Chosen-IV Statistical Attacks on eSTREAM Stream Ciphers

Markku-Juhani O. Saarinen

Information Security Group
Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK.
m.saarinen@rhul.ac.uk

Abstract. d-Monomial tests are statistical randomness tests based on Algebraic Normal Form representation of a Boolean function, and were first introduced by Filiol in 2002. We show that there are strong indications that the Gate Complexity of a Boolean function is related to a bias detectable in a d-Monomial test. We then discuss how to effectively apply d-Monomial tests in chosen-IV attacks against stream ciphers. Finally we present results of tests performed on eSTREAM proposals, and show that six of these new ciphers can be broken using the d-Monomial test in a chosen-IV attack. Many ciphers even fail a trivial (ANF) bit-flipping test.
Earlier version of this paper appeared in SASC 2006 workshop record under the title "d-Monomial Tests are Effective Against Stream Ciphers".
Keywords: Stream Ciphers, eSTREAM, Algebraic Normal Form, Möbius test, d-monomial test.

1 Introduction

Statistical testing has traditionally been a part of evaluation of stream ciphers. However, most cryptographers agree that generic tests such as the NIST 800-22 suite are appropriate mainly for catching implementation errors rather than determining the cryptographic strength of an algorithm [4, 5].

Usually these tests have been performed in a passive setting; a sequence of bits is generated under a (random) key, and these bits are then subjected to a generic statistical test. What is ignored in this approach is that stream ciphers equipped with an Initialization Vector (IV) should also be able to withstand chosen-IV attacks, where a sequence of data is generated by varying the IV value rather than the "counter" value (see Figure 1).

Stream ciphers are optimized for security, but also for speed and cost. Cost in many applications equates to the number of logical gates in a hardware implementation of the cipher, and hence designers usually attempt to minimize their gate complexity.

Most stream ciphers can be specified as a relatively simple iterated function. As a result of this, it has been observed that some keystream bits can be expressed as simple Boolean functions of the key and IV bits. In a chosen-IV attack, the key bits remain constant and the stream cipher can be viewed as a "black box" Boolean function of the IV alone.

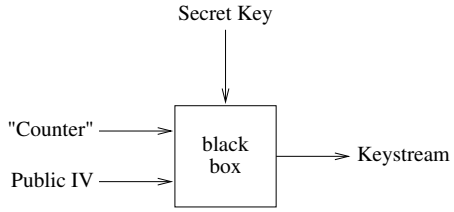

Fig. 1. A stream cipher can be seen as a black box Boolean function that takes in a secret key, a public IV, and a public "counter" to produce a single bit of keystream.

In a chosen-IV distinguishing attack, an attacker would wish to be able to determine whether or not a keystream bit (say, the first one after IV setup) is a simple Boolean function of some IV bits simply by making queries to this black box.

How would one automatically distinguish such a Boolean function of n bits from a random one? One solution is to examine its Algebraic Normal Form (ANF) representation for anomalies such as redundancy or bias. A test that utilizes this approach was first proposed by Eric Filiol in 2002 [2]. In this paper we will give further theoretical and experimental evidence of the applicability of ANF-based tests on stream ciphers.

The structure of this paper is as follows. In Section 2 we recall the Algebraic Normal Form and its basic properties. Section 3 contains an exposition of a variant of Filiol's d-monomial statistical test. Section 4 gives new, clear evidence of the relationship between Boolean gate complexity and the d-monomial test. Section 5 discusses a simple statistical attack based on flipping input bits that was found to be surprisingly effective against eSTREAM ciphers [3]. Section 6 contains new results on statistical tests on the 34 eSTREAM cipher proposals, followed by conclusions in Section 7.

2 Preliminaries

Let $\mathbb{F}_2^n$ be the vector space defined by n-vectors $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, where $x_i \in \mathbb{F}_2$, i.e. each of the n elements has either value 0 or 1 and computations are defined modulo 2. A Boolean function f of n variables is simply a mapping $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$. There are exactly 2^{2^n} distinct Boolean functions of n variables, each uniquely defined by its truth table.

