



ZK-Crypt Features - Revised Draft

*Answers to NIST Evaluation Attributes for SHA-3 Candidate Submissions
Includes the new Mersenne Prime LFSR/Binary HAIFA Count Current Consumption Revision*

Responding to the list of hard to find concepts and attributes in our August 31 draft submission.

Table of Contents – NIST Specified Attributes

- 0) Simplicity of Concept- Simple Tenets, Many Simple Permutations, Simple Rationales
- 1) The ZK-Crypt Pseudo Random Function (PRF)
- 2) Message Modification- Orthogonal 32 Bit Feedback Streams
- 3) Time Memory and Relevant Work Factor
- 4) Resynchronization, Related Key, and Rekeying Differential Cryptanalytic Attacking
- 5) Precluding Sliding Attacks
- 6) Preventing Message Extensions & Herding
- 7) Supporting Maximum Message Length of $2^{64} - 1$ bits
- 8) Precluding Collisions, Preimage and Second Preimage Attacks- (incl. a Conjectured Observation)
- 9) Evaluating Algebraic Complexity
- 10) HMAC_K or Authenticating Encrypted Messages
- 11) Fundamental Increase of Security; Parallelization for 64 bit Computers; 2 x Throughput
- 12) Flexibility for a Wide Variety of Applications
- 13) Tunable Implementations – Wait & Read; Pairing Engines; Locking Matrices; No IV/Scrambles
- 14) Long Chaining Value Security Despite Smaller Hash Values- - Conjecture poorly supported.
- 15) Provenance of the Origins and Attributes of Tables & Constants in the ZK-Crypt
- 16) Innate Advantages, Limitations and Demerits
- 17) Throughput, Size, Current Consumption, Memory Constraints, Cost**
- 18) Memory Requirements on Hardware and Software Legacy Applications
- 19) Hardware ZK-Crypt on 8 Bit CPUs, Satellites, Small Memory & Limited Real Estate

Authors: Carmi Gressel
Nicolas T. Courtois
Gregory V. Bard
Avi Hecht
Ran Granot

January 2009 Revision

0) Simplicity of Concept, Simple Tenets, Many Simple Permutations, Simple Rationales

It is simpler for an expert or layman to understand the ZK-Crypt its concepts, and derivatives of conventional pseudo random devices, if she/he first understands simple tenets, and continues reading the top-down architectural combination of simple to understand building blocks. There is little need to "get stuck" on a definition or a concept, as we have updated the A to Z guide and glossary [in supporting documents], with brief explanations of the individual modules, the concepts and the linear and non-linear algorithms.

Understanding the rationales, then, is simple- without scanning the lengthy proofs, and observations in the security analysis tome.

The formal algorithmic spec [zk-algo] should be saved for the programmer or chip designer. The lengthy [zk-secure] security manual, may be a trove for later research.

If you're ready for a little humor, you may start with the animated [zk-gold], 3 time winner of the 2nd prize at cryptographic conferences. 30 different sub devices working together, may not bother you after a hearty laugh.

1) The ZK-Crypt Pseudo Random Function (PRF)

The ZK-Crypt in all modes of operation is an AIS 20 compliant Deterministic Random Number Generator, a PRF. In all stages of development, and, especially in preparation for the SHA-3 contest, the designs have been exhaustively tested with tailored tests: Maurer [maurer-92], the NIST FIPS 140 [fips 140], the DieHard [diehrd], our enhanced and automated DieHard, Bernstein's [repword], our Repeated Word test suite, Schindler's on-line test of deterministic noise [schndlr] and our own '1's counting to detect bias in any of the 320 state variables in the 32 Bit Word Manipulator. (The first three are included in our automated DieHard suite.) The tests were executed with the Message¹ input variable locked on zero; and a simple deterministic PRF. During each stage of development the results were exceptional, whereas the final results of all stages were very close to or seemingly better than optimal. Each state 32 bit variable in the 13 register Word Manipulator is a pseudo random word, uncorrelated to other words, without inter word correlations between bits, without differentials, or impossible differentials.

The final design has undergone the most rigorous of the deterministic tests. No differentials or correlations were detected with the constant or random inputs.

In the final design a unique location marker [HAIFA] is dispersed into the two orthogonal feedback streams. The pseudo random 64 bit Mersenne prime/binary HAIFA counter output is an integral part of the chaining value, and is therefore part of the ZK-Crypt 32 bit (64 for 512 Hash Values) bit to 416 (832) bit expansion. The HAIFA pseudo random output salts the two orthogonal feedback streams. Note- there is neither compression nor truncation in the ZK-Crypt message digest.

