

Observations and Statistics of Combining Functions

The input to the Intermediate Store & XOR filter is a debiased string with a very slight residual correlation. The Intermediate Store & XOR filter typically reduces the residual correlation thereby generating a non-distinguishing Repeated Word merit number.

In the following exaggerated examples, we show a variety of cases where biased bits (unbalanced ratio of '1's to '0's) are processed in 3 input MAJ filter and will compare them with the functioning of 2 and 3 input XOR gates, ubiquitous in the ZK-Crypt.

We show why the MAJ gates often amplify imbalance, and when the comparable 3 input XOR gates may (or may not) serve to rectify syndromes caused by non-linearity.

Then, we will show that if only one of the values is biased, the output is immediately unbiased by an XOR of the two samplings. We leave it to the reader to see that if biases are binary mirror symmetric (one bit is heavily biased to '1', and the complement bit is heavily biased to zero), the result is the same as for same positive bias on '1' and similarly negative biased on '0' on the second input. We also leave it to the reader to see how with a "symmetric" imbalance, the MAJ function may do a better job debiasing than the 3 input XOR.

Non-linear components, and in many cases linear components develop syndromes that cause biased bits. Assume for simplicity, as in these embodiments, that two biased random (or pseudorandom) vectors are XORed together and that these biased vectors are then XORed again to other biased (or unbiased vectors). Note that in the following, in all cases, the bias is either theoretically reduced considerably or completely when combined in XOR gates, iff the inputs are not correlated.

Example: Two (0.7 to 0.3) Biased Input Generate a Less Biased Result

IN	PROBABILITIES		XORed to 0	XORed to 1
0⊕0	0.7	0.7	0.49	
0⊕1	0.7	0.3		0.21
1⊕0	0.3	0.7		0.21
1⊕1	0.3	0.3	0.09	

Average XORed output – 58% '0's to 42% '1's, a 60% reduction of bias.
and a second round-

IN	PROBABILITIES		XORed to 0	XORed to 1
0⊕0	0.58	0.58	0.3364	
0⊕1	0.58	0.42		0.2436
1⊕0	0.42	0.58		0.2436
1⊕1	0.42	0.42	0.1764	

with an average XORed output– 51.28% '0's to 48.72% '1's, an 85% bias reduction.

¹ References in Chapter 1



An example that demonstrates our ideal- a biased variable XORed to an unbiased variable produces an unbiased result.

Example of a biased bit XORed to an unbiased bit.

INPUT	PROBABILITY		XORed to 0	XORed to 1
0⊕0	0.7	0.5	0.35	
0⊕1	0.7	0.5		0.35
1⊕0	0.3	0.5		0.15
1⊕1	0.3	0.5	0.15	

Average XORed output bit – 50% '0's to 50% '1's

Proof: For a bias of ϵ , where one polarity, e.g., 0, has a probability of $0.5 + \epsilon$, the complement polarity would then be $0.5 - \epsilon$, where typical $\epsilon \ll 0.5$.

First polarity, e.g., '0', output for $0 \oplus 0$ and $1 \oplus 1$, would be the sum of a) and b):

$$a) (0.5 + \epsilon)(0.5 + \epsilon) = 0.5^2 + \epsilon + \epsilon^2$$

$$b) (0.5 - \epsilon)(0.5 - \epsilon) = 0.5^2 - \epsilon + \epsilon^2$$

with an average bias of $0.5 + 2\epsilon^2$. As $\epsilon \ll 0.5$, $2\epsilon^2 \ll \epsilon$, for $\epsilon = 0.02$ (a huge bias), $2\epsilon^2 = 0.0008 \ll 0.02$. (Note, ϵ is by definition less than 0.5, as $0.5 + 0.5$ defines a probability of one, and there can only be a single polarity, '1' or '0'.)

Now we will compare the results of an exaggerated bias in a MAJ gate (3 input) to the same bias in a 3 input XOR, where all three inputs are heavily biased to zero, as before.

INPUT	PROBABILITIES			MAJ 0	MAJ 1	XOR 0	XOR 1
0 0 0	0.7	0.7	0.7	0.343		0.343	
0 0 1	0.7	0.7	0.3	0.245			0.147
0 1 0	0.7	0.3	0.7	0.105			0.105
0 1 1	0.7	0.3	0.3		0.063	0.105	
1 0 0	0.3	0.7	0.7	0.105			0.105
1 0 1	0.3	0.7	0.3		0.105	0.105	
1 1 0	0.3	0.3	0.7		0.405	0.405	
1 1 1	0.3	0.3	0.3		0.405		0.405

With grossly disparate results: 78.0% 21.6% 53.2% 46.8%

The Majority function exaggerated the bias, whereas the 3 XOR function improved the results, as expected.

A more interesting example occurs when one input is unbiased, and two other inputs are grossly biased – assuming that there is no correlation between the unbiased bit and the two biased bits.

INPUT	PROBABILITIES			MAJ 0	MAJ 1	XOR 0	XOR 1
0 0 0	0.7	0.7	0.5	0.245		0.245	
0 0 1	0.7	0.7	0.5	0.245			0.245
0 1 0	0.7	0.3	0.5	0.105			0.105



0 1 1	0.7	0.3	0.5		0.105	0.105	
1 0 0	0.3	0.7	0.5	0.105			0.105
1 0 1	0.3	0.7	0.5		0.105	0.105	
1 1 0	0.3	0.3	0.5		0.045	0.045	
1 1 1	0.3	0.3	0.5		0.045		0.045
				70%	30%	50%	50%

This seems to say, that no matter how many variables are XOR combined, the result will be as unbiased as the least biased, whereas the MAJ function will never remove bias from the same combination. Neither conjecture is true.

See what happens to the MAJ if the bias to '1' is 0.7, 0.3, and 0.5 to the three input bits, respectively.

Correlated unbiased inputs into XOR functions will typically produce outputs which are strongly biased;

e.g., 01010101 XORed to 01010101 will result in an all zero string.

In the next section we will observe the statistical efficacy of the three types of filtering media; i.e., the MAJ gate filter, the 3XOR gate filter, and the stalwart Store & XOR.

Bias in Strings with Low Correlation

A first case in point, in the noise generator, [zl-ccc Fig. N04], QTA, is the LS cell of an up-counter which is randomly (a function of data bits in the Register Bank) set to a new up-count which may be even or odd generating a ...1010101011011001....type sequence, where the bold literal was the LS preset bit on an up-counter. Note that there is a natural prevalence of '1's, as a count sequence always ends with a '1', but may start with either a '1' or a '0'. This series has extreme correlation, as there are short sequences with same probabilities – if it's a '1', most probably the next bit will be a '0'; conversely a '0' is with a high probability followed by a '1'. As we are reasonably sure that there cannot be a strong cross correlation between the primitive nLFSRs in the Middle and Bottom Control Units in the Random Controller [zk-ccc Figs. 14 and 15xx] and the nLFSR in the noise generator and the pseudo random data bits affecting the toggling of the QTA signal; we prove to have found a good way to inject entropy into a pseudo random bit stream, despite the fact the QTA signal, in this case has about 9 '0's for every 10 '1's; and the signal derived from the nLFSRs is also at best locally biased and full of recurrent periodic correlations, so that we can assume that the QTA signals reduce bias and increase cryptocomplexity of the deterministic Random Controller output. The differential statistics of the forced MAJ filter output demonstrates that the 4th Toggle has no sensible bias. Note that the 0.53 bias of '1's when XOR summed to an uncorrelated pseudo random word is irrelevant, owing to the very low bias, and to the function which will typically minimize bias.

Forced Correlation in Maj Filters Hides Values in the Data Churn

A second far more vital revelation, is the affects of the 4 random EVNN permutation control signals from the Random Controller regulating the Hybrid MAJ filter. If the input bit to a MAJ function gate is 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 fairly random. But as each EVNN control signal 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 will be of like polarity, and these words would be repeated more frequently in the 10M samplings. The summary in Appendix A shows that in such a case there are almost 2M Repeated Words, but no differentials.

Natural & Forced Correlations in MAJ Functions vs Ordered Linear Combining Functions

We will observe the statistical efficacy of the three types of filtering media; i.e., the MAJ gate filter, the 3XOR gate filter, and the classic stalwart Store & XOR (a present result is XOR summed with a previous result.).

In the figures we show the correlating structure of the MAJ filter in the Data Churn. Each of the four EVNN bits are unbiased, emanating from the noise source in the Random Controller, so that the output of the MAJ filter is statistically close to unbiased, and the internal correlations are obviously dissimilar to the output of the Splash Matrix.

Security against Chosen-IVs

The ZK-Crypt is designed to accept variable length IVs, following a 32 MAC mode clocked scramble digest of the loaded secret key. The additional 16 cycle scramble following the MAC mode loading of the IV causes each bit of the IV to affect virtually all bits of the internal state. This strategy increases the security of the cipher against chosen IV attacks, as well as drastically reducing the possibility of discovering a weak key. We can say that we have made this into a one-way function.

DPA and Side Channel

Side Channel Attacks

The ZK-Crypt design is immune to power attacks, (no S-Boxes or look up tables), and is protected against power attacks with a current compensating strategy which maintains two distinct ranges of current compensation.

Algebraic Cryptanalytic Attacks

Algebraic Cryptanalytic Attacks

Using algebraic modelling methods from [13], Bard has analyzed the Data Churn alone, without its 64 bits of feedback and found it to have an over 50,000 monomial count. We estimate that the Register Bank will have a monomial count of over 70,000; while the affects of the Message, the Result/Feedback Process and the 4 distinct forms of Dual Tracks of orthogonal feedback have not been taken in account. A count of 6500 monomials has been shown to be intractable in the coming decades. An exposition of the methodology on the ZK-Crypt architecture is being submitted to SASC 2008, and will be archived.

BIAS ATTACKS

Differential Cryptanalytic Attack

[14] shows samples of the importance of "immunizing" stream cipher designs from differentials, wherein a distinguishable differential may be used as a valuable "hook" for an adversarial attack. The state variables in the ZK-Crypt naturally and provably have no sensed binary differentials. A simple and important test for differentials is to count the number of '1's of each indexed bit of a test word in a large number of test words. This test, however is not sufficient to discover internal correlations between bits in a string.

