ॐ भूर्भुवस्व तत् सवितूर्वरेण्यम् भर्गो देवस्य धीमहि धियो यो न प्रचोदयात्

Algebraic Structure Defectoscopy (ASD Tests)

Sean O'Neil
VEST Corporation S.A.R.L.
http://defectoscopy.com

Abstract: We present a novel instrument of automated cryptanalysis suitable for measuring the number of rounds that can build one PRF round, so that 4 such rounds could be recommended as a Luby-Rackoff cipher secure against adaptive attacks. ASD tests can detect structural flaws in all kinds of cryptographic primitives and their implementations. We present our results for some of the well-known ciphers and hash functions and for some of the eSTREAM candidates. Our tools can distinguish complete Achterbahn, Grain v1 and Grain-128 from random, detect weak keys in the complete IDEA cipher and find fatal structural flaws even in complete ciphers like LILI, KeeLoq or TEA in a matter of seconds. Cryptanalysts can save their valuable time by requiring that all new ciphers must pass not only randomness tests, but also automated cryptanalysis tests like ours before they could be considered interesting for manual cryptanalytic study.

1. Introduction

First, the term 'defectoscopy' means non-destructive analysis. It is the Latin 'defectus' (failure, lack, deficit) combined with the Greek '–scopy' (viewing). The term 'defectoscopy' first appears in a 1948 work of A.K. Trapeznikov, the Russian pioneer of non-destructive analysis by X-rays. X-rays and their medical use for non-destructive analysis of the body were first discovered by the German physicist Wilhelm Conrad Röntgen in 1895. 'Defectoscopy' is also the name of the annual international exhibition of non-destructive analysis equipment.

1.1 Background

There are numerous references in the cryptographic papers to the famous Luby-Rackoff[1] (LR) theorem recommending for block ciphers 4 rounds of a mysterious round function that can be proven to be a PRF. However, the separation between these theoretical results and the real world ciphers led to the division of block cipher designers and analysts into three main groups: 1) those who mistakenly* see a single Feistel round as one LR round and do not even find it surprising that much more than 4 rounds are required for Feistel ciphers to be secure, 2) those who mistakenly† believe that PRF means indistinguishable from random at any cost and thus recommend to use an excessive 4 times the number of rounds that cannot be broken by any known attacks, and 3) those who believe that these results are purely theoretical and cannot be applied to the real world ciphers.

We know that LR cipher combining three independent LR rounds is a construction secure[1] against non-adaptive attacks. LR cipher combining four independent LR rounds is a construction secure[1] against adaptive attacks. If we can find out when/whether a real world cipher succeeds at building a PRF, we can have a practical application of Luby-Rackoff and subsequent theoretical work to the actual real world designs, which could allow us to construct provably secure and practical ciphers.

So what is an LR round? – Our ASD tests measure the number of rounds r_{LR}, after which even the worst-case-scenario change in the key/IV/plaintext/ciphertext/counter/state would make the combined round functions indistinguishable from PRFs with reasonable[1] polynomial time[1] complexity. We have set the bound on "reasonable time" to $2^{2\sqrt{\text{security}}}$. Thus, r_{LR} rounds can be seen as a single LR round [or as we call it a "solid" round] that can guarantee solid polynomial growth, complete diffusion and many other important cryptographic properties. Of course, a cipher that perpetually fails to build a PRF [fails to "solidify"] is certainly fatally flawed.

---

* f^{32} used in DES is not by any stretch of imagination pseudorandom[1, p. 377]
† f^n is pseudorandom if there is no distinguishing probabilistic **polynomial in n** time algorithm for it[1, p. 377]

Practical results show that the number of rounds that resists all known and often new attacks seems to be a consistent linear factor of 4 times r_{LR}. We see it as a practical confirmation of correctness of the Luby-Rackoff theorem. There is not a single insecure block cipher that uses at least $4*r_{LR}$ rounds, yet most well studied ciphers that use fewer rounds have been broken. We do not consider attacks with total time*memory complexity $\geq 2^{\text{security}}$ to be successful at breaking the cipher.