There are many alternative representations for Boolean functions, such as Conjunctive and Disjunctive Normal Forms (CNF and DNF), which are widely used in automated theorem proving and other fields of theoretical computer science. We will focus on Algebraic Normal Form (ANF, also known as Ring Sum Expansion, or RSE [6]). [1]

Definition 1. *A function* $\hat{f} : \mathbb{F}_2^n \mapsto \mathbb{F}_2$ *satisfying*

$$\hat{f}(\mathbf{x}) = \sum_{\mathbf{a} \in \mathbb{F}_2^n} f(\mathbf{a}) \prod_{i=1}^{n} x_i^{a_i}$$

[1] This transform is sometimes confusingly called the Möbius transform [2], hence the name, "Möbius test" in Filiol's original paper.

is an Algebraic Normal Form representation of a Boolean function $f : \mathbb{F}_2^n \mapsto \mathbb{F}_2$.

Using transformed function $\hat{f}$, a multivariate polynomial representation of f can be obtained as can be seen from the following example (or directly from the definition).

Example 1. Consider the Boolean function $f : \mathbb{F}_2^3 \mapsto \mathbb{F}_2$ defined by the following table:

$$f(0,0,0) = 1,\ f(1,0,0) = 0,\ f(0,1,0) = 1,\ f(1,1,0) = 0,$$
$$f(0,0,1) = 1,\ f(1,0,1) = 1,\ f(0,1,1) = 0,\ f(1,1,1) = 1.$$

As indicated by Definition 1, we wish to find a $\hat{f}$ that for all $\mathbf{x}$ satisfies

$$f(x_1,x_2,x_3) = \hat{f}(0,0,0) + \hat{f}(1,0,0)x_1 + \hat{f}(0,1,0)x_2 + \hat{f}(1,1,0)x_1x_2 +$$
$$\hat{f}(0,0,1)x_3 + \hat{f}(1,0,1)x_1x_3 + \hat{f}(0,1,1)x_2x_3 + \hat{f}(1,1,1)x_1x_2x_3.$$

this corresponds to solving the following system of linear equations in $\mathbb{F}_2$:

$$\begin{pmatrix} 1&0&0&0&0&0&0&0\\ 1&1&0&0&0&0&0&0\\ 1&0&1&0&0&0&0&0\\ 1&1&1&1&0&0&0&0\\ 1&0&0&0&1&0&0&0\\ 1&1&0&0&1&1&0&0\\ 1&0&1&0&1&0&1&0\\ 1&1&1&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} \hat{f}(0,0,0)\\ \hat{f}(1,0,0)\\ \hat{f}(0,1,0)\\ \hat{f}(1,1,0)\\ \hat{f}(0,0,1)\\ \hat{f}(1,0,1)\\ \hat{f}(0,1,1)\\ \hat{f}(1,1,1) \end{pmatrix} = \begin{pmatrix} f(0,0,0) = 1\\ f(1,0,0) = 0\\ f(0,1,0) = 1\\ f(1,1,0) = 0\\ f(0,0,1) = 1\\ f(1,0,1) = 1\\ f(0,1,1) = 0\\ f(1,1,1) = 1 \end{pmatrix}.$$

The solution to this matrix equation is obtained easily with Gaussian elimination:

$$\hat{f}(0,0,0) = 1,\ \hat{f}(1,0,0) = 1,\ \hat{f}(0,1,0) = 0,\ \hat{f}(1,1,0) = 0,$$
$$\hat{f}(0,0,1) = 0,\ \hat{f}(1,0,1) = 1,\ \hat{f}(0,1,1) = 1,\ \hat{f}(1,1,1) = 1.$$

The ones in $\hat{f}$ directly give the five monomials in the polynomial expression for f:

$$f(x_1,x_2,x_3) = 1 + x_1 + x_1x_3 + x_2x_3 + x_1x_2x_3.$$

2.1 Properties of the Algebraic Normal Form

We briefly summarize some of the most important properties and concepts (facts) of ANF that are relevant to the present discussion:

