithm will admit around $\lg(m - n)$ zero rounds.

3.2 Key Weight and the Nonrandomness Threshold

For some ciphers we have found that the result of the MDM test depends heavily on the weight of the key. A typical example of this is Trivium, for which the test seems to work best for the all-zeros key and worst for the all-ones key. Fig. 5 shows the efficiency of the bit sets produced by the Greedy Bit Set Algorithm for Trivium, starting with an empty set, using zero IV fill for these two keys.

For Trivium it seems that the all-zeros and all-ones keys are extreme cases. All other keys we have tried end up producing a curve that lies between these two, and a curve produced by averaging over several randomly chosen keys certainly falls between as well. So which value is most interesting: the maximum, minimum or the averaged one? Which zero round count should be reported? An attacker working on a deadline might be

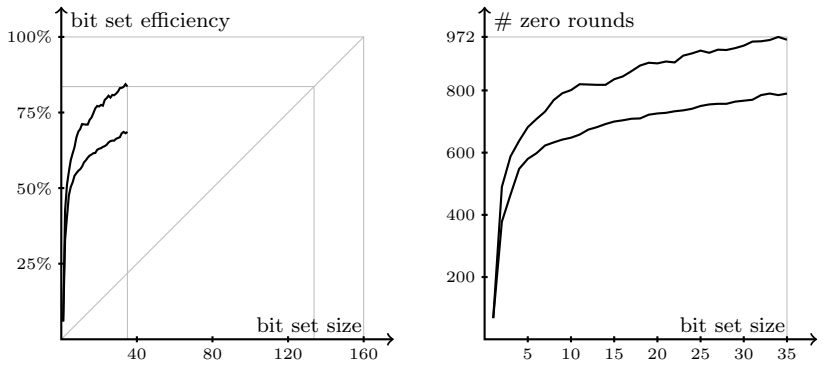

Fig. 5. The all-zeros key works better than the all-ones key for Trivium.

interested in the average performance over random keys, or possibly in the worst case performance if her deadline is really tight. But the algorithm designer may have quite other preferences.

Consider an algorithm analyst that needs to determine a reasonable number for how many initialization rounds that are needed for balancing initialization time and security in Trivium. Using the graphs in Fig. 5, the analyst can see that 1000 rounds will just barely withstand signs of improper mixing in this setting. At 972 rounds we start finding keys that allow us to prove that the bit mixing is inadequate. As we keep reducing the number of rounds, more and more keys show the same vulnerability. At 790 rounds, more or less all keys simultaneously chant "Inadequate mixing" in four-part harmony. The algorithm designer should, of course, in this case decide on an initialization round count well above 972. How much more is debatable.

Recall that we use a bit set $S = \{0,1\}^n$ which is a subset of the entire bit space $B = \{0,1\}^m$. The highest round count value 972 obtained above should really be viewed as a lower bound of a threshold - the Nonrandomness n-Threshold for bit sets of size n. That is the nature of the limit we are exploring here, a threshold for the existence of proof of inadequate bit mixing. Testing a specific bit set of size n over a single key and IV provides a lower bound for this threshold. The true threshold value is conceptually obtained by repeating the MDM Test several times taking the maximum over all possible keys, IVs and bit sets of size n for a total complexity of $\binom{m}{n}2^{m-n}$.

Definition 1. ***Nonrandomness n-Threshold***
The maximum number of zero rounds attainable according to

$$\max \ \text{numInitialZeroRounds}(\text{MDMsignature}(k, v, B, S)),$$

where the maximum is taken over $S \subseteq B$ with $|S| = n$, $k \in K$ and $v \in V$. B, K and V are the bit set-, key- and IV space, respectively.

4 Results

The algorithms are grouped according to susceptibility to the MDM test below, where particularly interesting algorithms are given room for elaboration. An algorithm is given a susceptibility rating high, significant, moderate or low according to its tendency to submit to the MDM test as the bit set size gets larger.

A direct comparison of our results to the previous best ones can be found in Table 2 in Section 5.