2) Resistance to Message Modification- Orthogonal 32 Bit Feedback Streams

The ability to modify Messages is possibly the holy grail of attackers. Precluding the ability to change numbers and names in a series of financial transactions was a central issue in the first ZK-Crypt MAC designs.

In a single feedback previous MAC, it was found that, in a very limited way, a Message could be modified, and 28 binary state variables in the 32 Bit Word Register Bank section could be reconciled to an original sequence of states with random non-meaningful Message Words generated for the remainder of the message. It was proved to be impossible to reconcile all Feedback Processor variables, because the single feedback was an XOR sum of the previous and present Results. Consequently, it was impossible to generate a valid MAC tag on a falsified message. [zk-secure, app 1]. The existence of the ability to gain limited control of state variables with random Message Words led to the development of the dual track orthogonal 32 bit feedback streams, fed into the Register Bank and Data Churn in five different forms. [zk-secure, app 2].

We define two data authentication feedback streams as orthogonal if a sequence of Message Words causes one feedback stream to successfully corrupt (to falsify the equations of a minimum of 144 bits in the 32 bit word manipulator) and reconcile (to flip falsified bits back to valid value) in one section of tiers in the Register Bank, whilst the "orthogonally" activated changes in the second feedback stream simultaneously irreconcilably corrupt at least one other section of the Register Bank; causing all state variables to be corrupted after no more than four rounds.

¹We refer to a longer than 32 bit data input operand as a message (small "m"). We refer to the 32 bit operand that is hashed, encrypted or decrypted (XORed to the Cipher Mask) as a Message Word, (capital "M").

We prove orthogonality if one feedback stream is a linear function of the present Message Word and the previous Message Word;

and if the second feedback stream is a linear function of the present Message Word only;
and where both feedback streams act on different internal data manipulator variables which are subsequently combined; where,

any attempt to reconcile (with false Message Words) the false bits in one section of internal word variables caused by any previous false Message Word, immediately corrupts the 32 Bit Word Manipulator; and provably corrupts the entire Word Manipulator after four rounds [zk-fbpat3][zk-secure app 2].

Stated simply, with the ZK-Crypt 64 bit dual track orthogonal feedback we prove that an adversary cannot eradicate traces of modified message aberrations in the Data Manipulator with false 32 bit Message Words.

3) Time Memory and Relevant Work Factor

A most efficient Time Memory attack requires dividing the number of variables in the memory table and in executable (time) variables into roughly equal parts. This quite clearly puts the Random Controller and the Register Bank in the memory table, as most of the calculations are on small odd length operands; and leaves the merged Data Churn and Result/Feedback Processor in the executable (time) division [zk-ccc fig. A01].

The Random Controller and Register Bank¹ merged together have 222 memory variables, 64 feedback inputs from the Result/Feedback Processor, and 2 inputs from the Data Churn; a total of 288 variables requiring 2^{288} addressable memory states to output 38 bits to the Data Churn.

The combined Data Churn and Result/Feedback Processor contain 256 memory variables, with 32 bits of input from the Register Bank and 32 bits of Message Word input, and 6 bits from the Random Controller; a total of 326 variables. This would require a Work Factor of $O2^{326}$ steps. We have not counted the redoubtable HAIFA Counter which is an integral part of the chaining value. The HAIFA value is known to an adversary.

A work factor of $O2^{326}$ steps is far larger than the industry standard of a safe $O2^{100}$ work factor, or the conservative estimate of $O2^{128}$ for future non-quantum computing attacks. (We estimate that quantum computing technologies will not be able to handle the algebraic complexity of the ZK-Crypt [bard].) Obviously, a brute force attack on a 544 variable, $O2^{544}$ work factor engine is out of the question. Any time memory attack on the 64 bit double engine with one of the two swapped feedback stream configuration is therefore unthinkable.

Bard finds that a minimum of $10^{56.1}$ "memories" would be needed. The weight of the planet is about 6×10^{24} kgs, divided by the weight of a nucleon, leaves us about 1.7×10^{27} nucleons. The maximum "available" number of nucleons in the planet is about 3.6×10^{51} or about 2^{171} nucleons, barely enough to solve Birthday Paradox problem on the ZK-Crypt. (One might estimate that each nucleon would have to be covered with millions of Christmas tree lights, to be valid memory units.) Without "interplanetary" travels, we'd never be able to assemble that much gray cell.

4) Resynchronization, Related Key, and Rekeying Differential Cryptanalytic Attacking

Running exhaustive tests on the ZK-Crypt PRF, Pseudo Random Functions, with random IVs, and Message words locked on all zero, statistically supported in all cases:

- never to generate a bias (a differential) on any binary state variable;
- never to generate a distinguishing feature;
- non-existence of a distinguishing feature, (an impossible differential)

in the 32 Bit Word Manipulator (the Register Bank, the Data Churn or the Result/Feedback Processor).

