

Understanding the ZK-Crypt- a Hash/Stream Cipher for (Almost) all Reasons*

Carmi Gressel¹ Nicolas Courtois² Gregory Van Bard³ Avi Hecht¹ Ran Granot¹
FortressGB Ltd.¹ University College London² Fordham University³

Introduction:

The ZK-Crypt algorithm was conceived as a compact hardware system on chip designed to efficiently authenticate encrypted data, additionally encrypt and decrypt data, and to supply highest quality true random numbers for key generation and other cryptographic functions. Used in tandem, units may simultaneously encrypt or decrypt input data whilst authenticating same input data and optionally validating the source of said data. The complete design optionally includes the FortressGB AIS 31 compliant random noise source driven by an unpredictably regulated frequency modulated ring oscillator. For generating IVs and keys, swapping the hash number generator noise source with the random FM noise source makes a true random number generator. The original design has been enhanced with expansion procedures to support a 64 bit Biham-Dunkelman design count framework [haifa].

Each incremental stage of the design was reviewed by international consultants. Their suggestions triggered larger chaining values to preclude pre-image resistance, innovative hybrid 2 of 3 Majority/3XOR/ Store & XOR filters; a bijective S Box to generate highly diffused, random, non-linearized outputs; and dual track orthogonal feedback, to provably preclude message modification and Mersenne Prime LFSR counters for proactive prevention of herded collisions. Increasingly effective proprietary and enhanced standard testing methods were developed and exhaustively used to support the basic diffusive hybrid linear and non-linear approach. Exhaustive testing has proven that no one of the 13 state 32 bit word variables has a differential, a distinguishable or an impossible distinguishable. A study of the equations of the Data Churn convinced us that an algebraic analysis of the 64 bit configuration would have an estimated 50 million monomials.

The ZK-Crypt algorithm was evolved as hardware architecture launched to be a single step 32 bit word Data Authenticator, Stream Cipher and Random Number Generator. By massively increasing the amount of diffusing random logic filters, the number of intermediary variables is more than three times the number of state variables (flip-flops), with consummate increase of algebraic cryptocomplexity, at little cost. Incremental improvements led to hashed loading of secret keys and finally to robust data authentication with proved resistance to message modification and herd collisions. The process was exceedingly difficult as we followed our since vindicated, non-conventional approach that both hash functions and stream ciphers (and TRNGs) should be based on massively diffusing feedback and non-linear permutations to "get the most out of" each binary state variable.

Understanding the ZK-Crypt is easier if one "lays back" realizing that the following 4 basic tenets; the validity and rationale of which we elucidate in the exposition of the concepts and circuitry of the modules, are the backbone of the device. We implement these tenets using straight forward proven and conventional artifacts.

- 1) Large scale diffusion, wherein each binary state variable is an immediate function of at least three preferably disparate state variables, leads to intractable algebraic cryptocomplexity; unbiased binary and word state variables; and to a stark reduction of state variable autocorrelation.
- 2) XOR summing of two or more slightly biased binary structures; i.e., bits, nibbles bytes or 32 bit words, results in less predictable, less biased output; if and only if there is no or very little detectable cross correlation between the structures; i.e., one structure may show strong signs of autocorrelation.
- 3) Judicious use of conventional and tailored non-linear and linear artifacts to simultaneously control: the levels of autocorrelation and bias within binary structures; and the acute reduction of cross correlation between linear combined binary structures.
- 4) Dual Track Orthogonal Feedback (2x32 bit) permuted streams are linearly combined into interacting state variables thereby amplifying the affects of both the randomized increments of the Message counter and the irreconcilable modifications of Message Words; the resulting increased diffusion effects preclusion of Herded Collisions and Message Modification attacks and effective resistance to Pre-Image attacks.

Analysts generally prefer simple "easily analyzed" algorithms (the antithesis of crypto-complexity) which we have replaced with pseudorandom statistically and mathematically proved constructs.

* This article was based on an exposé of the workings of the ZK-Crypt for eSTREAM- co-written with Orr Dunkelman. Research and development were supported entirely by Fortress GB Ltd., and Fortress GB Ltd. employees.

2 The ZK-Crypt Tenets Briefly Explained

The tenets with brief explanatory examples:

- 1) **Large scale diffusion, wherein each binary state variable is an immediate function of at least three preferably disparate state variables, leads to intractable algebraic cryptocomplexity; unbiased binary and word state variables; and to a stark reduction of state variable autocorrelation.**

A device with massive immediate first degree linear and non-linear diffusion, wherein each variable is affected by 3 or more state variables; where there is no sensible bias, no sensible end effect, and no sensible auto or cross correlation; where said device embraces a large state space of over 520 binary state variables, and more than double intermediary logic variables with tiers permuted by random displacements and jittered clocks; and where pseudorandom words are shifted and randomly linear and non-linearly combined; together cause a condition where an algebraic or quantum computing cryptanalysis is inherently intractable as a consequence of the huge diffused number of unbiased monomials in irreducible equations. [zk-secure, chap.2]

- 2) **XOR summing of two or more slightly biased binary structures; i.e., bits, nibbles bytes or 32 bit words, results in less predictable, less biased output; if and only if there is no or very little detectable cross correlation between the structures; i.e., one structure may show strong signs of autocorrelation.**

Examples:

The output of the Least Significant, LS, cell of an up-counter that randomly counts even and odd length pulse streams and thereby generates a random toggle (jittered oscillator) variable. This "irregular" oscillator output is XOR summed to the unpredictable outputs of randomized pseudo-Linear Feedback Shift Registers, nLFSRs, and to random data bits from unbiased word structures, to generate unpredictable unbiased permutation drivers. Such unbiased signals also modulate conventional random logic to generate "occasional" permutating pulses; e.g., to generate missing clock pulses; and to generate false feedback (slip) pulses on one or more of the 12 ZK-Crypt nLFSRs. [zk-ccc fig. P08].

The permuted 2 of 3 Majority Splash Output filter is forced into a strongly correlated condition, in which at every clock every fourth output bit is highly biased to the same random value. We prove statistically that the output is unbiased and strongly correlated. When linearly combined to a slightly correlated (even number of ones and zeroes) word, the output is unbiased, with trace autocorrelation.

- 3) **Judicious use of conventional and tailored non-linear and linear artifacts to simultaneously control: the levels of autocorrelation and bias within binary structures; and the acute reduction of cross correlation between linear combined binary structures.**

- a) Conventional correlation immunizers, e.g., a summation of present and past results; and pseudo-random linear displacements, reduce auto correlation; and,
- b) A non-linear transformation can force a biased input into an orderly correlated word, e.g., every n'th bit is forced, with high probability to be of same value. [zk-secure chap.3]

- 4) **Dual Track Orthogonal Feedback (2x32 bit) permuted streams are linearly combined into interacting state variables thereby amplifying the affects of both the randomized increments of the HAIFA Message counter [zk-secure chap. 8] and the irreconcilable modifications of Herded Collision and Message Modification attacks and effective resistance to Pre-Image attacks. Swapping orthogonal feedback between concatenated ZK-Crypt units insures that a modified Message in one unit will irreconcilably alter the state variables in both units.**

We show that any change of a valid message word aberrates orthogonal feedback streams fed into the Register Bank and the Data Churn, such that different parts of the Bank and the Churn are corrupted differently so that any reconciliation of the corruption caused by one stream will further corrupt (and continue to corrupt) parts corrupted by the second stream. Formally, we define 2 Hash/MAC feedback streams as orthogonal in a tiered word manipulation architecture where:

any non-valid sequence of message words causes a first feedback stream to corrupt at least one first bit in at least one word in a first tier on a first clock; and on a second clock a second false message word activates a reconciliation (a flip back) of the altered? at least one bit in said first tier's altered word to the valid previously unmodified second clock condition; and whereas simultaneously, the at least one altered bit in the first false message word of the message sequence altered at least one bit in the second feedback stream which simultaneously first corrupts at least one bit in at least one other tier simultaneously, such that the said second message word cannot reconcile the said altered bits in the second tier,

thereby modifying the state variables of said tiered word manipulation architecture from the valid condition

generated by a valid message sequence.

E.g., in an attempted message modification wherein an adversary changes a "\$100", USD value to a "£100", UK Pounds Sterling value in a first message word in a dual track orthogonal feedback configuration:

On the first cycle (clock), flipping false message bits wherein said flips cause an equivalent false "£" bit value in at least a first word in one first feedback stream affecting one specific tier; and in a second clock in a second sequence message, via said first feedback stream, flipping bits operative to reconcile relevant bits in said first tier; thereby to restore the "£" related bits back to the valid "\$" representation bits in said first tier, hiding the first word modification in said at least first tier; while simultaneously,

said flipping on the first clock false message bits causes an equivalent false "£" bit value in the second feedback stream causing an equivalent false "£" bit value in at least one second word in one second tier; wherein said second clock first restoration "\$" message bits cannot restore said second word in said second tier to a valid condition.[zk-secure- appxs 1-2] . Who knew, at the time of writing that the two currencies may some day be practically equivalent?

A Result is defined as an intermediate hash digesting value which is a linear function of an enciphered mask output of the engine core XOR summed to a message word.

We later prove generically, that if one feedback sequence is a function of a first clocked Result, and a second feedback sequence is a function of the (same) first clocked Result XOR summed to the previous clocked Result, the two feedback sequences are orthogonal, thereby, provably preclude unobserved message modification.

the ZK-Crypt was enhanced to a degree that we can also prove that by the 4th cycle after a bit of a false message in an attempted message modification, the entire ZK-Crypt variables has been massively corrupted.

The Dual Track Orthogonal Feedback streams when XOR summed to 64 disbursed output bits of an enlarged Biham-Dunkelman HAIFA counter, serve to prevent hash collisions for the first 2^{62} 32 bit message word sequences. The only degree of freedom available to an adversary in a secured environment is to modify message inputs. To engineer a collision, the attacker would have to forge message words that would complement the disbursed counter bits in both feedback streams. Any attempt to complement previous HAIFA counter bits with forged message bits in order to generate "valid" feedback, simultaneously corrupts both feedback streams and subsequent chaining values. As in the proof of message modification, after four cycles, we can prove that the entire ZK-Crypt variables will be massively corrupted.

For explaining the algorithm we use hardware parlance when referring to logic variables. A signal is a variable that serves to start or cause some action; e.g., a Slip which aberrates an nLFSR¹ feedback, a Splash Select signal which routes displacement bit in the Splash Matrix, or a clock signal. We are more explicit, however, when referring to clock signals. We generally call clock signals pulses, as they are reserved for "Sampling" memory cells, and are only effectively active when "rising" from zero to logic '1'. A clock pulse remains at logic '1', for only the first half of a clocking cycle. State variables are the outputs of memory cells, and they drive random logic gates, e.g., XOR, AND, and OR. During the initial rise time of a clock signal, the input value to the memory cell (a flip-flop) becomes or continues to be the output value for the remainder, at least, of the clock cycle.

The formal Specifications of the ZK-Crypt Algorithm can be found at [zk-algo].

2. Organization of this Document:

We recommend that a skip-through reader of our documentation may benefit keeping the annotated A to Z Guide to the ZK-Crypt [zk-a-z glos] as a quick reference for terms and concepts. Because of the hardware nature of the algorithm, it might be easier to have a better "picture" of the algorithm from the ZK-Crypt Circuit & Concept Drawings [zk-ccc], when trying to grasp the reasoning of the sections. Proofs are detailed in the ZK-Crypt Security Analysis; [zk-secure chaps 1-2] documents are a good start. A formal description of the modules can be found in the ZK-Algorithmic Specification [zk-algo]. The spec is a description of the permutations; the reader may loose perspective of the complete system.

2.1 The ZK-Crypt Algorithm Architecture – As the Algorithm design was intended for efficient implementation in hardware, the processes, functions and modules are easier to understand when described as hardware block modules. We have tried to bridge the gap for workers who shun hardware circuitry.

¹ Most of the so called "nLFSR" shift registers in the ZK-Crypt linearly receive external feedback; and may better be referred to as "pseudo linear feedback shift registers", pLFSRs.

2.2 Operational Configurations of ZK-Crypt Engines - All incoming Messages Words are XORed to an unpredictable word immediately on entering the ZK-Crypt; irrelevant of mode of operation. A preferred system consists of two engine sets wherein data is strongly encrypted prior to hash digesting.

The ultimate efficacy for secure and virtually transparent authentication of 32, 64 and larger bit data words requires a pair of ZK-Crypt engine (sets). Data can either be stored encrypted or in the clear in secure or unsecure memory. Following initialization, at each Primary Clock, a word is fed simultaneously to both engines (sets).

If the data is encrypted (as required for modes of implementation where Hashed data is encrypted), e.g., an encrypted boot, the "left" set decrypts, while the "right" set digests; and only the decrypted data is read; see Section 4]

If the stored data is in the clear, e.g., secured random access memory; the clear text is encrypted in the "left" engine and simultaneously read by the host; and transferred on the next clock cycle to the "right" set, to be hashed.

After the file has been processed, the Hash value/MAC is generated, and validated by the Host.

2.3 The Register Bank – Consists of 4 pairs of concatenated unique pseudo-Linear Feedback Shift Registers, nLFSRs, arranged in 4 tiers, each tier with a unique projected permutation; where all four tiers are filtered and non-linear combined to output a 32 bit word to the Data Churn. We call the four tiers our "Sanctus Sanctorum"², holy of holies vault. In the vault a Message bit is high entropy encoded, and diffused, so that the hybrid 4 tier combiner will remove any vestige of correlated differentials.

2.4 The Data Churn – Consists of tiers of hybrid linear/non-linear combiners with memory, and 2 four rule displacement matrices. The Churn inputs two versions of the Lower Feedback and outputs five 32 bit words to the Result/Feedback Processor.

2.5 The Result/Feedback Processor– Consists of one Result and 2 Feedback Registers and random logic to regulate the two main modes of operation, Hash/MAC mode and Cipher Mode. The Result Register contains the XOR sum of the Cipher Mask output of the Data Churn and Message Word. The Processor generates two orthogonal feedback tracks, the Lower Feedback and the Super Tier Feedback, and when relevant, a Result value to the Host.

The hashing processes are function of the present and past Result, and are used to initialize keys, IVs and all phases of data authentication.

The Stream Cipher generates two feedback streams, neither of which is a function of the Cipher Mask or the Message Word.

The Hash/MAC mode feedback is the essential component for indelibly encoding Message Words in the Register Bank. It is used for secured initialization, for hashing in Secret Ciphers and HMAC Keys, and Cipher and Hash IVs.

Mersenne prime LFSRs and a 6 bit binary counter [haifa] salt and mask both feedback streams preventing herding. As 32 bit Message words are reflected orthogonally in the 64 bits of feedback stores provably a unique place number cannot be forged in the chaining value.

2.6 The ZK-Crypt Random Controller – Consists of the 3 Control Units, the noise source, and the output permutation encoder. The Random Controller generates 14 permutation and clocking signals to the Register Bank and to the Data Churn. The Controller receives 10 balanced feedback signals from the Register Bank and the Data Churn.

2.7 HAIFA Counting, Page Counting, Entropy Validation & Pseudo Random Salting-

The 64 celled HAIFA Counter uniquely diffuses 64 randomized bits into the orthogonal feedback streams at each clock in the deterministic modes of operation. The dispersed 64 count bits prevent

² We intended no offence to believers or non-believers when describing the TMB Tiers as our "Sanctus Sanctorum" – Holy of Holies. We believe that the TMB Tiers are an irreconcilable, unobservable vault which uniquely reflects each bit of all Messages and into which "mortals" will never enter.

collision, pre-imaging and message modification. The Host will typically include the 6 binary count bits to synchronize Internet Pages in lengthy data transmissions, to assure orderly reception of transmitted cipher and clear text pages. In the TRNG mode, a random period of the autonomous oscillator output, **fr**, is ascertained by counting **fr** pulses in fixed Primary Clock periods.

2.8 Increasing Cryptocomplexity-Parallelizing, Scrambling and Multi-Step Wait & Read- ZK-Crypt engines can be concatenated for speed and higher security, or alternately may work side by side to simultaneously de/encrypt and authenticate. Concatenation multiplies bit rate, without increasing energy per processed bit. The Multi-Step Wait & Read function is the equivalent to multiplying the number of rounds in a Feistel cipher. The Results are sampled at user defined intervals. Typically, the only added cost is a linear increase of energy per processed bit, as we anticipate that in most applications, a single host cannot hope to service the ZK-Crypt operating at full speed. Scrambling, exercising the engine without sampling a Message Word and/or Reading a Result, is used to increase cryptocomplexity after or during the processes of Initializations and Message Digesting. **Multi-Step Wait & Read is not included in the first version.**

2.9 Hash/MAC & Cipher Procedures in the ZK-Crypts – This section outlines the initialization and execution of the Cipher and Hash/MAC Protocols. Additional proprietary methods for synchronizing and authenticating transactions are included in [zk-ccc].