F.1 A unique $\hat{f}$ exists for all Boolean functions f.
F.2 The ANF transform is its own inverse, an involution; iff $g = \hat{f}$, then $\hat{g} = f$.
F.3 We define a *partial order* for vectors $\mathbf{x}$ as follows: $\mathbf{x} \leq \mathbf{y}$ iff $x_i \leq y_i$ for all i. Using the partial order, Definition 1 can be written as $\hat{f}(\mathbf{x}) = \sum_{\mathbf{a} \leq \mathbf{x}} f(\mathbf{a})$.
F.4 The *Hamming distance* $\mathsf{d}(\mathbf{x},\mathbf{y})$ between $\mathbf{x}$ and $\mathbf{y}$ is the number of positions where $x_i \neq y_i$.
F.5 A norm, called the *Hamming weight*, $\mathsf{wt}(\mathbf{x}) = \mathsf{d}(\mathbf{0},\mathbf{x})$, is equivalent to number of positions in $\mathbf{x}$ where $x_i = 1$.

F.6 The *algebraic degree* $\deg(f)$ is the maximum Hamming weight $\mathbf{x}$ that satisfies $\hat{f}(\mathbf{x}) = 1$; this is equivalent to the length of the longest monomial (most variables) in the polynomial representation of f.

F.7 Functions of degree one are *affine functions*. If the constant term $\hat{f}(0, 0, \ldots, 0) = 0$, an affine function is simply a sum of some of its input bits and called a *linear function*.

F.8 A *d-Truncated Algebraic Normal Form* of Boolean function f, denoted $\hat{f}_d(\mathbf{x})$, is equal to $\hat{f}(\mathbf{x})$ when $\mathsf{wt}(\mathbf{x}) \leq d$, and zero otherwise. In essence, monomials of degree greater than d have been removed from the corresponding polynomial of the truncated ANF.

F.9 Since $\hat{f}(\mathbf{x})$ is the sum of f at all positions with smaller or equal partial order (and hence degree) than $\mathbf{x}$ (F.3), it can be seen that if we have tabulated $f(\mathbf{y})$ at all positions $\mathbf{y}$ with $\mathsf{wt}(\mathbf{y}) \leq d$, the d-truncated ANF can be completely determined.

2.2 Computing the ANF

Networks and algorithms for computing the complete ANF do not require more than $n2^{n-1}$ additions in $\mathbb{F}_2$.

Let $z : \mathbb{F}_2^n \mapsto \mathbb{Z}$ be the standard mapping from binary vectors to integers; $z(\mathbf{x}) = \sum_{i=1}^{n} 2^{i-1} x_i$. Let v be a binary-valued vector of length 2^n that contains the truth table of f; $v_{z(\mathbf{x})+1} = f(\mathbf{x})$ for all $\mathbf{x}$. Algorithm 1 gives a fast method for computing $\hat{f}$.

Algorithm 1 Compute the Algebraic Normal Form in vector v of length 2^n using two auxiliary vectors t and u of length 2^{n-1}.

```
for j = 1, 2, 3, ..., n do
  for i = 1, 2, ..., 2^(n-1) do
    t_i ← v_(2i-1)
    u_i ← v_(2i-1) ⊕ v_(2i)
  end for
  v ← t || u
end for
```

The complexity of Algorithm 1 is clearly $O(n \lg n)$. Variants of this algorithm can be implemented very efficiently using shifts and bit-manipulation operations.

3 The d-Monomial Tests

In [2] Filiol introduced "Möbius tests", which examine whether or not an ANF expression of a Boolean function has the expected number of d-degree monomials. With $d = 0$ the test is called the *Affine test* and for $d > 0$ a *d-Monomial test*.

Please note that the following exposition of the test / distinguisher is significantly simpler and less formal than that originally proposed by Filiol. Details have been modified for the purposes of this paper. The reader is encouraged to use [2] as a reference for Filiol's version of the test.