Previous and present statistical tests have consistently shown that all ZK-Crypt versions fed with random, structured or constant Message Words generated same excellent range statistics as with the same engine fed with the Message Word input locked on zero.

Biham [diffrentl] stipulates that cores "without differentials and impossible differentials are probably immune to resynchronization, related key, and rekeying differential cryptanalytic attacks".

Counting the number of '1's in all binary state variables in the Register Bank, Data Churn and Result/Feedback Processor, in many giga word samplings showed that there never was a probability larger than 10^{-4} of bias on any binary state variable in the 32 Bit Word Manipulator.

Repeated Word tests [bard] showed that there were no intra-word correlations in state variables in the 32 Bit Word Manipulator. The tests also show there are no inter or intra-word correlations in the state variables of the 32 Word Manipulator (save for the jittered clock affected nLFSR tiers in the Register Bank). For the two typical

¹ In the January 15, 2009 revision for NIST, a Register Bank Store, and two new Bottom decorrelation buffers were added; and the HAIFA counter was converted from a 56 bit binary up counter to a unique 64 bit composite of Mersenne prime LFSRs and a 6 bit binary up counter.

types of correlations: intra word correlations and inter-word correlations, (e.g., where bit i and bit j of the same word are same value with a more than 0.5 probability); the RW tests are extremely sensitive.

For testing purposed the nLFSR clock jitters were switched off (no jitter), to demonstrate that the PRFs of the Register Bank tiers were sound; allowing us to establish a compensating metric for testing the jittered clock tiers.

5) Precluding Sliding Attacks

The 64 bit HAIFA Counter is an integral part of the ZK-Crypt chaining value. Each counter "increment" generates a new unique one of more than 2^{61} deterministic pseudo random numbers, The counter output is diffused into the two orthogonal feedback streams, clearly leaving unique "location" marks on each 522 bit (554 bit with Message Word) chaining value.

The HAIFA Counter removes the self-similarity of the hashing process, thereby precluding sliding attacks.

The reader can convince him/herself by observing the diffusion maps in [zk-secure app 4] that no sliding attack is possible. The maps and the tables show that a single changed Message bit affects, in the first clocked cycle: the equations of 9 or 10 bits in the Register Bank; the equations of another minimum of 134 bits in the Data Churn and Result/Feedback Processor; which include all 32 bits of the Cipher Mask.¹

Obviously, on the next relevant round, we are assured that not only the Cipher Mask equation, but all equations in the 32 Bit Word Manipulator's 416 bits will be randomly affected.

6) Preventing Message Extensions & Herding

For the first more than 2^{61} Message digest words, the incremented Mersenne Prime/Binary HAIFA counter inserts unique 64 bit position values into the chaining value. This precludes the possibility of repeating a section of code, as replication can only occur if an "equivalent" chaining value is the first and last chaining value of a repeatable sequence of 32 bit words.

The HAIFA value precludes message extension, as the message length and the number of bits in the message is included in the final two chaining values. Obviously, a series of 32 bit Messages cannot reconcile the relevant series of pseudo random 64 bit HAIFA counter values to a new fraudulent location.

7) Supporting Maximum Message Length of $2^{64} - 1$ bits

Each ZK-Crypt expanding initialization, digestion and Hash Value generation clocked function steps diffuses a changed Cipher Mask or Message bit into the equations of at least 144 of the over 500 significant bits of chaining variables.

Each Message Word history is linearly XOR recorded into three or four 32 bit tiers in the Register Bank and into the Register Bank Store, the Top, Intermediate and Triple Bottom Store & XOR registers in the Data Churn; in four derived forms from the original feedback words. Said differently, a maximum of 2^{60} Message Word histories are linearly recorded into 2^{192} combinations in the Register Bank, and the Data Churn. If we only view the 160 bit Register Bank as a vault to hold 2^{160} combinations affected by 2^{64} bits of message, this is sufficient to assure that there is no long term period.

The ZK-Crypt Tail Word extension consists of two 32 bit words which record, together, the total number of bits in the message. The ZK-Crypt binary message bit count in the last two Tail Words is unformatted, therefore formally supporting a $2^{64} - 1$ bit count. In addition, the HAIFA counter indelibly feeds unique count values via orthogonal feedbacks into the Register Bank and Data Churn during the first (can be more than 2^{61}) Message Word digests, equivalent to a unique encoding of as many as 2^{66} message bits. This uniquely marks each Message Word's place in the first up to 2^{61} Message Word sequence.

8) Precluding Collisions, