Security Analysis of the ZK Crypt Data Authenticator and Stream Cipher against Algebraic Cryptanalysis, Differential & Correlation Attacks

Gregory V. Bard, Carmi Gressel, and Avi Hecht.

1 Summary

ZK Crypt is a three function system (Data Authenticator, Stream Cipher, and True Random Noise Generator) based on an extremely high-diffusion hardware number generator, with an orthogonal dual track feedback mechanism that precludes message modification, algebraic, side-channel, differential and correlation attacks. This device has only 9500 gates and is designed for simultaneous deciphering and data authentication with single or parallel concatenated units, maximizing throughput with low energy usage, and very small circuitry. This document expands previously used methodologies for robustness and security analysis with a description of the intractability of an algebraic attack. The simplified methods used herein show the nonexistence of useful differentials via repeated word analysis, and by counting high bits to detect bias, on all Data Manipulator state variables.

2 Overview

Accepted guidelines for pseudorandom number generation include large scale diffusion, and the use of permutations and non-linear transformations. Cryptanalysts prefer provably secure systems, which are easily analyzed, like simple easy-to-understand designs [12], [31]. A system with proved massive diffusion, where every internal bit is a complex function of disparate variables is the antithesis of the common conception of simplicity. In previous expositions we proved massive immediate diffusion, e.g., a single toggled bit in a word diffuses into at least 144 binary variables in the monomial equations of the 32-bit Word Manipulator in the first clock cycle. We also showed message modification was more strongly precluded by the dual-track orthogonal feedback mechanisms. In this exposition, we have shown the intractability of an algebraic attack on one part of the high diffusion system, where we have measured both the polarity balance of every word, and binary state variable, and the correlation between juxtaposed internal variables.

The purpose of this document is to show that using tests of algebraic complexity and repeated word tests we can prove multistep complexity, absence of differentials and the absence of correlation of internal and observable variables, generated by judiciously combining uncorrelated operands. In other documents we analyze the efficacy of our primitives, which underlie the methods of this document.

This document has essentially two parts. First, we show the algebraic cryptocomplexity analytically. Second, we prove with exhaustive testing with repeated word test and counting '1's in all index bits of all state word variables thereby showing that all internal state variables output balanced essentially random word streams.

3 Algebraic Analysis

3.1 What is Algebraic Cryptanalysis?

It is a fact that any map from a finite set to another finite set can be modelled as a polynomial system of equations over a finite field [2]. In fact, the field can be $\mathbb{GF}(p)$ for any p and thus trivially $\mathbb{GF}(p^n)$ for any n by the obvious embedding. In cryptanalysis, sometimes the cipher is written in terms of a particular finite field, and that is the natural choice. For example, the Algebraic Encryption Standard (AES) is written in terms of $\mathbb{GF}(256)$, and most attacks are attempted in that field, while Ferguson, Schroeppel, and Whiting [14] have a continued fraction expansion of the AES, over $\mathbb{GF}(16)$. However, in almost all

other cases, $\mathbb{GF}(2)$ is used, because $\mathbb{GF}(2)$ is binary. Since all circuits eventually become a set of logical gates prior to implementation, any circuit is a map from a set of inputs (a finite set) to a set of outputs (another finite set), and is therefore representable as a system of equations.

3.1.1 Measures of Hardness

A polynomial system of equations in m equations and n unknowns is hard to solve if $m \approx n$. If the system is degree d, and $m \approx n^d$, then the system is easy to solve by the XL algorithm [10]. Usually the number of equations and number of variables cannot be lowered by the attacker, though he/she may write additional equations. There must be at least one variable for each of the key and IV bits that one wishes to find. In this case, ZK-Crypt that is 256 or 320, since 128-bit or 160-bit keys and IVs are possible.

The degree of the system is crucial. If the degree is 1, then the system is linear, and well known techniques of linear algebra will work rapidly to solve the system in polynomial-time. Any system of degree higher than 1 can be rewritten into a quadratic system of equations [2, Ch. 2] by introducing new variables. However, this is not always wise, as the number of variables also has a crucial impact. For example, SAT-solvers do not appear to be concerned with the degree, but are very concerned with sparsity, explained in [11].

3.1.2 Building and Growth of Polynomials

There are primarily three ways to combine two polynomials, but they have significantly different impacts on the two measures of complexity: the degree of the polynomial and the monomial count.

Suppose p_1 has degree d_1 and m_1 monomials, likewise p_2 has degree d_2 and m_2 monomials. Then, in the absence of accidental cancelations, $p_1 + p_2$ has degree $\max(d_1, d_2)$, and it will contain $m_1 + m_2$ monomials. Over $\mathbb{GF}(2)$, this is equivalent to the logical operator XOR.

On the other hand, $p_1 p_2$ will have degree $d_1 + d_2$, which is obviously greater than $\max(d_1, d_2)$ unless one or the other is degree 0 (a constant). The number of monomials is still worse, because in the absence of cancellations it is $m_1 m_2$. In fairness, if the two polynomials have variables in common, the probability of a cancellation is not very low. Multiplication of the polynomials is equivalent to the logical operator AND.

The third possibility is $p_1 \circ p_2$ which basically means $p_1(p_2(x))$. In this case the degree is $d_1 d_2$ and so this makes the degree spike very rapidly. The number of monomials will be $m_1(m_2)^{d_1}$. So here it should be made very clear that while $p_1 + p_2 = p_2 + p_1$ and also $p_1 p_2 = p_2 p_1$, i.e. they are commutative, this is false for $p_1 \circ p_2 \neq p_2 \circ p_1$. Thus the monomial count can also be a non-symmetric function of the monomial counts of each of the inputs. The logical equivalent to this operation is making one sub-circuit the input to another sub-circuit.

3.2 Main Features of ZK-Crypt

In the next two sections we will examine a few features of ZK-Crypt that will render an algebraic attack essentially impossible. There are other features that are hard to model besides these, but we highlight here possibly the least challenging parts of the cipher, which are nonetheless sufficiently complex that algebraic attacks are infeasible. The idea being that if even the simplest part of the cipher cannot be modeled algebraically, there is no hope of modeling the entire system.

This modeled part is called the Data Churn (See Figure 1), sitting in the middle of the ZK-Crypt Cauldron, which while simple may be the most innovative major component, even without trying to analyze the versions of feedback and the virtually unobservable Top, Middle, Bottom (TMB) Tier elements in the Register Bank.

First, we will track some bits through the Top, Intermediate and Bottom sets of MAJ-double-XOR, Store-Xor, and splash matrices. These eight stages (three each of MAJ-double-XOR, and Store-Xor, and two each of the splash matrices) are denoted the Data Churn, and are typical of bit patterns in ZK-Crypt in that they are very hard to model.

Second, we will discuss several other regions of the circuit/algorithm which further confound the algebraic cryptanalysis process. The Data Churn alone is enough to render an algebraic attack infeasible,