In practical terms the d-Monomial test involves counting the number of ones $\hat{f}(\mathbf{x}) = 1$ of an ANF transformed function f at positions $\mathbf{x}$ with Hamming weight d. A d-truncated ANF is is sufficient for this purpose. A χ^2 statistical test is then applied to this count to see if the count is exceptionally high or low.

Theorem 1. *For a randomly chosen n-bit Boolean function f, $\mathsf{Pr}[\hat{f}(\mathbf{x}) = 1] = 1/2$ for all $\mathbf{x}$.*

Proof. Trivial. Since the ANF transformation is bijective on the truth table of f, $\hat{f}$ will be random if f is.

Consider an n - bit Boolean function f. Our null hypothesis is that the expected bitcount $\sum_{\mathsf{wt}(\mathbf{x})=d} \hat{f}(\mathbf{x})$ is $\frac{1}{2}\binom{n}{d}$ and the bitcount is binomially distributed. The alternative hypothesis is that there is a bias in this sum, up or down.

We can use Pearson's classic χ^2 test in this case. Suppose that we sample $\hat{f}$ at N distinct points (in this case with $\mathsf{wt}(\mathbf{x}) = d$) and in M of those $\hat{f}(\mathbf{x}) = 1$. Then we set

$$\chi^2 = \frac{1}{N}\left(2M - N\right)^2.$$

Since "0" and "1" cases in bitcount are mutually exclusive, there is one degree of freedom in the test. Using the cumulative degree-one distribution function of χ^2, we can determine a confidence level for f being distinguishable from random in our test. We call this the P value and its intuitive interpretation is the "probability that the null hypothesis is true". For example, if P is 0.01, there's still a 1% probability that the null hypothesis is true (and the function is, in this sense, "random").

Some "upper critical" values for χ^2 and the corresponding P values are given in the following table:

χ^2	P
6.635	0.01
10.83	0.001
18.70	2^{-16}
40.17	2^{-32}
24.02	2^{-40}
83.82	2^{-64}
105.8	2^{-80}

This type of test is dependent upon the sample size; even a very slightly biased function will yield a high χ^2 value by the test if the sample size is allowed to be arbitrarily large. The sample sizes are bound by computational restrictions, however. A distinguishing attack is not relevant unless its total expected computational complexity is smaller than the claimed security level of the cipher (typically equivalent to 2^{k-1} key trials, where k is the size of the secret key).

4 Gate Complexity and the d-Monomial Test

In this section we will give a formal definition for *gate complexity* and investigate its relationship with the d-Monomial test. Gate complexity is essentially equivalent to circuit complexity with realistic limitations [1, 6].

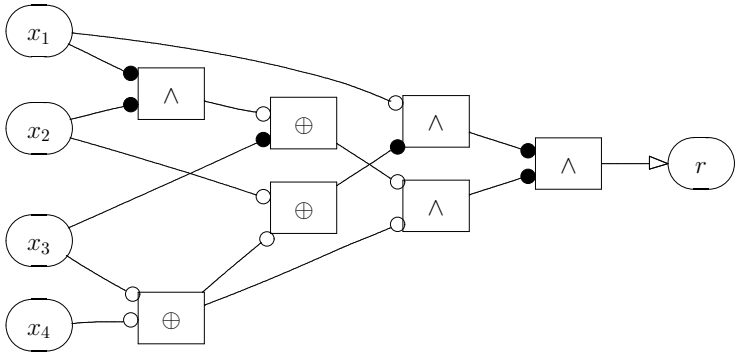

Fig. 2. An automatically generated picture of a Boolean function with gate complexity 7. In this picture a filled circle indicates that the given input is inverted. This function can not be implemented with, say, six gates (regardless of the choice of gates).

Definition 2. *Gate complexity of a Boolean function $f(x_1, x_2, \ldots, x_n)$ is the minimum number of gates required to implement it in an acyclic circuit network. A gate is a Boolean function with two inputs. The constant functions* 0 *and* 1*, together with trivial functions $x_1, x_2, \ldots$ have gate complexity 0.*

Note that all $2^{2^2} = 16$ two-bit functions count as a single gate, not just the standard ones ($\vee$, $\wedge$, $\neg$, $\oplus$).