A successful Repeated Word test result, where the number of repeated words is "close to" the number of words estimated by a Poisson calculation of a perfect random sequence of numbers, shows both the absence of differentials and an amplified estimate of the trace correlation between words in the sequence.

We use the Repeated Word test to assess the word state variables in the 32 Bit Word Manipulator, and a Chi Square test [7] to estimate the entropy generated by the ZK-Crypt deterministic/random noise generator.

CORRELATION BIAS

However, we show cases where we purposefully generate two logic unobservable strings each of which has a disparate internal correlation; so that the XOR summed results will have eliminated differentials, and reduced state variable correlations.

This XORing of two internally correlated variable strategy was used in randomizing the deterministic noise source in the Random Controller. In the Control Units there are randomly preset short cycle (between 4 and 12 counts, randomly starting with an even or odd number) counters. The LS flip-flop is toggled at each (P)Random Clock, except for when the pseudorandom clocked up-count presets at 15. The parity of the presets, and the up-count preset start value are affected directly by highly diffused bits from the Register Bank. However, despite the fact, that with strong probability, polarity is changed at each Primary Clock cycle; e.g., most of the time a '1' is followed by a '0', (extreme correlation between successive bits), the statistics of the output of XORing these bits to internal permutation regulators has markedly improved the random statistics (see [zk-ccc fig N04] and a typical noise generator report in Appendix F from [17]).xxx

The second importance issue is the distributions of '0's and '1's in the state words of the Data Churn.

As is shown in the Splash Hybrid Filters, if any one of the EVNN signal inputs to the filter is '1', the relevant 8 MAJ gates is output is '1' with a probability of 3/4; conversely if the input is '0'. If the EVNN signals are not biased, then the output of the the 32 MAJ gates has no sensed differential in any of the 32 output bits, but there is a strong correlation between any bit and another bit both four places "down or up the line". Also, if the permutation bits show strings with a dearth of single literals (few 0101s and many 111000s) then there is a strong correlation of bits $X_{(i,t)}$ and $X_{(i+1,t)}$ in the output of the MAJ filter, which would not be detected by simply "counting" the number of '1's. This correlation is most probably sensed in the Repeated Word test. The results have shown as expected, measured by repeated words; if two biased uncorrelated words are XOR combined, the degree of inner correlation of the output word is statistically less correlated than the lesser correlated of the two input words with internal correlation, and as we have shown, the bias is typically reduced.

Precluding Differentials

The nature of the ZK-Crypt 32 bit Word Manipulator precludes differentials, as the 7 32 bit Word states in the Register Bank and Data Churn outputs are right and left rotating rings of binary variables; accepting inputs from other rings dependent on right and left rotations and pseudo random displacements. We have shown that there is no appreciable bias in any binary state variable in the 32 bit Word Manipulator; proven by counting the number of '1's in hundreds of 10M samplings; averaging the results and calculating the standard deviation.

The architecture has proved to be amazingly robust. Not only are the variables in the Register Bank unobservable; but we have forced unreasonable differentials in the TMB tiers, which caused large differentials in

3 Tier Combiner output without seriously degrading the Register Bank Output; as there can be no correlation with the Super Tier's excellent statistical output. Conceptually, we consider the TMB Tiers to be the "sanctus sanctorum", the unobservable final repository of all aberrations; they record all, and reveal virtually nothing. In addition, whether the random enacting Tier Clocks, the Brown Image Controls or the Right and Left Slip permutation controls are locked or operative, there is no sensed affect on the 4 Tier Register Bank Combiner, we estimate because of the uncorrelated quality output of the Super Tier.

We have checked the affect caused by Message bits on any internal differentials. We ran the same Repeated Word sets of 10M samples, wherein, in each, a different (1 of 32) single '1' (with 31 '0's) Message Word is the input. We wanted to ascertain if there could be an asymmetrical affect causing a differential on one or more internal binary variables in either the Register Bank or the Data Churn. This only strengthens trust in the massive diffusions [Appendix B], and our observations of algebraic complexity.

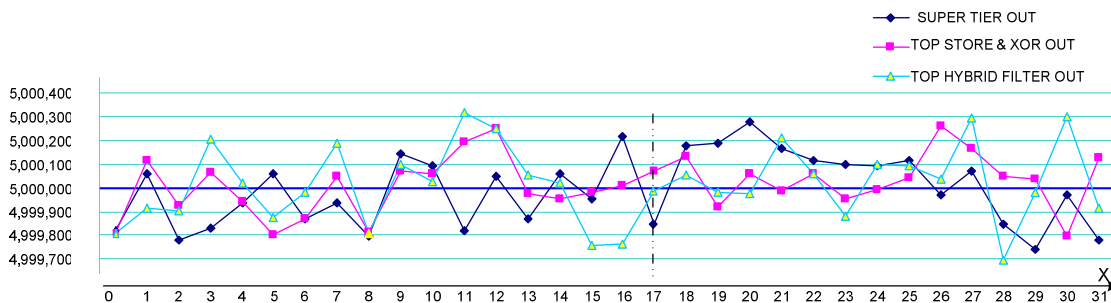
Either the Super Tier output or a dispersion of the output appears linearly in the immediate output equations of all 10 32 bit state variables of the 32 bit Data Manipulator. The Super Tier output is composed: of linear feedback combinations of loosely correlated internal words (and an additional permuted Result word in MAC mode); of the present pseudo random shifting state of the two 16 bit linear FSRs XORed to the internal One to Many internal feedback of the LFSRs; in the case of MAC data authentication, of the output of the Synch Counter; which are all XOR summed together in the nLFSRs' flip flop outputs; wherein said output is 7 left shifted into an Image output which is XORed to the the same nLFSR output.

The Cipher mode Super Tier Feedback shows signs of correlation in its inputs (see the Repeated Word results in Appendix A).

We see that the outputs of the Super Tier's nLFSRs, and the Super Tier's output (undiscerningly alike for Cipher and MAC mode), give excellent statistical results when measured by DieHard, Repeated Word, and bit counts.

The Super Tier output is the output of its two nLFSRs XORed to the 7 left rotated outputs of said nLFSRs. Such a result typically lowers any long term residual bias. The Repeated output Word of the Super Tier is an ENS, even number string word; i.e., there will always be an even number of '1's and '0's, and any output is therefore one of 2^{31} possible numbers (not one of 2^{32} unconstrained numbers). The reduced number of possible word combinations does not designate a loss of unpredictability, as we cannot know if the concatenated Super Tier's nLFSR words' input was X or the complement (inverse) of X. This aberration is detected (see the Stream Cipher Imaged output) by the Repeated word test, as the pool of words is halved, causing the number of repeated words to be doubled.

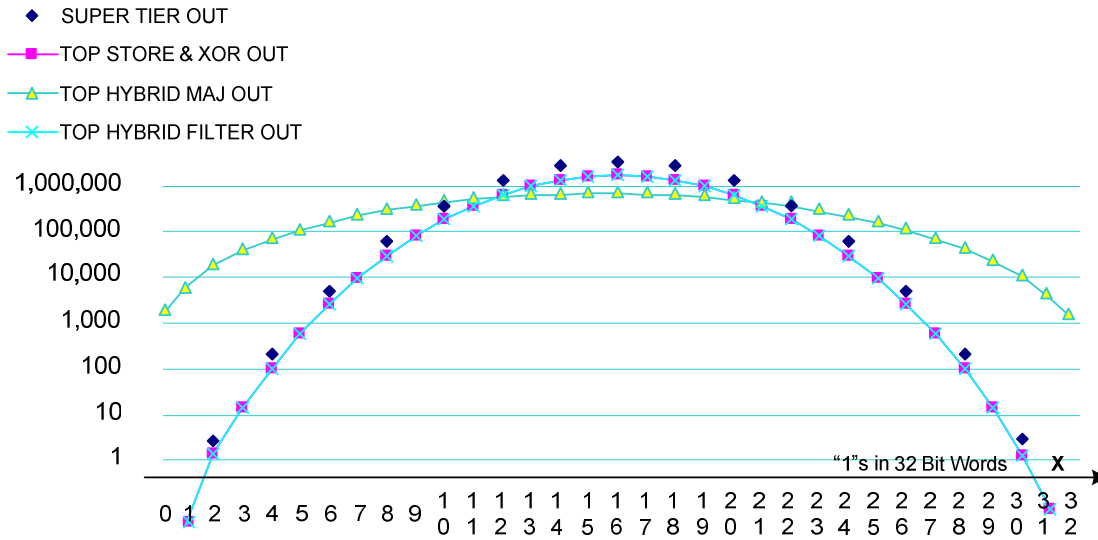
The following graph shows the result of a search for a sense of binary differentials on three words of particular interest. If a Splash output filter consists of the XORed sum of every two adjacent bits, the result shows a higher degree of correlation, see $H_{(i,t)} \oplus H_{(i+1,t)}$, Fig. 21HYBF in Appendix A. We intended to present explicit results at SASC2008.



The Average Number of "1"s in Index Position X in 100 Samplings of 10M Rounds;
 e.g., in Index Position 17, there were an Average of 4,999,847 "1"s in Output of the Super Tier;
 in Index Position 17, there were an Average of 5,000,072 "1"s in Output of the Top Store & XOR; and,
 in Index Position 17, there were an Average of 4,999,985 "1"s in Output of the Hybrid MAJ/3XOR Filter.

Visibly, 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 strong internal correlations. We see that in all of the relevant 100 10M sampled tests, the number of ones is close to 5M. The maximum average

deviation bit count value is 5,000,316; e.g., $316/(5 \times 10^6)$ which is less than 10^{-4} . Remember, we are only analyzing an infinitesimal part of an assumed 2^{379} binary sequence so that "things happen".



The Average Number of Words with X "1"s in Samplings of 10M Rounds
 e.g., there were 1,317,375 Words with 17 "1"s in the FFs of the TOP STORE & XOR;
 and 689754 Words with 17 "1"s in the Output of the Top MAJ Gate in the Hybrid Filter;
 and 0 words with 17 "1"s in the Output of the Super Tier (an EVNN Vector)

The graph shows the distribution average of the "counted" number of occurrences of '1's in any one 32 bit word in 10 M samples. Note that in a logarithmic scale, number 0 is way down beyond nowhere, explaining the buried diamonds from the ENS (even number string) Super Tier.