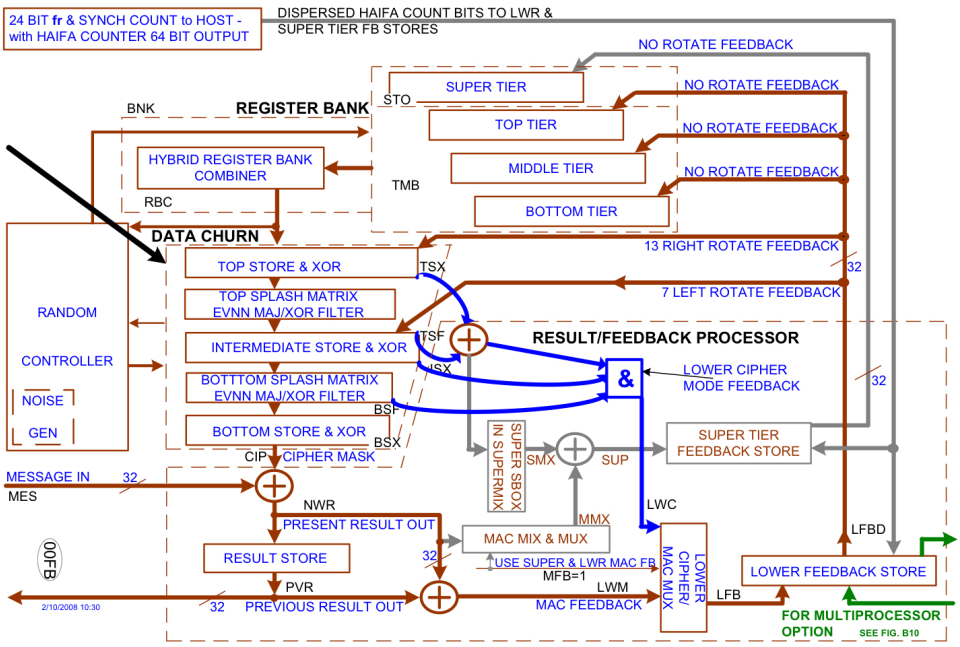

Figure 1: Data Churn in the ZK-Crypt Cauldron

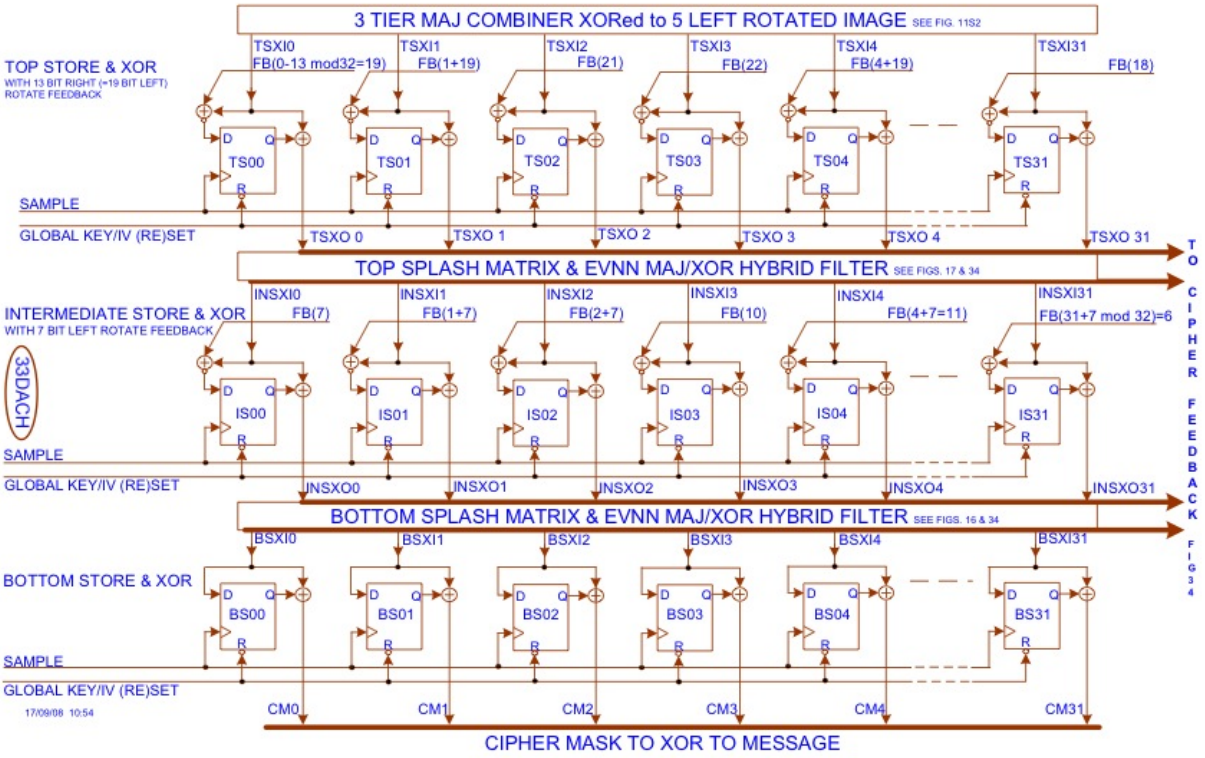

Figure 2: Eight Stages of the Data Churn

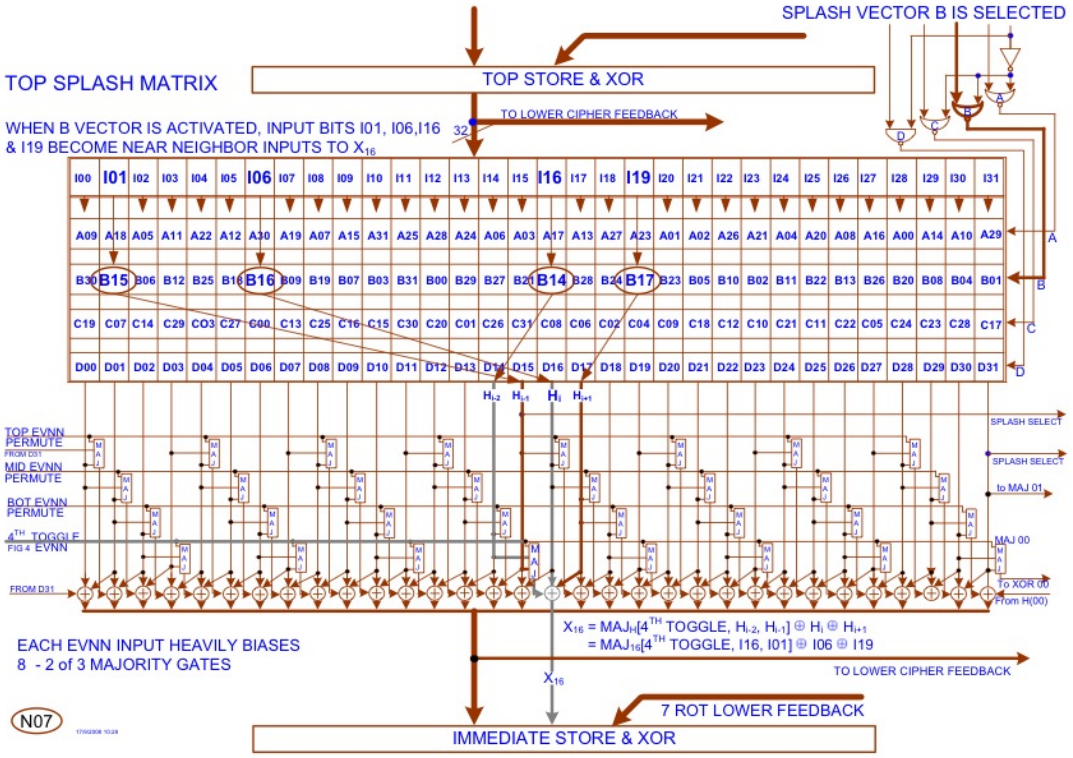

Figure 3: Hybrid Combined 4 B-Vector Deranged Data Bits

but these additional hurdles may lengthen the "nails in the attacker's coffin." Needless to say, the concatenated ZK-Crypt pair, with swapped feedback streams, used in 512-bit and 768-bit lengths, is more than exponentially stronger, but also double the ZK-Crypt throughput.

3.3 The Data Churn

Before we track bits in the Data Churn, (See Figure 3) we must identify the algebraic form of a few of its basic elements. In all cases, the $x_1, \ldots, x_n$ represent n input bits and $y_1, \ldots, y_m$ represent m output bits, viewing the function as $\mathbb{GF}(2)^n \rightarrow \mathbb{GF}(2)^m$.

3.3.1 Splash Matrices

The Splash Matrix consists of four permutations of 32 elements (i.e. elements of the group S_{32}. One of them, denoted "D" is the identity permutation. The other three are derangements, because $\sigma(x) \neq x$ for all x.