4.1 High Susceptibility

TEA and The Bit Flip Test TEA is the top candidate. Starting with an empty bit set, we reach a full 100% zero round count after only two key bits have been added. It is the key bits that are the weak link, and this is a previously known deficiency in TEA (see [18]). The picture becomes quite different when one considers IV bit mixing. Allowing IV bits only results in a susceptibility that seems to be inherently low.

It seems that the shortcomings in key bit mixing have been properly dealt with for XTEA, as the Greedy Bit Set Algorithm cannot show anything beyond a low susceptibility level for any bit type. There is something we can learn from TEA. The TEA flaw is revealed by flipping two key bits, in which case the output does not change. We can devise an automated test for these simple symmetry faults. A Bit Flip Test can be defined by adding the two output sequences produced before and after flipping all bits in a given bit set. Trying all bit sets of small size will catch design flaws such as the one in TEA. The Bit Flip Test is, in fact, a MDM test for a bit set of size 1 with a prior change of basis. Instead of summing over a perfect cube, we sum over the "tilted" cube that is the result of a linear transformation of the basis.

Two such two-bit configurations are known for TEA, and we have verified that no other ones exist. We have also verified that none of the

other algorithms we are considering here show any such bit flip weaknesses for small bit set sizes (five or so).

The Bit Flip Test should really be part of every algorithm designer's toolbox. This test, and many others, should be used routinely to check for errors or unexpected behavior.

Grain-128 For Grain-128, IV bits have a tendency to be more efficient than key bits and, as with Trivium, low weight keys work better than high weight keys. Running the Greedy Bit Set Algorithm on the all-zeros key with the Add2-strategy up to bit set size 40, IV bits only, we produced a nonrandomness detector for the full Grain-128 with its 256 initialization rounds. The successive development from the optimal 6-set to bit set size 40 is shown in Fig. 6.

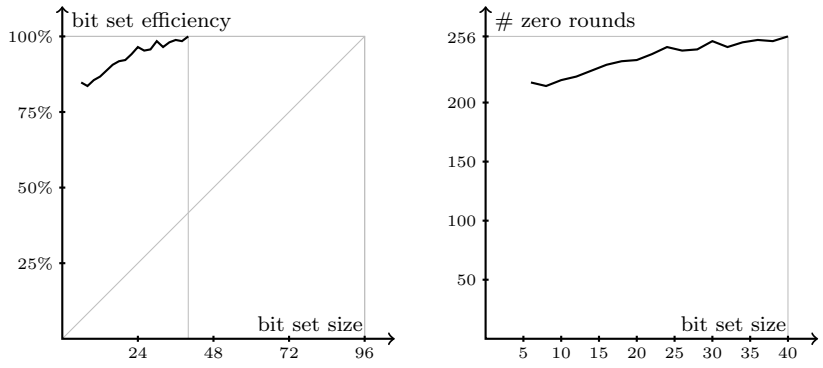

Fig. 6. Insufficient IV bit mixing in full Grain-128 (all 256 rounds).

We now turn our attention from nonrandomness detectors to distinguishers. The best previous distinguisher result on Grain-128 was due to Aumasson et al. [1]. Taking the maximum number of rounds over 64 random key trials, they found a 237-round distinguisher for a bit set of size 40.

Our greedy bit set of size 40 turns out to provide a 246-round distinguisher, measured by taking the maximum zero round count observed over 16 random key trials for a complexity of 2^{42}. The bit set is given below (zero indexed), and the order in which the bits have been added to the set has been preserved. The remaining IV bits were set to zero. The first six bit indices form the optimal 6-set.

34	59	63	64	67	69	55	61	25	85	35	58	2	73	30	38	5	6	10	44
24	50	3	77	91	95	12	13	41	72	19	29	15	79	7	37	21	45	8	71

To summarize the case for Grain-128, we have found one greedy non-randomness detector showing that 256 (out of 256) rounds are insufficient for mixing the IV bits. This detector uses a bit set of size 40 and has complexity 2^{40}.

We also found a greedy 246-round distinguisher with complexity 2^{42}. This distinguisher uses the 38 first bits of the bit set above, taking the maximum zero round count over 16 random keys. The two last bits did not improve the distinguisher.

Trivium There are several interesting observations for Trivium, apart from the importance of key weight that we have already established in Section 3.2. Key and IV bits are equally effective, but allowing both kinds in our bit set will take us much further. To see why this is not a contradiction, have a look at Fig. 7, which depicts the case where we allow only IV bits.

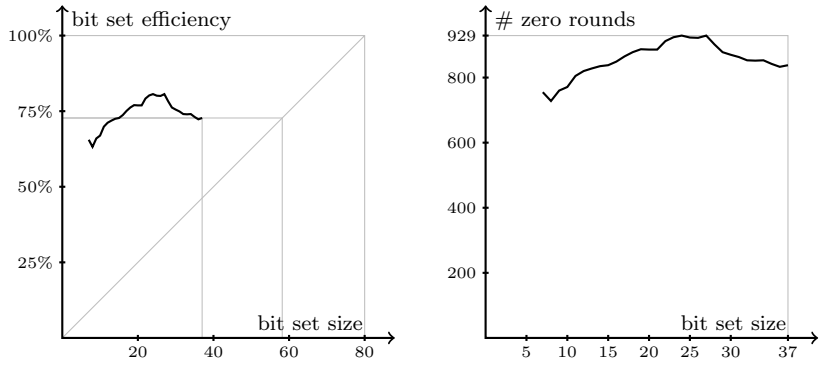

Fig. 7. GreedyBitSet(Add1, Opt7, Trivium, {IV}, **0**, **0**)

This graph is unique in that the curve drops significantly after the bit set has been built to size 27. Using *every third* bit for our bit set turns out to be the most effective choice. This is due to the threefold structure of Trivium, and this is not a new observation (see [21]). It doesn't seem to matter much which third we choose, but once we have started to build up our set we do best if we stick to that implicit third. After 27 bits we run out of bit space, but we can allow both key and IV bits.

Our best greedy nonrandomness detector using both key and IV bits takes us 1026 out of 1152 rounds. This is for the zero key, which we noted before was heavily biased. The greedy strategy was to start from the optimal 5-set and to use the Add2-strategy up to bit set size 29, via the Add1-strategy up to bit set size 37, to finally just guessing the last few bits for a total bit set size of 45. The resulting bit set is

Key bits	1 4 7 10 12 16 19 22 25 31 34 37
	40 43 46 49 52 55 58 61 64 70 73 76
IV bits	1 4 7 10 16 19 25 28 31 34 37 40
	43 46 49 52 55 58 64 67 70

The every-third-structure is evident in this bit set, so it would be interesting to measure the zero round performance of the corresponding 54-bit set with 27 key and 27 IV bits. Considering the bit set performance drop we saw in Fig. 7 above, it is reasonable to assume that we will see the same effect once we try to go beyond this supposedly near-optimal 54-bit set. More than one million core hours of computation on a cluster showed that we get 1078 zero rounds after 2^{54} encryptions.

We also present a distinguisher for 806-round Trivium. As noted before, one can use the internal structure of Trivium by using every third IV bit for the bit set. Unfortunately, we run out of bits after 27 of them have been added. We can, however, skip exploiting the threefold structure and, instead, just use the fact that multiplication is always performed between neighboring state variables. Using *every second* IV bit for the bit set will avoid fast initial term growth and take us 803 rounds over randomly selected keys. This was the *minimum* number of rounds obtained over 16 trials, so the resulting complexity is 2^{40}. Taking the maximum produces an 806-round distinguisher with complexity 2^{44}.

4.2 Significant Susceptibility

Grain v1 is definitely susceptible, as one can see from the direction of the curve in Fig. 8. The level of susceptibility seems limited, however, as the extrapolated greedy-curve will not hit the roof for any bit set of relevant size.