The points on the parabolic curves show an average counted number of occurrences of '1's in 10M 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 Super Tier output, being an XOR sum of the outputs of the concatenated "composed" Super Tier nLFSRs and their 7 left rotated Image, checks out nicely with a comparative low Repeated Word sequence. The number of expected Repeated Words is slightly 23,200- double of an 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 close to double the value of the underlying square symbol of the Top Store & XOR.

As a result of the biased sets forced by the 4 EVNNs, see Fig. 17TSM in Appendix A; 'many '0' and many '1' output words are hardly rare in the output of the MAJ filter in the Top Hybrid Filter, as seen in the present graph.

Debiasing, Diffusing, Delinearizing and Decorrelating in One Filter

The following figure typifies 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. The 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.

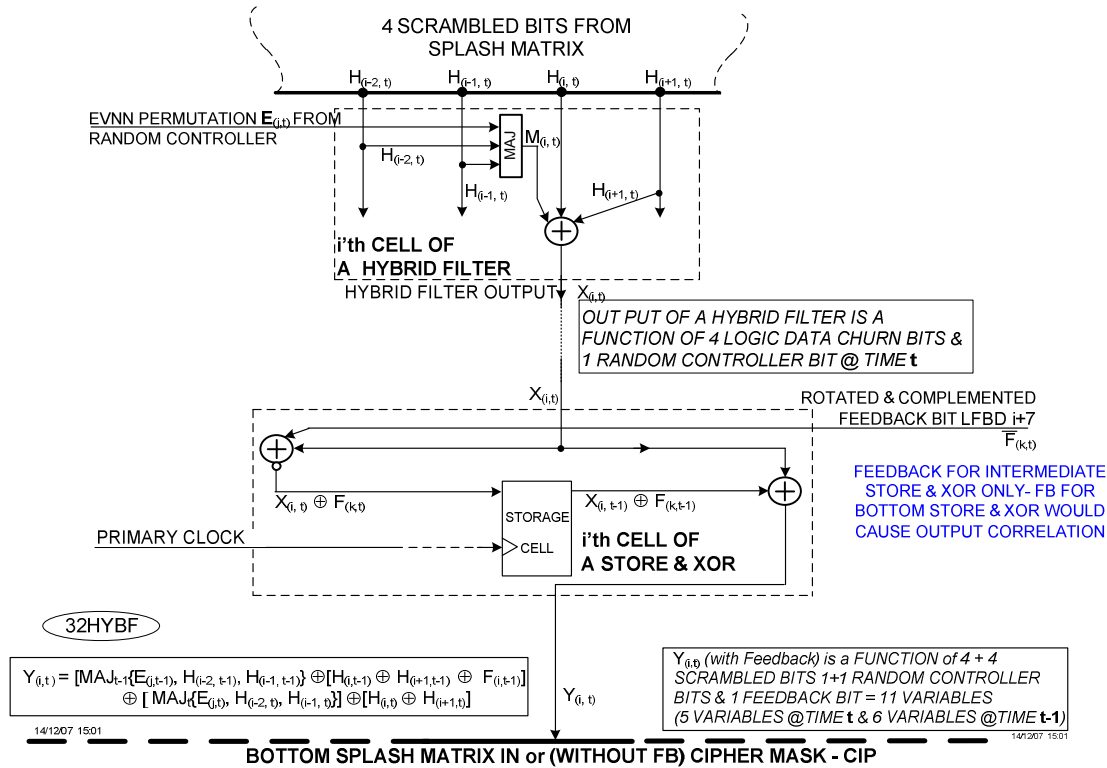


Fig. - Combining, Diffusing, Correlation Immunizing & Debiasing in the Data Churn

The Store & XOR cell serves as a classical correlation immunizer, while it doubles the number of variables in the output equations. The feedback bit, $F_{(k,t)}$, is a function of the "present output", so that it is stored to be output as $F_{(k,t-1)}$ on the next clock. In our analyses in the next section, we only "count" the diffusive variables appearing during a present clock cycle (1st degree), not the cumulative effect (higher degrees) from all of the previous clock cycles. We will see that two clock cycles "down", every memory variable in the Word Manipulator's equations carries a "history" of all the previous variable values, starting from the cipher initialization. Note that $Y_{(i,t)}$ is a function of 11 variables.

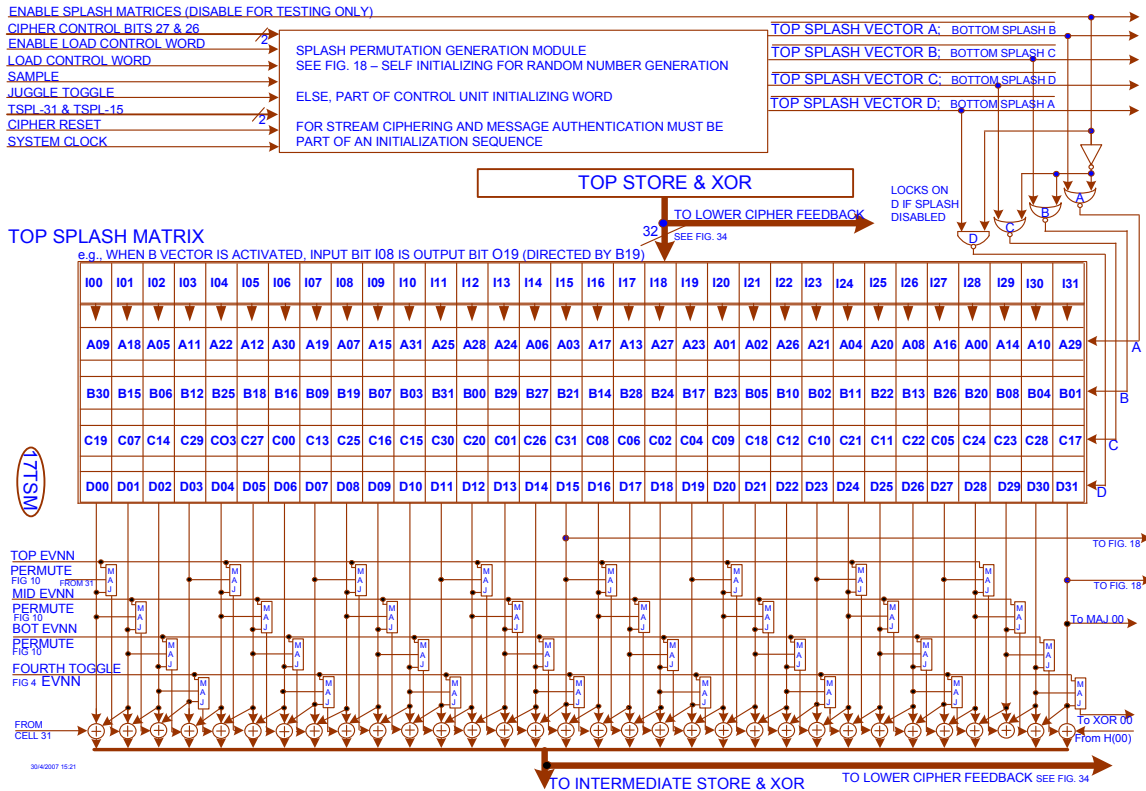


Fig. Hybrid Filter - Each EVNN_j Forces a Same Value Output on every 4th X_i with Prob = 3/4
 In Figure 1 each of the EVNN signals strongly biases every fourth MAJ gate output. The output of the MAJ filters therefore is an unbiased auto-correlated string.

The XORed sum of two near Splash Matrix neighbor bits inputs an ENS string (an even number of '1's and '0's in a word with an even number of bits) into the Hybrid Filter.

Diffusion – Correlation Immunity – Non-Linearity – Quality and Quantity

In a well designed compact hardware solution modules with the highest level of diffusion, non-linearity and correlation immunity devices are most advantageous building blocks. A simple Measure of non-linearity is trivial; count the MAJ gates and the random permutation signals. Maintaining the quality of non-linearity is often illusive, and a cause of distinguishing features and biased output. A characteristic of all functions in the Word Manipulator is their "circularity" – with no end effects, typically seen in normal and modular multiplication.

MAC Mode Security Features in Cipher and TRNG

At the initial stages of developing the ZK-Crypt a chosen word attack was suggested against the MAC mode, see Appendix C. This has subsequently shown to be a false alarm, but prompted the designers to introduce the basis for the present ZK-Crypt with a Dual Track feedback mechanism. In short time the enhancement generated better statistics, proof of the inability of an attacker to gain partial control of the Register Bank, quick proof of resistance against Message Modification, providing the back bone for precluding collisions with the 56 bit HAIFA counter.

In earlier ZK-Crypt developments, ultimate protection against fraudulent word attack relied on the non-linear complexity of the internal state, and the fact that the six feedback streams were an XORed combination of a Present and a Previous Message affected Result. Once changed, there is no rectification or reconciliation, as the Result Store keeps a record of all fraudulent Message words, used to rectify a first

untrue Message Word; assuming that the attacker must maintain valid feedback to sustain any reconciled condition..

Changing of one plaintext bit can affect up to 170 bits in the internal state. The problem an attacker faces when he (or she) tries to complement one (or several) bits in a ciphertext word is the fact that the effects of his (or her) actions are completely unpredictable. This follows from the fact that any change in the internal state keeps affecting the machine in the following rounds, and due to the nonlinear nature of the effect, predicting these effects cannot be done with probability larger than random.

We note that it is easy to see that changing only one word of ciphertext cannot lead to the same internal state. Thus, each attempted attack must use at least two ciphertext words. As we noted before, due to the nonlinear nature of the development of the effects of the previous message word on the internal state, it is impossible to predict the effect, and thus, to find the correct sequence of words that compensate for the first change.