1.2 General description

In short, our ASD tests work as follows: a small set of d bits of the cipher state, key, IV or data input is looped through all possible combinations. After each round, either bits of output or bits of state are linearly summed for all the 2^d combinations thus removing all the monomials of degrees less than d from all the polynomial relationships. In this way we extract a subset of monomials of a given degree d. Calculation of all the monomials of that degree may be computationally infeasible, so we reconstruct only a sufficiently large fraction of them, after which we perform randomness tests on the resulting stream. It allows us to determine very quickly how soon algebraic structure of the iterated feedback either on its own or with other components [such as input or output combiners] no longer contains local non-randomness. The reader can find details of the process of reconstructing ANF from the output of a black box in the literature[3],[4].

We use two kinds of ASD tests that work equally well: testing all possible small sets of bits and grouping bits in a small sliding window. One of the two will produce slightly better results than the other depending on the cipher structure. Our third test for similarity between rounds detects the cipher's vulnerability to slide attacks. In fact, according to [1], each solid round must rely on an independent PRF. ASD tests verify that by skipping a given number of rounds to test other rounds of the cipher, not necessarily the first few.

Of course, passing automated tests like ours cannot be used as a blind proof of security (the cipher may still be vulnerable to guess-and-determine attacks, have insufficiently long periods or hard to find weak keys), but a cipher's failure to pass such tests should be taken very seriously because such distinguishers can be used to mount key recovery attacks.

1.3 Similar work

Results of similar but much simpler tests of algebraic structure of cryptographic primitives were previously published by *Eric Filiol*[3] and *Markku-Juhani O. Saarinen*[4]. They limited randomness testing to merely counting the number of logic gates or low-degree monomials present in the polynomial relationships of complete ciphers. It cannot possibly be called a sufficient measure of pseudorandomness or super-pseudorandomness and does not show how large the security margin is if the complete cipher simply cannot be distinguished from random with polynomial time tests.

2. Results of ASD tests

Our tests are capable of detecting structural flaws in some complete ciphers such as IDEA, TEA, Achterbahn, Grain v1, MICKEY2, KeeLoq, LILI, Polar Bear, in most LFSR-based ciphers, in ciphers vulnerable to Mod N attacks, decimation attacks, slide attacks, algebraic attacks, etc. These tests can show significantly different results for constructions that may appear quite similar. For example, Rijndael passes ASD tests only after 5 rounds, but Whirlpool after 3 rounds even with a much larger state that actually requires much greater diffusion.

Ciphers that suffer from entropy loss seem to fail ASD tests perpetually. If a cipher fails ASD tests perpetually, we automatically assume that it is due to an implementation error [which turns out to be correct 99% of the time] and check every part of the source code to make sure that it is not an accident. This high implementation error rate must be the reason why *Eric Filiol* believed that he had discovered shocking flaws in ciphers like Rijndael, which are simply not there.

Although some ciphers may be able to get away with partial non-randomness in their states, at least the bits of state that get combined to produce the output, as well as the output itself must pass ASD tests to form a solid round. Practice also shows that a cipher's perpetual inability to turn its secret state into indistinguishable from random is a sure sign of an exploitable structural weakness. Good examples of that are LILI, TEA and KeeLoq.

2.1 Well-known ciphers and hash functions

In this paper we will present only the most notable or controversial results for the DES, Skipjack, AES, IDEA, RC6, Serpent, Twofish, KeeLoq, TEA, XTEA, MD4, MD5 and SHA. The complete list of all the tested ciphers and their results can be found on http://defectoscopy.com/results.html. The infinity symbol ∞ in the tables below means those ciphers can be distinguished from random or broken if iterated for any number of rounds (perpetually).

2.1.1 Results:

Cipher	r_{LR} *key*	r_{LR} *data*	*Rounds*	*Broken*	$3*r_{LR}$	$4*r_{LR}$
DES	6	6	16	20?	18	24
Skipjack A/B	6/11	7/14	32	31?	31.5	42
AES	4	5	10	7?	15	20
IDEA	9, ∞	4-8, ∞	8.5	8.5?, ∞	∞	∞
RC6	3	5	20	16?	15	20
Serpent	3	3	32	7?‡	9	12
Twofish	2	4	16	7?	12	16
KeeLoq	∞	∞	528	∞	∞	∞
TEA	∞	∞	64	∞	∞	∞
XTEA	15	9	64	26?	45	60
RTEA§	≤8+*w*	8	32+4**w*	–	≤24+3**w*	≤32+4**w*

Table 1. ASD tests results for some well-known block ciphers.