We have determined the gate complexity of all $2^{2^4} = 65536$ four-bit Boolean functions. This was done by performing an exhaustive search over all circuits with one gate, two gates, etc, until circuits for all functions had been found. The task was computationally nontrivial, even though we optimized the code to take various symmetries and isometries into account. The maximum gate complexity turned out to be 7 (see Figure 2).

Table 1 gives the distribution of functions by gate complexity. In it, G_i is the number of functions of gate complexity i. These sum to $\sum_i G_i = 65536$. Here $g_{i,d}$ is the number of monomials of degree d and gate complexity i. These sum to $\sum_d g_{i,d} = G_i$. The maximum possible value for $g_{i,d}$ is $G_i\binom{4}{d}$. The expected number in a d-monomial test is half of this value. The table contains the "bias" fraction $q_{i,d} = g_{i,d}/(G_i\binom{4}{d})$.

Note how in Table 1 the d-Monomial "bias" $q_{i,d}$ tends to be strongly increasing as the gate complexity i grows (apart for anomaly at $q_{6,4}$). This is clear evidence of a correlation between the complexity of a Boolean circuit and the d-monomial test. It is plausible to expect that a similar phenomenon is exhibited by Boolean functions with $5, 6, \ldots$ inputs. However, the exact degree of this bias is currently an open problem for $n > 4$. We can expect simple functions to be distinguishable in a d-monomial test even when n is large.

It is interesting to note that it is even possible to test the opposite; to distinguish a *complex* function from a randomly chosen one, as the following example illustrates.

		$d=0$		$d=1$		$d=2$		$d=3$		$d=4$	
i	G_i	$g_{i,0}$	$q_{i,0}$	$g_{i,1}$	$q_{i,1}$	$g_{i,2}$	$q_{i,2}$	$g_{i,3}$	$q_{i,3}$	$g_{i,4}$	$q_{i,4}$
0	6	1	0.167	4	0.167	0	0.000	0	0.000	0	0.000
1	64	34	0.531	76	0.297	48	0.125	0	0.000	0	0.000
2	456	228	0.500	648	0.355	672	0.246	256	0.140	0	0.000
3	2474	1237	0.500	3912	0.395	5136	0.346	3264	0.330	832	0.336
4	10624	5312	0.500	18960	0.446	26976	0.423	17536	0.413	4608	0.434
5	24184	12092	0.500	47888	0.495	71328	0.492	47616	0.492	13216	0.546
6	25008	12504	0.500	52992	0.530	83232	0.555	55744	0.557	12576	0.503
7	2720	1360	0.500	6592	0.606	9216	0.565	6656	0.612	1536	0.565

Table 1. Distribution of the 65536 four-bit Boolean functions by gate complexity and the results of d-monomial tests on Boolean functions of given gate complexity.

Example 2. With the 2720 functions of gate complexity 7, all d-Monomial counts appear to be biased *upwards*; $q_{7,d} \geq 0.5$. We will use a d-Monomial test to create a distinguisher based on this fact, particularly that $q_{7,1} = 0.606$.

Consider the following game. There is a list L containing binary vectors of length 5. Entries in L are may have been generated with one of the following two methods:

1. Choose a random 4-bit Boolean function of gate complexity 7 for each entry, and add the following vector to the list

$$(f(0,0,0,0), f(1,0,0,0), f(0,1,0,0), f(0,0,1,0), f(0,0,0,1)).$$

2. Choose a completely random Boolean function (one of the 65536 possibilities) and create a vector in similar fashion.

We pose the following question: How long does L need to be for us to see which type of list it is ?

We first note that the vectors contain sufficient information for computation of 1-Monomial test (e.g. $\hat{f}(1,0,0,0) = f(0,0,0,0) + f(1,0,0,0)$). Each 1-Monomial test is simply the sum of 4 bits in the ANF result. The expected sum after n list entries is $2n$ for a random function and based on our exhaustive search, $g_{7,1}n/G_7 = 6592/2720n \approx 2.424n$ for a gate complexity 7 function. Our distinguisher will simply return "a" if the sum is greater than $2n$ and "b" otherwise.