2.10 True Random Number Generation, TRNG – Stream Ciphers and Hash Generators are both classic examples of pseudo-random number generators, which when randomly activated may generate true random number sequences. The fully implemented ZK-Crypt is more compact than competing TRNGs, is much faster, with an all digital noise source (not mixed analog/digital), with 4 activated noise outputs. The noise source is BIS AIS 31 compatible with a proprietary on line entropy test mechanism. Silicon-wise, the ZK-Crypt grants an implementer the three basic symmetric functions for the price of a first class TRNG. All security devices will need on board true random number generators, with provable on line testing of the noise source. The ZK-Crypt true noise source loads entropy many thousands of times faster than competing devices.

2.11 Testing; Past, Present, and Future: We have progressed from our own and standard and proprietary PRF test packages for Repeated Words, Noise, intra word and inter word correlations, bias counting of all state bits.

We assiduously combed the design and the output statistics in search of local bias, and potential weak spots (have found none) in general to be used in more rigorous types of classical differential and linear/correlation attacks which may include:

- Algebraic attacks with Grobner Bases,
- Algebraic attacks with improved ElimLin algorithms,
- Algebraic attacks with 3-SAT solvers,
- Algebraic attacks with various generalized T' methods (proprietary),

2.11 Cost, Computational Efficiency, Memory Requirements (NIST 4.B i, ii)

HW implementation - The ZK-Crypt Size, Timing, Current Consumption

A single engine ZK-Crypt HW block consists of an estimated less than 10K two input NAND gate equivalents for full implementation including a small Finite State Machine (FSM).. Concatenating a second engine will double the estimated size. The enhanced ZK-Crypt (single) engine has over **550** proactive flip-flops, all used in hashing and ciphering, wherein logic blocks have been buffered to reduce the critical path by one third, allowing us to estimate safe operation at 300 MHz, with present technology.

Deterministic timing – ZK-Crypt Hardware

engines operating on 100 MHz Bus (Prior implementations of the design ha exceeded 200 MHz)

IV/Hash Value Bit Size	IV/Key Initialization Includes: Configuration Changes, Hashing in IVs, & Scrambles		Hash Value Generation Includes: Configuration Changes, 1000 Byte Digests, 16 Scrambles & Hash	
	Machine Cycles	Time @ 100MHz	Machine Cycles	Time @ 100MHz
Standard Single Engine	32 Bit Bus Machine			
224	24	240 nano sec	277	2770 nano sec
256	25	250 nano sec	278	2780 nano sec
384	26	260 nano sec	279	2790 nano sec
Total			256 bit total time hash	3.05 µsec
Standard 64 bit Concatenation	64 Bit Bus Machine			
384	23	230 nano sec	151	1510 nano sec
512	25	250 nano sec	153	1530 nano sec
768	29	290 nano sec	157	1570 nano sec
Total			512 bit total time hash	1.78 µsec

Smart Card Implementation

8 bit CPU 32 bit Message No Scrambles	4Cycles for Each 32 Bit Auto-Load No Scrambles	Time @ 5 MHz DMA CPU- 1 MHz Engine	4 Cycles for Each 32 Bit Auto-Load No Scrambles	Time @ 5 MHz DMA CPU- 1 MHz Engine
256 (Re)set IV	No Load- 5 Cycles	5 µsec	1047	1047 µsec
Total			256 bit total time hash	1047 µsec

Estimated current consumption per processed bit:

For a single or double ZK-Crypt Engine at 0.09 micron technology, 941 Mbits per milliwatt second. Note: the double engine 64 bit configuration doubles speed and more than exponentially increases strength, with the same current consumption per processed bit. The current consumption per processed bit in a popular SHA-1 implementation is 8 Mbits per milliwatt second.

Timing the ZK-Crypt Algorithm on a PC for Legacy Applications

(These figures are for the version with a simple binary HAIFA counter- for legacy implementations we will suggest elimination of the HAIFA counter).

The timing tests were executed on a Dell-Latitude D620, with a T2400 Intel CPU (1.83GHz) and 0.99GB of RAM with Windows XP Professional OS.

In all tests the set up time was measured separately from Message digestion and Hash Value/Tag generation. We timed 4 and 1000 byte random messages. Each test was repeated 10 times for each bit length.

We see no advantage to a 64 bit platform, as all computations (including double engine are on 32 bit operands.

Memory Requirements on Hardware and Software Legacy Applications

The following includes Hashing, HMAC, Stream Ciphering, (and the TRNG without online verification).

Est. total external memory requirements for a single engine hardware implementation- 3 KB.

Est. total external memory requirements for concatenated engine hardware implementations- 4 KB.

Est. total external memory requirements for software implementations on any platform- 18KB.

RAM required to hold intermediate values and all tables – 652 bytes.

Average Performance Times for High Speed PC Computing of the ZK-Crypt Algorithm on 4 Byte (1 block) and 1000 Byte Messages

IV/Hash Value Bit Size	Set up Includes Hashing in IV/Key and 8 Scrambles		Digestion, 16 Scrambles and Hash Value Generation			
			4 byte		1000 bytes	
	Cycles	Time	Cycles	Time	Cycles	Time
1 Engine						
224	28,149	0.015ms	26,868	0.0146ms	414,445	0.226ms
256	27,915	0.015ms	28,408	0.0155ms	413,882	0.226ms
384	27,062	0.015ms	34,103	0.0183ms	418,866	0.229ms
					256 bit total time	241 µsec
2 Engines						
512	54,211	0.029ms	54,514	0.0298ms	442,724	0.242ms
768	54,434	0.029ms	66,841	0.0365ms	453,912	0.248ms
					512 bit total time	271 µsec

Hardware performance with single or double (concatenated) engines is therefore 80 to 150 times faster than software PC speeds. Hardware constants and tables are implemented in random logic with efficient reduction of critical propagation time.

The all-digital ZK-Crypt is designed to replace popular mixed signal non-AIS 31 compliant TRNGs, facilitating a "no cost" silicon upgrade to the ZK-Crypt on new designs.

3. The ZK-Crypt Algorithm Architecture as implemented in hardware:

In this section we describe the interaction of the five main modules of the ZK-Crypt; the Random Controller, which drives the 32-bit Register Bank and Data Churn, the Result/Feedback Processor which generates orthogonal feedback into the Bank and Churn, and the HAIFA Counter.

The Random Controller which includes a permutation Clocking Mechanism; the 32-Bit Word Manipulator, aka the Word Manipulator, which includes the Register Bank, the Data Churn; and the Result/Feedback Processor which outputs Results, and processes Data Churned variables to generate two tracks of either cipher or data authentication feedbacks randomized by the 64 Bit HAIFA Counter.

The Random Controller outputs pseudo-random clocking pulses to the Register Bank, and regulates permutations in both the Register Bank and the Data Churn. The Random Controller receives 8+2 pseudo-random binary feedbacks from the Register Bank and the Data Churn. The Controller outputs 8 clock and binary permutations signal back into to the Register Bank and generates 6 permutation control signals to regulate Data Churn permutations. The Random Controller selects 4 displacement rules in two displacement Data Churn displacement matrices, and regulates unpredictable forced correlated internal outputs from non-linear filters in the Data Churn. Figure 1 shows the basic interaction of the main components.

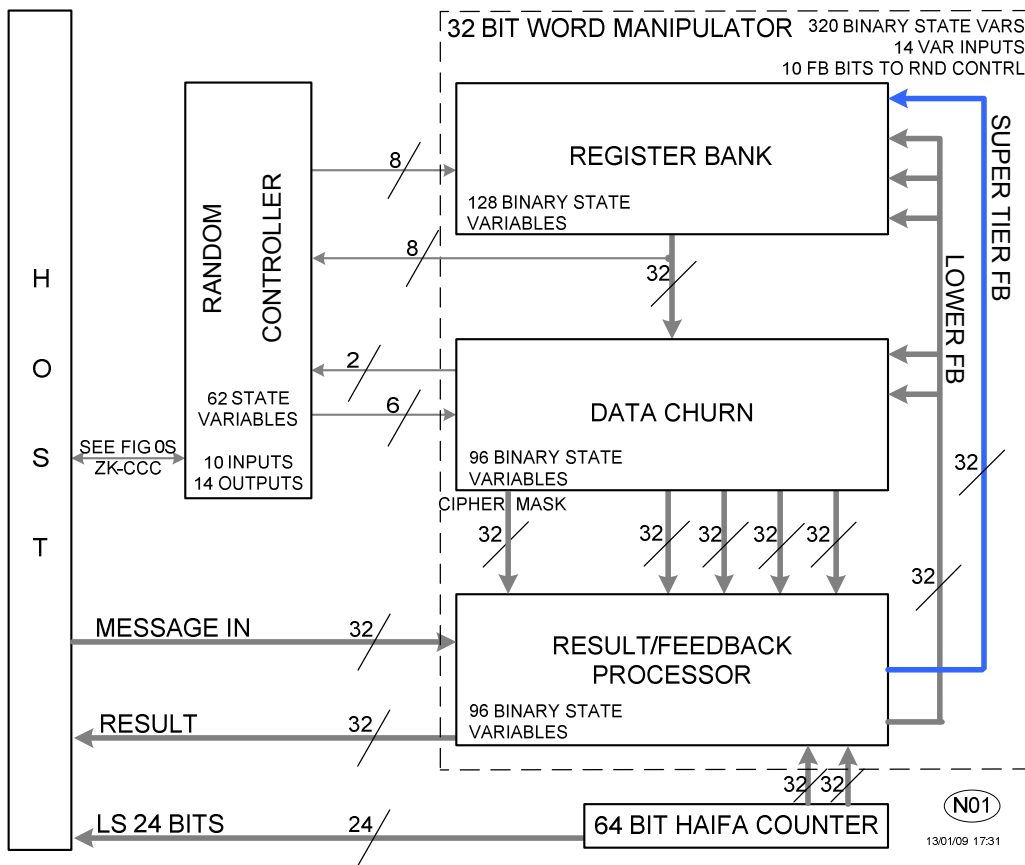


Fig. 1: The Basic ZK-Crypt Algorithm Architecture

The HAIFA Counter is a device for preclusion of hash/MAC herded collisions and for generating user defined synchronizing pages in long data transmissions.

Except for unkeyed hash, we view the Register Bank, as a Sanctus Sanctorum secret "record" of all previous operations. A single feedback aberration or a changed Random Controller signal immediately corrupts all data bits in the 32 bit Word Manipulator. Each of the 32 parallel output bits of the Register Bank randomly affects most or all of the "observable" Cipher Mask bits. [zk-secure appx 4]

In unkeyed hashing, where the attacker is omniscient, we show that the Register Bank cannot be reconciled to a true state, after once receiving a modified message. The Data Churn was designed to assure massive displaced diffusion, and to control any auto or cross correlated data combinations in any or between any two intermediately generated 32 bit words. Each of the tiers in the Data Churn or the Register Bank in Fig. 2 is a statistically top quality random number source; as each state variable is configuration consistently emits hi-quality statistics. An aberration of a false bit in a Message, on the first clock cycle, "appears" in all bit variable equations in the second clock cycle, and in more than 140 binary state variables in the first clock aberration as shown in a 20 graphic case study of possible permutation configurations of the engine [zk-secure appx 4].

The Register Bank as shown in Fig. 2A consists of four tiers of unique pairs of pseudo-Linear Feedback Registers, aka nLFSRs in these documents. Each tier's concatenated nLFSR pair outputs are projected (rotated) Images which are bit wise XOR summed with the nLFSR pairs' outputs. At each sampling a non-linear combiner process combines the Top, Middle and Bottom tiers, which is then masked by a Super Tier linear permutation. The 32-bit word virtual no leakage output of the Register Bank is non-linearly combined and stored in the Register Bank Store. At the next clock the store value is input into the Data Churn.

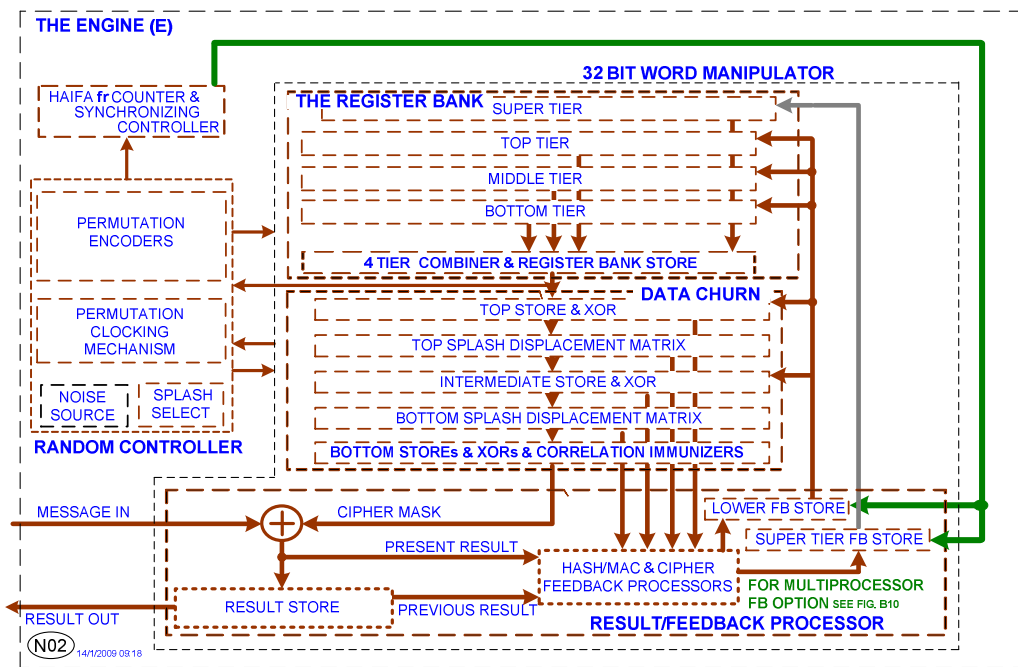


Fig. 2A: The ZK-Crypt Architecture with Dual Track Hash/MAC & Cipher Feedback

The Data Churn receives the output of the Register Bank and outputs the Cipher Mask and three more internal words for Lower and Super Tier Feedback generation. The Churn consists of three tiers of non-linear combiners with memory, and two 4 rule Splash Matrices each of which displaces input bits into four different output strings and then diffuses the outputs such that each output bit is a function of four near neighbor outputs and a permutation bit signal from the Random Controller; detailed explanation follows.

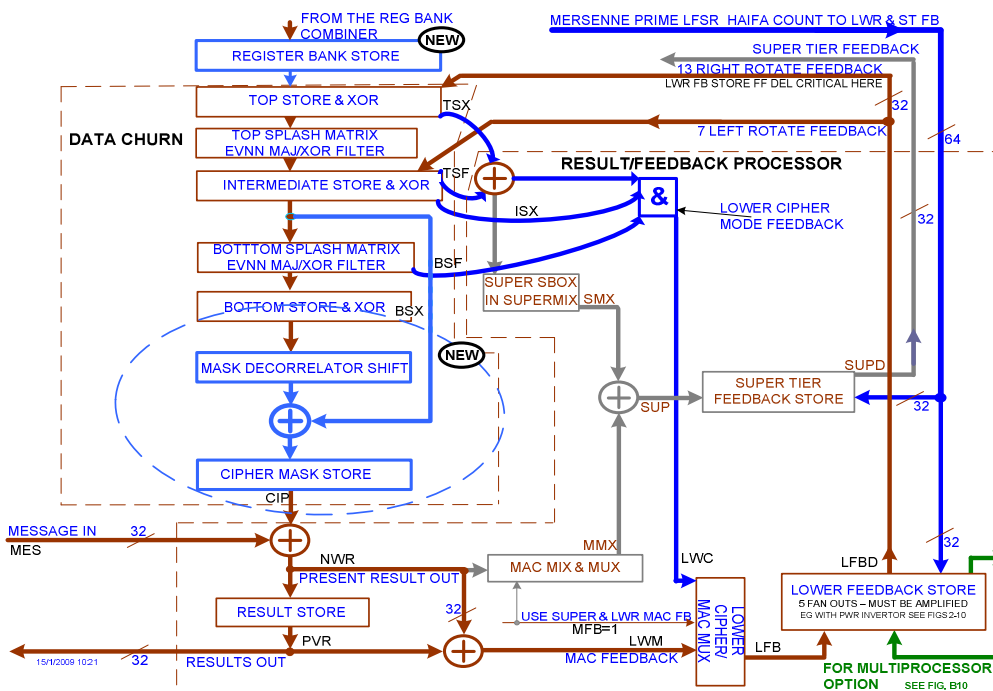


Fig. 2B: Revised Dual Feedback Circuitry with the Decorrelated Register Bank & Cipher Mask

The Result/Feedback Processor has two modes of operation- Hash/MAC Mode, for Hash and Cipher Initialization, and Cipher Mode, for Stream Ciphering.