Hash	r_{LR} *data*	r_{LR} *hash*	*Rounds*	*Broken*	$3*r_{LR}$	$4*r_{LR}$
MD4	3/2/2	2/2/1	3	>3?	7.5	10
MD5	2	1	4	>4?	6	8
SHA-0/1	2	1	5	>5?	6	8
SHA-256	2	1	4	2?	6	8
SHA-512	2	1	5	2?	6	8

Table 2. ASD tests results for some well-known hash functions.

2.1.2 Description:

DES: The best known linear cryptanalysis of the DES claims to break up to 20 rounds. Our tests recommend 24 rounds for the DES. No structural flaws were found.

Skipjack: The best known attack claims to break 31 out of 32 rounds of Skipjack. Its two round functions have a very different structure, one strong during encryption but weak during decryption and the other strong during decryption but weak during encryption. Our tests recommend 42 rounds for their combination instead of the authors' proposed 32. No structural flaws were found.

AES: At present time, the best known attack against the AES breaks 7 out of 10 rounds, but our tests suggest that 20 rounds are required for the AES to be secure. So far this is the only inconsistency between the results of known cryptanalysis and results of our tests, although some cryptologists agree that the AES needs 20 rounds to be secure. We hope that the future research will help us resolve this inconsistency. No structural flaws were found.

IDEA: The best known attack claims to break 6 out of 8.5 rounds of IDEA. Our tests immediately show presence of large classes of weak keys, for some of which IDEA is broken perpetually and for some of which it is distinguished by our tests from random up to the full 8.5 rounds. Key schedule of IDEA must be redesigned.

---

‡ [5] claims to have broken up to 10 rounds of Serpent, but the total time*memory complexity of that attack is well over 2^{256}. Parallel brute-force is clearly faster. Thanks to the papers like [5] or [6], we have to question validity of all published attacks as possibly being more complex than parallel brute-force, while code-makers are forced to overdesign their primitives in fear of seeing them declared "broken" by such attacks.

§ Our proposed correction for the TEA cipher, which is much simpler and faster than the TEA or XTEA

RC6: The best known attack claims to break 16 rounds of RC6. Our tests suggest that the authors have chosen 20 rounds for RC6 correctly. No structural flaws were found.

Serpent: The best known attack[5] claims to break 7 rounds of Serpent. Our tests recommend 12 rounds for Serpent instead of 32. It is obviously overdesigned. No structural flaws were found.

Twofish: The best known attack claims to break 7 rounds of Twofish out of 16. Our tests suggest that the authors have chosen 16 rounds for Twofish correctly. No structural flaws were found.

KeeLoq: Algebraic attacks break KeeLoq with any number of rounds. KeeLoq fails our tests perpetually exposing an obvious structural flaw in its NLFSR construction.

TEA: An equivalent-key attack breaks TEA with any number of rounds. TEA fails our tests perpetually when the difference is in the key. Its key schedule is obviously flawed.

XTEA: The best known attack claims to break 26 rounds of XTEA. Our tests recommend 60 rounds for the XTEA instead of 64. Much better attacks must exist. No structural flaws were found.

RTEA: As an example of application of our tests to cipher design, we propose our own RTEA variant that is notably much simpler and much faster than the TEA or XTEA:

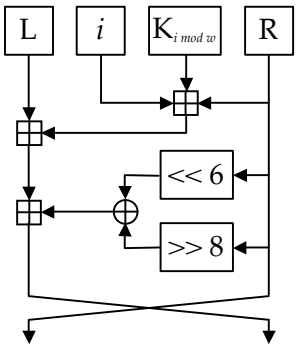

Fig 1. RTEA

(L and R are 32-bit data words, i is round index, K is the key, w is the number of 32-bit key words)

RTEA is an example of how ASD tests can help automate cipher design by finding optimal parameters such as the best choices of shifts, rotations and other operations. The choice of shifts in the TEA is not optimal. Out of all the best combinations of shifts, we chose 6-8 for its simpler implementation on 8-bit processors. Even with its much simpler round function, RTEA passes our tests faster than the more complex XTEA. Thus according to our tests, RTEA needs only (32+4**w*) rounds to resist all adaptive attacks with total time*memory complexity < 2^{32*w}. That is 48 rounds for 128-bit keys or 64 rounds for 256-bit keys. Its (24+3**w*) rounds should be able to resist all non-adaptive attacks. We have not performed any other cryptanalysis of RTEA, so it makes a good challenge for comparing our automated cipher design method with the modern manual approach.