Since there are 4 of these, we can imagine four bits, a, b, c, d, which are selector bits, and say that exactly one of these should be 1 and the others 0 at any time (indicating that the permutation of the corresponding letter has been selected for the splash). We then can write,

$$\begin{aligned} y_0 &\leftarrow ax_9 + bx_{30} + cx_{19} + dx_0 \\ y_1 &\leftarrow ax_{18} + bx_{15} + cx_7 + dx_1 \\ \vdots &\quad \vdots \quad \vdots \\ y_{31} &\leftarrow ax_{29} + bx_1 + cx_{17} + dx_{31} \end{aligned}$$

3.3.2 The Majority Function

A function $\mathbb{GF}(2)^m \to \mathbb{GF}(2)$ is said to be balanced if half the output values are one and the other half are zero. The majority function of $2n+1$ bits is the balanced function with highest algebraic immunity from the set of all balanced functions $\mathbb{GF}(2)^{2n+1} \to \mathbb{GF}(2)$, see [9].The majority of $a, b,$ and c is given by

$$\mu(a,b,c) = ab + bc + ac$$

3.3.3 MAJ-Triple-XOR

This stage consists of a series of "majority" functions. Then each output is given by an XOR of three inputs. Those inputs are two majority functions, and also some input bit. See Page 34 of [25]. There are four control bits here marked "top", "middle", "bottom", and "fourth". We will denote them τ, m, b, f. In particular we can write, as two isolated examples,

$$\begin{aligned} y_8 &= \mu(x_6, x_7, \tau) + x_8 + x_9 \\ y_{18} &= \mu(x_{16}, x_{17}, b) + x_{18} + x_{19} \end{aligned}$$

Resolving the μ, we obtain instead

$$\begin{aligned} y_8 &= x_6x_7 + x_6\tau + x_7\tau + x_8 + x_9 \\ y_{18} &= x_{16}x_{17} + x_{16}b + x_{17}b + x_{18} + x_{19} \end{aligned}$$

3.3.4 Store-and-XOR

See Figure 4. Here we need to use some sort symbol for time. Let x_i^t represent x_i in the current clock cycle, x_i^{t+1} in the next, and x_i^{t-1} in the previous.

In each clock tick, the Store-and-XOR recalls the input value from the previous clock tick, x^{t-1}, and XORs it with the present input x^t. The present input, in its original form, is stored. With the Lower Feedback bit rotated into the Intermediate Store-And-XOR (not into the Bottom Store-and-XOR). So we have

$$y^t \leftarrow x^{t-1} + x^t$$

3.3.5 Combining these into One Single Layer

Suppose we were interested in bit 8, for some reason. The Bottom Store-and-XOR would output the XOR of the values of bit 8 at time t and $t-1$. Above that is the Bottom MAJ-double-XOR which tells us

$$\begin{aligned} y_8^t &= x_6^t x_7^t + x_6^t\tau^t + x_7^t\tau^t + x_8^t + x_9^t x_6^t x_7^t + x_6^t f^t + x_7^t f^t \\ y_8^{t-1} &= x_6^{t-1} x_7^{t-1} + x_6^{t-1}\tau^{t-1} + x_7^{t-1}\tau^{t-1} + x_8^{t-1} + x_9^{t-1} + x_6^{t-1}x_7^{t-1} + x_6^{t-1} f^{t-1} + x_7^{t-1} f^{t-1} \end{aligned}$$

And therefore our bit of interest is

$$\begin{aligned} y_8^t + y_8^{t-1} = {} & x_6^t x_7^t + x_6^t\tau^t + x_7^t\tau^t + x_8^t + x_9^t + x_6^t x_7^t + x_6^t f^t + x_7^t f^t + \\ & x_6^{t-1} x_7^{t-1} + x_6^{t-1}\tau^{t-1} + x_7^{t-1}\tau^{t-1} + x_8^{t-1} + x_9^{t-1} + x_6^{t-1}x_7^{t-1} + \\ & x_6^{t-1} f^{t-1} + x_7^{t-1} f^{t-1} \end{aligned}$$

Now the Splash Matrix table allows us to write each of these x_i's as a quadratic expression, just as we wrote x_0 and x_1 in the section on that topic. Since terms like $x_6^{t-1}x_7^{t-1}$ exist, then it is clear that there will be quartic terms.

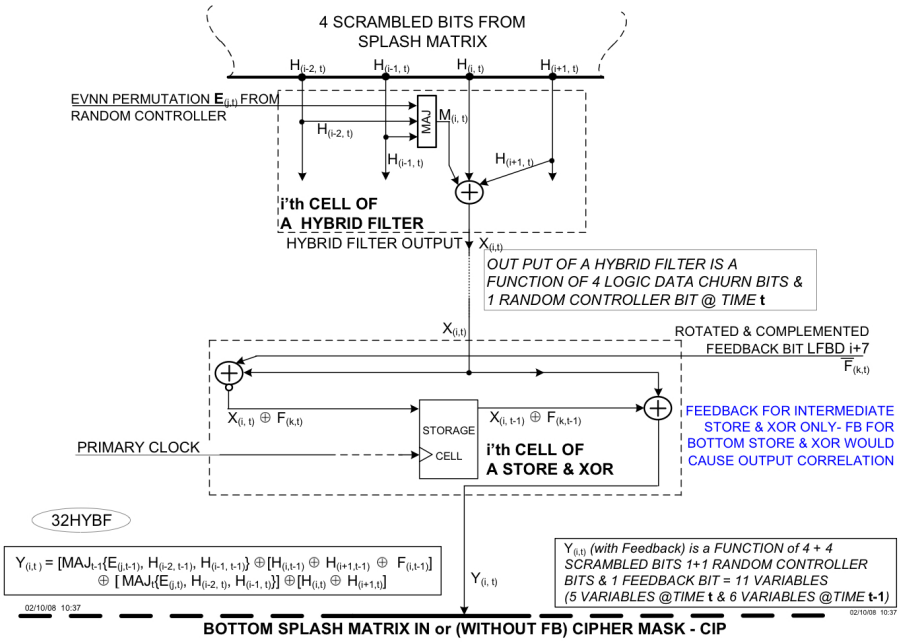

Figure 4: What Happens in the Hybrid Filter

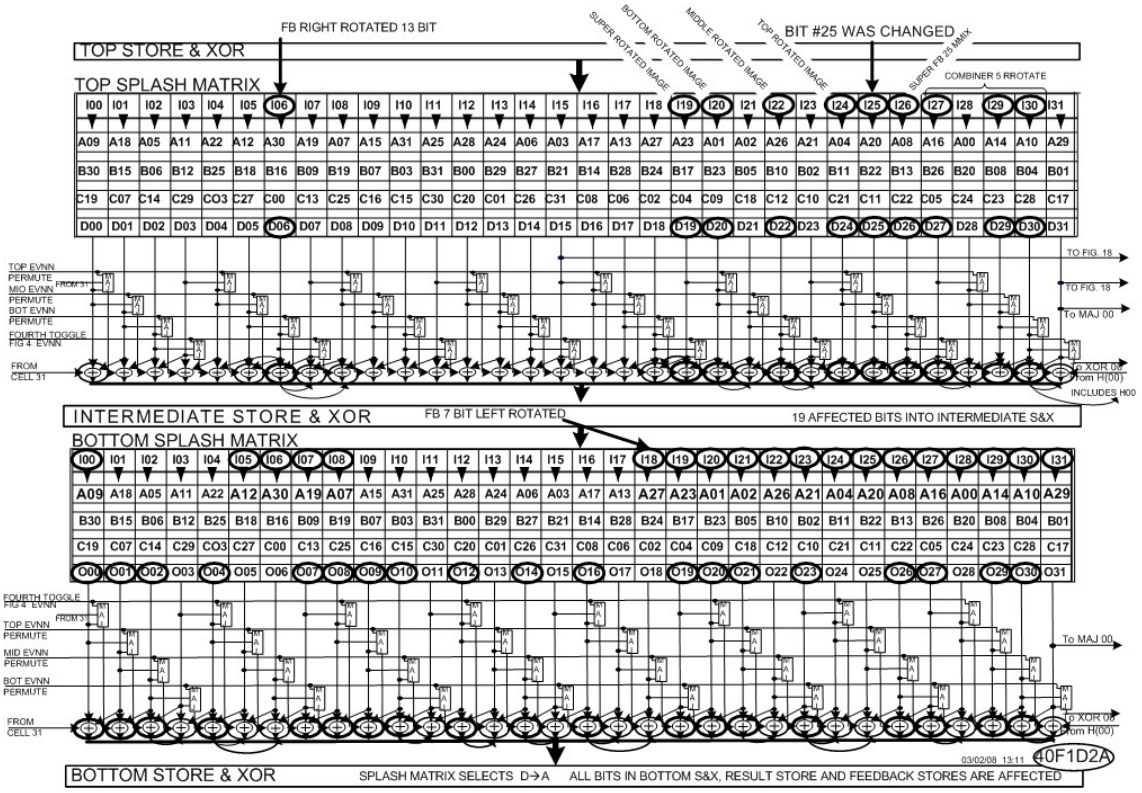

Figure 5: Diffusion of Message Bits in the Bottom Store-And-XOR

It is also interesting to count the number of monomials. Each splash matrix output is a quadronomial, and four terms are products of splash matrix outputs (i.e. of the form $x_i x_j$). These produce 16 monomials each, or 64 monomials total. There are 14 monomials in our equation, the remaining 10 are either linear or the product of a x_i with a control bit. These will give rise to 4 monomials each, or 40 monomials total.

Thus the equation for bit 8 will have 104 monomials and be of degree 4. Graphically, in Figure 5, we show how 9 bits from the Data Churn (derived from 1 message bit) offset in at least one way or another all 32 bits of the input to the Bottom Store-And-XOR.

3.3.6 Modeling More of the Data Churn

The Data Churn, however, is not one layer, but three layers. In fact, the middle Store-and-XOR will require $t-1$ and $t-2$ to be modelled, since we need information from $t-1$ in the bottom layer. This means in the top layer, we need $t-3$, $t-2$, $t-1$ and t to be modelled.

Since there are terms of the form $x_i x_j$ in our equation, we can predict that the total degree will be exactly 12. The number of monomials is harder to predict, but ignoring accidental cancellations, which should be unlikely, a good estimate is $12^3 = 1728$ monomials.

It is important to keep this in perspective. Tracking one single bit through the Data Churn, requires a degree 12 polynomial with roughly 1728 monomials. Yet there are 32 bits. Clearly, these equations will be long, of high degree, and there will be many of them. The problem is completely intractable.

For comparison, in one version of the algebraic cryptanalysis of Keeloq, the "frontal assualt" which failed, there were 1152 *quadratic* equations, the longest of which had 15 monomials [2, Ch. 2], to represent one encryption from start to finish. Only 384 of the equations were long, the others consisted of two monomials each. Thus the total monomial count was $384 \times 15 + 2 \times 2 \times 384 = 6528$. This was intractable.

Yet in ZK-Crypt we observe one trip through "the Data Churn" entails $1728 \times 32 = 55,296$ monomials. Clearly, this completely out of the question.

A model of the four-tiered Register Banks with 128 binary state variables, 8 permuting inputs, and where the 4 tiers are fed by two 32 orthogonal feedback streams all with higher degree inputs, where the hybrid four-tier combiner would certainly exceed the complexity of the Data Churn.

Modelling the Result/Feedback Processor entails 96 binary state variables, hybrid functions: e.g., Serpent S Boxes, Store-and-XOR, and HAIFA Counter's 64 deranged dispersions.

A model of the 62 binary state variables which incorporate 10 disparate data dependent feedbacks in the Random Controller and which was designed as a post-processor for a noise source may prove to be no less complicated.

The monomial estimate of the dense Data Churn was over fifty-five thousand, and we assume that the Register Bank would generate another (at absolute minimum) fifty-five thousand monomials, while the Result/Feedback Processor and the Random Controller might both be simplified "creatively" to fifteen thousand monomials each.

This might lead to a staggering problem with 10^5 to 10^{15} monomials. We estimate that this is well-out of reach for all computers in the decades to come. (Even 10^3 is beyond the realm of feasibility at the time of this writing.)

The 64-bit ZK-Crypt configuration, with two concatenated engines, are natural for 64- or 128-bit CPUs, and should satisfy the demand for high speed with the same energy used per bit processed, see Figure 5. We choose not to make any estimate or model for such a complex machine. See Figure 6.

3.4 Other Challenging Features

3.4.1 Highly Irregular Clocking

The highly irregular clocking is a serious issue. The standard if-then-else construction is "if X then Y else Z". Algebraically, this is written

$$XY + Z + XZ$$

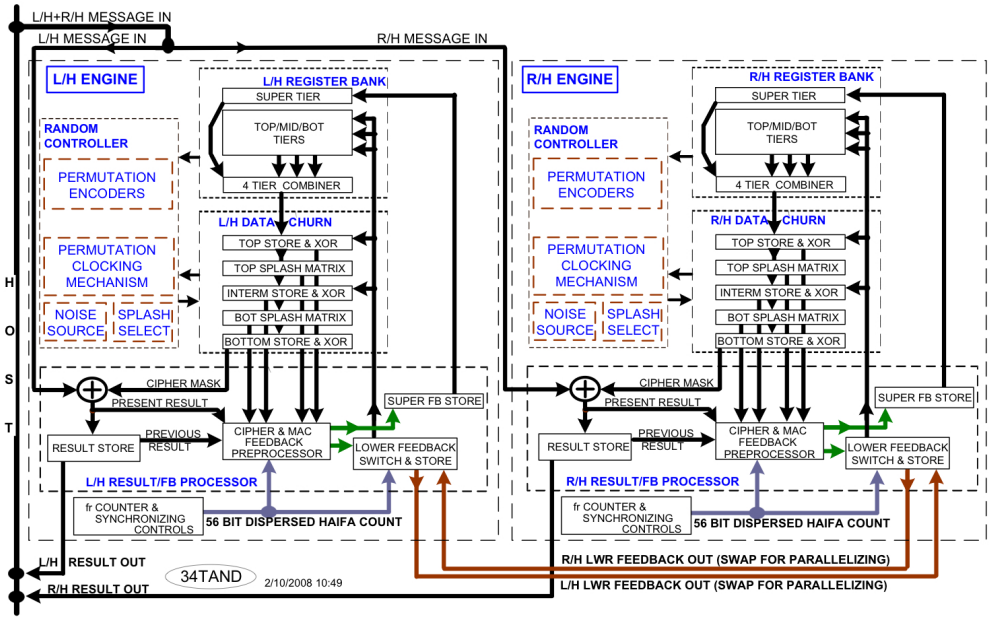

Fig. B10: 34TAND Pair of Concatenated ZK-Crypt with Swapped Lower Feedback Tracks

Figure 6: Two Engine Configuration with Swapped Feedback

and one can verify that if $X = 1$ then one gets Y and if $X = 0$ then one gets Z. But the degree of the entire equation is the sum of the degrees of X and Y, or of X and Z, which ever is worse. The monomial count will be the product of the count of X and the sum of the counts of Y and Z. Thus if each of X, Y, and Z, is even remotely complex, then the result formula will be huge.

3.4.2 The 56-Bit HAIFA/Synchronization Counter Dispersing 64 Deranged Feedback Bits

One would imagine that since the increment function is so simple to conceptualize that it is simple for algebraic attacks to handle. In reality, this is not quite so. If one has a long bit string which is being incremented, the carry bits have to look at each previous input, all the way to the least significant. Thus the 31st carry bit will include all previous 31 inputs. In fact, it will be a 31-input AND-gate, and thus a 31-degree polynomial.

3.4.3 Large Internal State with Massive Diffusion

There are 320 binary state variables in the Data Manipulator, 62 state variables in the Random Controller, 32 binary states in the Message, and finally 56 state variables in the HAIFA counter which is deranged, expanded to 64 bits, and dispersed to the 64 bits of the Lower Tier and Super Tier Feedback Stores. If we only talk about 16 bits from the HAIFA counter, and constrain our Domain to 2^{16} 32-bit input Message Words, then we have what is now called a "chaining value" of 430 binary state variables.

While the counter updates in a predictable way, (it is a counter after all), the remaining 374 bits are important and non-neglectable state variables.

For any stateful function f, the state can be thought of as a B bit string if there are 2^B possible states or fewer, regardless of how the state is actually stored. The internal state of this system is $B = 374$ bits of entropy, or 2^{374} states.

3.4.4 nLFSR Length Co-Primality

However, it should be noted that the internal state is not really a proper measure. Compare one nLFSR of length 64 bits to 4 nLFSR's of length 13, 23, 17 and 11 bits. If we know all the internal state in one stage, in the first case, after t iterations we know $64 - t$ bits. But in the second case, after t iterations we would know $64 - 4t$ bits. Furthermore, having mutually prime lengths would make it occur that the least common multiple of the register lengths (i.e. the length of full cycle of simulation), would be optimal, namely the product of the lengths.

Of course, while this is useful for security, it is not sufficient. The GSM cipher called A5/1 has been broken by Gendrullis, Novotnỳ and Rupp [16].