Key bits seem to work a little better than IV bits, in general, but our best nonrandomness result is 96 zero rounds for the all-zeros key with the optimal IV bit set of size 7 given by (zero-indexed)

1 22 26 37 45 47 55

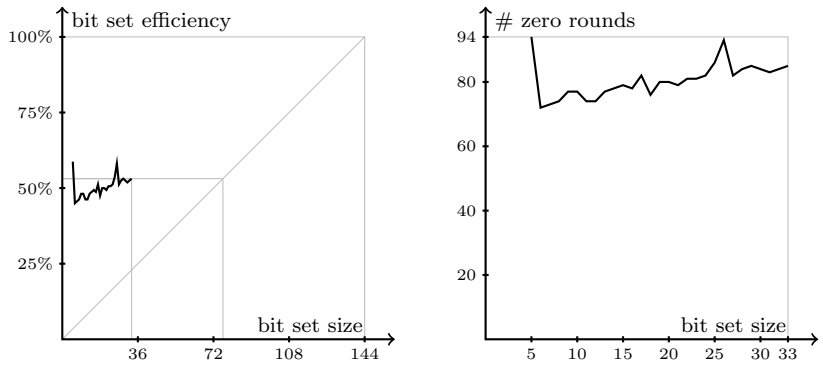

Fig. 8. GreedyBitSet(Add1, Opt5, Grain v1, {Key, IV}, **0**, **0**)

A 90-round greedy distinguisher was derived from a bit set of size 35 by taking the maximum zero round count over 16 random keys for a complexity of 2^{39}. The zero-indexed IV bit set is

1	22	26	37	45	47	55	12	16	4	28	29	36	0	39	31	34	10
11	7	32	9	50	13	25	59	5	3	57	53	51	42	33	38	8	

4.3 Moderate or Low Susceptibility

AES, DES, CLEFIA and HIGHT all start at and stay within a bit set efficiency in the range 25-50%. These algorithms show only very slight or no sign of budging as the bit set size increases.

The remaining ciphers have a bit set efficiency below 25%. Edon80 deviates from the norm by having a somewhat erratic curve, but it seems to stay within the 0-25% efficiency range. Sosemanuk does show a tendency to be affected by the MDM test, but all other algorithms seem to be more or less inherently non-susceptible.

It is interesting to see that the bit set efficiency for IV bits in RC5, and for IV bits in RC6 and key bits in XTEA to a lesser extent, show a *decreasing* tendency as the search progresses and bit set sizes increase. The curve for RC5 IV bits can be seen in Fig. 9.

HC-128 and HC-256 set a record of sorts at the low end by showing no significant susceptibility while producing an extremely large amount of initial data.

The yet unmentioned and remaining algorithms (Camellia, MICKEY v2, PRESENT, Rabbit, Salsa20/12, SEED and SMS4) all seem to be inherently non-susceptible.

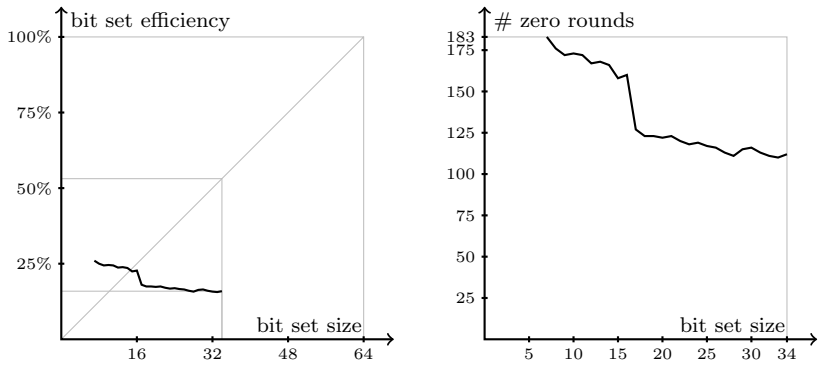

Fig. 9. IV bit sets for RC5 show decreasing efficiency.