In the second, fully random case, the distinguisher has no advantage as the bits in the vector are random too; "a" and "b" will both be returned with probability $1/2$ regadless of the length of L.

In case 1, after $n = 34$ steps, the sum can be expected to reach $2.424*34 = 82.4$. "a" will be returned by the distinguisher with probability 99%. Hence we can distinguish the list of (partially computed and randomly chosen) "complex" functions with significant certainty with a list of only 34 entries! Note that the probability here was computed exactly using binomial sums, rather than using the χ^2 test.

5 The (ANF) Bit-Flip Test

The bit-flip test is a simple statistical test that measures the effect of flipping one of the input bits on a Boolean function. The test can be performed either on the function f itself or its ANF counterpart $\hat{f}$.

The same "bit-counting" χ^2 test with one degree of freedom can be applied as in d-Monomial test (Section 3).

Given a vector with $\mathbf{b}$ with $\mathrm{wt}(\mathbf{b}) = 1$, we sample $f(\mathbf{x})$ (or $\hat{f}(\mathbf{x})$) at N distinct points with $x_i = 0$ and count the number of occurrences M where $f(\mathbf{x}) = f(\mathbf{x} + \mathbf{b})$ (or, respectively, $\hat{f}(\mathbf{x}) = \hat{f}(\mathbf{x} + \mathbf{b})$). The statistic is again

$$\chi^2 = \frac{1}{N} (2M - N)^2$$

and the confidence level P is computed in the same fashion as with d-monomial test.

This simple test is useful for measuring the basic mixing properties of the function and was therefore employed in our tests of eSTREAM proposals as discussed in the following section.

6 Chosen-IV Tests on eSTREAM Proposals

As there were as many as 34 proposals for eSTREAM [3], some with poor documentation, we decided to make certain assumptions about their structure in order to facilitate "automatic" d-Monomial and bit-flipping testing.

1. We wish to find a subset of input bits that is likely to receive less mixing during the IV setup process than other bits. This is likely to be either at the beginning or the end of the IV bit-vector.
2. After the bits for a d-Monomial test have been chosen, the remaining constant IV bits also greatly affect the probability that the keystream will exhibit bias. We chose to run the tests with these bits set as 0 and also when they are set to 1.
3. Rather than running the test on some low-degree limit d (In [2] $d \leq 3$ and $d \leq 5$ are mentioned), we limit the number of bits n to some manageable number and compute all d-Monomial tests on those bits.

There are four d-Monomial tests in total; {bits in beginning, bits in the end} $\times$ {rest of bits set to 0, rest of bits set to 1}. In practice the black box function (IV setup) was run with increasing values of n until a time or memory limit was exceeded. An ANF was then computed and monomials of various degrees counted. The same data was also subjected to bit-flipping tests as described in Section 5.

The testing code was integrated into the "eSTREAM speed testing framework", which allowed the test to be easily run on most eSTREAM ciphers. The test code simply utilizes the eSTREAM API and treats each cipher as a black box function.

There appears to be bugs in some cipher implementations, that resulted in exceedingly high biases. Those cases are ignored in the discussion below. We only mention ciphers where definitive evidence of statistical anomaly was detected (positive results

are not reported). All tests were run at least 10 times with randomized keys. We only report anomalies that reoccurred in a consistent pattern in distinct tests. Note that when the same tests were run on reference ciphers such as AES-CTR, no anomalies were found.

All specifications of the ciphers are available from the eSTREAM web site [3]. The following list of results is not exhaustive, but just relates to the current status of the tests.

6.1 MAG, Frogbit, and F-FCSR

MAG is a stream cipher designed by Rade Vuckovac that uses a 128-bit key and a 32-bit IV. Frogbit is a "cipher, data integrity algorithm" designed by Thierry Moreau with 128-bit key and IV values. F-FCSR is a family of stream ciphers designed by Thierry Berger, François Arnault and Cédric Lauradoux.