In Hash/MAC Mode, the Hash/MAC Result/Feedback Processor also XOR sums the output of the Data Churn (the Cipher Mask) and the Message Input. In Hash/MAC Mode, the Message may be all '0'. The Lower Feedback Hash/MAC Mode process consists of combining (XOR) the Present and Previous Results; which is loaded into the Lower Feedback Store. The Super Tier Feedback process consists of combining the Super Tier Cipher Feedback with the Present Result. The Super Tier Feedback is loaded into the Super Tier Feedback Store on the next Primary Clock cycle. In addition disparate output bits from the HAIFA Counter are simultaneously diffused into the Lower and Super Tier Feedback Stores.

When stream ciphering is required, the Message is XOR summed with the Cipher Mask and the processed word is loaded into the Result Store at the next clock cycle. The Result is either the Clear or Cipher Text version of the Message. The Cipher Feedback Processor simultaneously generates two uncorrelated pseudo-linear random feedback words, which are loaded on the next Primary Clock Cycle into the Lower and Super Tier Feedback Store. In addition disparate output bits from the HAIFA Counter are simultaneously diffused into the Lower and Super Tier Feedback Stores.

We have proved that as one feedback stream is a linear function of the Present and Previous Results, and the second feedback stream is a linear function of the Present Result, the feedback streams are orthogonal; i.e., changed bit or bits of a Message cause irreconcilable aberrations in at least one of the two linear input feedbacks.[zk-fbpat3]

The Random Controller generates the clocking pulses that are used in the Word Manipulator and is composed of four clocking mechanisms which drive the triple Control Unit permutation encoder. By pseudo-randomly clocking the 24 permutations and correlation immunizers in the 32-bit Word Manipulator we amplify the crypto-complexity of the Manipulator.

In a single clocked cycle, one bit of feedback typically is diffused (affects the equations) into 9 of the 32 bits which are output by the Register Bank. These 9 bits in the first version are further diffused by the 10 levels of displacement permutations and non-linear memory combiners into an average of 31 (!) of the output bits (virtually complete diffusion). In the implemented version, the Cipher Mask coverage is complete in the new round, as are more than 200 of the 416 bits in the Word Manipulator. In the next round, all of the 416 bit Word Manipulator bit equations are affected.

In the new version, the Bottom Store & XOR has been enhanced with two more buffers where the Mask Decorrelator Shift receives the output of the Bottom Store & XOR, and the Cipher Mask is the XOR sum of the output of the output of the Intermediate Store & XOR and the Mask Decorrelator Shift buffer. See Fig. 2B.

Each of the Churn's Top, Intermediate and Bottom Store & XOR filters and also the Result buffer in the Result/Feedback Processor Store & XOR filter, accept a current clocked 32 bit input value from a preceding combining function, while outputting the result from the previous clocked cycle. The output of each Store & XOR cell is the bit wise XORed value of the previous input to the memory store and the present output bit of the preceding combining function.

In addition, the Top and Intermediate Store & XORs also combine (into the relevant memory store) a shifted feedback from the Lower Feedback Store. Therefore, each Store & XOR memory cell output is a function of many disparate input variables.

There are 5 outputs from the Data Churn into the Result/Feedback Processor; the Cipher Mask which is always XOR summed to the Message and loaded into the Result Store, and four words that are used to generate Lower and Super Tier Feedback.

The 64 Bit Mersenne Prime LFSR/binary HAIFA Counter output bits are XOR summed to the Lower and Super Tier Feedback Stores, thereby uniquely encoding 128 bits of the chaining value of each Cipher Mask in all three modes of operation, unaffected by true or false Hash/MAC Messages.

4. Operational Configurations of ZK-Crypt Algorithm:

4.1 The Single Engine ZK-Crypt Standalone Configuration for Constrained Environments

The estimated 9.5K gate single ZK-Crypt standalone engine can be a very cost effective addition for the smallest RFID and conventional contact smart card devices which need low energy encryption and data authentication. Smart cards may be used in high security systems. In many instances, there is no need for

simultaneous en/decryption and hash authentication. Because of flexible configuration, two or four engines simultaneously operatable on 64 bit Hosts are both cost effective and downward compatible. When increased security is required, "Wait & Sample" procedure may be used, which is an equivalent for multiplying the number of rounds; in proprietary FSM implementations.

4.2 The Paired ZK-Configurations

Though a single engine ZK-Crypt is secure for data authentication of sensitive Clear Text, we realize that for practically no cost, a paired unit can cost effectively transparently decrypt stored data from unsecure memory, and authenticate previously encrypted data.

Similarly, data held in the clear, when being moved by the host to secondary memory, can be encrypted by a L/H twinned ZK-Crypt engine while being transparently digested for preferred Data Authentication.

The ultimate efficacy for secure and virtually transparent authentication of 32, 64 and larger bit data words requires a pair or n concatenated ZK-Crypt engine (sets). Data can either be stored encrypted or in the clear in secure or unsecure memory. We assume that simultaneous de/encryption and authentication is a necessity for future hi-speed transactions.

In the following six sections, we show the basic methods for attaining speed with paired units.

4.3 Initializing the ZK-Crypt Cipher/Data Authenticator Pair

For data authentication and encryption, both units are configured in Hash/MAC Mode, where Message Words can be efficiently loaded with keys and IVs by hashing in Message Words. All Key and IV (Message) words are hashed into the ZK-Crypt engines. [zk-ccc fig. 7]

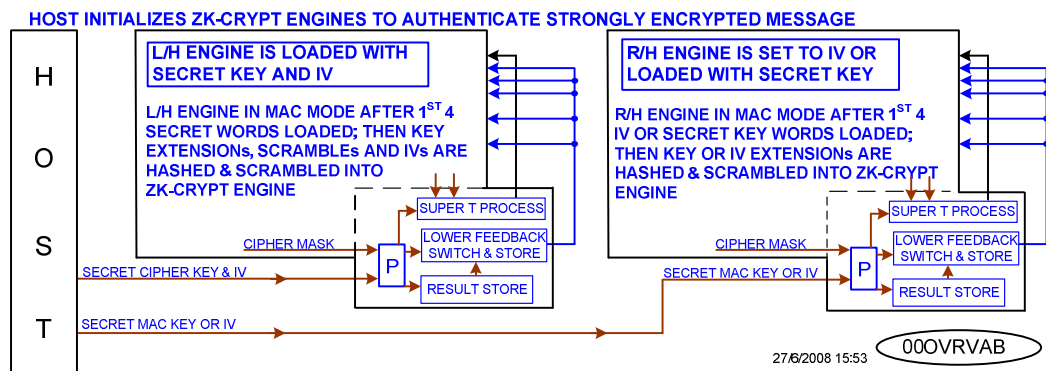


Fig. 3: Host Initializes 2 ZK-Crypt Engines to Authenticate Encrypted Data; e.g., an encrypted Boot B03

All data lines are 32 bit words.

In Hash/MAC mode, both the Super Tier and the Lower Feedback Streams are functions of the loaded Message Words. The "P" processor/switch outputs the Present Result into each of the blocks in the Result/Feedback Processor, the lower right hand block in the engines. The Present Result is an XOR summation of the Cipher Mask and the incoming Message; in Initialization, either a key or an IV. The Lower Feedback is an XOR summation of the Present Result and a Previous Result residing in the Result Store. The Super Tier Feedback stream is salted both by the HAIFA counter and the MAC MIX permutation of the Present Result.

In Fig. 4 we deal with transparent decrypting and authentication of Message data. Typically, this may be an encrypted Boot, where the ZK-Crypt engine pair will simultaneously decrypt each Message word and the Host will place the data in random access memory, whilst the R/H engine will digest each Message in preparation for generating a validated hash value/MAC tag.

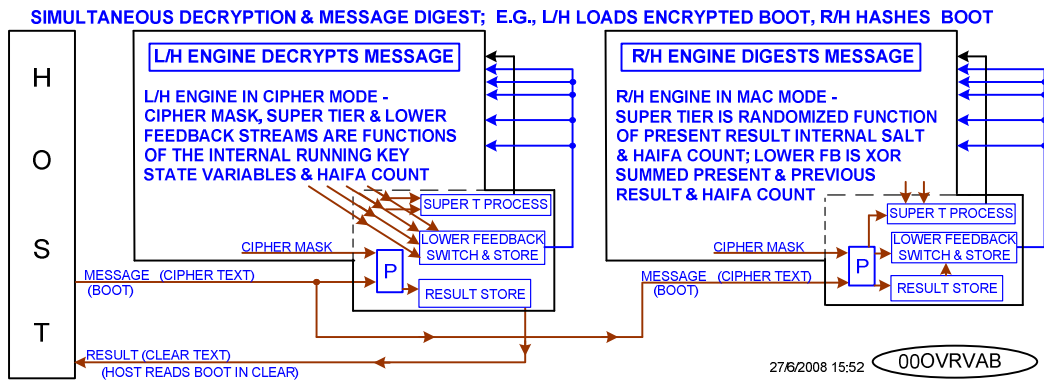


Fig. 4: Simultaneous Decryption & Hashing; e.g., L/H Loads Encrypted Boot, R/H Hashes Boot Fig.B00

At each Primary Clock cycle, the Host inputs the same cipher text Message into L/H and R/H engines.

In Fig. 5 we demonstrate the robust combination of transparently encrypting clear text data in the L/H engine, which is simultaneously being moved to secondary fast access memory, whilst the R/H engine (hash) digests the enciphered data.

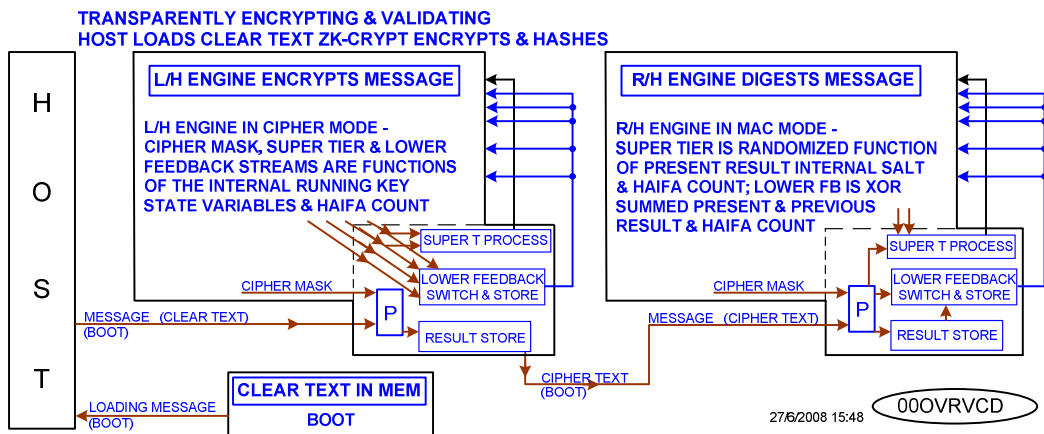


Fig. 5: Simultaneous Encryption & Hashing; e.g., L/H Loads Clear Text, R/H Hashes Encrypted Data

This configuration is most effective with a firmware embedded FSM command. As a clear text Message is moved from source to destination, the data word is encrypted in the L/H engine then in the next Primary Clock cycle, the enciphered word is hashed into the R/H engine. Note that during the Message Digest, in a proper hardware implementation, Results cannot be read by the Host.

In Fig. 6 we see the configuration for the last stage of data authentication; i.e., generation of the Hash Value/Tag. No data is input; the Message input is locked on '0'. Following the Message Digest the R/H engine is activated for 16 clock cycles; effectively scrambling the contents; thereby increasing the robustness of the operation. During this last phase, the L/H device is in Sleep Mode whilst the R/H engine is exercised. After the scrambling phase, the engine generates and the Host reads the Hash Value/Tag from the Result Store.

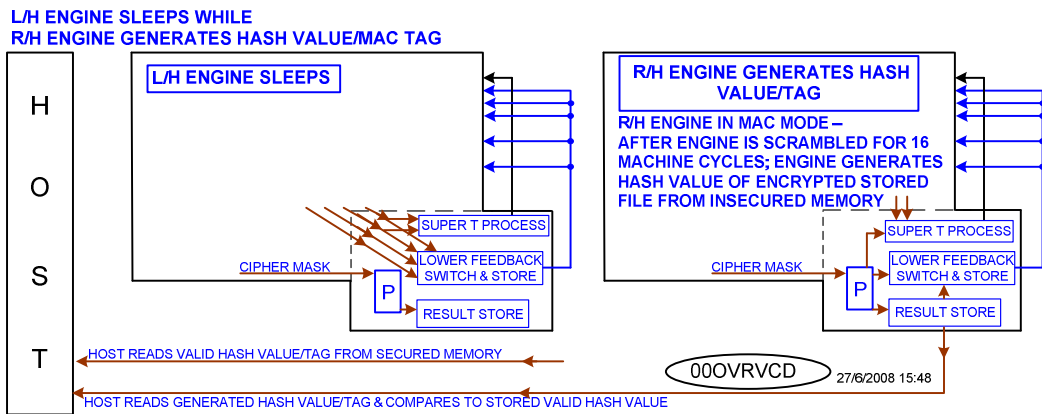


Fig. 6: R/H Engine Scrambles Digest Data and Generates Hash Value/Tag; while L/H Engine Sleeps B00

In Figs. 7 and 8, we show the basic configurations for parallelizing two ZK-Crypt engines. Parallelizing the engines with swapped feedback streams effectively joins the units into a unit with 1108 state variables/chaining value including the Mersenne composite HAIFA counters; for exponentially higher security, faster, encryption and hashing without increasing energy per processed bit.

In practice, the L/H and R/H engines may be effectively configured as parallelized concatenated engine pairs, as shown in Figs. 7 and 8. Theoretically, any number, $0 < n$, of units can be linked for servicing $n \times 32$ length encrypted words as shown in [zkccc-fig. B11].

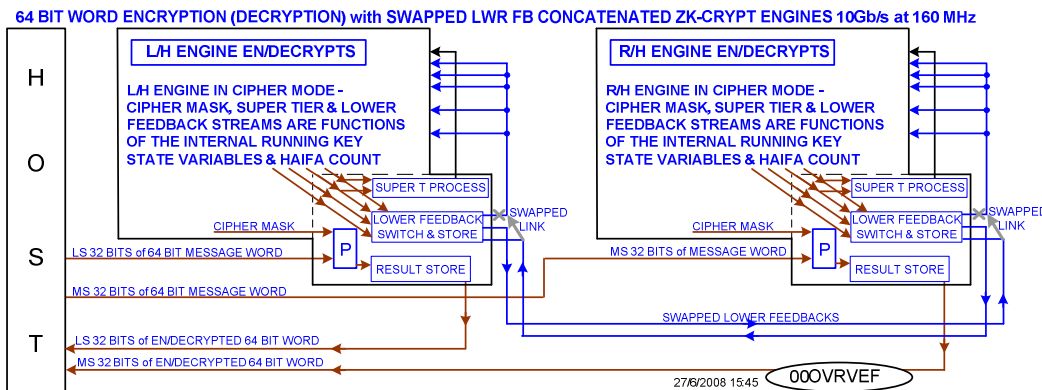


Fig. 7: 64 Word En/Decryption; with Swapped Lower Feedback B000VRVEF

In both implementations, the LS 32 bit of the Message Word is input into the L/H engine, and the R/H engine accepts the MS 32 bit word. These configurations comfortably support cipher key and IV lengths of 512 bits and hash value/MAC tag lengths of 1024 bits. (Remember, our state variables work harder!)

In Fig. 8, the L/H Lower Feedback Switch links the Present and Previous Result from the L/H engine to the R/H Lower Feedback Switch input. Conversely the R/H Lower Feedback is fed into the L/H Register Bank and Data churn.