5 Results Summary

We have shown how to find efficient bit sets in a systematic and deterministic way by using the Greedy Bit Set Algorithm. The record-breaking distinguishers and nonrandomness detectors derived from using the Greedy Bit Set Algorithm show that this algorithm outperforms all other bit set selection schemes we have seen so far. Table 2 compares the previous best results to ours for Trivium, Grain-128 and Grain v1.

We presented a nonrandomness detector showing that Grain-128 with full 256-round initialization does not behave sufficiently random. This detector uses an IV bit set of size 40 and has a complexity of 2^{40}. We also presented a 246-round distinguisher over random keys with complexity 2^{42}.

For Trivium we found a greedy 1026-round nonrandomness detector with complexity 2^{45}. Using a cluster, we went on to find a nonrandomness detector for 1078 out of 1152 rounds with 2^{54} comlexity. We also presented a 806-round distinguisher with 2^{44} complexity.

For Grain v1 we showed nonrandomness up to 96 rounds with complexity 2^7, and a 90-round distinguisher with complexity 2^{39}.

6 Concluding remarks

With the exception of TEA, all block ciphers we have tested seem reasonably resistant to the maximum degree monomial test. Due to differences in how zero rounds are measured in stream and block ciphers, one should, however, not immediately draw the conclusion that block ciphers are safer than stream ciphers.

Table 2. Comparison to previous results.

Algorithm	Attack type	Rounds	Time	Authors	Rounds	Time	Authors
Trivium	distinguisher	790	2^{30}	[2]	806	2^{44}	this paper
Trivium	nonrandomness	885	2^{27}	[2]	1078	2^{54}	this paper
Grain-128	distinguisher	237	2^{40}	[1]	246	2^{42}	this paper
Grain-128	nonrandomness	-	-	-	256	2^{40}	this paper
Grain v1	distinguisher	81	2^{24}	[1]	90	2^{39}	this paper
Grain v1	nonrandomness	-	-	-	96	2^{7}	this paper

The Greedy Bit Set Algorithm can be examined with more elaborate strategy variants, bit selection schemes, randomness tests, cryptographic algorithms, allowing plaintext bits in the bit set, and so on. The most urgent and constructive goal, however, would be to explain why the MDM test fails miserably for some algorithms. What minimal set of properties is guaranteed to render the MDM test useless?

Let us elaborate on the concept of "weak" bits, see [2, 13]. Weak bits are such that they significantly increase the efficiency (the number of zero rounds) of a bit set if they are added to it. The first question one might ask is: Do weak bits exist at all? The Greedy Bit Set Algorithm answers this question and reveals some deeper insight into the concept of weakness. Our algorithm successively builds larger bit sets by repeatedly adding the weakest remaining single bit (Add1 strategy). For Trivium, bits at every third bit position eagerly reappear among the top ranked bits again and again as the bit set size steadily increases. The bits at other (off-third) positions do not show up as top ranked at all. This zero round distribution regularity is clear evidence that Trivium *has* weak bits. Other algorithms show no sign of weak bits. This does not prove their non-existence in any way, but we surmise that any bit selection strategy for a truly perfect algorithm should not perform much better than random choice. For Grain-128, there are signs of bit weakness, but they are much less conclusive than for Trivium.

The existence of weak bits is algorithm dependent. Also, when we use GreedyBitSet we successively expand a bit set with the *currently* weakest bit. This means that the existence of weak bits does not only depend on the choice of test, but also on the current state of the test. As for drawing conclusions on the existence of globally weak bits, defining how to measure bit weakness is only the first step into a rather non-trivial enterprise.

One consequence of this is that one cannot prove any general performance guarantees for GreedyBitSet stating that we will obtain a good bit with some supposedly high probability. As we have seen, for Trivium we do, for RC5 we don't.

Also, more intelligent analysis of the zero round distribution over the remaining bit space could lead to better practical assessment measures for bit weakness that could be used to improve The Greedy Bit Set Algorithm.