These ciphers exhibited extreme biases. In some cases flipping a particular bit in IV did not affect the first keystream bits at all. The designers of these ciphers appear to have failed to consider the implications of chosen-IV attacks.

6.2 DECIM

Decim is a stream cipher with a 80-bit key and a 64-bit IV designed by Come Berbain et al. Decim is highly vulnerable to d-Monomial distinguishers. Biases that occur with $P < 2^{-96}$ (our implementation precision limit) were consistently found. Decim also appears to be susceptible to a bit-flipping attack, although to a lesser degree. In a typical run of 2^{18} IV setups, a bit-flipping bias with $P < 2^{-16}$ could be found.

6.3 ZK-Crypt

ZK-Crypt is a stream cipher designed by Carmi Gressel, Ran Granot and Gabi Vago. With a 128-bit key and a 128-bit IV it is highly vulnerable to both bit-flipping and d-Monomial distinguishers. Biases with $P < 2^{-96}$ were consistently found in bit-flipping attacks. In d-Monomial attacks the bias was in $P < 2^{-12}$ range, although in one case $P < 2^{-37}$ was observed. A typical test run would involve 2^{21} IV setups.

6.4 POMARANCH

POMARANCH is a stream cipher designed by Cees Jansen and Alexander Kolosha. With a 128-bit key and a 112-bit IV it is susceptible to bit-flipping tests when the flipping occurs at the end of the IV vector. Biases with $P < 2^{-96}$ were consistently observed in such attacks. Typical run would involve 2^{17} IV setups.

6.5 NLS and TSC-3

NLS is a stream cipher designed by Gregory Rose, Philip Hawkes, Michael Paddon and Miriam Wiggers de Vries. TSC-3 is a stream cipher proposed by Jin Hong, Dong Hoon Lee, Yongjin Yeom, Daewan Han and Seongtaek Chee.

These ciphers fall into "borderline category". Some strong biases were found, but not strong enough to indicate a clear design flaw. We suspect that improved attacks are possible by hand-crafting the test parameters to exploit particular features of the design of these ciphers.

In NLS with a 128-bit key and a 128-bit IV, a bias with $P < 2^{-20}$ was observed in one d-Monomial test run of 2^{24} IV setups. Multiple lesser d-Monomial biases occur in a consistent pattern.

In TSC-3 with a 160-bit key and a 128-bit IV, a bit flipping bias with $P < 2^{-18}$ was observed and lesser biases occur in a consistent pattern.

7 Conclusion

We have discussed the application of Algebraic Normal Form and d-Monomial tests to chosen-IV attacks against stream ciphers. It has been demonstrated that these tests appear to be highly effective in distinguishing "simple" Boolean functions as well as (rather surprisingly) complex functions from random ones.

In an experiment with eSTREAM stream ciphers, we found that the output of six of the 34 candidates could be distinguished from random with our methods, with additional few being borderline cases and requiring further investigation. Ciphers with poor mixing properties even fail a simple bit-flipping test (or its ANF variant).

8 Acknowledgments

The author wishes to thank Keith Martin for his valuable comments. This research was supported by a grant from *Helsingin Sanomain 100-Vuotissäätiö*.

References

1. Clote, P., Kranakis, E.: Boolean Functions and Computation Models. Springer-Verlag, 2002
2. Filiol, E.: A New Statistical Testing for Symmetric Ciphers and Hash Functions. Proc. ICICS 2002, LNCS 2513, Springer-Verlag 2002. pp. 342 – 353.
3. ECRYPT: The home page eSTREAM, the ECRYPT Stream Cipher Project. `http://www.ecrypt.eu.org/stream/`
4. Murphy, S.: The Power of NIST's Statistical Testing of AES Candidates. AES Comment to NIST, April 2000.
5. Rukhin, A. et al.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22 (revised May 15, 2001)
6. Wegener, I.: The complexity of Boolean functions. Wiley-Teubner series in computer science. Wiley, Teubner, 1987