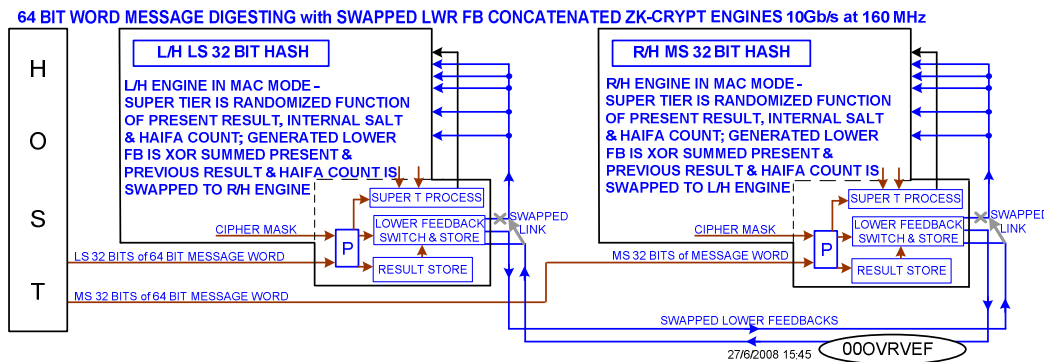


Fig. 8: 64 Word Message Digesting with Swapped Lower Feedback; supporting 1024 bit Hash Values
000VRVEF

5. The Register Bank:

The Register Bank in Fig. 9 linearly accepts the two (Lower and Super Tier) orthogonal feedback streams which linearly affect the Super Tier and the unique Top, Middle and Bottom (TMB) Tiers.

Flipped bits in a Message affect both the Super and Lower Feedback. Super Feedback linearly changes the memory cells in the Super Tier. Lower Feedback randomly affect at least two of the three TMB tiers; randomly chosen, one of three tiers "misses" about 46% of the activating clocks.

Simultaneously, the inputs are linearly affected by the previous "shifting in" state of the unique pseudo-linear feedback shift registers (nLFSRs) and by the linear internally generated feedback of the nLFSRs and the "false" slip feedback from the Random Controller.

Eight random signals from the Register Bank Output randomize the Control Units of the TMB Control Units, Figures 1, 17A and [zk-ccc figs. P02 –P04].

The Register Bank's 4 tier outputs are divided into two sets. The "unpredictable" output of the Super Tier linearly masks the hybrid linear/non-linear combined output of the randomly clocked TMB tiers. The TMB Tiers outputs are combined in a 2 of 3 Majority filter. The output of the 2 of 3 Majority Combiner is rotated 5 bits to the right and XORed with itself; and also XORed with the output of the Super Tier in the 4 Tier Combiner to generate the output of the Register Bank. As any 32 bit value which is rotated and XOR summed to itself is an ENS (an Even Number String has an even number of '1's and '0's), and ENS strings XORed to ENS strings are ENSes, so that the output of the Register Bank is an ENS value. The output of the Register Bank is stored in the Register Bank Store, and is output to the Data Churn at the next Primary Clock tick. The isolating Register Bank Store shortens the critical propagation path by at least 35%.

In a demonstration of the 20 cases of diffusion [zk-secure appx 4] we show that that each TMB nLFSR output bit is a first order (immediate) function of an average of 5 state variables. The tiers all include linearly combined rotated Images, each of which double monomials in the equations. In the TMB tiers, the images are randomly summed, and randomly clocked, so that Images, nLFSRs, and combined outputs may be a function of a previous clock cycle. The number of bit variables is more than quadrupled to more than 26.

The triple input of the 2 of 3 Majority Combiner will have an average of more than 72 first order state variable monomials, which is doubled by the 5 Right Rotated Image to an average of more than 140 state variable monomials on the output of the 3 Tier Combiner. [zk-ccc fig. R13]

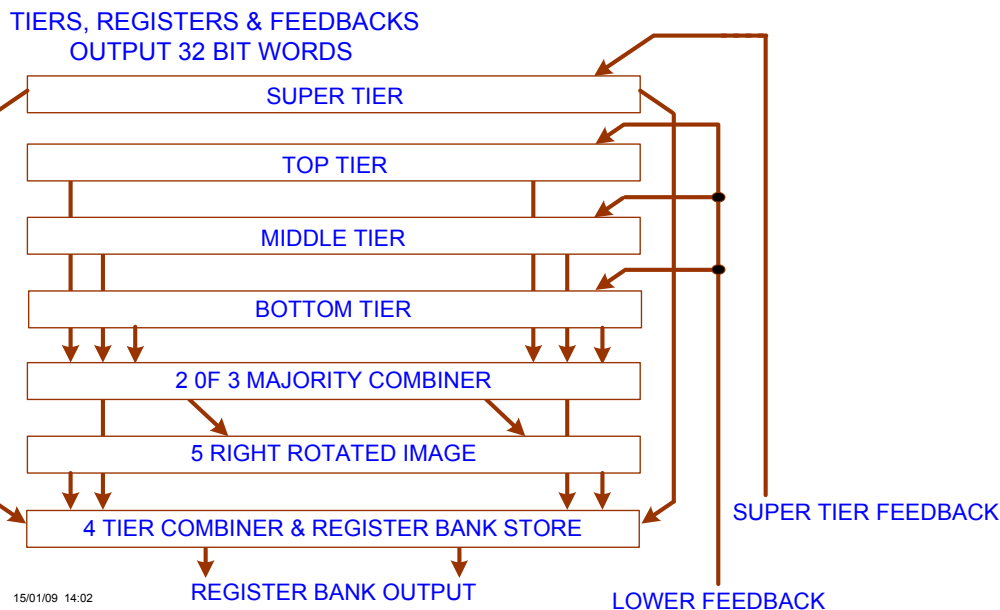


Fig. 9: The Register Bank

The first order number of output variables on each bit of a Super Tier nLFSR is less critical, and consists of the near neighbor "shove in", the single Super Tier Feedback bit, and about a 37.5% probability of an internal feedback, as described in the next two sections. The output of the Super Tier nLFSRs are 7 left rotated, to an "Image" and XOR summed to the Image, therefore the average number of monomials on each output cell of the Super Tier is 4.75; which adds to at least 144 monomials. Therefore, at least 148 monomials are output to each of the 32 inputs of the Data Churn. Stated differently, $148 \times 32 = 4736$ monomials, minimally, multiply the more than 5000 first degree monomials in the output equations of the Data Churn [secure- chap 2].

6. A Tier Architecture:

All four tiers are essentially the same construction, as described in Fig. 10. The Super Tier and its Image are activated on every clock, whereas the TMB (Top Middle and Bottom) Tiers and their Images are randomly clocked. A minimum of two tiers is clocked at every cycle; each TMB Tier is clocked on an average 84.7% of the cycles. A tier that is not clocked is stationary for the un-clocked cycle. The outputs of the TMB nLFSRs are therefore unchanged when a tier is not clocked. The TMB Tiers differential statistics also "benefit" from randomly combining Images to their nLFSR outputs

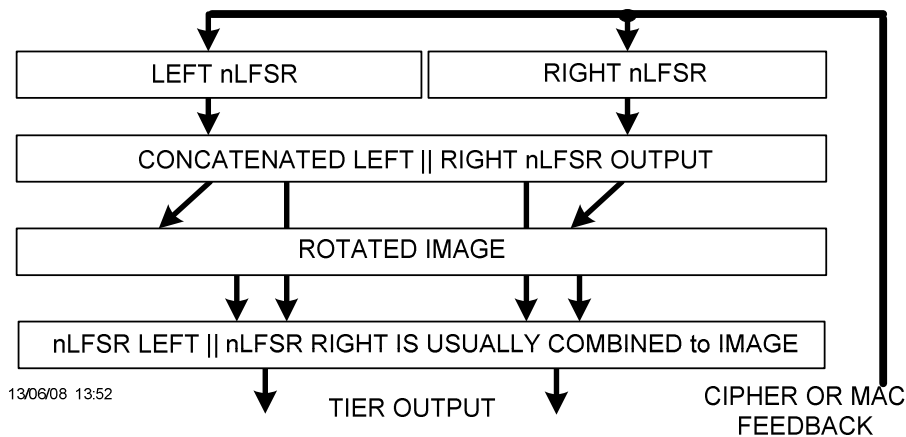


Fig. 10: A ZK-Crypt 32 Bit Tier with two Unique nLFSRs

All tiers receive 32 bit unbiased pseudorandom word inputs from the Feedback/Result processor. Image combining, nLFSR clocking and false internal nLFSR feedback of the TMB tiers is driven by 8 signals from the Random Controller, see Figure 1.

The TMB tier outputs are combined, irrelevant of a tier's being stationary or clocked at a given Primary Clock cycle. XORing the Image to the concatenated nLFSR output reduces to a minimum a trace of the left to right movement of data in the tier nLFSRs.

In each of the four 32 bit tiers as typically described in Fig. 10; there are two unique concatenated pseudo-linear Galois feedback shift registers, nLFSRs, described in the next section. The concatenated output of the nLFSR pair is left rotated into an Image.

The Super, Top, Middle and Bottom Tier Images are generated by left rotations of 7, 1, 3, or 5 bits respectively.

6.1 One to Many (Galois) pseudo-Linear Feedback Shift Registers (nLFSRs)

One to Many nLFSRs are a family of efficient shift register based pseudorandom number generators where feedback from the MS memory cell, see Fig. 11, is fed back into predefined XOR taps located between memory cells. At each clock cycle bits flow from left to right. If the nLFSR feedback is a '1', bits moving from left to right via the XOR tap, are complemented. In a classical nLFSR, a pseudorandom deterministic sequence is generated. In the ZK-Crypt, multi input XOR gates are located between all memory cells, and the natural sequence is aberrated by Lower or Super Tier Feedback XOR summed in the multi input XOR gates. Likewise, Keys and IVs are also be loaded into the TMB (Top Middle and Bottom) Tier nLFSRs via multi input XOR gates.

The most relevant inputs into the One to Many nLFSRs in the Register Bank are the pseudorandom feedback words generated by the Result/Feedback Processor. The Super Tier Feedback is input into the Super Tier, and the Lower Feedback into the TMB Tiers (and also into the Data Churn).

Each ZK-Crypt nLFSR has a unique construction, defined by the number of memory cells and the placement of feedback taps between cells, e.g., 8 cells and 1, 2, 4, 7 feedbacks in Fig. 11. At each activating clock, data content of each cell moves one cell to the right. Note that the MS cell is the rightmost cell. The MS cell output, XORed to a less frequent false feedback Random Controller generated signal, a "Slip", generates a feedback signal into the designated "taps" (affecting the input to adjacent memory cells). For example in Fig. 11, if a present nLFSR stage is 01010100, the feedback would be '0', and the next stage would be 00101010. However, if the Slip caused a '1' feedback, the next stage would be 10011110; it would have changed 4 bits from the normal bits in the stage. The second stage would be distanced on the average $2^{n/2}$ stages from the normal anticipated nLFSR stage.

In the TMB Control Unit nLFSRs in the Random Controller [zk-ccc figs P02-P04], we chose the simpler Many to One configuration, wherein the XOR gates are external, and there are no XOR gates between cells. Here, the feedback bit progresses, unchanged from left to right. Because of the short length of the registers, and the sampling logic, the results are similarly unpredictable; in each case the Random Brown signals are a function of the three last nLFSR slip/MS bit feedbacks.

The construct of the Register Bank "One to Many" pseudo Linear Feedback Registers, nLFSRs, is similar to non-extended LFSRs where, here, the internal nLFSR state is not a deterministic sequence of stages linear dependent on the previous state of the memory cells. The hi-entropy non-biased random Hash/MAC or Cipher Feedback which is XOR summed to the memory cells assures that all cells will not suffer from the "Stuck on Zero" syndrome. The Super Tier and Lower Feedback values which are linearly summed to the tiers, surpass any normal analysis of the nLFSR contents.

Note that if there were no random external feedback, e.g., as in the Random Controller's nLFSRs, and the output of each cell were '0', at the next clock cycle all cells would remain in the all zero stage, i.e., "Stuck on Zero". In the 3, 5 and 6 celled nLFSRs of Top, Middle and Bottom Control Units of the Random Controller [zk-fig 13-15], therefore erratic Slip pulses may provoke a stuck on zero syndrome in an nLFSR, causing a dormant all zero state for a number of clock cycles. In each Random Controller nLFSR we input the output of the n-1 left hand cells into a NOR- NFIX gate, such that the NFIX gate generates a '1' if all n-1 cells are '0'.

THE pseudo LINEAR FEEDBACK SHIFT REGISTER nLFSR CONCEPT

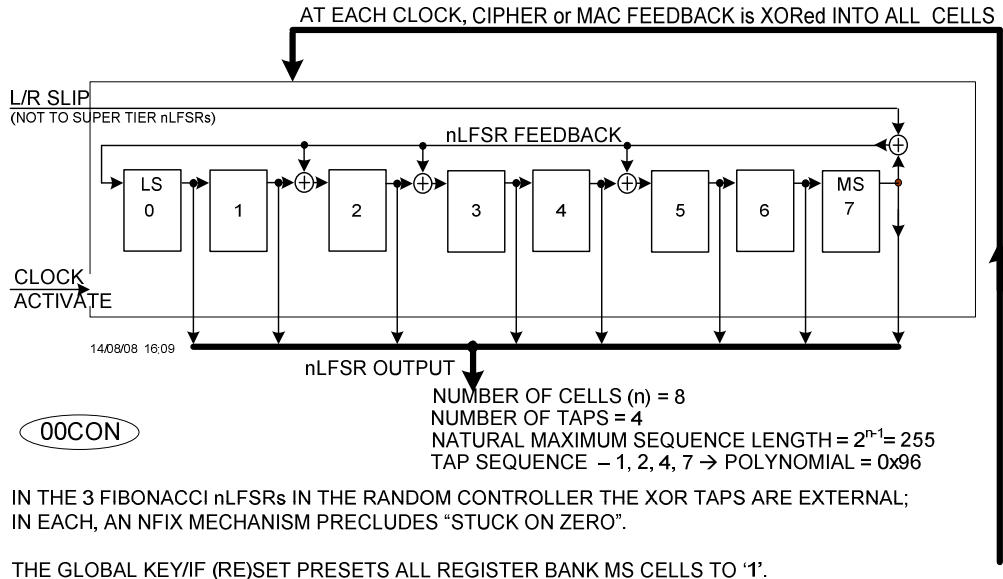


Fig. 11: A ZK-Crypt Type "One to Many" pseudo Linear Feedback Shift Register, nLFSR

The Global Key/IV (Re)set presets all Register Bank nLFSRs' MS cells to '1', to assure immediate wake-up (no stuck on zero syndrome) of the Word Manipulator, if a user chooses to use the Global (Re)set condition as an initial condition. In Table 1 we show the construct of all of the nLFSRs in the Register Bank.

Table 1: Configuration of the Register Bank's "One to Many" nLFSRs

	Left Hand		Right Hand	
Tier	# Cells	Tap Configuration	# Cells	Tap Configuration
Super	16	0 2 5 6 10 11 12 15	16	5 8 11 15
Top	13	2 3 5 8 9 12	19	1 4 6 7 8 9 11 14 16 18
Middle	18	2 4 6 7 10 11 12 13 15 17	14	1 4 5 8 10 13
Bottom	15	0 1 5 6 10 14	17	1 4 7 9 10 12 13 16

7. The Data Churn:

In Fig. 12, we show how the Data Churn receives: a) the combined output of the Register Bank; b) two rotated versions of the Lower Feedback; c) 4 rule Splash Matrix vector displacement selectors; and 4 EVNN correlation MAJ filter "correlation forcers". After 9 levels of processing the Data Churn outputs the Cipher Mask and four 32 bit words into the Result/Feedback Processor, see Figs. 2 and 1.

There are four Store & XOR filters in the Data Churn. At clocked cycles, each Store & XOR receives a 32 bit data vector from one (or several, in the case of the Cipher Mask) level above, and stores the contents in its 32 memory cells, see Fig. 12. At the next clock cycle the output is a previous XOR input value XORed with a present input value. This memory stored previous bit output XORed to the present input effectively decreases correlation and bias between the near neighbor bits in the sequence of words.

7.1 The Splash Displacement Matrices

Interspersed between the TMB levels of the Store & XOR/ MAJ/3XOR Hybrid Complexes in Fig. 12 are the Top and Bottom Splash Displacement Matrices. Each matrix is composed of four row displacement vectors. The A, B, and C vectors each signify a different pseudo-random displacement rule for directing input bits into the output word. The D vector causes the input word to be output without change; i.e., the identity rule vector.