3.5 The Time-Memory Relationship

In block ciphers, a classical time-memory trade-off can be found in an attack on a one-way function f. This comes from [26, 3]. For example, let $f_k(P) = C$ be the encryption of a plaintext P into a ciphertext C under a key k. In a pre-computation step, $f_k(n_0)$, $f_k(f_k(n_0))$, $f_k(f_k(f_k(n_0)))$, $f_k(f_k(f_k(f_k(n_0))))$, ... is calculated, starting with a random ℓ-bit string n_0 for a length of $2^{\ell/2}$. Only the last value is stored—call these the terminal values—along with the starting point n_0 that lead to each terminal value. This is repeated $2^{\ell/2}$ times, for a total of 2^ℓ computations. The total storage is $2^{\ell/2+1}$

When an actual value of $f_k(P) = C$ is found, then one starts at C and computes $f_k(C)$, $f_k(f_k(C))$, $f_k(f_k(f_k(C)))$, ... and so forth, for $2^{\ell/2}$ steps. At any time, if one of the $2^{\ell/2}$ "terminal values" is encountered, then start again at the initial value which lead to that terminal value and keep repeatedly evaluating f_k. Until one reaches the value C. The immediately preceeding value is P. At worse, this requires $2^{\ell/2+1}$ computations.

Thus, by investing $2^{\ell/2+1}$ memory, at 2^ℓ computation time as a pre-computation step, we can calculate any pre-image under f_k in $2^{\ell/2+1}$ steps instead of 2^ℓ steps. This is a great speed-up.

However, in most cases 2^ℓ time is infeasible. (Note this process is highly ammenable to parallel programming). However, using arguments based on the birthday paradox, one can only partially execute

the above attack. If enough $f_k(P) = C$ are available, eventually one of them will fall into a "chain" whose initial and terminal value have indeed been stored, and then the pre-image of that point can be found.

3.5.1 Application to ZK-Crypt

Since the internal state of the function is B bits, having $2^{B/2}$ memory is completely infeasible.

The application to ZK-Crypt is not to perform this attack on the entire ZK-Crypt system but rather to pick some important point in the circuit that forms a function easy to compute in one direction but hard to find pre-images (i.e. a one-way function). This particular "choke-point" then would become the f and the time could be invested to make the chains and store the initial and terminal values.

Consider, then that $B = 374$ excluding the HAIFA counter. Then $B/2 = 187$, and of course $2^{187} \approx 10^{56.1}$. It is simply impossible to have this much memory without interplanetary travel because there are

$$\frac{5.97 \times 10^{24}\text{kg/earth}}{1.67 \times 10^{-27}\text{kg/nucleon}} = 3.57 \times 10^{51}\text{nucleons/earth} \approx 2^{171}\text{nucleons/earth}$$

and this even neglects the fact that a nucleon (or electron for that matter) can model a bit or several bits, but not the entire state of this machine, which would take at absolute minimum 374 bits.

3.6 Feasibility of an Algebraic Attack

Performing an algebraic attack then would be slower than brute force, and for the digest sizes of 224, 256, 384, 512 and 768 bits, brute force is completely impossible. Even for a 224-bit digest, one would expect a collision to take 2^{112} evaluations to discover. With one evaluation per nanosecond (which is absurdly fast), this would require 1.64×10^{17} CPU-years.

4 Relationship between Diffusion and Algebraic Cryptanalysis

Consider two possible worlds where the inputs to the circuit/algorithm, are identical in some way, with the exception of one particular bit. One can then imagine watching the bits go through the circuit/algorithm and track what later values are different between the two worlds. In a block cipher, for example, after one block it is expected that at least half, if not more, of the bits of the ciphertext should be different if one plaintext bit is flipped.

We showed in Figure 6 that all 32 result bits are affected.

Systems like the ZK-Crypt have very high diffusion. In the specific case of ZK-Crypt, it is shown in [25] that after a few clock ticks, essentially all bits have been made different in the two worlds. This applies more generally to algebraic cryptanalysis in the following sense. If all the bits are known to the attacker, except a single bit, which he must consider to be *either* a 1 or a 0, then after a few clock ticks, the attacker knows nothing!

We can be more mathematically precise. For any particular bit y, either the value changes when some other bit x is changed, or it does not. If y does change, then that changing bit x must appear somewhere in the formula for y. If not, then it does not appear. However, in ZK-Crypt, if one looks four clock cycles into the future, we observe that essentially every bit affects every other bit. So the formula for any of the outputs must contain *all of the input variables*.

At the very least, this results in an increase in the monomial count. The "best case" for the algebraic attacker would be if all the variables occur linearly. But if they appear together, (in the extreme case, all in one monomial), then the degree might increase as well.

But monomial count, as we discussed in Section 3.1.1 is a principle measure of the difficulty of solving a system of equations and this is sufficient to show that the work in [25] which shows that diffusion is high (thus confounding certain forms of differential cryptanalysis) will also confound the algebraic attacker.

5 Repeated Word Tests

5.1 Full Operand Repeated Word a Quality Check for Differentials and Correlations

We were introduced to the 32-bit Repeated Word [4] testing when Bernstein tested our ZK Crypt I. He discovered an average of about 12,500 RWs in 10 million samples, instead of the 11,625 range as measured with RD4 or the Linux RNGs. Luckilly for us, our second version was already on the table and checked out, comfortably, in the acceptable range of 11,630. However, we were surprised, because all previous development versions were tested with Marsaglia's rigorous DieHard [28] tests and received extremely good results. Our latest figures with colored Random Messages yield 16,714 Repeated Words.

Bernstein's RW [4] test is simple. Generate 10 million test random 32-bit words, and count all repeats. The RW counter is incremented each time a word seen before is seen again. Run the test 100 times, each time with a different random initial condition. Average all results, and calculate the Standard Deviation, STD. Poisson estimates that you'll find 11,614 pairs and 9 triplets—a total of 11,632 discrete RWs [19].

The following back-of-the-envelope calculation can also be used to estimate the expected number of repeated words [19], by ignoring the possibilities of triples, quadruples, and higher repeats which are quite rare. If there are n words generated, then there are $\binom{n}{2} = n(n-1)/2$ pairs of words. Each pair is a repeated word with probability 2^{-32}, since they are 32-bit strings. Thus the expected number of repeated words is $2^{-32}\binom{n}{2} \approx \frac{2^{-31}10^7(10^7-1)}{2} = 11641.53$

If the pool of possible words is limited, e.g., if one bit is stuck on "1", there are only 2^{31} possible bits, and there is a greater probability $1/2^{15}$ that one will find any particular number. If a word with an even number of bits and its rotated image are XOR summed, the result is an even number string (or ENS); i.e. it has an even number of '1's and an even number of '0's. This is an operation used in all tiers and every hybrid combiner. The result is a pool with half of the possible numbers, leading to twice the number of RWs. If a bit in a word has a bias toward either '1' or '0', the number of words that are likely to be chosen is reduced, and the RW number would be larger than the expected value.

The strong correlation sensed in the output of the MAJ filters is reflected in the output of the Hybrid Filters. This slight bias is successfully reduced by the following classical correlation immunizer, the Store & XOR. This is essentially in line with previous exhaustive statistical testing which led to this configuration [28].

The bias of a binary variable, the difference between the expected and actual values, is the "hook" that a cryptanalyst needs to begin "resynch attacks", related key attacks and re-keying attacks [7]. Biham states that "a design that has no highly probable, or highly improbable differential is expected to be immune to such attacks." We say an unsatisfactory deviation occurs if it differs by 1 in 10,000 from the mean. We have ascertained by counting and by repeated word tests that no word variable in the ZK-Crypt has a sensed differential. We show that none of our state variables has a sensed differential. We do show that many intermediate logic generated words have strong internal correlations; e.g., in the Hybrid Filters every fourth MAJ gate receives the same regulating signal, thus causing every fourth gate to generate an output equal to that regulating signal, with a probability of 3/4.

We also use the RW test to detect trace correlation, and to spot biases. Strictly speaking, instead of counting the bias, we are measuring for differentials. It is simpler, because one need only count the '1's on every index bit and see how close it is to half the number of samples. This is a lengthy but worthwhile procedure for checking software, and helps to qualify many kinds of trace and gross correlations. Three results are shown in Figures 7 and 8.

A characteristic of all functions in the ZK Crypt 32-bit Word Manipulator is "circularity"—with no end effects, as would typically be seen in normal and modular multiplication. Words in virtually every module rotate left, or right, by 1, 3, 5, 7, or 13 steps. As a result of extensive diffusion, each binary monomial is a function of many monomials, with both a linear and a non-linear component. Orthogonal feedback streams circulate fresh unpredictable words, statistically uncorrelated with the recipient's words, into rotating combiners.

These processes minimize but do not preclude existence of differentials. In the world of random numbers "things happen and have happened", and must be checked. For these reasons we engaged in

the experiments described below.

5.2 The 32-bit Word Manipulator

The ZK Crypt 32-bit Word Manipulator, in Figure 1 is comprised of three modules—the Register Bank, composed of 4 Tiers of complex nLFSR based primitives, and disparate linear and nonlinear combining mechanisms.

The Data Churn is composed of two sets debiasing, delinearizing, decorrelating filters, see Figure 2, separated by two 4 rule 32-bit displacement matrices. Five "quality" words from the Churn are fed into the Result/Feedback Processor. The Result/Feedback Processor prepares Results and generates two orthogonal streams of feedback to the Register Bank and the Data Churn.

The Data Churn and the Register Bank's clocking and permutations are regulated by the Random Controller. We check the quality of the Random Controller's triple binary-signal-noise source using our enhanced [30] "local randomness" test, which is an equivalent repeated word test. We also check the correlation of the three binary sources using the same repeated word test. The Word Manipulator generates 10 internally derived binary feedback signals which have a slow acting, randomly delayed, and mutually uncorrelated effect on the Random Controller's encoder's binary variables. The Random Controller generates 14 unique clocking and permutation signals which de-corelate clock permutations and derangements in the Register Bank and Data Churn word variables.

Each block in Figure 1 serves as a combiner, typically of 3 word variables. In each block there are state variables (memory in the form of flip-flops) and intermediate logic variables. We show that each state variable and intermediate logic word does not generate differentials, but some may purposefully generate correlations. State variable outputs show no (or in one case no relevant) sensed internal correlation and little correlation between juxtaposed words in a sequence. We show that it is judicious to generate intermediate internal logic correlation before combining the derived outputs of the state variables. The XOR gate is an excellent combiner if the input variables are not correlated; then the word bias is typically reduced quickly. The MAJ function combiner, much as it defies the algebraic cryptanalyst, can amplify differential correlations, but in the long run, apparently with "good" random inputs, does not generate new differentials.

Note well that Exclusive-OR combining two correlated words invites disaster, e.g., 01010101 XOR'ed to 01010101, will result in an all zero string.

Figure 3 and Figure 4 typify the ZK Crypt proprietary hybrid diffusing, delinearizing, debiasing and decorrelating of four scrambled near neighbor data bits, one permutation bit, and one feedback bit, into one output binary variable. Such 32-bit Hybrid Filters, in similar configurations, are strategically placed before each of the Top, Intermediate and Bottom Store & XOR correlation immunizers. The top Hybrid Filter is found in the 4-Tier combiner of the Register Bank, and the other two precede the Intermediate and Bottom Store & XORs.

We know from experience that there is often inexplicable correlation between four adjacent bits, e.g., the scrambled bits from the Splash Matrix, in Figure 3 and Figure 4.

If we simply input XOR summed bits $H(i,t) \oplus H(i+1,t)$ into the Store & XOR cell, correlation would most likely be amplified, and may be manifested in both $Y(i,t)$ and $Y(i+1,t)$, and similarly in $Y(i-1,t)$. Viewing the 32 sums of the form $H(i,t) \oplus H(i+1,t)$ as the output of the Splash Matrix, we actually see a Word output rotated one bit to the left into an Image, and then see that the Word output is XORed to itself. The result is an ENS word, with an even number of '1's. In Figure 3 we see the whole Hybrid Filter vector, where every fourth MAJ gate is controlled by a single uncorrelated permutating EVNN signal, from the Random Controller.

The EVNN controllers are seen on the left side of Figure 3. The 4 random EVNN controllers serve to generate correlation between every fourth bit on the outputs of the MAJ gates. If any particular input bit to a MAJ function gate is known to be a '1', with a probability of 0.75 the output of the MAJ gate will be a '1', assuming that the other two bits are uniformly random (i.e. if the MAJ input is 1, there is only one combination of the other two inputs that output a '0'.) Stated differently, each EVNN control signal affects every fourth output bit in the MAJ filter output of the Splash Matrix; there is a strong correlation between every fourth output bit; an average of almost 3/4 of the output bits will be of like polarity, and

these biased words would be repeated more frequently in the 10M samplings of the output of the MAJ Filter. Intuitively, such a masking permutation may be thought to be ill-concieved, but we show, in the following exercise, that the results are as desired, both in the short term and in the long term. There will be about 1.8 million Repeated Words in the MAJ vector output, (as compared to about 11,630 RWs in a word with no sensed internal correlation) but the output will show no measured differentials. We show that there is no sensed correlation between the two XORed vectors, as the RW sum of the output filter Hybrid Filter, in all instances, averages nicely in the normal range.

The Store & XOR filter typically lowers the RW count, as expected information theoretically. We suspected and observed that there is a very slight correlation between the t'th and t+1'th results (Cipher Masks) emanating from the Bottom Store & XOR. Before the last tweak, after establishing a fairly consistent average number of Repeated Words in the sampled Results tests (with standard deviations on the state variables all the realm of 100) we pushed our hypothesis to the extreme.

By locking three of the four EVNN signals to '1', we generated a highly biased and correlated string, with 22 '1's on the average in every Hybrid Filter MAJ output word. (One set of 8 bits would have no bias, the other three would generate 3/4 '1's on 24 bits.) As the Results of the RW test were satisfactory; we proceded to execute the two following tests (using same randomly chosen Initial Conditions) where we generated:

- the sequences of (10M + 1) ZK Crypt Cipher Mask Results, where from every t(0) to t(10M+1)th CM Result, at every t+1'th clock, each Y(t) CM Result is XORed to each (next) (Y(t+1)) CM Result to generate a new tth test Result; i.e., each CM Result is XORed to the following CM Result to generate a 10M test sampling, and,

- the sequences of (10M +7) ZK Crypt Cipher Mask Results, where from every t(0) to t(10M+7)th CM Result, at every t+7'th clock, each Y(t) CM Result is XORed to each (next) (Y(t+7)) CM Result to generate a new tth test Result; i.e., each CM Result is XORed to a "distanced by 7" CM Result to generate a 10M test sampling,

... in an attempt to support the hypothesis: "juxtaposed words have nearly infinitesimal correlation".

In the first instance, wherein near neighbors were XOR summed, there was a miniscule degradation of 4 more Repeated Words. This proves that there is (or at least was) a very small correlation between nearest neighbor outputs.

In the second instance, wherein Result words were "distanced"; there was an appreciable improvement; 16 less Repeated Words to generate our previous lowest recorded result, 11,614 RWs. All previous testing was with null word messages. We are now working on tests with colored random message inputs where we get similar results. From this we concluded: if we stored 7 output words and XOR summed the "7'th distanced" Results which we could assume had no correlation between them, we could "squeeze out" the last bit of crypto-complexity; essentially without degrading performance. (If we sacrifice performance and only sampled every 7'th output we might enjoy the same benefits).

Was the effort to reduce the number of Repeated Words by 16—indiscernible to an adversary—worth the cost? We would have to save 7 intermediate 32-bit words in the Result/Feedback Processor; i.e., 234 added flip-flops; increasing the size of the engine by 20%. The exercise was a practical step in understanding the kind of "things that happen".

As expected, if we halve the number of randomly chosen possible Result words to 2^{31}; we essentially double the number of Repeated Words which we see:

- obviously, if one bit in a word is a "Stuck On" on polarity (always '1' or '0'), the subset of possible words is halved- or if one bit has a high polarity probability; there is a loss of unpredictability, and even more important to us;

- if a "good word", X, is rotated 7 bits and XORed to itself, as we generate the Super Tier output, the near neighbour bits are distanced, and the pair of XORed bits are less correlated than near neighbor bits; and the output is an ENS word. Once again, the subset of possible words is halved. However, in this case we have not lost unpredictability (entropy) as we do not know if the result

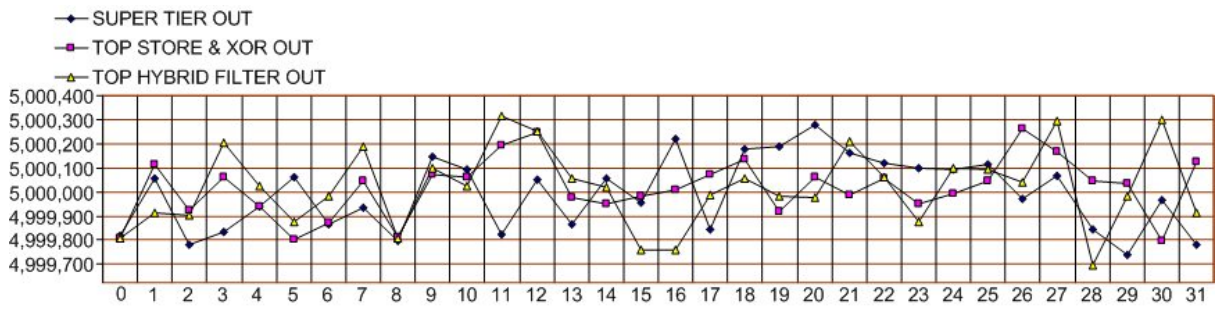

Figure 7: The Average Number of '1's in Three Key Variables

is caused by X or the complement of X. This ENS word is a "perfect" word to be XOR summed to any word uncorrelated to it. (If the second word is also an ENS, the two would be correlated, and the generated word would, once again, be an ENS.) As the Super Tier word is combined both with the ENS combination of the MAJ and its XORed Image and also XORed to a Lower Feedback word that is predictably neither an ONS[1] nor an ENS; the input to the Intermediate Store & XOR is an unconstrained typically uncorrelated word; and,

- conceptually, we are very dependent on the Super Tier's ENS randomness, as its dispersions appear immediately in every equation in the Data Churn and the Result/Feedback Processor. The Super Tier does not directly affect the unobservable TMB Tiers; which serve as the "secret archive" the ZK Crypt's Data Authentication Sanctum Sanctorum. Two or three of the scribes simultaneously record aberrations from the Message Words; together they release no clues of the TMB tiers.

Note the inherent resilience of the ZK Crypt using the triple locked EVNN RW test syndrome used in the previous "distance" test. We show that the Hybrid Filters "smoothed out" exaggerated input aberrations, so that the final results are unaffected.

As before, each of the four EVNN MAJ control signals affects every fourth output bit in the Hybrid Filter output of the Splash Matrix; there is a strong correlation between every fourth output bit; an average of almost 3/4 of the output bits are of like polarity, and here, with three EVNN signals locked at '1', we will expect an unusually high RW output of the MAJ filter. The following results in Cipher Mode, where the Super Tier Cipher Feedback is the XORed sum of two apparently correlated words (as opposed to the Super Tier MAC feedback is an XOR sum of 3 possibly correlated words). In Cipher mode, the Lower Feedback is non-linear and sparse, with an average of 4 '1's. Thus it is a smaller pool of probable words generating a huge RW.

An all '1' word sequence XORed to a second unbiased word sequence, would simply complement the "good" word, with no deleterious affect. The colored random average of 22 '1's in a word sequence XORed to a slightly colored random second word sequence generated a satisfactory sequence.

Figure 8 shows the average number of ones in three significant internal output words of the five Locked EVNN test sequences.

We see that the suspect output of the Hybrid Filter leaves us with a string with no sensed differentials, despite the fact that the filter is an XORed sum of two operands each with small and strong internal correlations. We see that in all of the relevant 100 sampled tests of 10^7 words each, the number of ones is close to 5 million. The maximum average deviate bit count value is 5,000,316; i.e. relative error is $316/(5 \times 10^6) < 10^{-4}$. Remember, we are only analyzing an infinitesimal part of an assumed 2^{380} binary sequence so that "things happen." Figure 8 graph shows the super-imposed distribution of the average number of '1's in significant words taken from the previous tests, with 3 EVNN inputs locked on '1', and the same test wherein all 4 EVNN inputs are uncorrelated and random. Note that a zero or close to zero fraction cannot be represented on this logarithmic graph.

[1] An ONS is an odd number string, i.e. any string that is never an ENS, or more simply, one which always has an odd number of ones.

Operand	Number of RWs	Sensed Differentials	Constraint & Remarks	Notes
Super Tier nLFSR Pair Out	11,635	None	Un/Locked EVNNs—Cipher & MAC mode	
Top Tier nLFSR Pair Out	11,623	None	Locked Clock & (No Missed Pulse)	a,b
Top Tier nLFSR Pair Out	143,784	None	Std. Irregular Clock & (Random Missed Pulse)	
Mid Tier nLFSR Pair Out	11,643	None	Locked Clock & (No Missed Pulse)	a,b
Mid Tier nLFSR Pair Out	143,805	None	Std. Irregular Clock & (Random Missed Pulse)	
Bot Tier nLFSR Pair Out	11,636	None	Locked Clock & (No Missed Pulse)	a,b
Bot Tier nLFSR Pair Out	143,874	None	Std. Irregular Clock & (Random Missed Pulse)	
Input Top Store & XOR	11,612	None	Cipher Mode—Sparse Lower Feedback	c
Input Top Store & XOR	11,632	None	MAC Mode—Saturated Lower Feedback	c
Top MAJ Filter	4,446,490	e	3 Locked EVNNs—All 3x8 MAJs Usually '1'	f
Top MAJ Filter	1,847,870	None	Strong Correlation Between Every 4th Bit	
Top Hybrid Filter Out	11,747	None	Typical MAJ Correlation	
Top Hybrid Filter Out	12,569	None	3 Locked EVNNs—All 3 MAJs Usually '1'	
Input Intermediate Store & XOR	11,626	None	Cipher Mode—Sparse Lower FB—unsaturated	c
Input Intermediate Store & XOR	11,638	None	MAC Mode—Saturated Lower FB	c
Bottom Hybrid Filter Out	12,577	None	Cipher Mode—No Lower Cipher Feedback	
Bottom Hybrid Filter Out	11,737	None	MAC Mode—No Lower MAC Feedback	
CipherMask—Bottom Store & XOR	11,630	None	All Modes no Lower FB into Bottom Store & XOR	c
CipherMask—Bottom Store & XOR	11,614	None	Result(t) $\oplus$ Result(t+7); distanced	d
CipherMask—Bottom Store & XOR	11,634	None	Result(t) $\oplus$ Result(t+1); juxtaposed	d
Super Tier Feedback	11,640	None	MAC Mode (3 Words Includes Result)	c
Super Tier Feedback	12,784	None	Cipher Mode (2 Close Words Combined)	c
Lower Feedback	11,641	None	MAC Mode (2Prev & Pres ResultsCombined)	c
Lower Feedback	8,207,387	g	Cipher Mode (Sparse Average 4 '1's)	c,g

a	Analytical Tests
b	In the analytic mode, the TMB tiers are clock ticked at every Primary Clock pulse; and the Image is XOR combined to the nLFSR outputs with a probability of about 0.55. There are naturally no Repeated Words. In both Cipher and MAC ZK-Crypt Modes with a probability of about 0.45, 1 of the three TMB Tiersis not clocked. In the normal ZKCrypt, one of the TMB tiers randomly "misses" a clock tick with a probability of about 0.45. The Image is XOR combined to the nLFSR outputs with a probability of about 0.55. When a tier is not clocked, with a probability of about 0.55 the previous word is repeated.
c	Analytic and normal values with small deviations were averaged.
d	The analytic averaged Repeated Words when Result(t) is XOR'ed to Result(t+1) or Result(t+7).
e	3/4 Prob 3/4,
f	footnote missing,
g	All bits are 1 with prob 1/4

Table 1: Repeated Word (RW) Tests: Comparison of Analytical and Standard Modes—Raw Averages