The Dual Track feedback is used in all functions. Secret Cipher and MAC keys which are longer than 128 bits are "digested" using the MAC mode [Ref 9 Fig. 7], (the first 128 bits are loaded directly into the Top, Middle and Bottom Tiers). It is used in the MAC function, for which the Dual Track was designed, to preclude "Fraudulent Word" insertion. Conversely, the added randomizing functions have shortened the TRNG random initialization process.xxx

The ZK-Crypt III - MAC mode incorporates two types of feedback. The simple FB transfers the whole incoming word combined with the ciphers resultant mask and the previous word back into the cipher. The complex FB transposes the individual nibbles in the simple FB and combines them with the Super Tier's complex FB. The combination of the dual feedbacks ensures that tailored words sent into the MAC will affect, in a single round, many neighbouring bits both near and far making the calculation of the tailored false feedback impossible. See explanations and explicit proofs in Appendices C & D.

Figure 2: Hybrid Filter with Store & XOR Decorrelator Processes a Diffused Unbiased Output

We are reasonably sure that successive word outputs of the hybrid filter are not highly correlated, and that the Store & XOR filter significantly reduces autocorrelation in the output.

The following figures include demonstrations and proofs of Hybrid Filtering which is the backbone of the robustness of the 32 bit Register Bank and the Data Churn data manipulators. The guiding design concept was to assure massive diffusion and highest quality randomness in all ten 32 bit state variable buffers. In each of the three main functions, three 32 bit words are combined in a 2 of 3 Majority filter. We have taken adequate steps to assure that the worst case output is loosely correlated to two of the input words. The intermediate result is further decorrelated in a Store & XOR filter, (the classic Present Result XOR summed to the Previous Result).

For the demonstration, we randomly chose the same source vector B in the examples in Figures 3, 6, 7 and 8 to be a concatenation two pseudorandom 32 bit sequences. The A vector in all examples is an XOR sum of 2 adjacent bits of B; i.e., $A_i = B_i \text{ XOR } B_{i+1}$.

Figure 3: An Example of String Extension Generated by a MAJ Filter on a 64 bit Word

Figure 3 typifies the structure of the MAJ Filter in the 3 Tier TMB Combiner of the Register Bank. The source Vector B is a concatenation of two 32 bit pseudorandom sequences. The 3 vector inputs are the three tiers' outputs, each of which is generated by mutually exclusive functions, where we can assume that adjacent bits are unpredictable.

Vector C, the MAJ output is composed of an average string length of 4. (The average string length in the pseudorandom source Vector B is 2).

Vector E typifies the imaging of internal correlated functions, in that bias and internal correlations are typically increased when a MAU output is rotated and XOR summed to itself. In the case of the four tier

combiner, the output of the Super Tier, a high quality which is XOR summed to Vector E, is of highest quality, assuring a reasonable output from the Register Bank into the Top Store & XOR.

Figure 4: String Extensions Generated by MAJ Filters-Probable Outputs with Random Inputs

Figures 4 and 5 show the probability estimates of string extensions, e.g., the probability of the next bit being a '1', if the present bit is a '1'. Note, the statistics assume six bit core inputs, where, in each case the MS and LS bit of the 6 bit word are not previously defined, and we judge the likelihoods of each of the six bit outputs, where left (right) most output bit is the largely affected by the MS and LS bits of the core nibble. We are interested in the included string lengths of the inputs compared to the string lengths of the outputs. Note that the statistics for a nibble are identical to both the mirror and the complement of the nibble; e.g., '0001' → S₁ statistics are similar to '0111' → S₇, '1000' → S₈, and '1110' → S₁₄. The following should explain the symbols and statistical measures.

Assume that we want to define the MAJ function outputs C_i for input bits B_i i= 0 to 5 for $\emptyset \underline{0} \underline{0} \underline{1} \underline{1} \emptyset$ where there are only four possible B inputs- $\underline{0} \underline{0} \underline{1} \underline{1} 0$, $\underline{0} \underline{0} \underline{1} \underline{1} \underline{1}$, $1 \underline{0} \underline{0} \underline{1} \underline{1} 0$, $1 \underline{0} \underline{0} \underline{1} \underline{1} \underline{1}$; note that the included strings are underlined, and that the average total included string length is 5. A string is defined as groups of same literal pairs or longer; i.e., no single '1's or '0's in a string.

If C_i = MAJ(B_{i-1}, B_i, B_{i+1}), and B = $\emptyset \underline{0} \underline{0} \underline{1} \underline{1} \emptyset$ then we know for certain that bits C₁, C₂, C₃ and C₄ respectively are 0011, as in all cases at least two of 3 test bits are single value. However, all we know of C₀, for example is that that one input bit, B₁, to the MAJ function is 0. If one bit is known, 0 in this instance, we know that in ³/₄ of the cases the combination of B₀ and B₋₁ will include at least one '0'; e.g., 00, 01, and 10 have a '0' only 11 could affect a '1' MAJ output. We embolden and underline the C₀ '0' in the output string; i.e., as in $\underline{\underline{0}} \underline{0} \underline{0} \underline{1} \underline{1} \underline{0}$. In this instant, the right hand 0 is less likely, i.e., Prob (C₅ = 0) = ¹/₄.

The probability of combination $\underline{\underline{0}} \underline{0} \underline{0} \underline{1} \underline{1} \underline{0}$ is therefore ³/₄ x 1 x ¹/₄ = ³/₁₆, and the included string length is 5 (the C₅ is a lone literal). By such measure, $\underline{\underline{0}} \underline{0} \underline{0} \underline{1} \underline{1} \underline{\underline{1}}$ has a likelihood of ³/₄ x 1 x ³/₄ = ⁹/₁₆, and its included string length = 5. The nibble core, 0011, is defined with a probability of 1.

Figure 5: String Extensions Generated by MAJ Filters – with Weighted Analysis

The weighted probabilities for all 64 six bit inputs, leads to a probability of extending a string is 72.229%, whereas in the example in Figure 3, we found 75% string extension. **Elena, Avi is corroborating the results with lengthy tests.**

Figure 6: EVNN Vector (1100) Generates Internal Unrelated Correlation to a Random Word

In Figures 6, 7 and 8 we use the source vectors A and B, as in the first randomly chosen test, which seemed to corroborate with our estimation statistics on six bit literals. Here vector B', a nibble EVNN input into the MAJ filter forces the C' output vectors to have strong nibble autocorrelation, which in practices leads to miniscule bias, as the EVNN inputs are unbiased. There are four types of rings, each leading to identical statistics, 0000, 0001, 0011, 0101; e.g., in a ring, ...00110011001100011... nibbles can be sampled as 0011, 0110, 1001, or 1100. EVNN Vector (0011) led to an extension E factor of 0.125; a probability that the next bit in a sequence will be the same value, with a probability of 0.625.

Figure 7: EVNN Vectors (0001 & 0100) Generate Unrelated Correlations to Random Words

EVNN Vector types (0001) and (0100) led to extension E factors of 0.267 and 0.0 respectively.

Figure 8: EVNN Vector (0000) Correlations & Weighted Results

EVNN Vector types (0000) and (1111) led to an extension E factors of 0.25.

The weighted average of all four vector type showed an E factor estimate of 0.206. We can assume from this weighted assessment that EVNN type autocorrelation slightly lowers the string extension factor.

Figure 9: Can Pure EVNN Vectors Reduce Correlations with Random Vectors?

This experiment seemed to show that randomly concatenating EVNN vectors would generate a slightly reduced ϵ factor.

We might only assume from this experiment is that it corroborates assumption from previous experiments, that the EVNN signal input into the MAJ function reduces the string extension factor.

Figure 10: The Efficacy of Exclusive ORing of Statistically Uncorrelated Binary Strings

Simply stated, this shows why linear uncorrelated combinations in random number generation produce good statistical results.

Figure 11: The Efficacy of MAJ Filtering Wherein 1 of 3 Inputs is Unbiased

Herein we show the probability of each of 8 input combinations occurring, where two input bits are biased by ϵ_1 and ϵ_2 , and the third bit is unbiased.

Then we show that the MAJ output is biased by $\epsilon_1 + \epsilon_2$, and that if they polarity of one is reversed, the bias is less than $|\epsilon_1| + |\epsilon_2|$.

Figure 12: The Ideal Result of 3XOR Filtering Wherein 1 of 3 Inputs is Not Biased

This result of MAJ efficacy on bias in Figure 11, is to be compared to the 3XOR output with the same inputs where the uncorrelated unbiased third input eliminated 3XOR output bias.

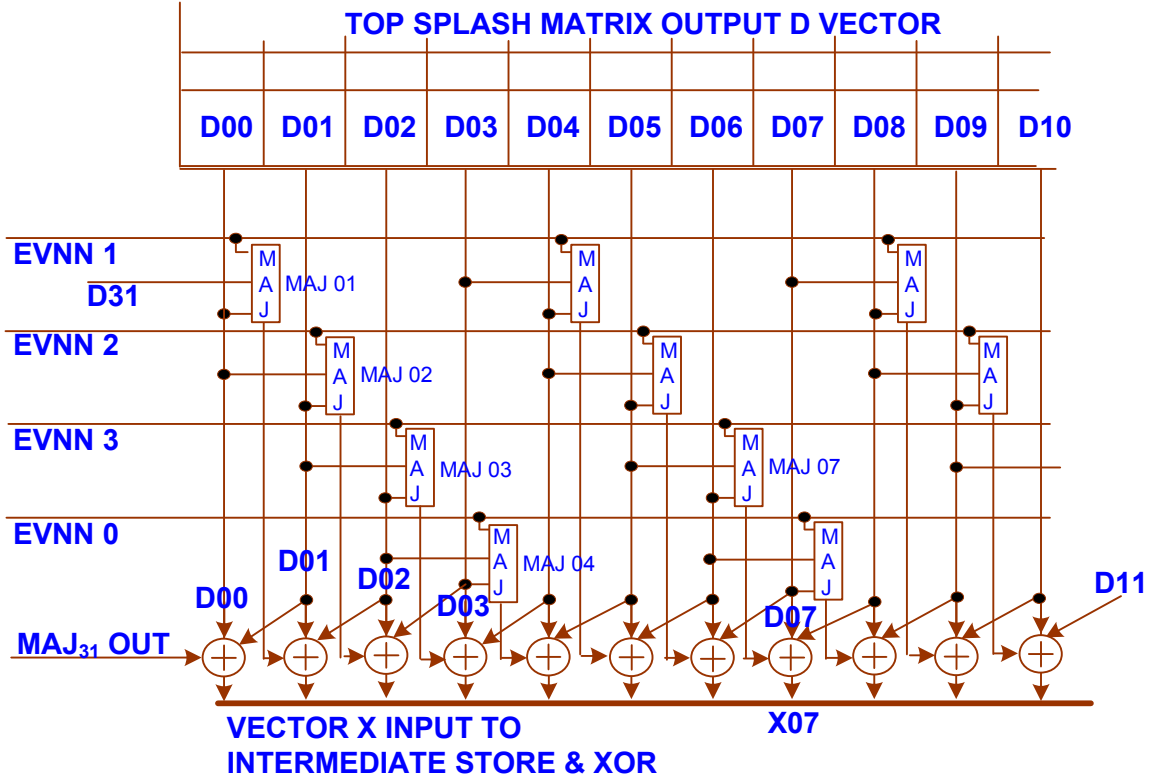
Figure 13: Comparing Results of 3XOR & MAJ Filtering Wherein All 3 Inputs are Biased

Here we assume that three equally biased bits; $0.5 + \epsilon$, are input into 3XOR and 2 of 3 MAJ gates.