At each clock cycle, the Splash (Rule) Selector in the Random Controller enables one vector from the Top Matrix, e.g., the A vector; and a different vector; i.e., the Bottom B Matrix vector. In

Fig. 13, the B rule vector is enabled, input bit I14 is displaced to output Index 27 by B27 (in column 14). Ideal pseudo-random sequence vectors were altered slightly to assure that an index i'th input will "exit" a different j'th output for each of the rules.

The Splash Select is a function of the Noise Source, [zk-ccc fig P06] the 2 previous Splash Selector memory outputs; and the Index 15 and 31 outputs from the Top Splash Matrix. The four selectable Rule Pairs are A & B, B & C, C & D and D & A; in each case for the Top and Bottom Splash Matrices respectively.

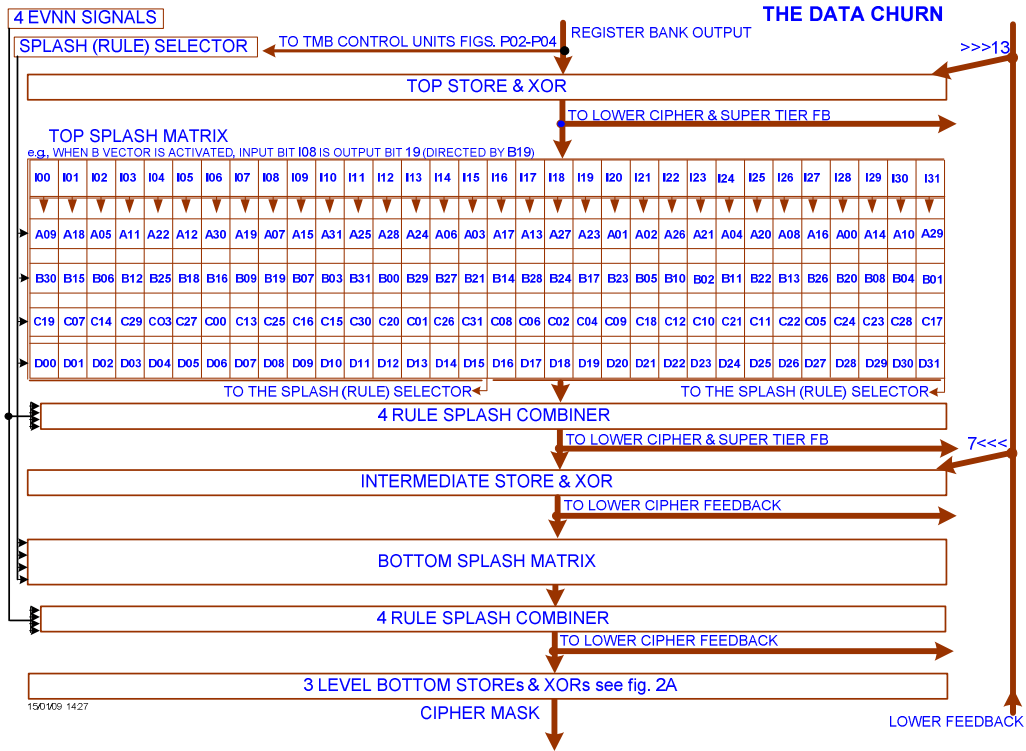


Fig. 12: The Data Churn with Explicit Top Splash Matrix

The output of the Top Splash Matrix is input into the Top 4 Rule Splash Combiner. If the B vector is selected, four inputs into the Splash Matrix, I01, I06, I16 and I19 are displaced to be near neighbors via B14, B15, B16 and B17. These four near neighbors, in Fig. 13 are input into MAJ cell 16 of Figs. 13 and 14. The 4th Toggle EVNN signal is the correlation forcer of MAJ16.

We assume, with reasoning [secure chap 3], that the displacements in the Splash Matrix will reduce correlation between near neighbor bits in the outputs, and supplement the other diffusion processes.

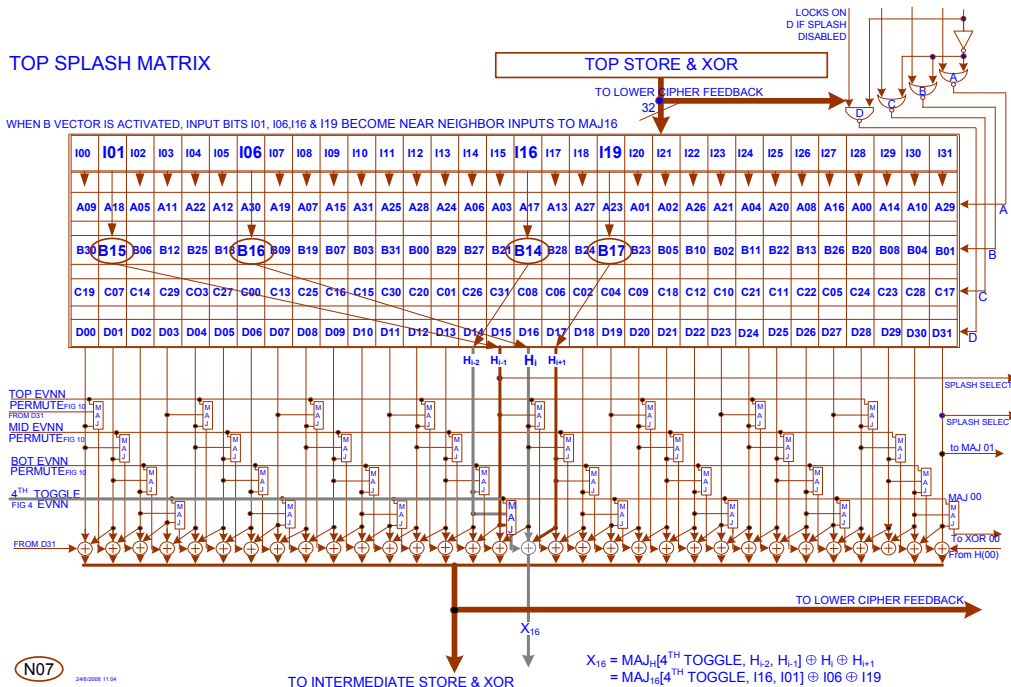


Fig. 13: Top Splash Matrix/4 Rule Combiner- 5 Disparate Binary Variables into one Store & XOR input

Tests have shown that if both the Top and Bottom Splash Matrices are locked on D, the identity vector, (which does not displace input data) the statistics are still very good (and comparable to the standard configuration). We assume that the "Lock on D" mode will be useful with deployed ZK-Crypt software on legacy systems.

7.2 Store & XOR/ Hybrid MAJ/3XOR Complex

In Figs. 13 and 14 we see how four diffused near neighbor outputs from the Top Splash Matrix and one permuting signal from the Random Controller diffuse, with one feedback bit into one cell of the Intermediate Store & XOR. At each clock cycle the output is a diffusion of eleven binary variables. The complex is composed of two subsystems: a MAJ/3XOR filter and the Store & XOR combiner.

The MAJ/3XOR filter receives four inputs, in this example, from the Top Splash Matrix, and one of four EVNN signals from the Random Controller. The two "left hand" Splash Matrix variables ($H_{i-2,t}$ and $H_{i-1,t}$) and the EVNN variable from the "Random Controller" are the input to the non-linear MAJ gate. The $H_{i,t}$ bit from the Splash Matrix, its right hand near neighbor, $H_{i+1,t}$, and the output of the MAJ gate are combined in the 3XOR gate; to create the MAJ/3XOR filter 5 variable result.

The Intermediate Store & XOR receives three inputs, the two last MAJ/3XOR outputs and a 7 left rotated Lower Feedback bit. The previous (t-1), MAJ/3XOR output XORed to the feedback bit is saved from the last cycle in the memory buffer, to be output on the next clock cycle.

The XOR of the Store & XOR, the right hand XOR gate, combines the present MAJ/3XOR filter output (a function of five variables) with the Intermediate Buffer's output (from the last clock cycle, a function of six variables) to generate an output bit comprising 11 diffused variables. Due to the Splash Matrix bit displacements, neighboring inputs to the MAJ and 3XOR are a changing mix of near neighbor variables.

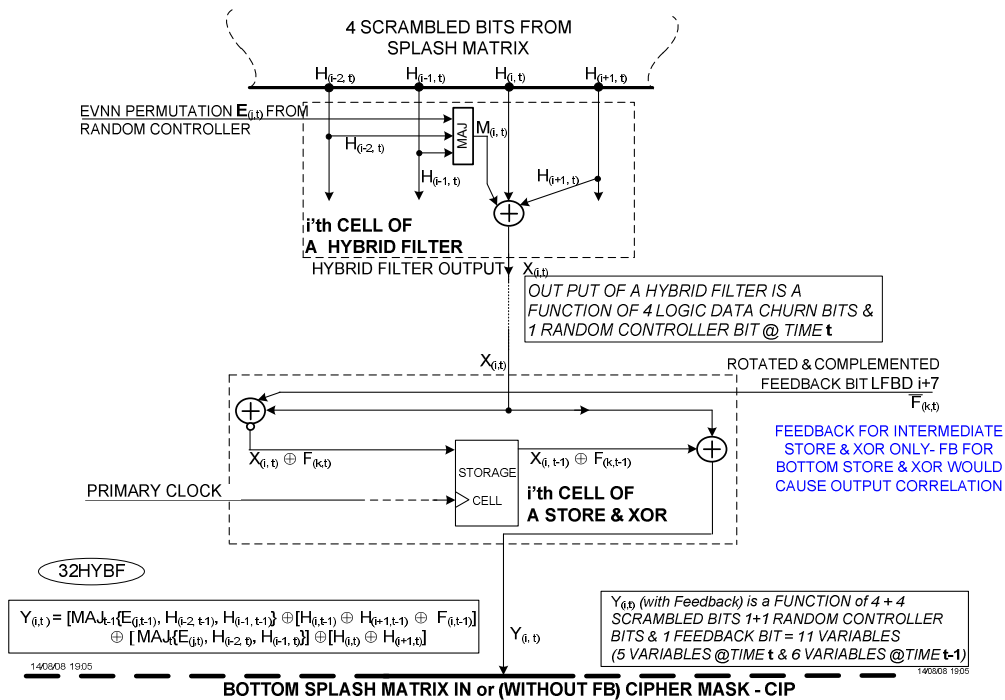


Fig. 14: The MAJ/3XOR Filter/ Store & XOR Complex

The 8 or 9 output bits output from the Register Bank which are functions (the equations of) one Message bit subsequently affect an increasing number of variables as the logic bits "descend" and diffuse into new near neighbors at each level of the Data Churn. We have shown [zk-secure appx 4] that one Message bit affects virtually all Result bits in the first cut, and a minimum of over 140 binary variables in the Word Manipulator.

Each bit leaves an immediate trace in the tiers of the Register Bank, in three Store & XOR memories, and in the Result Store. The affect of each Hash/MAC Message bit on the 61 variables of the Random Controller is less immediate, but no less troublesome to a potential attacker. The counters in the TMB (Top, Mid and Bottom) Control Units [zk-ccc figs P02-P04] each process feedback signals, from the output of the Register Bank, on an average of about $1/7$ of the Primary Clocks. The TMB Control Unit nLFSR Slip signals each process Slip feedbacks close to 50% of the Primary Clock occurrences. The Splash vector select is active on every Primary Clock cycle.

8. The Result/Feedback Processor:

The Result/Feedback Processor inputs 3 or 5 intermediary 32 bit words from the Data Churn; in Hash/MAC and Cipher mode, respectively. The 64 Bit Output of the HAIFA Counter is dispersed into two 32 bit words which are linearly combined (so as not to loose entropy) to the Lower and Super Tier Feedback Stores. In all modes the processor generates a Result, which is an XOR sum of the Cipher Mask output and a Message Word. In many phases, i.e., when the engine is scrambling the internal contents, the Message Word, is locked on zero.

Hash/MAC mode Hashing, also called Message Digesting is a process wherein each Message Word must leave an indelible unique "foot print" in the ZK-Crypt engine via the two orthogonal feedback tracks, the Lower and the Super Tier Feedbacks. Generically, we provide proof that the two tracks, each feeding different tiers of the Register Bank, are orthogonal if only one track is a function of the Present Result and the second track is a function of an XOR summing of the Present Result with the Previous Result [zk-fbpat3]. In addition, we salt the Super Tier Feedback with the SuperMIX displacement of the XOR sum of two intermediary "statistically proven quality" pseudo random words.

Although the bias statistics of the Super Tier Feedback (input to the Super Tier) are already excellent, the functions are Linear. Therefore to delinearize and internally decorrelate, the SuperMIX component of the feedback which is the XOR sum of two internal hi-entropy Data Churn state words. The output of these words are delinearized, decorrelated, and randomized in the Serpent bijective S Box; wherein the output is subsequently rotated and each nibble is reversed. Nibble bits A, B, C&D, in nibble [ABCD] are reversed into nibble [DCBA]).

8.1 Feedback Modes in the 32 Bit Word Manipulator:

There are two strategies of word feedback in the ZK-Crypts, Cipher Mode and Hash/MAC mode. The Hash/MAC Mode feedback must be a function of the Cipher Mask and the Message Word in order to diffuse every bit of each Message as quickly as possible into the ZK-Crypt variable equations. The Cipher Mode feedback(s) are displacement transformed internal Data Churn word variables.

Both Cipher and Hash/MAC Feedbacks are output (recirculated) to the four tiers of the Register Bank (not rotated) and to the Top Store & XOR and to the Intermediate Store & XOR (right rotated 13 bits and left rotated 7 bits, respectively), see Fig. 15.

The Dual Track Feedbacks with dispersed HAIFA Counter output enhancements add diffusion in both Hash/MAC and Cipher modes, preclude herd collision in Hash/MAC mode and obviate false word reconciliation in Data Authentication.

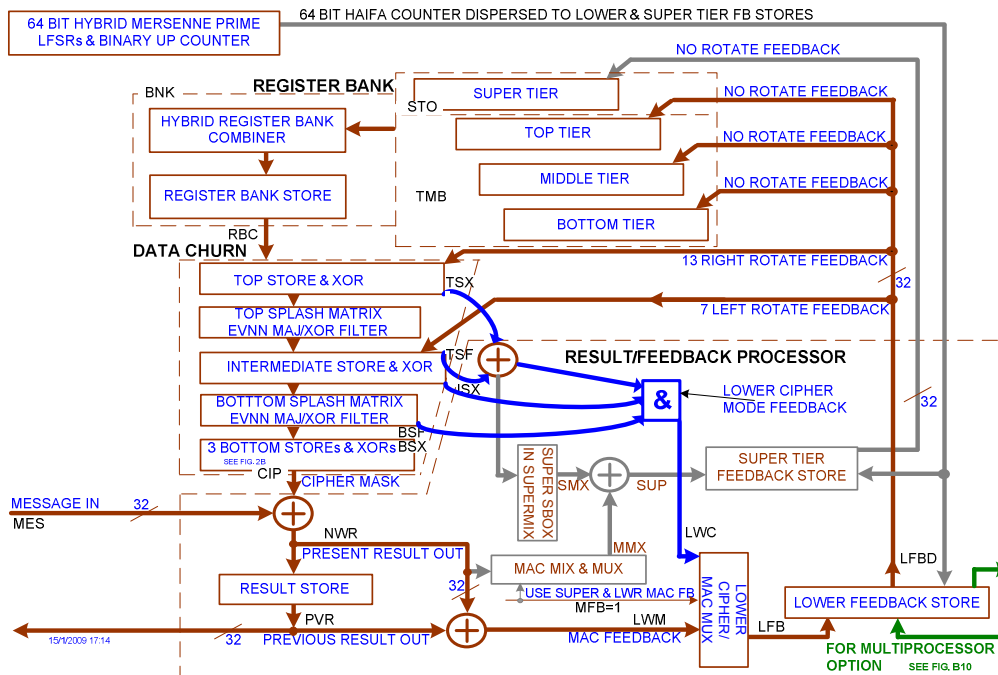


Fig. 15: The ZK-Crypt Feedback Strategies B01A

8.2 ZK-Crypt Hash/MAC and Cipher Feedbacks

Dual Track feedback streams significantly enhance the Data and Cipher Authentication functions by:

- recirculating dense feedback in Cipher Mode without degrading random statistics, thereby increasing the cipher's crypto-complexity;
- recirculating to the Super Tier a (proved) linearly debiased feedback word, assuring that the Super Tier's output will effectively mask and debias the TMB Tier's non-linear combined output;
- virtually assuring that there can be no leakage of knowledge from the TMB Tiers;
- causing a linear debias of every XORed word equation "descending" each level of the Data Churn;
- assuring that any false bit or word in a Hash/MAC Message Word will cause uncontrollable aberrations in (either or both) the TMB Tiers and the Data Churn, or in the Super Tier, and,
- incorporating into both feedback streams a deterministic scramble of all 64 bit outputs from the HAIFA Counter, further adding to the proved impossibility of reconciling a falsified Previous Result, see [zk-secure appx 2]

Both modes of Super Tier feedbacks include the SuperMIX transformation based on the Serpent S Box transformation followed by the Nibble Reversed transformations of permuted Data Churn words; e.g., a Nibble Reversal of bits ABCD is DCBA. The MMX transformation on the Result word, used in all

Hash/MAC mode functions reverses all 8 nibbles without changing the positions of the Reversed Nibbles.

8.3 ZK-Crypt Cipher Feedback (see nomenclature in Fig. 15):

In Cipher Mode, the Feedback to the Word Manipulator cannot be a function of the Message Word, therefore, see Fig. 16, the MAC MIX output, MMX, is set to zero, and the input to the Super Tier Feedback Store, SUP, is equal to SMX.

Table 2: Reverse Nibble Serpent Bijective S Box Transformation in the SuperMIX Super Tier FB

REVERSED NIBBLE SERPENT S BOX

```
SR0[16] = {12, 1, 15, 8, 5, 6, 10, 13, 7, 11, 2, 4, 14, 0, 9, 3};
SR1[16] = {15, 3, 4, 14, 9, 0, 10, 5, 8, 13, 7, 1, 6, 11, 12, 2};
SR2[16] = { 1, 6, 14, 9, 12, 3, 5, 15, 11, 8, 7, 2, 0, 13, 10, 4};
SR3[16] = { 0, 15, 13, 1, 3, 9, 6, 12, 11, 8, 4, 2, 5, 14, 10, 7};
SR4[16] = { 8, 15, 1, 12, 3, 0, 13, 6, 4, 10, 2, 5, 9, 7, 14, 11};
SR5[16] = {15, 10, 4, 13, 2, 5, 9, 3, 0, 12, 7, 1, 11, 6, 14, 8};
SR6[16] = {14, 4, 3, 10, 1, 2, 6, 13, 7, 9, 8, 15, 11, 12, 5, 0};
SR7[16] = { 8, 11, 15, 0, 7, 1, 4, 13, 14, 2, 3, 5, 9, 12, 10, 6}.
```

In Fig: 16 we show the explicit SuperMIX S Box transform of the LS reversed nibble output of the XOR sum of the i'th sum of Top Store & XOR 0'th nibble, TSX_{i0} , and the 0'th nibble of Top Splash Matrix Filter, TSF_{i0} :

$$SBX_{0i0} = f_{sbi0}[TSX_{i0} \oplus TSF_{i0}],$$

and the i'th rotated and reversed nibble output:

$$SUP_{i0} = f_{RNx0}[SBX_{0i0}].$$

The i'th Cipher Feedback to the Super Tier is therefore the SuperMIX transform on the outputs of the Top Store & XOR and the Top Splash Matrix Filter:

$$SUP = f_{STMX}[TSX_i \oplus TSF_i]$$

If we define the 32 word input bits to the SuperMIX reverse nibble, RNx, transformation-
[ABCD EFGH JKLM NPQR STUV WXYZ abcd efgh];

then the displacement transform, f_{RMX} , with reversed and rotated nibbles-

$$f_{RNx}[ABCD EFGH JKLM NPQR STUV WXYZ abcd efgh]; \text{ outputs the displacement-} \\ SMX = [dcba hgfe DCBA HGFE MLKJ RQPN VUTS ZYXW].$$

This dense feedback (an average of 16 '1's in each feedback word) is an effective non-linear, decorrelating randomizing input to the Super Tier which masks the non-linear 3 Tier combiner output and assures virtually no "leakage" from the unobservable secured vault in the Register Bank.

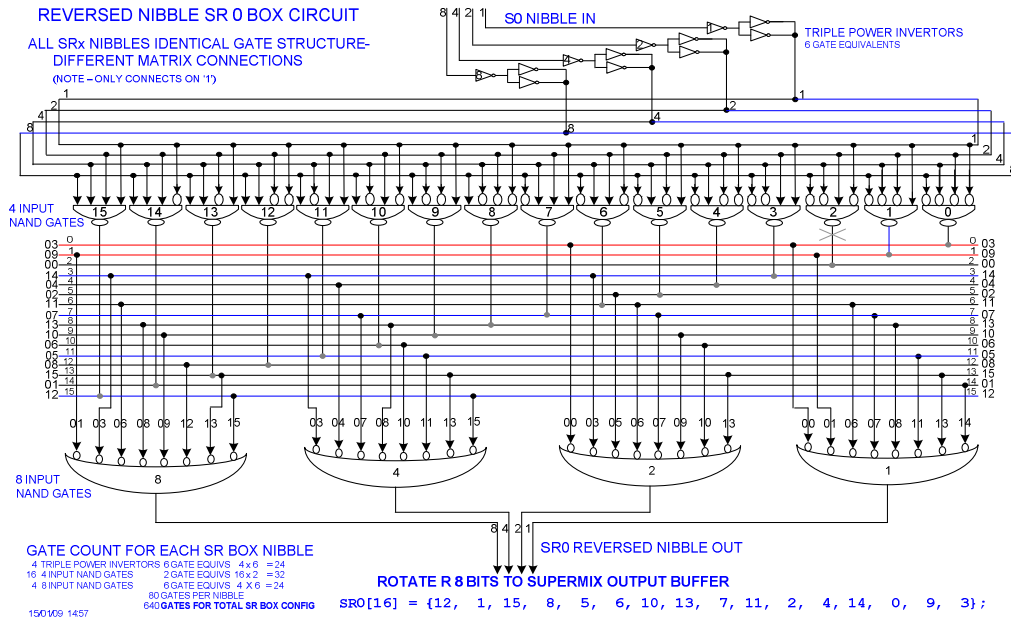


Fig. 16: The Bijective Serpentine S Box Transformation on the LS input Nibble

In the ZK-Crypt Cipher Mode, LWC, the Lower Cipher Word is simultaneously recirculated into the TMB Tiers and the Data Churn:

$$LWC_i = (ISX_i \oplus BSF_i) (TSX_i) (TSF_i) \text{ as shown in Fig. 15.}$$

8.4 ZK-Crypt Hash/MAC Feedback

If we designate the 32 word input bits to the MAC MIX, MMX, transformation-

$$[ABCD EFGH JKLM NPQR STUV WXYZ abcd efgh];$$

then the MMX displacement transform, f_{MMX} with reversed nibbles-

$$f_{MMX}[ABCD EFGH JKLM NPQR STUV WXYZ abcd efgh] \text{ outputs the displacement-}$$

$$MMX = [DCBA HGFE MLKJ RQPN VUTS ZYXW dcba hgfe].$$

The generated Result Word, NWR_i , in both ciphering and data authentication is the XORed sum of the Message Word, MES_i , and the Cipher Mask, CIP_i . At every Primary Clock cycle, in all operations, the Result Word is sampled into the Result Store. Therefore, at the present clock cycle, the output of the Result Store, PVR_i , is the NWR_{i-1} value; so that:

the Lower Feedback is composed of:

$$NWR_i = CIP_i \oplus MES_i; \text{ and } PVR_i = (CIP_{i-1} \oplus MES_{i-1}) \oplus \text{a HAIFA dispersion.};$$

and the i'th Lower Feedback to be recirculated to the Data Churn and to the TMB Tiers:

$$LWM_i = (NWR_i \oplus PVR_i) = (CIP_i \oplus MES_i) \oplus (CIP_{i-1} \oplus MES_{i-1}) \oplus \text{a HAIFA dispersion}$$

Simultaneously, the Super Tier is fed the SMX, the Super Tier Cipher Feedback, XORed to the MMX nibble transformed Present Result (a function of the Message Word):

$$SUP_i = f_{SMX}[ISX_i \oplus BSF_i] \oplus f_{MMX}[CIP_i \oplus MES_i],$$

Therefore, if $MFBI=1$, the two tracks of generated ZK-Crypt feedback at step i where:

$$LWM_i = (NWR_i \oplus PVR_i) = (CIP_i \oplus MES_i) \oplus (CIP_{i-1} \oplus MES_{i-1}) \oplus \text{a HAIFA dispersion}$$

is the Hash/MAC feedback linearly combined to the TMB Tiers and the Data Churn, and,

$$SUP_i = f_{SMX}[ISX_i \oplus BSF_i] \oplus f_{MMX}[CIP_i \oplus MES_i] \oplus \text{a HAIFA dispersion}$$

is the Hash/MAC feedback which is input (only) into the Super Tier.

9. The ZK-Crypt Random Controller – Fig. 17A depicts the interaction of the 3 Control Units, the noise source, the output permutation encoder, the Register Bank and the Data Churn. The Random Controller generates 14 permutation and clocking signals to the Register Bank and to the

Data Churn. The Controller receives 10 balanced feedback signals from the output of the Register Bank and the Data Churn.

The ZK-Crypt Random Controller clocks and drives the permutations in the Register Bank; and regulates the matrix displacements and the 2 of 3 Majority function permutations in the Data Churn.

The Random Controller is designed to be remotely and intermittently affected by the Register Bank and the Data Churn, such that sensing any permutation activity in either the 32 Bit Word Manipulator or the Random Controller would be difficult to correlate to, or leak knowledge of, the internal processed data.

The Random Controller consists of:

- the Deterministic Noise Source which generates 3 binary signals affecting the other modules, and the (P)Random clock which drives the three Control Units;
- the Top, Mid(dle) and Bot(tom) (TMB) Control Units, each of which generates 4 signals whose principal tasks are to remotely affect the configurations and permutations of the Register Bank and the Data Churn;
- the Activation, Permutation & Control Encoder, the AP&C Encoder, which is a buffered output block of random logic which translates the logic outputs of the TMB Control Units into all of the clocks and permutations which affect the Register Bank; and,
- the Splash Selector which regulates the displacement vectors in the Splash Matrices.

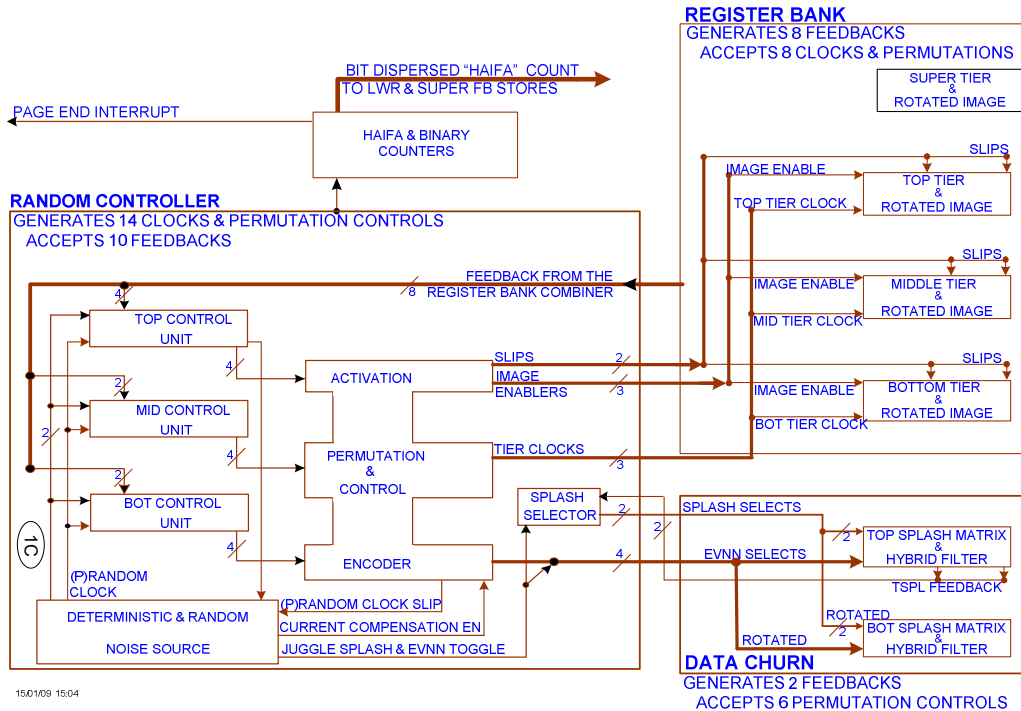


Fig. 17A: The Random Controller Drives the Register Bank & Data Churn (not the Result/Feedback)

The Pseudo Random Noise Source, is a complex of nLFSRs, Store & XOR filters and random logic. The nLFSRs and the filters are clocked by the Host's Primary Clock, which is also the Sample command of the Data Churn and Result/Feedback Processor. The Noise Source outputs 3 unbiased signals and the (P)Random Clock (a random missing clock pulse). The (P)Random Clock drives the Random Controller TMB Control Units, only [zk-ccc figs P02-P04]. The (P)Random Clock, and all internal Tier clocks are synchronized to the Host's Primary Clock. The (P)Random Clock "passes on" an average of about 11/12 of the Primary Clocks to the Control Units; each of the AP&C Encoder generated Register Bank TMB Tier Clocks randomly replicates about 5/6 of the Primary Clocks to each of the TMB Tiers (either 2 or 3 TMB tiers are activated at each clock).

Each TMB Control Unit consists of an interacting Programmable Up-Counter and a different length nLFSR; 3, 5 and 6 celled respectively. At each (P)Random Clock, the nLFSR generates unbiased signals, which may or may not affect the up-counter and/or the AP&C Encoder. At count 15 the

up-counter resets to a new start count; i.e., $4 \leq \text{start count} \leq 15$. The start count is determined by internal logic and feedback from the Register Bank. At count "15", a Control Unit counter trips, and via the AP&C Encoder, typically generates a new configuration of TMB Tier Clocks, TMB Tier Image Controls; and activates Left or Right TMB Tier Slip signals. The TMB Control Units receive 8 binary Feedback streams, emanating from the output of the Register Bank, and 2 logic bits from the Random Controller's Noise Source. These input bits to the Control Unit affect the start count of the Programmable Up-Counter and serve to generate Slip permutations to the Register Bank's nLFSRs.

The LS cell output of the Top Control Unit's Up-Counter serves as a debiasing signal, Q_{TA} , for the 3 outputs of the Deterministic Noise Source. The LS counter cell is toggled (complemented) at every (P)Random clock, except for the "after count 15" programmed start, when the LS bit may be set to either '0' or '1'. This generates a stream of toggled zeroes and ones, e.g., 0101, which on an average of about one in thirteen Primary Clocks, will generate a double one, e.g., 1010110101....; causing a slightly biased "debiasing" signal, operative to be XOR summed to uncorrelated internal variables, which typically lowers bias while increasing unpredictability of permutation signals.

The Top, Mid and Bot (TMB) Control signals are fed forward to the AP&C Encoder, which re-encodes the signals and:

- changes the Tier Image configurations at Host generated clock cycles, [zk-ccc P08];
- with a probability of 0.5 does not clock one of the TMB Tiers, e.g., the un-clocked Tier's nLFSRs (only) are static and the tier outputs previous stored data; and
- about once in four clocks generates a Left and/or a Right Slip signal to aberrate the left and/or the right hand TMB Tier's nLFSRs.

In the ZK-Crypts Data Authentication and Cipher modes, all driving clock signals (rising edge of a clock pulse) are either the Host's generated Primary Clock or derivations thereof.

The Host generates a clock pulse (the Primary Clock) that, at arbitrary instants, steps the ZK-Crypt engine; e.g., typically processes a Message Word and may output a Result word; i.e., in Cipher Mode. The Control Units in the Random Controller are clocked by the (Pseudo)Random Clock pulses generated by the Noise Source, as explained previously.

9.2 How the Random Controller Drives the Register Bank:

The Super Tier is not affected by any of the Random Controller's permutation or clocking signals. The Super Tier's nLFSRs are clocked at every Primary Clock, and the rotated Image is XORed to the outputs of the Super Tier's nLFSRs, at every clock. The designers view the Super Tier as a randomizing mask of virtually every word variable in the Data Churn. The Hash/MAC mode ZK-Crypt Super Tier Cipher Feedback is a linear, XORed, function of four statistically uncorrelated (permuted to be uncorrelated) word variables, virtually assuring (and proved) that the output of the Super Tier has hi-entropy statistics. Note that a linear derivation of the Super Tier's output appears linearly in the input equation of every state of the Data Churn.