MD4: The weakest hash function of the MD family. The proposed 3 cycles have been very badly broken. Our tests recommend 12 full cycles for MD4 to be secure. No structural flaws were found.

MD5, SHA: Recent practical attacks have exposed their insecurity. Our tests demonstrate that 8 full cycles (128 rounds) are required for them [and for **SHACAL** ciphers] to be secure instead of the proposed 4-5. No structural flaws were found.

SHA-2: There are no known attacks against SHA-2 at this time, but our tests suggest that at least 8 full cycles are required for it to be secure instead of the proposed 4-5. No structural flaws.

2.2 Achterbahn-128/80

2.2.1 Results:

Due to the low maximal degree of the polynomial relationships between bits of its state, Achterbahn can **never pass** our high-degree ASD tests. However, Achterbahn **perpetually fails** to turn the state

and thus the output of any of its registers and subsequently the output of the whole cipher into indistinguishable from random by our low-degree ASD tests as well.

2.2.2 Description:

The first and the most obvious observation from the state sizes (297 bits for Achterbahn-80 and 351 bits for Achterbahn-128) is that the security margin of Achterbahn-80 is significantly higher than that of Achterbahn-128. Unfortunately, a stateless nonlinear output combiner cannot secure a cipher with a weak or linear state unless the output combiner is as good as a block cipher of the same security level. Therefore Achterbahn cannot be fixed without feedback between its registers or a much stronger stateful output combiner.

2.3 Grain v1 and Grain-128

2.3.1 Results:

With the change localised in the key (NFSR), the output of Grain v1 begins to pass our ASD tests after **168** rounds and the output of Grain-128 begins to pass our ASD tests after **276** rounds. With the change localised in the IV (LFSR), the output of Grain v1 begins to pass our ASD tests after **214** rounds and the output of Grain-128 begins to pass our ASD tests after **313** rounds.

2.3.2 Description:

The number of sealing rounds (160 for Grain v1 and 256 for Grain-128) is extremely low, so low that our tests work as an extremely cheap chosen-IV distinguishing attack that completely **breaks** both Grain v1 and Grain-128. The first few bits of keystream even for a number of *known* IVs should be enough to recover the key. The numbers of sealing rounds should be at least 4*214=856 for Grain v1 and 4*313=1252 for Grain-128.

The key is loaded into the NFSR and the IV is loaded into the LFSR. If the difference is in the IV alone, Grain fails our tests longer than with different keys. Feeding IV bits into the NFSR during initialisation and choosing NFSR and LFSR with co-prime sizes could strengthen the cipher. Another possible solution could include feeding NFSR and LFSR off each other during the IV loading and state sealing stages in a way similar to MICKEY2. With these modifications, Grain would require a much smaller number of sealing rounds.

The proportion between the state size and the output size is also too low. To make both Grain ciphers secure against state attacks, one bit of output should be produced on every 6 rounds.

2.3.3 Other notes:

The IV is smaller than the key size in both ciphers. It makes them inherently vulnerable to TMDTO attacks. Considering TMDTO attacks, Grain v1 cannot be considered more secure than (80+64)/2=72 bits, and Grain-128 cannot be considered more secure than (128+96)/2=112 bits.

The reference implementation only stores the pointer to the key in the context, not a copy of the key as it should. It needs to be corrected: the key storage may get freed or overwritten before the next call to the *ivsetup*. Speeds of both implementations can also be improved dramatically.

2.4 MICKEY2-80 and MICKEY2-128

2.4.1 Results:

In mixing mode: A change in register R **fails** to affect either R or S in a way indistinguishable from random in both ciphers **perpetually**. Register S passes ASD tests after **65** rounds in MICKEY2-80 and after **102** rounds in MICKEY2-128. A change in S also affects R in a way indistinguishable from random after **56** rounds in MICKEY2-80 and after **80** rounds of MICKEY2-128. Without mixing, a change in R or S in both ciphers is distinguishable from random **perpetually**.