The MAJ bias is equal to $1.5 \epsilon - 2 \epsilon^2$; e.g., a large input bias of 0.2 would generate an output bias of $0.3 - 0.016 = 0.284$ – thereby amplifying the input bias.

The 3XOR output bias equal to $4 \epsilon^3$; e.g., a large input bias of 0.2 would generate an output bias of 0.032 – thereby reducing the input bias.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX



$$MAJ_{07} \text{ OUT} = MAJ[(EVNN 7 \bmod 4), D05, D06]$$

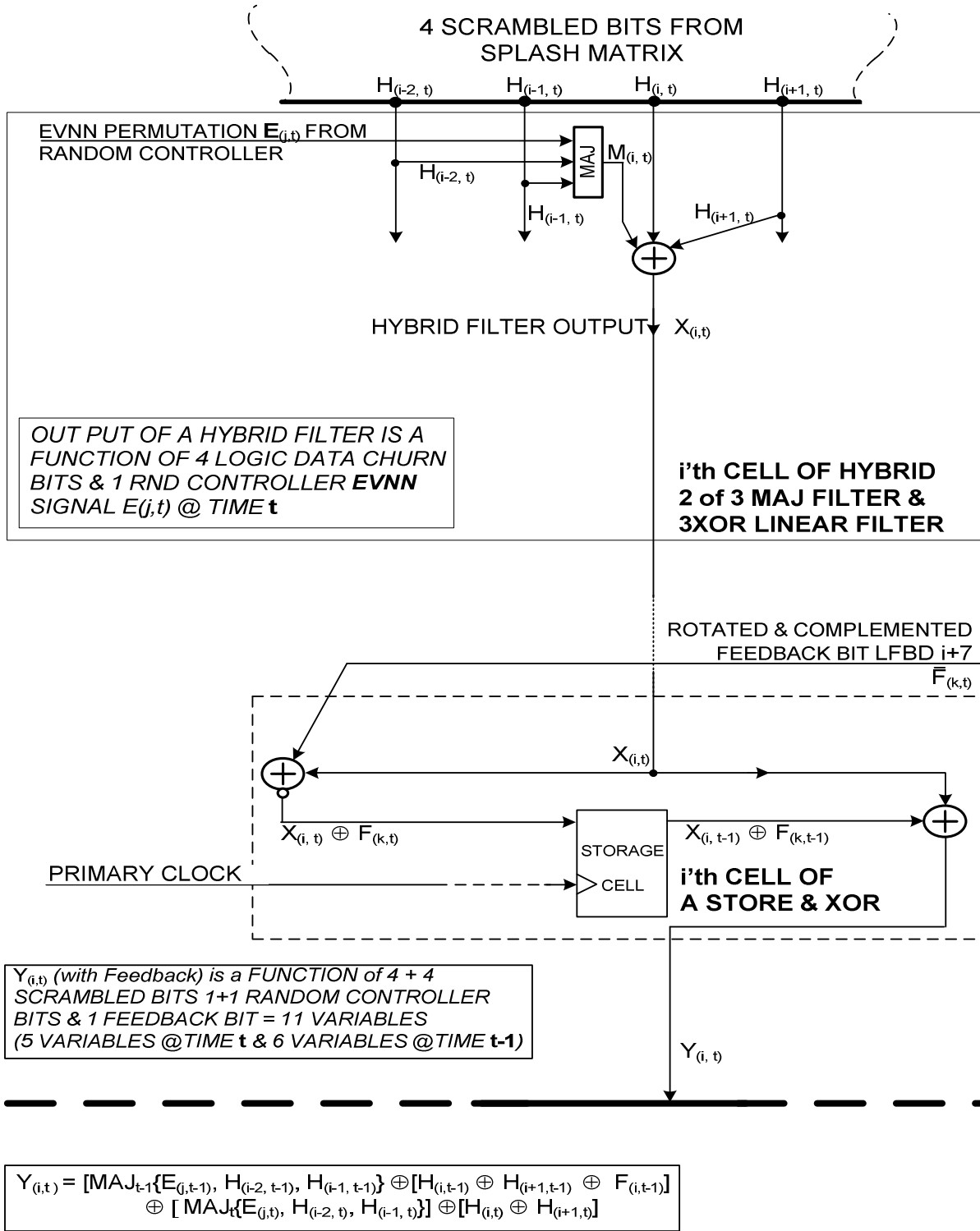
$$X07 = MAJ[EVNN 3, D05, D06] \oplus D07 \oplus D08$$

SEE FIGURE 2

EVNNFORC

15/5/2008 16:09

Fig. : Hybrid Filter - Each EVNN_j Forces a Same Value Output on every 4th X_i with Prob = 3/4



15/05/08 14:30

HYBFILT

Fig Hybrid Filter with Store & XOR and Lower Feedback

XXXXXXXXXXXXXXXXXXXX

Only S_5 & $S_{10} \rightarrow 192/64$ literal extensions.

Input $S_5 =$	\emptyset 0 1 0 1 \emptyset Aver stringed input literals = 2.0	Probabilities	
		Prob(y_i)	Prob(i'th extension over single pair)
Output $-y_0 =$	0 0 0 1 0 0	$1/4$	Prob(y_i , extension) = $1 \times (5-2)/4 \rightarrow 3$
Output $-y_1 =$	0 0 0 1 1 1	$1/4$	Prob(y_i , extension) = $1 \times 4/4 \rightarrow 4$
Output $-y_2 =$	1 1 0 1 0 0	$1/4$	Prob(y_i , extension) = $1 \times 2/4 \rightarrow 2$
Output $-y_3 =$	1 1 0 1 1 1	$1/4$	Prob(y_i , extension) = $1 \times 3/4 \rightarrow 3$

There are only four possible inputs- $001010, 001011, 101010, 101011$; $(2+4+0+2)/4$ - therefore for the four MAJ $\emptyset 0101\emptyset$ inputs there is an average of only 1 included double literal; eg, 00, 11; on the average of the said 6 bit MAJ outputs there is an average extension of $12/4 = 192/64$ literals.

Only S_6 & $S_9 \rightarrow 128/64$ literal extensions.

Input $S_6 =$	\emptyset 0 1 1 0 \emptyset Aver stringed input literals = 4.0	Probabilities	
		Prob(y_i)	Prob(i'th extension over 4 bits of literal string)
Output $-y_0 =$	0 0 1 1 0 0	$1/4$	Prob(y_i , extension) = $1 \times 2/4 \rightarrow 2.0$
Output $-y_1 =$	0 0 1 1 1 1	$1/4$	Prob(y_i , extension) = $1 \times 2/4 \rightarrow 2.0$
Output $-y_2 =$	1 1 1 1 0 0	$1/4$	Prob(y_i , extension) = $1 \times 2/4 \rightarrow 2.0$
Output $-y_3 =$	1 1 1 1 1 1	$1/4$	Prob(y_i , extension) = $1 \times 2/4 \rightarrow 2.0$

There are only 4 possible inputs- $001100, 101100, 001101, 101101$; $(6+4+4+2)/4$ - therefore for the 4 MAJ $\emptyset 0111\emptyset$ inputs there is an average of $16/4 = 4$ single polarity bit strings; see above. on the average of these 6 bit MAJ outputs there is an average extension of $8/4 = 128/64$ literals.

S_7 (S_8 & S_{14}) $\rightarrow 80/64$ literal extensions. Statistics like S_1 on previous page.

Input $S_7 =$	\emptyset 0 1 1 1 \emptyset Aver stringed input literals = 4.5	Probabilities	
		Prob(y_i)	Prob(i'th extension over 4.5 bits of literal string)
Output $-y_0 =$	0 0 1 1 1 0	$1/2 \times 1/4 = 1/8$	Prob(y_i , extension) = $1 \times 0.5/8 \rightarrow 0.5$
Output $-y_1 =$	0 0 1 1 1 1	$1/2 \times 3/4 = 3/8$	Prob(y_i , extension) = $3 \times 1.5/8 \rightarrow 4.5$
Output $-y_2 =$	1 1 1 1 1 0	$1/2 \times 1/4 = 1/8$	Prob(y_i , extension) = $1 \times 0.5/8 \rightarrow 0.5$
Output $-y_3 =$	1 1 1 1 1 1	$3/4 \times 1/2 = 3/8$	Prob(y_i , extension) = $3 \times 1.5/8 \rightarrow 4.5$

There are only 4 possible inputs- $001110, 001111, 101110, 101111$; $(5+6+3+4)/4$ - therefore for the 4 MAJ $\emptyset 0111\emptyset$ inputs there is an average of $18/4 = 4.5$ single polarity bit strings; see above. on the average of these 6 bit MAJ outputs there is an average extension of $10/8 = 80/64$ literals.

Input complements &/or mirror images have same statistics, eg $S_1, S_7, S_8, \& S_{14}$. See S_7 .