The TMB Tiers are regulated by the following aberrations [zk-ccc fig. P08]-

- Right and Left Hand nLFSR Slip Signals (see Fig. 11). On an average of about once in 4.7 Primary Clocks, a Left Slip pulse complements the Left hand TMB Tier's 3 nLFSRs internal feedbacks; likewise the Right Slip aberrates the 3 Right hand TMB Tier's 3 nLFSR's internal feedbacks;
- Top, Middle and Bottom Brown (BRN) Image Select. [zk-ccc fig P08]. The Brown Image Select activates the bit wise XOR sum of a tier's nLFSRs' outputs and its rotated Image. When a tier's Brown Image is not activated, the tier's output is the concatenated left and right hand pair of nLFSRs' output, only; i.e., not affected by the Image.

The three Brown Image Selects are never activated simultaneously. Randomly, one or two of the TMB tiers' Images are selected at each clock cycle. The nLFSRs of each tier are clocked by the AP&C Encoder. At each Primary Clock; with a probability of slightly less than 0.5, one of the 3 TMB Tiers is not clocked.

- A random missing clock cycle "unpredictably stalls" one of the three tiers, (whose output, therefore, remains unchanged).

9.3 How the Random Controller Drives the Splash Matrices:

At each Primary Clock, the Splash Matrix selector designates uniquely, in both the Top (see Fig. 13) and Bottom Splash Matrix, one of four displacement vectors. The valid vector choices are:
AB, BC, CD, and DA, for the Top and Bottom Splash Matrices, respectively.

9.4 How Binary Feedbacks from the 32 Bit Word Manipulator Affect the Random Controller:

- a) 8 designated index bits of the Register Bank's 4 Tier Combiner are fed back to the Random Controller's Top, Middle and Bottom Control Units. [zk-ccc- figs. F01 to figs P02-P04].
- c) Index bits 15 and 31 (TSPL-15 & 31) from the output of the Top Splash Matrix in Fig. 12 are fed back to the Splash Control Selector.

9.5 A More Detailed Description of the Random Controller's Components:

- a) The Top, Middle and Bottom Control Units each consist of a unique nLFSR coupled to a 4 bit pseudo-random programmed Up-Counter, not shown, see [zk-ccc- figs. figs P02-P04].
- b) The TMB Control Units output signals which affect AP&C Encoder outputs, see Fig. 17 relating to the Brown (BRN) Images (the Left Rotated Images of Fig. 4); relating to the R/L Slips of the TMB Tiers in the Register Bank, see Figs. 11 & 16; and, relating to the internal configuration of the AP&C Encoder [zk-ccc fig P01]; and, relating to 3 of the 4 EVNN toggles each of which uniquely affects 8 bits in each of the MAJ/XOR Filters of the 4 Rule Splash Combiners.
- c) The Deterministic "Noise Source" emits three highly unpredictable bit signals and one (P)Random clock signal which activates the 3 Control Units in the Random Controller, see Fig. 16:
the Juggle Splash Toggle which affects the Splash Selector;
the 4th EVNN Toggle affects 8 bits in the 4 Rule Splash Combiners, Figs. 13 and 14; and,
the (das- not shown) Slip Toggle affects the L/R Slip decision in the Control Units.
- d) The "Noise Source" accepts the (P)Random Clock Slip to its nLFSR, and a debiasing toggle from the Top Control Unit. [zk-ccc – figs. N00-N03]
- e) The AP&C Encoder accepts permutation signals from the TMB Control Units and re-encodes all of the clocking and permutation logic which are generated to the TMB Tiers and their outputs and to 3 of the 4 EVNN signals feeding the Top and Bottom Splash Combiner Filters. [zk-ccc- figs. P06-P09]
- f) In the TRNG mode of operation, the Physical Random Noise Source is an autonomous random ring oscillator emitting the same nominal signals as c). [zk-ccc fig. N04-N07].

10. The 64 bit composite Mersenne Prime LFSR & binary HAIFA Counter – The 64 bit composite counter, subsequent to initialization, uniquely records in the orthogonal feedback streams a new unique number at each generation of the Cipher Mask. See Fig. 17B.

In both the Hash/MAC and the Cipher mode of operation, the HAIFA counter's output bits are XOR summed to the Lower and Super Tier Feedback Stores.

The unique 64 pseudo random count bits preclude herd collision in the first (more than) 2^{57} 32 bit inputs by marking each chaining value with a unique number in the immediate "Message expansion" into the chaining value. There is no compression (loosing entropy) at every clock cycle. Finding a collision where the colliding Message words can reconcile the new count numbers is impossible; as each 32 bit Message cannot reconcile each new 64 random change in the ZK-Crypt's chaining value.

In both the Hash/MAC and the Cipher mode of operation, the HAIFA counter's 64 output bits are XOR summed to the Lower and Super Tier Feedback Stores.

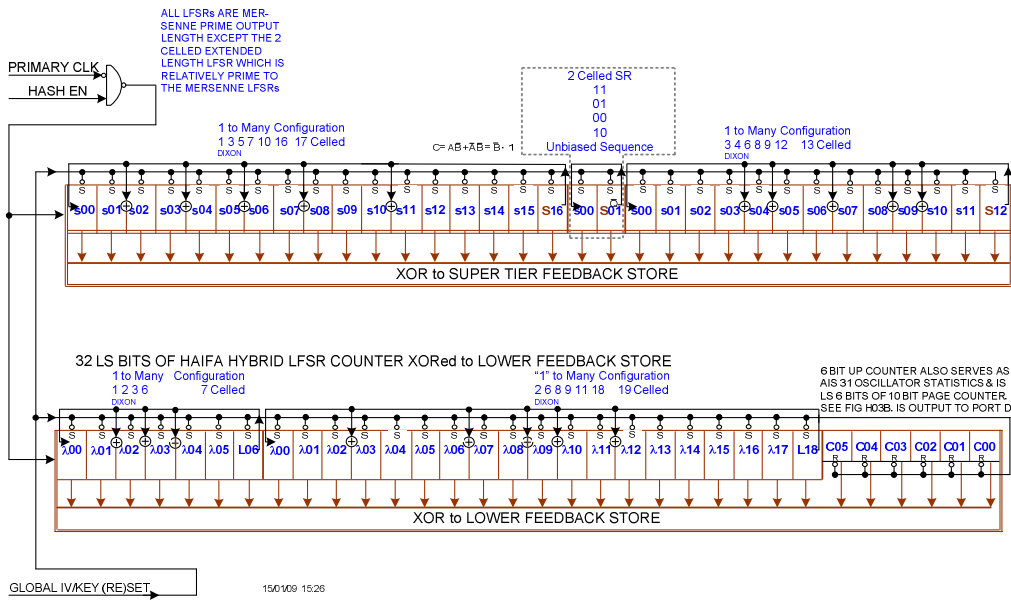


Fig. 17B: The Composite Mersenne Prime LFSRs and Binary HAIFA Counter

11. Increasing Cryptocomplexity-Parallelizing, Scrambling and Multi-Step Wait & Read ZK-Crypt engines can be concatenated for speed and higher security, or alternately may work side by side to simultaneously de/encrypt and authenticate, as shown in Figs. 4-8. Concatenation multiplies bit rate, and exponentially increases security without increasing expended energy per processed bit. To increase cryptocomplexity, in many instances we use the "scramble" operation. A scramble is a simple hashing into the engine via the dual track feedback streams of the previous Cipher Mask output value. Similarly, the Multi-Step Wait & Read function uses a defined number of Scrambles between a Message input and/or Result to emulate increasing rounds of a Block Cipher, which must be programmed by the host (not an FSM command).

11.1 Parallelizing ZK-Crypt Engines- In Figs. 4-6 we showed configurations of paired engines working side by side, for transparent 32 bit data flows, where one engine de/encrypts and its mate simultaneously Hash/MAC digests the same original data as cipher text.

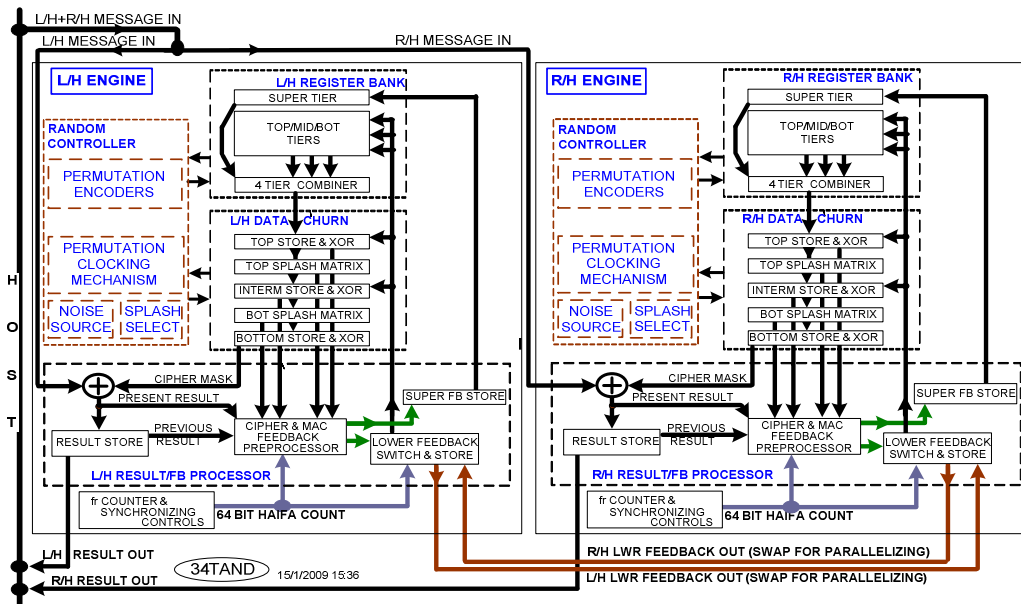


Fig. 18 Switching L/H and R/H Lower Feedback Manipulations in Paired Concatenations

In Fig. 18 we see the switching configuration for paired swapping of Lower Feedback, wherein Lower Feedback from the R/H and L/H engines are switched and swapped. n ZK-Crypt engines can be parallelized to linearly increase word size and exponentially increase cryptocomplexity, without increasing energy per processed bit. The hardware link between adjacent cores is the Lower Feedback stream.

For $n=2$, Lower Feedbacks are swapped; e.g., the generated left hand Lower Feedback stream is switched into the R/H Lower Feedback stream, and vice versa. Fig. 19 depicts an n engine configuration.

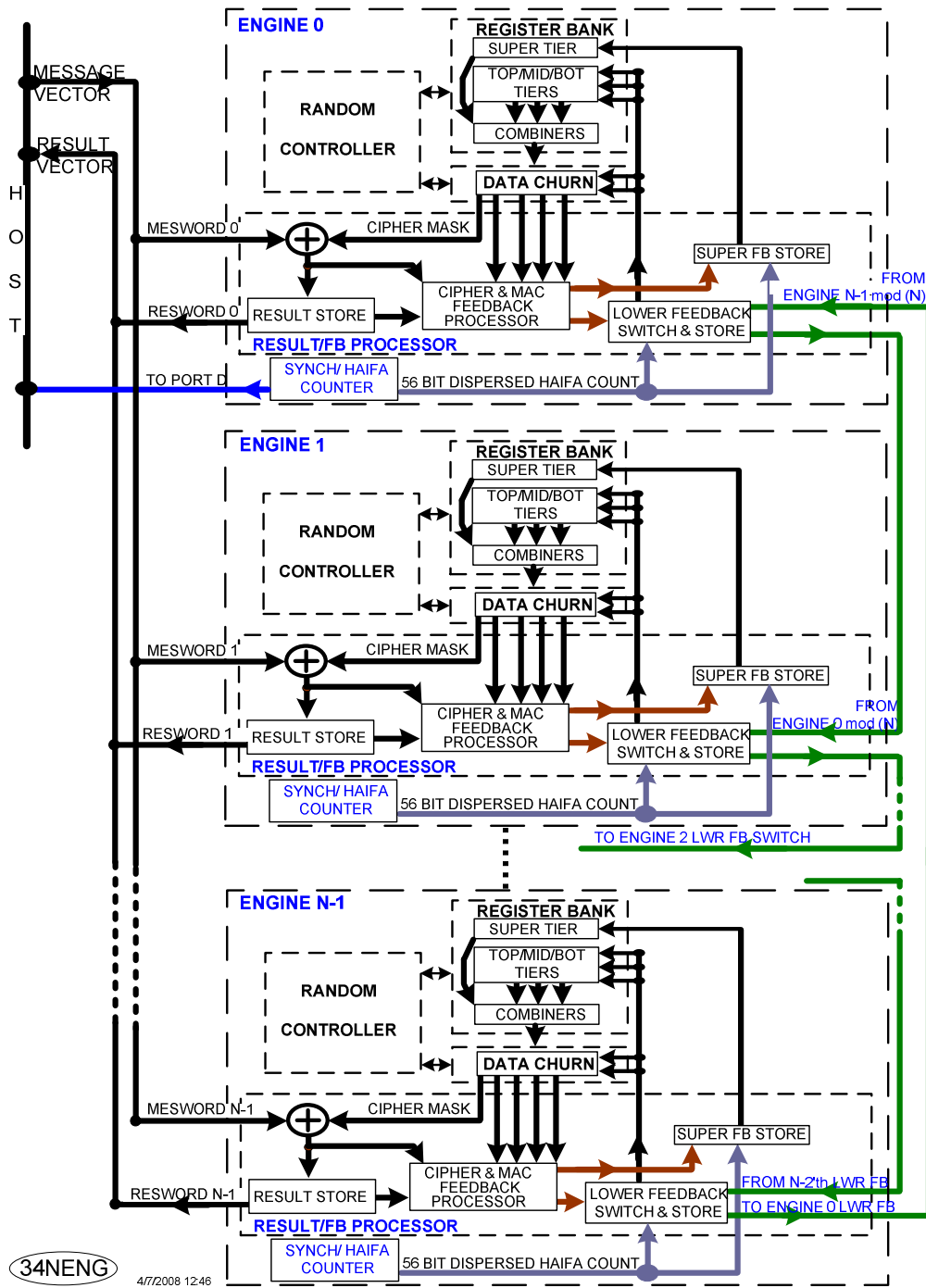


Fig. 19: n ZK-Crypt Engines; $x \bmod n$ Engine Accepts Lower Feedback from the $(x-1 \bmod n)$ Engine

In Fig. 19, an n engine configuration, the x 'th mod n ($0 \leq x \leq n-1$) engine's generated Lower Feedback is solely linked to (input into) the $x+1$ 'th mod n engine's Lower Feedback Store.

We have proved in [zk-secure] that a false Message Word in engine x mod n corrupts x 's Result Store beyond reconciliation, and simultaneously corrupts $x+1$ 'th mod n 's TMB Tiers and Data Churn, beyond reconciliation. Subsequently on the next Primary Clock, engine $x+1$ mod n will corrupt the contents of engine $x+2$ mod n beyond reconciliation; continuously corrupting all n engines beyond reconciliation.

11.2 Scramble-The Scramble function in the ZK-Crypt is a diffusion mechanism which we use to increase cryptocomplexity of initialization (also precluding weak keys) prior to ciphering, and prior to Message Digesting, and prior to Hash Value/Tag generation, and to enable increased security in constrained hardware.

Simply stated, a single scramble is a single Primary Clocked step procedure in Hash/MAC Mode, with the Message Word input and the Result output to the Host locked to zero.

Our statistics prove that with the exception of tier n LFSR outputs at missed TMB clocks, when a result is "repeated", each binary state variable (flip-flop) looks like a perfect distribution random number.

11.3 The Wait and Read Command. This multi-stepped command can best be implemented with a hardware finite state machine, an FSM, in proprietary implementations. This procedure consists of a sequence of a series of s scrambles ($0 < s < 32$) where Message in and Result to Host are locked to zero, preceding a valid defined procedure.

The Wait & Read is the equivalent to multiplying the number of rounds in a Feistel cipher, where Results are sampled at user defined intervals. The Message is fed in one cycle previous to the Read cycle. Typically, the only added cost is a linear increase of energy per processed bit, as we anticipate that in most applications, a single host cannot hope to service the ZK-Crypt operating at full speed. In the Wait and Read cipher or hash, the engine is Scramble cycled a defined number of times before the Host inputs a Message and/or reads the Result.