Automatic cryptanalysis can be performed on many cryptographic primitives. A toolbox of various tests, MDM-based and others, should be at the disposal of every algorithm designer. Such a toolbox can be used to reveal unexpected design weaknesses and to give better estimations on the required number of initialization rounds. The interested reader is referred to [29].

We wish to thank the anonymous reviewers for their insightful comments.

References

1. J.-P. Aumasson, I. Dinur, L. Henzen, W. Meier, and A. Shamir. Efficient FPGA Implementations of High-Dimensional Cube Testers on the Stream Cipher Grain-128. Available at *http://eprint.iacr.org/2009/218/*, Accessed June 17, 2009, 2009.
2. J.-P. Aumasson, I. Dinur, W. Meier, and A. Shamir. Cube Testers and Key Recovery Attacks on Reduced-Round MD6 and Trivium. In O. Dunkelman, editor, *Fast Software Encryption 2009*, volume 5665 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2009.
3. S. Babbage and M. Dodd. The MICKEY Stream Ciphers. In *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 191–209. Springer-Verlag, 2008.
4. C. Berbain, O. Billet, A. Canteaut, N. Courtois, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert. Sosemanuk, a Fast Software-Oriented Stream Cipher. In *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 98–118. Springer-Verlag, 2008.
5. D. J. Bernstein. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer-Verlag, 2008.
6. M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius. Rabbit: A New High-Performance Stream Cipher. In *Fast Software Encryption 2003*, volume 2887 of *Lecture Notes in Computer Science*, pages 307–329. Springer-Verlag, 2003.
7. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe. PRESENT: An Ultra-Lightweight Block Cipher. In *Cryptographic Hardware and Embedded Systems—CHES 2007*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer-Verlag, 2007.

8. C. De Cannière and B. Preneel. Trivium. In M. Robshaw and O. Billet, editors, *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 244–266. Springer-Verlag, 2008.
9. W. Diffie and G. Ledin. SMS4 Encryption Algorithm for Wireless Networks. Available at *http://eprint.iacr.org/2008/329.pdf*, Version 1.03, Accessed Feb 18, 2010, 2008.
10. I. Dinur and A. Shamir. Cube Attacks on Tweakable Black Box Polynomials. In A. Joux, editor, *Advances in Cryptology—EUROCRYPT 2009*, volume 5479 of *Lecture Notes in Computer Science*, pages 278–299. Springer-Verlag, 2009.
11. H. Englund, T. Johansson, and M. S. Turan. A framework for chosen IV statistical analysis of stream ciphers. In K. Srinathan, C. Pandu Rangan, and M. Yung, editors, *Progress in Cryptology - INDOCRYPT 2007*, volume 4859/2007 of *Lecture Notes in Computer Science*, pages 268–281. Springer-Verlag, 2007.
12. E. Filiol. A new statistical testing for symmetric ciphers and hash functions. In R. Deng, F. Bao, J. Zhou, and S. Qing, editors, *ICICS—2002*, volume 2513/2002 of *Lecture Notes in Computer Science*, pages 342–353. Springer-Verlag, 2002.
13. S. Fischer, S. Khazaei, and W. Meier. Chosen IV Statistical Analysis for Key Recovery Attacks on Stream Ciphers. In S. Vaudenay, editor, *Progress in Cryptology—AFRICACRYPT 2008*, volume 5023 of *Lecture Notes in Computer Science*, pages 236–245. Springer-Verlag, 2008.
14. D. Gligoroski, S. Markovski, and S. J. Knapskog. The Stream Cipher Edon80. In *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 152–169. Springer-Verlag, 2008.
15. M. Hell, T. Johansson, A. Maximov, and W. Meier. A Stream Cipher Proposal: Grain-128. In *International Symposium on Information Theory—ISIT 2006*. IEEE, 2006.
16. M. Hell, T. Johansson, and W. Meier. Grain - a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing, Special Issue on Security of Computer Network and Mobile Systems.*, 2(1):86–93, 2006.
17. D. Hong, J. Sung, S. Hong, J. Lim, S. Lee, B.-S. Koo, C. Lee, D. Chang, J. Lee, K. Jeong, H. Kim, J. Kim, and S. Chee. HIGHT: A New Block Cipher Suitable for Low-Resource Device. In *Cryptographic Hardware and Embedded Systems—CHES 2006*, volume 4249 of *Lecture Notes in Computer Science*, pages 46–59. Springer-Verlag, 2006.
18. J. Kelsey, B. Schneier, and D. Wagner. Related-key cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, NewDES, RC2, and TEA. In *Information and Communications Security, First International Conference, ICIS '97 Beijing, China, November 1114, 1997 Proceedings*, volume 1334 of *Lecture Notes in Computer Science*, pages 233–246. Springer-Verlag, 1997.
19. L. R. Knudsen and V. Rijmen. Known-Key Distinguishers for Some Block Ciphers. In *AC07*, volume 4833/2008 of *Lecture Notes in Computer Science*, pages 315–324. Springer-Verlag, 2007.
20. H. J. Lee, S. J. Lee, J. H. Yoon, D. H. Cheon, and J. I. Lee. The SEED Encryption Algorithm. Available at *http://tools.ietf.org/html/rfc4269*, Accessed Feb 18, 2010, 2005.
21. A. Maximov and A. Biryukov. Two Trivial Attacks on Trivium. The State of the Art of Stream Ciphers, Workshop Record, SASC 2007, Bochum, Germany, January 2007.
22. NTT and Mitsubishi Electric Company. Specification of Camellia - A 128-bit Block Cipher. Available at *http://info.isl.ntt.co.jp/crypt/eng/camellia/dl/01espec.pdf*, Version 2.0, Accessed Feb 18, 2010, 2000.