Aver Extension for $-S_0$ & $S_{15} = \frac{32}{64} \rightarrow 2 \times \frac{1}{2} = \frac{16}{16} \rightarrow 1$
 Aver Extension for $-S_1, S_7, S_8, \& S_{14} = \frac{80}{64} \rightarrow 4 \times \frac{5}{4} = \frac{80}{16} \rightarrow 5$
 Aver Extension for $-S_2, S_4, S_{11} \& S_{13} = \frac{80}{64} \rightarrow 4 \times \frac{5}{4} = \frac{80}{16} \rightarrow 5$
 Aver Extension for $-S_3$ & $S_{12} = \frac{32}{64} \rightarrow 2 \times \frac{1}{2} = \frac{16}{16} \rightarrow 1$
 Aver Extension for $-S_5$ & $S_{10} = \frac{192}{64} \rightarrow 2 \times 3 = \frac{96}{16} \rightarrow 6$
 Aver Extension for $-S_6$ & $S_9 = \frac{128}{64} \rightarrow 2 \times 2 = \frac{64}{16} \rightarrow 4$

MAJSTAT2

13/05/08 15:30

22/16 = 1.375 aver output string extension for 6 bit inputs into 2 of 3 MAJ Gates

1.375/6 = 0.229 the average string extension for each output bit -

∴ If a given output bit is a '1'/0', disregarding all previous output bits,

on an average of 0.5 + 0.229 ≈ 73% of all occurrences, the next bit will also be a '1'/0'.

The chosen random sample in DISPRS1 generated an average extension of 0.25, reasonably close to the estimated 22.9% extension.

Fig: String Extensions Generated by MAJ Filters – with Weighted Analysis

THE EVNN VECTOR HYBRID FILTERS BETWEEN THE SPLASH MATRIX & THE OUTPUT STORE & XORs

```

A → 010100100001011111011001110000100111110111000010101101000011001001
B → 0000011010100100010111100010101101000001100100111101110001101110
B' → 110011001100110011001100110011001100110011001100110011001100110011
C' → 0000110011001100110011000001101110000001100100111101110000111111
D' → 0101111011011011000001111100111110111101110100001011010110100010110110
  
```

VECTOR **A** – XORed ADJACENT BITS OF **B** (SAME AS PREVIOUS) $A_{i:7} = B_i \oplus B_{i+1}$
 VECTOR **B** – RND CONCATENATED SECTIONS OF 32 BIT nLFSRs (SAME AS PREVIOUS SOURCE)
 VECTOR **B'** – EVNN VECTOR IS COMPOSED OF A REPEATED RANDOM NIBBLE eg...0001 0001 0001
 THIS FORCES THE **C'** VECTOR INTO A STRUCTURE THAT IS NOT STATISTICALLY CORRELATED TO A TRUE RANDOM DISTRIBUTION
 VECTOR **C'** – 2 of 3 MAJ FUNC ON 2 ADJACENT BITS OF **B** & 1 BIT OF **B'** $C'_{i:7} = \text{MAJ}(B'_{i:7}, B_{i:7}, B_{i+8})$
 VECTOR **D'** – 7 RIGHT SHIFTED **A** XORed TO VECTOR **C** $D'_{i:7} = A_i \oplus C'_{i:7}$

EVNN VECTOR TYPE - ...0110011001100110... SEE NIBBLES S₃ S₆ S₉ S₁₂

```

>>> A → 11 11 2 1 4 11 5 1 2 2 3 4 1 2 5 1 3 4 1 111 2 11 4 2 2 1 1
010100100001011111011001110000100111110111000010101101000011001001
Ai:7} = Bi} ⊕ Bi+1}

B → 5 2 1 1 1 1 2 1 3 1 1 5 3 1 1 1 1 2 1 1 5 2 2 1 2 5 1 3 3 2 1 3 1
000001101010010001011111000101011010000011001001111101110001101110
B' → 110011001100110011001100110011001100110011001100110011001100110011

C' → 4 2 2 2 2 2 2 2 1 4 5 2 1 3 6 2 2 1 2 4 1 3 4 5
000011001100110011011110000011011100000011001001111011100001111111
>>> A → 010100100001011111011001110000100111110111000010101101000011001001
C'_{i:7} = MAJ(B'_{i:7}, B_{i:7}, B_{i+8})

D' → 1 1 1 4 1 2 1 2 1 2 5 5 2 5 1 4 1 1 4 1 1 2 1 1 1 2 1 1 3 1 1 2 1 1
010111101101101100000111110011111011110100001011010110100010110110
D'_{i:7} = Ai} ⊕ C'_{i:7}
  
```

VECTOR A 18 – 1 SINGLE (0,1) LITERALS 7 – 2 PAIRS OF 1s OR 0s 2- 3 4- 4 2- 5 33 '1's & 31 '0's 33 POLARITY FLIPS in 64 BITS →PROB(A _{i} = A_{i-1}}) ≈ 0.484}	VECTOR B (SOURCE-SPLASH MATRIX OUT) 16 – 1 8- 2 4- 3 4- 5 PAIRWISE XORing B to A SLIGHTLY 31 '1's & 33 '0's 32 POLARITY FLIPS in 64 BITS →PROB(B _{i} = B_{i-1}}) = 0.5}	VECTOR B' (RND NIBBLE EVNN) IN THIS CASE 32 – 2 (PAIRS OF 1s OR 0s) 32 '1's & 32 '0's 32 POLARITY FLIPS in 64 BITS →PROB(B'_{i} = B'_{i-1}) = 0.5
--	---	--

VECTOR C' (MAJ OUT) 4 – 1 11 – 2 2 – 3 4 – 4 2 – 5 1 – 6 32 '1's 32 '0's NOT BIASED 24 POLARITY FLIPS IN 64 BITS →PROB(C'_{i} ≠ C'_{i-1}) = $2^4/64 = 0.375$ →PROB(C'_{i} = C'_{i-1}) ≈ 1-0.375 = 0.5+0.125	VECTOR C' SEMBLANCE TO B' 5 NIBBLES 4 HITS 31% vs EST 31.6% 5 " 3 5 " 2 1 " 1 71% HITS vs EST 75%
--	--

VECTOR **D'** (C' XOR A)
20 – 1
7 – 2
1 – 3
3 – 4
3 – 5
∴ BIAS IS INCREASED
36 '1's & 28 '0's
34 POLARITY FLIPS in 64 BITS
→PROB(D'_{i} ≠ D'_{i-1}) ≈ $3^4/64 = 0.531$
→PROB(D'_{i} = D'_{i-1}) ≈ 1-0.531 = 0.5 - 0.031

13/05/08 15:34 **DISPRS2A**

EVNN VECTOR TYPE - ...0110011001100110... PROB(S₅ EXTENSION FACTOR) = 0.125

Figure 6: EVNN Vector (1100) Generates Internal Unrelated Correlation to a Random Word

XXXXXXXXXXXXXXXXXXXX

THE HYBRID FILTER BETWEEN THE SPLASH MATRIX AND THE STORE & XORs - CONTINUED

VECTOR A – XORed ADJACENT BITS OF B (SAME AS PREVIOUS) $A_{i+7} = B_i \oplus B_{i+1}$
 VECTOR B – RND CONCATENATED SECTIONS OF 32 BIT nLFSRs (SAME AS PREVIOUS SOURCE)
 VECTOR B' – EVNN VECTOR IS COMPOSED OF A REPEATED RANDOM NIBBLE eg...0001 0001 0001
 THIS FORCES THE C' VECTOR INTO A STRUCTURE THAT IS NOT STATISTICALLY CORRELATED TO A TRUE RANDOM DISTRIBUTION
 VECTOR C' – 2 of 3 MAJ FUNC ON 2 ADJACENT BITS OF B & 1 BIT OF B' $C'_{i+7} = \text{MAJ}(B'_{i+7}, B_{i+7}, B_{i+8})$
 VECTOR D' – 7 RIGHT SHIFTED A XORed TO VECTOR C $D'_{i+7} = A_i \oplus C'_{i+7}$

EVNN VECTOR TYPE - ...000000000...111111111... SEE EVNN NIBBLES S₀ & S₁₅

```

11 11 2 1 4 1 1 5 1 2 2 3 4 1 2 5 1 3 4 1 111 2 11 4 2 2 1 1
>>A → 010100100001011111011001110000100111110111000010101101000011001001
Ai+7 = Bi ⊕ Bi+1
5 2 1 1 1 1 2 1 3 1 1 5 3 1 1 1 1 2 1 1 5 2 2 1 2 5 1 3 3 2 1 2
B → 00000110101001000101111100010101101000001100100111110001101110
B' → 0000000000000000000000000000000000000000000000000000000000000000
5 1 13 4 8 1 8 1 6 4 2 2 4 1 2 2
C' → 00000100000000000000111100000000100000000100000011110011000010011
>>A → 010100100001011111011001110000100111110111000010101101000011001001
C'_{i+7} = MAJ(B'_{i+7}, B_{i+7}, B_{i+8})
1 1 1 1 1 2 4 1 1 5 3 5 4 2 1 5 1 1 1 1 4 2 1 1 1 1 2 1 3 1 4 1
D' → 010101100001011111000111110000110111101010000110101001000100001
D'_{i+7} = Ai ⊕ C'_{i+7}
    
```

VECTOR C' (MAJ OUT)		VECTOR C' SEMBLANCE TO B'
4-1		6 NIBBLES 4 HITS 37.5% vs EST 31.6%
4-2		6 " 3
3-4		2 " 2
1-5		2 " 1
1-6		
2-8		
1-13		
16 '1's & 48 '0's - STRONGLY BIASED		75% HITS vs EST 75%
16 POLARITY FLIPS IN 64 BITS		
→PROB(C _i ≠ C _{i+1}) ≈ 1 ¹⁶ / ₆₄ = 0.25		
→PROB(C _i = C _{i+1}) ≈ 1-0.25 = 0.5+0.25		

VECTOR D' (C' XOR A)
 19-1
 4-2
 2-3
 4-4
 3-5
 31 '1's & 33 '0's - NOT BIASED
 32 POLARITY FLIPS IN 64 BITS
 →PROB(D_i = D_{i+1}) = 0.5