2.4.2 Description:

Both the key and the IV are loaded in the same way, added one bit at a time into both registers. Therefore performance of each register and their contribution to the output were tested separately. The register R seems to be significantly weaker than S, but since both the key and the IV are loaded into both registers and since their outputs are linearly combined, MICKEY2 can be seen at least as secure as S, so we can treat the weak R as a small addition to the cipher's security.

The number of sealing rounds in MICKEY2 (180 in MICKEY2-80 and 288 in MICKEY2-128) is much smaller than the number that would be required according to our ASD tests for the mixing mode to be secure (65*4=260 for MICKEY2-80 and 102*4=408 for MICKEY2-128).

After loading the IV and the key in mixing mode, both ciphers then switch to a **flawed** mode of operation in which they probably continuously lose entropy. Increasing the size of the state will not help secure the ciphers. MICKEY2 ciphers have a greater chance to operate securely in mixing mode, but they cannot release more than one bit of output on every other clock cycle.

In either case, the subsequent key recovery would not be trivial, but with only 180 rounds of MICKEY2-80 and only 288 rounds of MICKEY2-128 to attack, it would be faster than the brute force. To prevent key recovery, the number of sealing rounds including key loading needs to be increased to at least 260 for MICKEY2-80 and to at least 408 for MICKEY2-128.

2.4.3 Other notes:

MICKEY2 ciphers load the key after loading the IV, which would increase their resistance against state recovery attacks – a successful attack on the state may not result in trivial key recovery. Such approach is great for low area applications, but the low resynchronisation speed can make the cipher unsuitable for scalable high-speed applications in CTR mode or operating on small packets.

Software speed of the submitted source code can also be significantly improved.

2.5 Trivium-80

2.5.1 Results:

Trivium **perpetually fails** to turn an arbitrary change in its state into indistinguishable from random. A change in the key turns the output into indistinguishable from random after **544** rounds and the state into indistinguishable from random after **599** rounds. A change in the IV turns the output into indistinguishable from random after **544** rounds and the state into indistinguishable from random after **574** rounds.

2.5.2 Description:

The last few outputs or states fail only every 3rd round and only the randomness tests that analyse every 3rd bit of the stream. It's a clear indication that decimation attacks against Trivium selecting properly spaced bits to attack and combining properly spaced bits of the output stream would be more successful.

Trivium's number of sealing rounds (1152) is significantly lower than the number recommended by our tests: 4*544=2176. The proportion between the size of the secret state and the output size in Trivium is also too low according to our ASD tests that recommend producing one bit of output on every 8 rounds ($\geq 2176/288$).

Trivium's perpetual failure to turn its state into indistinguishable from random can be easily fixed with inverters in the feedback function. Trivium with three NAND gates instead of the three AND gates in its feedback function passes our ASD tests turning any change in its state into indistinguishable from random after 512 rounds. With these two tweaks we could recommend Trivium as a secure cipher. It would also occupy the same or smaller area in ASIC.

2.6 VEST4-80, VEST8-128, VEST16-160 and VEST32-256

2.6.1 Results:

The following table shows the number of rounds required for a change in one part of VEST ciphers to affect another part in a way indistinguishable from random:

A change in:	Affecting:	VEST4-80	VEST8-128	VEST16-160	VEST32-256
Input	Output	8	8	9	9
Input	Core	6	7	7	8
Counters	Core	5	6	6	7
Core	Core	6	6	6	7
Core	Output	5	5	6	6
Core, AE	Core	5	6	6	7
Core, AE	Output	5	5	6	6
Plaintext, AE	Output	6	6	7	7

Table 3. ASD tests results for VEST ciphers.

2.6.2 Description:

On their own, even combined RNS counters **fail** ASD tests **perpetually**, so security of VEST depends mostly on the size and complexity of the polynomials produced by the core accumulator.