23. U.S. Department of Commerce and NIST. DATA ENCRYPTION STANDARD (DES). FIPS Publication 46-3, 1999.
24. U.S. Department of Commerce and NIST. Announcing the ADVANCED ENCRYPTION STANDARD (AES). FIPS Publication 197, 2001.
25. R. L. Rivest. The RC5 encryption algorithm. In *Fast Software Encryption'95*, volume 1008 of *Lecture Notes in Computer Science*, pages 86–96. Springer-Verlag, 1995.
26. R. L. Rivest, M. J. B. Robshaw, R. Sidney, and Y. L. Yin. The RC6 Block Cipher. Available at *http://people.csail.mit.edu/rivest/Rc6.pdf*, Version 1.1, Accessed Feb 18, 2010, 1998.
27. M.-J. O. Saarinen. Chosen-IV statistical attacks on eSTREAM stream ciphers. eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, 2006. *http://www.ecrypt.eu.org/stream*.
28. T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata. The 128-Bit Blockcipher CLEFIA (Extended Abstract). In *Fast Software Encryption 2007*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer-Verlag, 2007.
29. P. Stankovski. Maximum Degree Monomial Toolkit with Source Code. Available at *http://www.eit.lth.se/staff/paul.stankovski/phdprojects*, Accessed September 24, 2010, 2010.
30. M. Vielhaber. Breaking ONE.FIVIUM by AIDA an Algebraic IV Differential attack. Available at *http://eprint.iacr.org/2007/413/*, Accessed June 17, 2009, 2007.
31. D. J. Wheeler and R. M. Needham. TEA, a Tiny Encryption Algorithm. Available at *http://www.cix.co.uk/~klockstone/tea.pdf*, Accessed Feb 18, 2010.
32. D. J. Wheeler and R. M. Needham. TEA extensions. Available at *http://www.cix.co.uk/~klockstone/xtea.pdf*, Accessed Feb 18, 2010.
33. H. Wu. A New Stream Cipher HC-256. In *Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 226–244. Springer-Verlag, 2004.
34. H. Wu. The Stream Cipher HC-128. In *New Stream Cipher Designs*, volume 4986 of *Lecture Notes in Computer Science*, pages 39–47. Springer-Verlag, 2008.