EVNN VECTOR TYPE - ...000000000...111111111... PROB(S₀ EXTENSION FACTOR) = 0.25

ACCRUED STATISTICS FROM THE 16 EVNN VECTOR MAJ OUTPUT

THE VECTOR EXTENSION STATISTICS

TYPE	INCLUDES EVNN NIBBLE VECTORS	# of VECTORS	EXPANSION FACTOR - ε	WEIGHTED PROBABILITY
S ₀	S ₀ S ₁₅	2	0.25	2 X 0.250 = 0.50
S ₁	S ₁ S ₂ S ₄ S ₇ S ₈ S ₁₁ S ₁₃ S ₁₄	8	0.217	8 X 0.287 = 2.30
S ₃	S ₃ S ₆ S ₉ S ₁₂	4	0.125	4 X 0.125 = 0.50
S ₅	S ₅ S ₁₀	2	0.000	2 X 0.000 = 0.000

THE AVERAGE PROB (ALL EVNN VECTOR EXTENSION) = 3³/₁₆ = 0.206

LITERAL OCCURRENCES [O] IN ALL 64 BIT VECTORS

# LITS	S ₀ LITS x 2[O]	S ₁ LITS x 8[O]	S ₃ LITS x 4[O]	S ₅ LITS x 2[O]	TOTAL [O] EACH LIT	NORMALIZED [O] EACH LIT	# LITS
1	4 X 2 = 8	6 X 8 = 48	4 X 4 = 16	20 X 2 = 40	8+48+16+40= 112	7	1
2	4 X 2 = 8	4 X 8 = 32	11 X 4 = 44		8+32+44=84	5.25	2
3		5 X 8 = 40	2 X 4 = 8	8 X 2 = 16	40+8+16=64	4	3
4	3 X 2 = 6		4 X 4 = 16		6+16=22	1.375	4
5	1 X 2 = 2	2 X 8 = 16	2 X 4 = 8	4 X 2 = 8	2+16+8+8=34	2.125	5
6	1 X 2 = 2		1 X 4 = 4		2+4=6	0.375	6
7		2 X 8 = 16			16	1	7
8	2 X 2 = 4				4	0.25	8
9							9
11		1 X 8 = 8			8	0.50	11
13	1 X 2 = 2				2	0.125	13

13/05/08 15:41

DISPRS2C

Fig : EVNN Vector (0000) Correlations & Weighted Results

DOES THE EVNN VECTOR REDUCE THE AVERAGE MAJ STRING EXTENSION?

A maximum length pseudo random sequence of 2^n bits ($n > 5$) includes:

- $n/4$ single literals ('1's or '0's) strings,
- $n/8$ double same literal strings, e.g. 00 or 11
- $n/16$ triple literal strings;

.....
 double less than n bit same literal compensating string(s)
 2ⁿ n bit literal strings.

there are exactly the same number of '1's and '0's in the 2^n sequence.

Ex: A $2^7 = 128$ BIT MAX LENGTH LFSR OUTPUT
 THERE ARE:

- 32 - 1 SINGLE BIT LITERAL STRINGS 0 or 1
- 16 - 2 BIT LITERAL STRINGS
- 8 - 3 BIT LITERAL STRINGS
- 4 - 4 BIT LITERAL STRINGS
- 2 - 5 BIT LITERAL STRINGS
- 2 - 7 BIT LITERAL STRINGS

IF THE 128 LFSR STRINGS ARE
 CONCATENATED INFINITELY, IN EACH 128
 SEGMENT, WE SEE-

- 64 POLARITY FLIPS in IN EACH 128 BITS
- PROB ($D_i \neq D_{i-1}$) = $64/128 = 0.5$
- PROB ($D_i = D_{i-1}$) = $1 - 0.5 = 0.5$ WITH NO BIAS

There are four EVNN variables, each of which is input into every fourth MAJ gate output of the Splash Matrix. See Figs. 16 and 32 in the Circuit and Concept Manual.

Each bit of the EVNN repeated nibble vector is combined to 2 adjacent output bits from the Splash Matrix by a MAJ gate. In about $3/4$ of the instances, the output of the MAJ gate will be equal to the value of the respective EVNN vector bit. Therefore, on the average $(3/4)^4$ or about 32% of the time, the output MAJ nibbles are equal to the input EVNN nibbles; the output word has a semblance to the EVNN word.

We will make an estimate by replicating and concatenating each of the 16 EVNN vectors into a 64 bit ring to mitigate end effect.

If we examine each of the rings, we note that we have four sets of statistical observations-

A Rings) The ident rings, i.e., ...0000 0000 0000... and ...1111 1111 1111..., relevant to **2** nibbles S_0 and S_{15} .
 Each 64 bit string occurs once (1 [O]) in 2 vectors

B Rings) A lone '0' ('1' with 3 '1's ('0's) ...001000100010001 ... & ...1101110111011101...
 relevant to **8** nibbles $S_1 S_2 S_4 S_7 S_8 S_{11} S_{13}$ and S_{14} .

C Rings) The double '0'/double '1' oscillator, i.e., ...011001100110011... relevant to **4** nibbles $S_3 S_6 S_9$ and S_{12} .
 Each 2 bit string occurs paired (2 times) - (16 x 2)[O] in 4 vectors

D Rings) The single '1'/single '0' oscillator, i.e., ---01010101010101... relevant to **2** nibbles S_5 and S_{10} .
 A 1 bit literal appears 64 [O] times in each of the 2 vectors

Number of Literals	A Ring	B Ring	C Ring	D Ring	Sum of all Occurrences	Normalized to 64 Bit Vectors
1L		8 [S _i] x 16[O] 128[O]		2[S _i]x(32x2[O]) 128[O]	2x128=256[O]x1[L]	16[O]→ 16Bits
2L			4[S _i] x (16x2[O]) 128[O]		128[O]x2[L]	8[O]→ 16Bits
3L		8 [S _i] x 16[O] 128[O]			128[O]x3[L]	8[O]→ 24 Bits
64L	2 [S _i] x 1[O] 2[O]				2[O]x64[L]	* ² / ₁₆ [O]x64[L]=8

NORMALIZED 64 BIT EVNN VECTOR
 16 - 1 SINGLE (0,1) LITERALS
 8 - 2 PAIRS OF 1s OR 0s
 8 - 3
²/₁₆ - 8 BIT SINGLE LITERAL STRING
 32 '1's & 32 '0's
 32.125 POLARITY FLIPS in 64 BIT WORD

13/05/08 15:44

MAJSTAT3

→ PROB($C_i \neq C_{i-1}$) = $32.125/64 = 0.502 = 0.5 + 0.02$

→ PROB($C_i = C_{i-1}$) = $1 - 0.502 = 0.498 = 0.5 - 0.02$ - PROBABLE AVER NEGATIVE, -0.02, STRING EXTENDER
 ∴ EVNN NIBBLES PROBABLY LOWER THE AVER. (0.299) RANDOM SPLASH MAJ STRING EXTENSIONS.

Fig : Can Pure EVNN Vectors Reduce Correlations with Random Vectors?

XOR GATE & MAJ GATE COMPARATIVE EFFICACY FOR REMOVING UNCORRELATED BIAS

Correlation between two input words or internal correlation of one of two words can lead to unexpected results which typically amplify cross and auto correlation, respectively. While developing the ZK-Crypt design, we have been careful to exhaustively test the absence of internal and cross correlations of state variables in the Data Manipulator using controlled Repeated Word tests;

e.g., locking out Missing Clock controls on the TMB Tiers, which would have registered maximum correlation between two juxtaposed words whilst a tier is stalled.

An example of maximizing correlation resulting from exclusive OR summation of two 6 bit strings:

(010101) XOR (101010) generates (111111).

MAJ Gate Filters increase correlation between adjacent bits in a string, as was demonstrated in previous figures.

The following examples in this section either assume that operands are not correlated, or demonstrate, using randomly activated EVNN vectors, a method to force the output of a MAJ filter to be internally correlated, prior to XOR summation with words generated from the same source. The output of such loosely correlated combinations is further de-correlated in a Store & XOR filter. (See the section on the ZK-Crypt.)

PROBABILITY OF A BINARY SIGNAL OCCURRING WHEREIN TWO OF 3 INPUTS ARE BIASED

INPUT nmc	PROB(XOR INS)		PROB (XOR OUTPUT = 0)	PROB (XOR OUTPUT = 1)
	n	m		
00	$0.5 + \epsilon_1$	$0.5 + \epsilon_2$	$0.25 + 0.5 \epsilon_1 + 0.5 \epsilon_2 + \epsilon_1 \epsilon_2$	
01	$0.5 + \epsilon_1$	$0.5 - \epsilon_2$		$0.25 + 0.5 \epsilon_1 + 0.5 \epsilon_2 - \epsilon_1 \epsilon_2$
10	$0.5 - \epsilon_1$	$0.5 + \epsilon_2$		$0.25 - 0.5 \epsilon_1 + 0.5 \epsilon_2 - \epsilon_1 \epsilon_2$
11	$0.5 - \epsilon_1$	$0.5 - \epsilon_2$	$0.25 - 0.5 \epsilon_1 - 0.5 \epsilon_2 + \epsilon_1 \epsilon_2$	

SUMMATED- $0.5 + 0.0 \epsilon_1 + 0.0 \epsilon_2 + 2 \epsilon_1 \epsilon_2$ $0.5 + 0.0 \epsilon_1 + 0.0 \epsilon_2 - 2 \epsilon_1 \epsilon_2$

GIVEN: PROB (INPUT n or m = 0) = $(0.5 + \epsilon_1)$ or $(0.5 + \epsilon_2)$

E.G., FOR 10, $(0.5 - \epsilon_1)(0.5 + \epsilon_2) = 0.5^2 - 0.5 \epsilon_1 + 0.5 \epsilon_2 - \epsilon_1 \epsilon_2$

IF ϵ_1 or $\epsilon_2 = 0$, (NO BIAS); THE OUTPUT PROBABILITY FOR UNCORRELATED (n XOR m) IS 0.

IN ALL CASES THE MAXIMUM BIAS IS $2 \epsilon_1 \epsilon_2$, ALWAYS SMALLER THAN EITHER, AS BIAS IS BY DEFINITION LESS THAN 0.5.

EXHAUSTIVE TESTING HAS SHOWN THAT NO BINARY STATE VARIABLE IN THE DATA MANIPULATOR WAS BIASED MORE THAN $1/(0.36 \times 10^4)$, WHERE WE ARBITRARILY SET 10^{-4} AS THE MAXIMUM ALLOWABLE AVERAGE BIAS ON ANY "EQUIPROBABLE" STATE VARIABLE.