In all VEST ciphers, a controlled or a worst-case change in their core accumulator state affects the output in a way indistinguishable from random after 5-6 rounds. Although some of the bits of the core accumulator state fail ASD tests for one more round, the cipher output passes the tests sooner thanks to the wide output combiner. The input bits need three extra rounds to propagate linearly through the counters to the core accumulator, which means that three rounds must be added to the sealing process on top of the number recommended by the ASD tests. This delayed input propagation is common in stream ciphers and does not affect the state/output proportion.

As can be seen in the table below, the overdesigned VEST4-80 could be safely made 3 times faster, VEST8-128 could be safely made more than 2 times faster, VEST16-160 could be made 30% faster and VEST32-256 is at its speed limit. Sealing could also be reduced to 23 rounds for VEST4-80 and VEST8-128 and to 27 rounds for VEST16-160 and VEST32-256.

	VEST4-80	VEST8-128	VEST16-160	VEST32-256
Security:	80	128	160	256
State Size:	256	384	512	768
Core Size:	83	211	331	587
Required Degree:	66	64	70	106
Required Sealing Rounds:	20+3=23	20+3=23	24+3=27	24+3=27
Actual Sealing Rounds:	32	32	32	32
Required State/Output Proportion:	20	20	24	24
Actual State/Output Proportion:	64	48	32	24
Actual Core/Output Proportion:	20	26	20	18
Maximum Advised Output Width:	12	19	21	32

Table 4. Interpretation of ASD tests results for VEST ciphers

3. Testing implementations and black-box ciphers

The main difficulty in testing cryptographic primitives with automated tools lies in understanding their inner structure. It is also the most time consuming part of the process. Automated tools can be used directly and their results can be interpreted trivially only for the ciphers with one mode of operation and with the same round function used for all the rounds. The easiest ciphers to analyse automatically are of course those with a single round function, a clearly defined internal state, clearly defined inputs and outputs and a very simple FSM, in other words with as few structural changes between different stages as possible. Ciphers with several different round functions, ciphers with internal states of unclear size or structure and ciphers that switch between different modes of operation are the hardest to test. Thus results for ciphers like Mosquito with a number of completely different in size and strength round functions are nearly impossible to interpret correctly. Ciphers like Trivium or Phelix with complex internal states of their software implementations also make the worst-case scenario because they require either complete rewriting of their software implementations or their tedious manual study and adaptation.

It always takes more time to understand the cipher and to adapt its source code than to execute our tests. Therefore we include here an informal request to all the code-makers to consider making software implementations of their ciphers cryptanalysis-friendly. The number of components (counters, diffusers, LFSRs, NLFSRs, combiners, whitening, substitution, transposition, compression and expansion layers, etc.) has no effect on the difficulty of testing the primitive with our tools. Only complexity of the finite state machine switching between a number of different modes of operation and different round functions makes automated cryptanalysis difficult.

3.1 Source code requirements

In general, our tests can be applied to any black-box implementation of any cryptographic primitive and the source code is not required, only access to the round function, to all of its inputs and outputs and to the initialisation process. However, at this stage there is no standardised API to automate ASD testing of black box functions completely. We present our current API in (3.2). Our requirements for the usual software implementations are:

3.1.1 State (both fixed and variable) must be very clearly defined. It's okay if its size is an odd number or if it consists of several different parts. As long as it is very clear what those parts are and when/how they are updated. Please have them clearly marked with comments like:

```
ulong       ctr;    // round counter
ulonglong   ctr;    // keyed secret 64-bit counter, linearly updated by ctr_update()
uchar       key[STATE_BYTES];       // 4321-bit keyed nonlinear secret state
ulong       key[STATE_WORDS];       // 4321-bit fixed key schedule
ulong       x[4];   // 128-bit previous round ciphertext
```

Without such information it may be very difficult to test attacks against the state if the software implementation is too complex, in which case only the primitive's input and output can be tested fairly quickly. If there are any gaps in the state, please have them listed very clearly. Only C and Intel assembly implementations are acceptable. No Basic, C#, C++, Java, Pascal, Perl, Python, etc.

3.1.2 Round function must also be implemented as clearly as possible. Performance is not important. A bit-by-bit implementation of Trivium with a 288-bit state is much easier to test than its 64x optimised implementation with a 320-bit state with unspecified gaps in it.