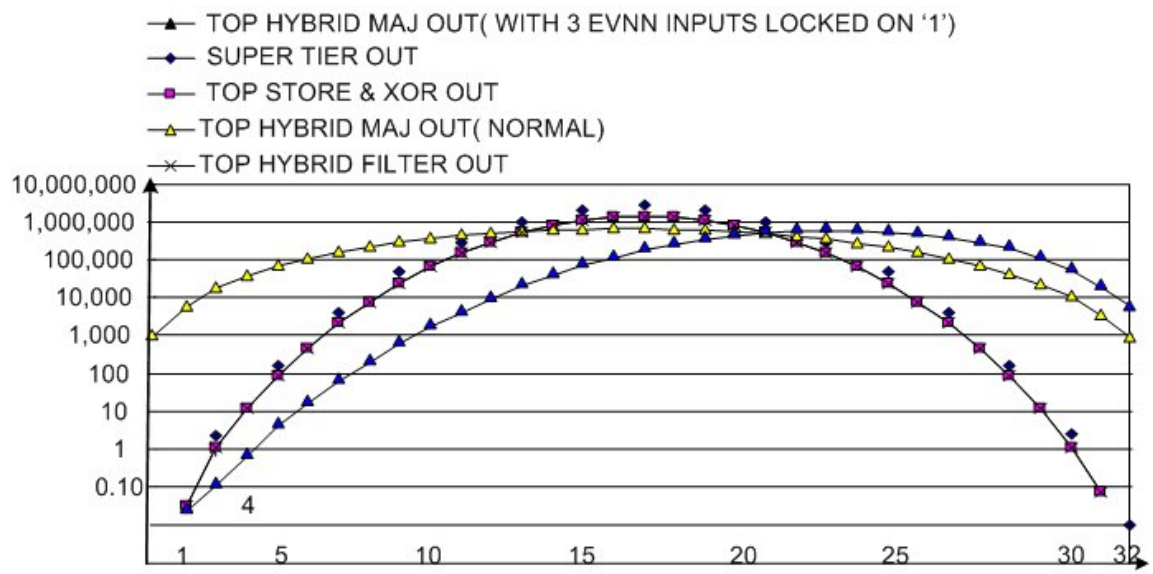

Figure 8: How many words with X Number of '1's in Five Key Variables

The points on the parabolic curves show an average counted number of occurrences of '1's in 10^7 samples, typical of four classes of state variables in the Register Bank and Data Churn, three of which demonstrate a close to perfect distribution of 32-bit numbers. The skewed curve shows the MAJ filter output where 3 of 4 EVNN permutation variables are locked '1'. The Super Tier output, being a linear XOR ENS sum of the outputs of the concatenated "composited" Super Tier nLFSR's and their 7 left rotated Image, checks out nicely with a comparative low Repeated Word sequence. The number of expected Repeated ENS Words is slightly more than 23,200—double the count of an "excellent" unconstrained variable; leading us to assume that the 7 Left Rotated Image successfully distances internally correlated bits in the XOR sum. Therefore, as expected, each Super Tier (ENS) diamond is double the relevant even-indexed value of the Top Store-And-XOR outputs of the underlying square symbol of the Top Store & XOR (and the missing odd diamonds are unseen).

References

[1] Steve Babbage, "Improved 'Exhaustive Search' Attacks on Stream Ciphers," Proc. IEE European Convention on Security and Detection, *IEE Conference publication 408*, pp. 161-165, 55.

[2] G. Bard. "Algorithms for the Solution of Linear and Polynomial Systems of Equations over Finite Fields, with Applications to Cryptanalysis." PhD Thesis. University of Maryland, at College Park, Department of Mathematics. 2007.

[3] Elad Barkan, Eli Biham, and Adi Shamir, "Rigorous Bounds on Cryptanalytic Time/Memory Tradeoffs" Advances in Cryptology—Crypto 2006. 2006.

[4] Daniel Bernstein, "Does ZK-Crypt version 1 flunk a Repetition Test?", eSTREAM Website, 2005.

[5] Norman Biggs. *Algebraic Graph Theory*. Cambridge University Press.

[6] Eli Biham, and Orr Dunkelman, "A Framework for Iterative Hash Functions", Proc. NIST Hash Forum 2006, August, 2006.

[7] Eli Biham, and Orr Dunkelman, "Differential Cryptanalysis in Stream Ciphers," Technion Computer Science Department Technical Report CS-2007-10-2007. 2007.

[8] Alex Biryukov, Sourav Mukhopadhyay, and Palash Sarkar, "Improved Time-Memory Tradeoffs with Multiple Data," Proc. Selected Areas in Cryptography 2005, *Lecture Notes in Computer Science*, 2005.

[9] Claude Carlet. "A Method of Construction of Balanced Functions with Optimum Algebraic Immunity." IACR E-print 2006/149.

[10] Nicolas Courtois, Adi Shamir, Jacques Patarin, and Alexander Klimov. Efficient algorithms for solving overdefined systems of multivariate polynomial equations. In Advances in Cryptology— Proc. of EUROCRYPT, volume 1807 of Lecture Notes in Computer Science, pages 392407. Springer-Verlag, 2000.

[11] Nicolas Courtois, Gregory V. Bard, and Chris Jefferson. "Efficient Methods for Conversion and Solution of Sparse Systems of Low-Degree Multivariate Polynomials over GF(2) via SAT-Solvers", eprint/2007/024/. Also presented at ECRYPT workshop "Tools for Cryptanalysis" in Krakow, 24-25 September.

[12] Thomas Cusick, Cunsheng Ding, and Ari Renvall, North *Stream Ciphers and Number Theory*. Holland Mathematical Library, Elsevier, 2004.

[13] Orr Dunkelman, Carmi Gressel, Avi Hecht, Ran Granot, "The ZK Crypt III Algorithm and Implementation," eSTREAM Website, 2006.

[14] Niels Ferguson, Richard Schroeppel, and Doug Whiting. *A simple algebraic representation of Rijndael*. Draft—technical report. 2001.

[15] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* Series of Books in the Mathematical Sciences. Freeman. 1979.

[16] T. Gendrullis, M. Novotnỳ, and A. Rupp. "A Real-World Attack Breaking A5/1 within Hours." Proc. Cryptographic Hardware and Embedded Systems. (CHES '08). *Lecture Notes in Computer Science*. Vol. 5154. August, 2008.

[17] Carmi Gressel, Gregory Bard, Orr Dunkelman, Avi Hecht, and Ran Granot, Patents-PCT/IL/2007001101, Priority US 60/842612, filed September 9, 2006.

[18] Carmi Gressel, Gregory Bard, Orr Dunkelman, Avi Hecht, and Ran Granot, "ZK Crypt Circuit & Concept Drawings," `www.FortressGB.com` , version 5, December, 2007.

[19] Carmi Gressel, Gregory Bard, Orr Dunkelman, Avi Hecht, and Ran Granot. "The A-Z Guide to the ZK Crypt, An Annotated Glossary," `www.FortressGB.com` , version 5, December, 2007.

[20] Carmi Gressel, Orr Dunkelman, and Avi Hecht, "Understanding the ZK Crypts-Ciphers for (Almost) all Reasons," `www.FortressGB.com` , December 2007.

[21] Carmi Gressel, Avi Hect, and Ran Granot, Patents-WO 2007/049268, Priority US 60/730,555, filed October 2005.

[22] Carmi Gressel, Avi Hecht, Michael Rivkin, and Ran Granot, "Annotated Test Result Reports from 'The ZK Crypt Noise Generator Design Parameter Emulator"' `www.FortressGB.com` December, 2007.

[23] Carmi Gressel, Michael Slobodkin, Roy Krotman, Yonatan Bick, Gabriel Vago, and Mark Fiterman, Patents-US 11/578,909 Priority 60/624,463, filed October 2004.

[24] A. Hecht, G. Bard, O. Dunkelman, and C. Gressel. "The ZK-Crypt II and III Security Analysis." *FortressGB*. 2007.

[25] A. Hecht, G. Bard, O. Dunkelman, C. Gressel, and R. Granot. "The ZK-Crypt Algorithm Specification." *FortressGB*. 2007.

[26] Martin Hellman, "A Cryptanalytic Time-Memory Tradeoff," *IEEE Transactions on Information Theory*, Vol. 26, No. 4, pp. 401–406, 1980.

[27] Alexey Lubkin, Boris Ryabko, "The Distinguishing Attack on the ZK-Crypt Cipher," eSTREAM website, 2005.

[28] G. Marsaglia, "DieHard Tests" as posted on
`ftp://ftp.csis.hku.hk/pub/random/source.2004`

[29] Markku-Juhani Saarinen, "Chosen-IV Statistical Attacks on eSTREAM Stream Ciphers," Proc. Stream Ciphers Revisited SASC 2006.

[30] Werner Schindler, "Efficient Online Tests for True Random Number Generators," Proc. Cryptographic Hardware and Embedded Systems 2001, *Lecture Notes in Computer Science*.

[31] Serge Vaudney, *A Classical Introduction to Cryptography*, Springer, 2006.