ASSUME $\epsilon_1 = \epsilon_2 = 10^{-3}$ THEN $2 \epsilon_1 \epsilon_2 = 2 \times 10^{-6} < 10^{-4}$

16/05/08 14:16

2XOR2B

Fig : The Efficacy of Exclusive ORing of Statistically Uncorrelated Binary Strings

EFFICACY OF XOR GATES & 2 of 3 MAJ GATES FOR REMOVING BIAS- CONTINUED

PROBABILITY OF A BINARY SIGNAL OCCURRING WHEREIN TWO OF 3 INPUTS ARE BIASED

INPUT nmc	PROBABILITIES OF EACH LITERAL			PROBABILITY OF EACH OCCURRENCE
	n	m	c	
000	$0.5 + \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
001	$0.5 + \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
010	$0.5 + \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
011	$0.5 + \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
100	$0.5 - \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
101	$0.5 - \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
110	$0.5 - \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
111	$0.5 - \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$

GIVEN: PROB (INPUT n or m = 0) = $(0.5 + \epsilon_1)$ or $(0.5 + \epsilon_2)$ RESPECTIVELY.

E.G., FOR 101, $(0.5 - \epsilon_1)(0.5 + \epsilon_2)(0.5) = 0.5(0.25 + 0.5 \epsilon_1 + 0.5 \epsilon_2 - \epsilon_1 \epsilon_2) =$
 $= 0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$

THE BIASED OUTPUT OF A 2 of 3 MAJ GATE, WHEREIN 2 OF THE INPUTS ARE BIASED

INPUT nmc	PROB (2 of 3 MAJ GATE OUTPUT = 0)	PROB (2 of 3 MAJ GATE OUTPUT = 1)
000	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$	
001	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$	
010	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$	
011		$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
100	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$	
101		$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
110		$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
111		$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
SUMMATED-	$0.5 + 0.50 \epsilon_1 + 0.50 \epsilon_2 + 0.0 \epsilon_1 \epsilon_2$	$0.5 - 0.50 \epsilon_1 - 0.50 \epsilon_2 + 0.0 \epsilon_1 \epsilon_2$

∴ IF $\epsilon_1 = \epsilon_2$ THE BIAS IS DOUBLED; IF HOWEVER, IN A RARE CASE -

$|\epsilon_1| = |\epsilon_2|$ AND WHERE ONE IS POSITIVE & THE OTHER NEGATIVE; THE BIAS IS REMOVED.

21/05/08 14:21

MAJ1G2B

Fig : The Efficacy of MAJ Filtering Wherein 1 of 3 Inputs is Unbiased

EFFICACY OF XOR GATES & 2 of 3 MAJORITY GATES FOR REMOVING BIAS- CONTINUED

PROBABILITY OF A BINARY SIGNAL OCCURRING WHEREIN TWO OF 3 INPUTS ARE BIASED

INPUT nmc	PROBABILITIES OF EACH LITERAL			PROBABILITY OF EACH OCCURRENCE
	n	m	c	
000	$0.5 + \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
001	$0.5 + \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
010	$0.5 + \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
011	$0.5 + \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
100	$0.5 - \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
101	$0.5 - \epsilon_1$	$0.5 + \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
110	$0.5 - \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
111	$0.5 - \epsilon_1$	$0.5 - \epsilon_2$	0.5	$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$

GIVEN: PROB (INPUT n or m = 0) = $(0.5 + \epsilon_1)$ or $(0.5 + \epsilon_2)$

E.G., FOR 101, $(0.5 - \epsilon_1)(0.5 + \epsilon_2) (0.5) = 0.5(0.25 + 0.5 \epsilon_1 + 0.5 \epsilon_2 - \epsilon_1 \epsilon_2) =$
 $= 0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$

THE REMOVAL OF BIAS WHEN 2 SIGNALS ARE BIASED & A 3RD IS NOT CORRELATED

INPUT nmc	PROB (3 IN XOR GATE OUTPUT = 0)	PROB(3 IN XOR GATE OUTPUT = 1)
000	$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$	
001		$0.125 + 0.25 \epsilon_1 + 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$
010		$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
011	$0.125 + 0.25 \epsilon_1 - 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$	
100		$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$
101	$0.125 - 0.25 \epsilon_1 + 0.25 \epsilon_2 - 0.5 \epsilon_1 \epsilon_2$	
110	$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$	
111		$0.125 - 0.25 \epsilon_1 - 0.25 \epsilon_2 + 0.5 \epsilon_1 \epsilon_2$

SUMMATED- $0.5 + 0.0 \epsilon_1 + 0.0 \epsilon_2 + 0.0 \epsilon_1 \epsilon_2$ $0.5 + 0.0 \epsilon_1 + 0.0 \epsilon_2 + 0.0 \epsilon_1 \epsilon_2$

IF ONE OR MORE INPUTS OF A 3XOR GATE HAS A 0.0ϵ BIAS THE OUTPUT IS UNBIASED.

WE ASSUME THAT NEITHER OF THE BIASED SIGNALS IS STATISTICALLY CORRELATED TO THE UNBIASED SIGNAL. (PROVABLE WITH 32 BIT REPEATED WORD TYPE TESTS; E.G. MISSING CLOCKS CAUSE MISLEADING REPEATED WORDS).

∴ THE 3 XOR GATE SUCCESSFULLY REMOVES BIAS IF 1 OF 3 INPUTS IS UNBIASED AND IS STATISTICALLY NOT CORRELATED TO THE OTHER INPUT SIGNAL.

CONCLUSIONS- ASSUMING UNCORRELATED INPUTS, THE 3 INPUT XOR GATE IS A "PERFECT" STATISTICAL REMOVER OF BIAS.

IT IS EXTREMELY UNLIKELY THAT A 2 of 3 MAJ GATE WILL SUPPRESS
 LOCALLY BIASED INPUT SIGNALS.

13/05/08 15:57

3XOR1G2B

Fig : The Ideal Result of 3XOR Filtering Wherein 1 of 3 Inputs is Not Biased

EFFICACY OF XOR GATES & 2 of 3 MAJ GATES FOR REMOVING BIAS- CONTINUED

BINARY OUTPUT PROBABILITIES WHEREIN ALL INPUTS ARE EQUALLY BIASED

INPUT nmp	PROBABILITIES OF EACH LITERAL			PROBABILITY OF EACH OCCURRENCE
	n	m	p	
000	$0.5 + \epsilon$	$0.5 + \epsilon$	$0.5 + \epsilon$	$0.125 + 0.75\epsilon + 1.5\epsilon^2 + \epsilon^3$
001	$0.5 + \epsilon$	$0.5 + \epsilon$	$0.5 - \epsilon$	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
010	$0.5 + \epsilon$	$0.5 - \epsilon$	$0.5 + \epsilon$	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
011	$0.5 + \epsilon$	$0.5 - \epsilon$	$0.5 - \epsilon$	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
100	$0.5 - \epsilon$	$0.5 + \epsilon$	$0.5 + \epsilon$	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
101	$0.5 - \epsilon$	$0.5 + \epsilon$	$0.5 - \epsilon$	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
110	$0.5 - \epsilon$	$0.5 - \epsilon$	$0.5 + \epsilon$	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
111	$0.5 - \epsilon$	$0.5 - \epsilon$	$0.5 - \epsilon$	$0.125 - 0.75\epsilon + 1.5\epsilon^2 - \epsilon^3$

GIVEN: PROB (INPUT n or m or p = 0) = $(0.5 + \epsilon)$

E.G., FOR 101, $(0.5 - \epsilon)(0.5 + \epsilon)(0.5 - \epsilon) = 0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$

BIASED OUTPUT OF A 2 of 3 MAJ GATE, WHEREIN ALL INPUTS ARE EQUALLY BIASED

INPUT nmp	PROB(2 of 3 MAJ GATE OUTPUT = 0)	PROB(2 of 3 MAJ GATE OUTPUT = 1)
000	$0.125 + 0.75\epsilon + 1.5\epsilon^2 + \epsilon^3$	
001	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$	
010	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$	
011		$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
100	$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$	
101		$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
110		$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$
111		$0.125 - 0.75\epsilon + 1.5\epsilon^2 - \epsilon^3$

SUMMATED- $0.5 + 1.5\epsilon + 0.0\epsilon^2 - 2\epsilon^3$

$0.5 - 1.5\epsilon - 0.0\epsilon^2 + 2\epsilon^3$

∴ THE OUTPUT IS A STRONGLY AFFECTED FUNCTION OF THE BIAS, AS OPPOSED TO THE MINISCULE BIAS IN THE 3XOR OUT (FOR UNCORRELATED WORDS).

UNBIASED OUTPUT OF 3 INPUT XOR GATES WHEREIN INPUTS ARE EQUALLY BIASED

INPUT nmp	PROB(3 IN XOR GATE OUTPUT = 0)	PROB(3 INPUT XOR GATE OUTPUT = 1)
000	$0.125 + 0.75\epsilon + 1.5\epsilon^2 + \epsilon^3$	
001		$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
010		$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
011	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$	
100		$0.125 + 0.25\epsilon - 0.5\epsilon^2 - \epsilon^3$
101	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$	
110	$0.125 - 0.25\epsilon - 0.5\epsilon^2 + \epsilon^3$	
111		$0.125 - 0.75\epsilon + 1.5\epsilon^2 - \epsilon^3$

SUMMATED- $0.5 + 0.0\epsilon + 0.0\epsilon^2 + 4\epsilon^3$

$0.5 - 0.0\epsilon - 0.0\epsilon^2 - 4\epsilon^3$

∴ THE OUTPUT BIAS IS A FUNCTION OF ϵ^3 . STATISTICALLY THERE IS NO SENSED BIAS IN ANY STATE VARIABLE OUTPUT IN THE DATA MANIPULATOR RESULTING FROM THE SIMULTANEOUS EQUIPROBABLE INPUTS INTO EACH STATE VARIABLE IN THE DATA MANIPULATOR.

13/05/08 16:43

3BXORMAJ

Fig : Comparing Results of 3XOR & MAJ Filtering Wherein All 3 Inputs are Equally Biased