3.1.3 Output must be produced as soon as it is ready, not buffered to improve performance.

3.1.4 Unrolled implementations are fine, as long as it is clear where the state/output is stored and what parts of it get updated. However, to facilitate automated analysis of a large number of ciphers simultaneously, it is best to have a separate round function or a round-dependent loop.

3.1.5 Key schedule alone is not interesting. The way it is implemented is of no importance. Its quality is automatically tested by its effect on the output in combination with the round function.

3.2 Automated Cryptanalysis API

Currently, all the cryptographic primitives are tested separately for every variation listed in Table 5 below. The API for automating this process completely is work in progress and can hopefully be simplified further. Any black-box function is acceptable. It can be implemented as a C or Intel Assembly source code or as a static or dynamically linked library, and even in hardware. The library must implement at least one function of the following type:

typedef void __cdecl__ ASD_cipher (u32 *input, const u32 in_bits, u32 *output, const u32 out_bits, const u32 first_round, const u32 skip_rounds, const u32 rounds);

The functions accepting key as their input must perform key initialisation. ASD_cipher functions must begin with the 'first_round' round (the first round number is 0). They must first process input for 'skip_rounds' rounds not saving any outputs. After that they must iterate saving their outputs for 'rounds' more rounds. If out_bits < 8, the output bits of all the rounds must follow each other directly, otherwise each round's output must occupy ((out_bits+31)/32) 32-bit words.

If possible, all the functions must be implemented twice, iterated in both directions. Automated round function reversal is not implemented yet. A separate ASD_cipher function must be implemented for all the necessary combinations of 'input' and 'output' parameters, such as:

input	output
key	state
key	output
data/IV/hash	state
data/IV/hash	output
state	state
state	output

Table 5. List of common input/output variations to test

The following initialisation function must return the list of all the implemented functions to test, along with their respective names, input and output widths and maximum supported numbers of rounds (0 means unlimited). The function itself must return the number of functions to be tested. If it receives NULL as any of its parameters, it must not initialise anything and must simply return the number of implemented functions:

u32 __cdecl__ ASD_init (ASD_cipher *ciphers, char **names, u32 *in_widths, u32 *out_widths, u32 *max_rounds);

Names of all the ASD_cipher functions must begin with the same primitive's name followed by a single dash ('-') and short but clear and distinct labels for every function.

4. Future work

Our current results include testing of the forward (encryption) functions only. However, iteration of the ciphers in both directions must be tested. It is particularly difficult for stream ciphers since it requires automated reconstruction of the reverse round function from a black-box forward round function. The whole purpose of automated cryptanalysis is to save time on tedious manual study of the cipher structure and to eliminate human factor as the main source of errors in both design and analysis. Therefore manual reversing of the round function for each cipher is not an option.

Another aspect that has been stressed numerously is the possibility of extending our tests to find good distinguishers and implement key recovery, possibly breaking[1] $< 3*r_{LR}$ rounds automatically. It is one of the prime directions of our future work.

5. Acknowledgements

We express our deep appreciation to everyone who has contributed his recommendations that helped us improve the quality of this work. Our special thanks go to Prof. Nicolas Courtois, Dr. John Kelsey, Prof. Daniel J. Bernstein, Dr. Joan Daemen and Dr. Praveen Gauravaram.

6. References

[1] M. Luby, C. Rackoff, "How to Construct Pseudorandom Permutations from Pseudorandom Functions", SIAM Journal on Computing, April 1988, Vol. 17, No. 2, pp. 373-386.

[2] M. Naor, O. Reingold, "On the Construction of Pseudo-Random Permutations: Luby-Rackoff Revisited", Journal of Cryptology, 1996.

[3] E. Filiol, "A New Statistical Testing for Symmetric Ciphers and Hash Functions", ICICS 2002, vol. LNCS 2513, Springer-Verlag 2002, pp. 342-353.

[4] M.-J. O. Saarinen, "Chosen-IV Statistical Attacks on eSTREAM Stream Ciphers", ECRYPT 2006/013, http://www.ecrypt.eu.org/stream/papersdir/2006/013.pdf

[5] E. Biham, O. Dunkelman, N. Keller, "The Rectangle Attack – Rectangling the Serpent", EuroCrypt 2001.

[6] O. Dunkelman, N. Keller, J. Kim, "Related-Key Rectangle Attack on the Full SHACAL-1", SAC 2006.