12 Hash/MAC and Cipher Procedures in the ZK-Crypts

In the block diagrams of Fig. 20 we see the essential configurations for Data Authentication processes, TRNG and Stream Ciphering. The TRNG after initialization is a Hash/MAC mode process with a physical AIS 31 compatible Noise Source. The Stream Cipher initialization process is essentially a Hash/MAC mode process where keys and IVs are hashed into the engine as Messages. In stream ciphering, the engines operate as synchronized deterministic random number generators, where the sender complements clear text to cipher text; and the receiver complements cipher text thereby reconciling the stream into clear text. Obviously, the ciphering process does not include the Message Word in its feedback. Maximum diffusion is achieved in Data Authentication wherein initialization, digesting of messages and tagging all benefit from Hash/MAC feedback.

The first step of any deterministic algorithm must start with a Global Key/IV (Re)set. At Power On, flip flops assume random (typically biased) values. The (Re)set causes all flip flops in the engine to assume a deterministic value; where most of the State Variables are reset to '0' value, and some to a '1', to assure easy "wake up". We claim that this State is a reasonable initial state for cost effective security Hash procedures.

The second phase of deterministic initialization consists of "hashing in" loading of 8 to 16 32 bit key or IV words (for standard hashing). In Hash mode this is followed by 8 Scrambles.

Key or Hash IV extensions in both Cipher and Hash/MAC protocols are "hashed into" the ZK-Crypt engine in MAC mode, as Message Words.

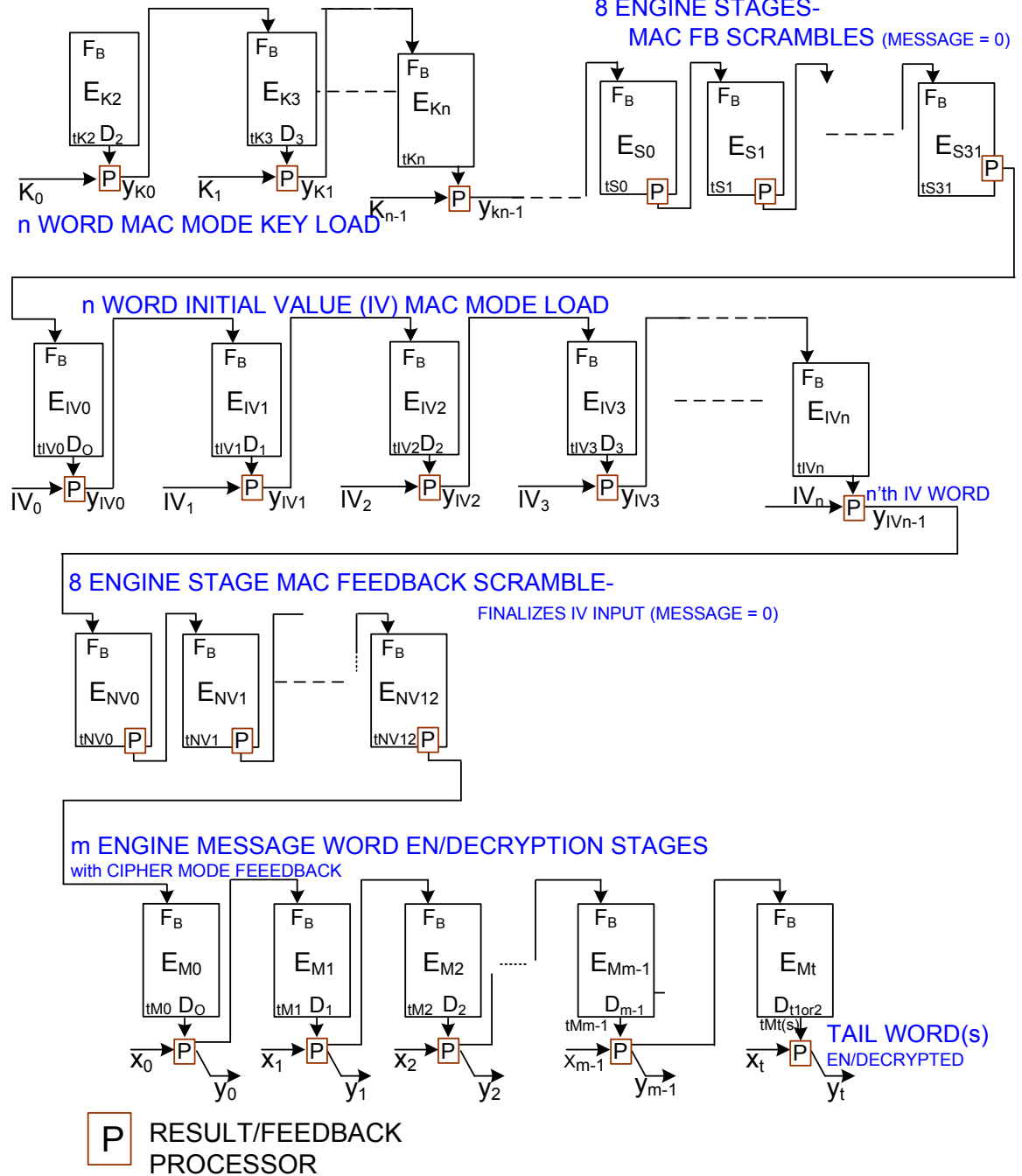
Third phase initialization of Cipher and Hash protocols, consists of a 24 cycle Scramble in Hash/MAC mode. For packet-synchronizing cipher protocol flow chart, see [zk-cc figs. C04 & C05].

12.1 Cipher Protocol

Fig. 10 graphically describes the algorithm.

ZK-CRYPT EN/DECRYPTPT PROTOCOL

GLOBAL (RE)SET



SEE [zk-cc figs. 35 EFC & DFC] for INTERNET PAGE SYNCHRONIZATION PROTOCOL

25/08/08 18:04



Fig. 20: The Stream Cipher Protocol Hashes in Keys, IVs & Scrambles in Hash/MAC Mode

12.3 Hash/MAC Protocol with Options

ZK-CRYPT MAC DATA AUTHENTICATION PROTOCOL WITH OPTIONAL 2ND KEY

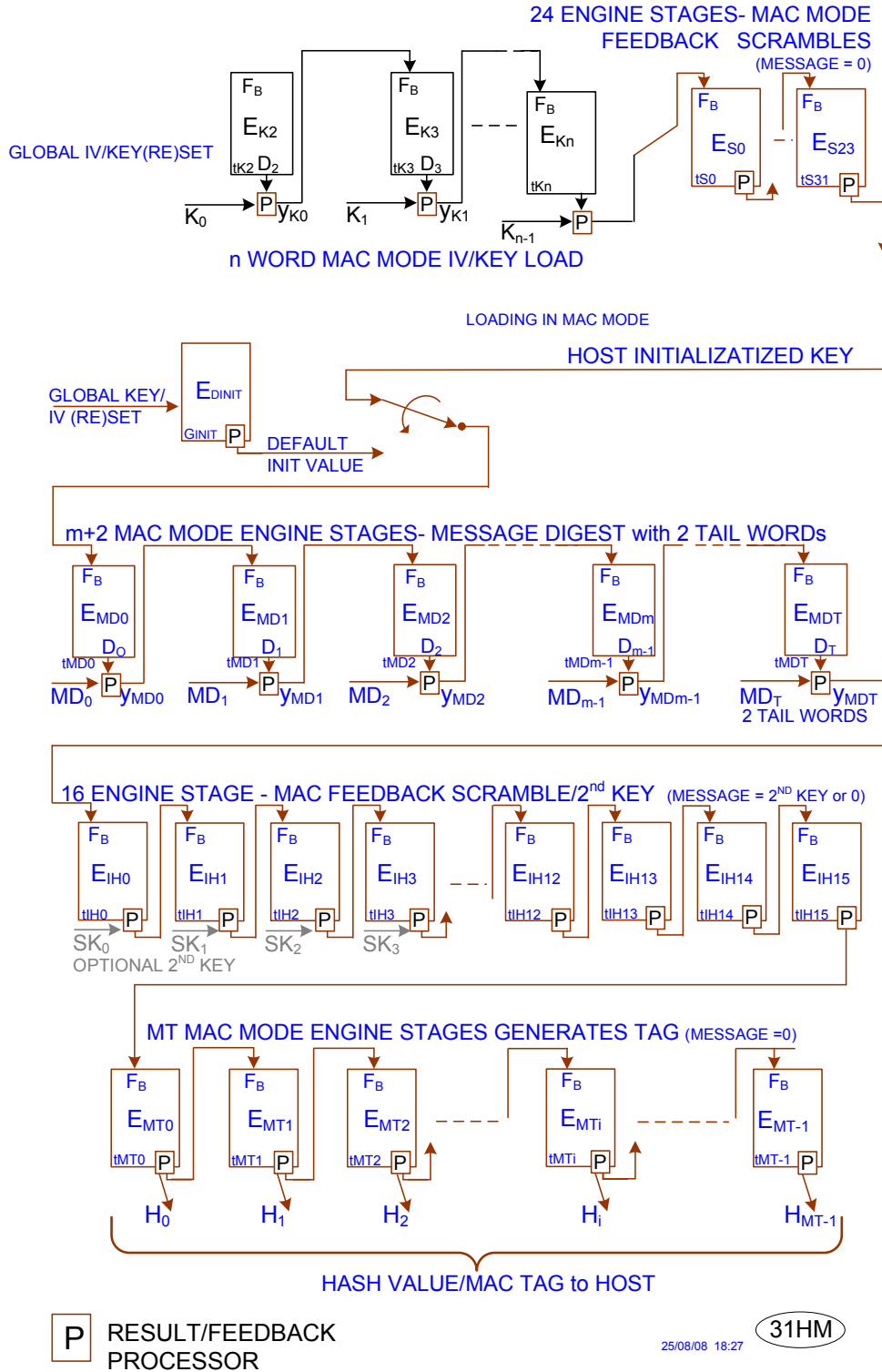


Fig. 21: Hash/MAC Protocols with Optional Second Secret Key

- 14. Security evaluation and Testing; a History:-** Numerous Exhaustive Random, entropy, Auto and Cross Correlation, Bias, Feedback Orthogonality tests were standard practice, prior to our being joined by Nicolas Courtois, the known expert on finding vulnerabilities of highly diffused large variable systems.

The final Zk-Crypt design has excelled in ALL tests available, with results significantly beyond the required criteria.

Testing has progressed from Knuth's [knuth-2] semi-algorithmic programs, to Maurer's universal tests [maurer-92], nicely programmed with graphics in NIST's [fips 140], and finally in the ubiquitous series of DieHard tests [diehrd], the benchmark for Pseudo Random Numbers and commercial stream ciphers. With DieHard we developed present strategies, wherein we were able to progress from 7 step to single step highest quality random string generation.

We ran numerous Repeated Word tests which consists of repeatedly (with different initial conditions) counting how many 32 bit words are repeated in a 10 million 32 bit word sampling, [zk-a-z glos]. We counted '1's and '0's (to check for bias) on all state variables and logic variables in the 32 Bit Word Manipulator, in the deterministic Noise and the permuting signals from the Random Controller. Results were generally better than the expected criteria (fewer repeated words than the statistical probability).

We assiduously combed the design and the output statistics in search of local bias, timing basis for second order side channel attacks, and weak spots in general to be used in more rigorous types of classical differential and linear/correlation attacks which may include:

- Algebraic attacks with Grobner Bases,
- Algebraic attacks with improved ElimLin algorithms,
- Algebraic attacks with SAT solvers,
- Algebraic attacks with various generalized T' methods (proprietary),
- Embedding an impossible to detect trapdoor, and,
- Various other proprietary algebraic attacks.

- 15. ZK-Crypts – a Short Hardware Spec:** Includes Size, Throughput, Binary Variables, Security Attributes, Random Statistics, Side Channel Attack resistance and a Comparison to AES 128.

15.1 Size- Full implementation of the ZK-Crypt includes a Finite State Machine, FSM (in proprietary implementations), which allows for simple or direct memory access activation– a single ZK-Crypt engine has over 470 flip-flops, almost all of which are used in all three functions. The ZK-Crypt has a total gate count of about 9500 gates; 2 input NAND gate equivalents, subject to implementations and timing for lower energy power control, and side channel attack protection of the loading procedure.

15.2 Hashing in IVs and Keys Initializing the ZK-Crypt is a straight forward "hashing in" digest, with an initializing overhead of 8 cycles for Hash/MAC and 16 cycles for Stream Ciphering.

15.3 Pipe Lined Throughput at Standard Single Step Operations- One 32 bit word processed (RNG, Stream Cipher or Message Digest) every clock (machine cycle) typically- 3.2 Gbits/Sec at 100 MHz Primary Clock Speed. A parallel linked ZK-Crypt pair at 160 MHz can process 10 Gbits/Sec, compliant with new Japanese data transmission standards. With submicron technologies which will be available at time of implementation, the ZK-Crypt can safely operate at 350 MHz.

15.4 Divide and Conquer Attributes- The Random Controller has 62 binary variables, and receives 10 binary feedbacks from the 32 Bit Word Manipulator. The 32 Bit Word Manipulator utilizes 522 binary variables, and receives 14 permutation logic signals from the Random Controller. In a twin configuration, the two machines swap 32 bits of feedback, doubling the number of state variables.

15.5 Comparing ZK-Crypt to AES (in Hardware) The ZK-Crypt is more compact, and has less than one fifth of the gate count of an excellent well thought out full scale AES design [zk-vs AES]. When pipelined, this fast AES block cipher outputs 16 bytes in 10 cycles, whereas the single

engine ZK-Crypt Stream Cipher outputs 4 bytes in one machine cycle; two and one-half times faster, with a small fraction of the energy expended per en/decrypted bit.

The basic ZK-Crypt hash has a 554 bit chaining value length (double that for a 64 bit Host) vs. AES 128 wherein the chaining value and hash value are the same.

15.5.1 Comparative Energy Consumption- For low power applications, battery and mobile applications, where total energy consumption per word processed is important, a more straightforward benchmark would be Mbit/(mWatt•second). Here with the 0.09 μ technology, at 1.8 volts, we anticipate 941 [Mbit/mWatt•second] against side channel attack protected AES with about 13 [Mbit/mWatt•second] or A/51 with about 273 [Mbit/mWatt•second]. Note that the paired concatenated configuration is twice as fast, more than exponentially stronger, and expends the same energy per processed bit, virtually in both Hash/MAC and Cipher Mode.

15.5.2 Side Channel Attacks- The ZK-Crypt is inherently resistant to known first and second order side channel attacks.

References:

- [diehrd] DieHard Tests, to be found at <ftp://ftp.csis.hku.hk/pub/random/source>, 2004.
- [fips 140] Federal Information Processing Standard Publication ,FIPS 140-2, NIST, Washington, May, 2001.
- [haifa] E. Biham & O. Dunkelman, A Framework for Iterative Hash Functions, NIST Hash Forum 2006, Santa Barbara, August, 2006.
- [knuth-2] D.E. Knuth, The Art of Computer Engineering, vol. 2, 2nd Ed., Addison-Wesley, Reading, 1981.
- [maurer-92] U.M. Maurer, "A Universal Statistical Test for Random Bit Generators", Journal of Cryptography, vol. 5 Springer-Verlag, Heidelberg, 1992.
- [repwrdr] D.J. Bernstein, "Does the ZK-Crypt I flunk the repetition test", eSTREAM website, March, 2006.
- [schndlr] Werner Schindler, "Efficient Online Tests for True Random Number Generators", Testing for AES 31 Compatibility, CHES 2001, Springer-Verlag, Berlin.
- [trichina] E. Trichina, Personal Conversations, 2005/6.
- [zk-algo] A. Hecht, O. Dunkelman, C. Gressel, ZK-Crypt Algorithmic Specification, www.fortressgb.com, London & Omer.
- [zk-a-z glos] The A-Z Guide to the ZK-Crypt, An annotated glossary, www.fortressgb.com, vers 4, January 2009.
- [zk-ccc] C. Gressel, ZK-Crypt Circuit Concept Drawings, www.fortressgb.com, London & Omer, January 2009.
- [zk-gold.ppt] C. Gressel, M. Slobodkin, The Gold Rush Cipher Revisited, www.fortressgb.com, vers SASC 2007, Bochum.
- [zk-intrfac] "What the Host Sees in the ZK-Crypt", Interfacing the ZK-Crypt, www.fortressgb.com, vers October 2008.
- [zk-fbpat3] US Patent Application 60/84612, September 7, 2006.
- [zk-secure] C. Gressel, N.T.Courtois, G.V.Bard, A.Hecht, The ZK-Crypt Security Analysis,www.fortressgb.com, London & Omer, Chapter 2, September 2008
- [zk-vs AES] "Comparing the ZK-Crypt to the NIST-AES Block Cipher", www.ecrypt.eu.org, February 2007.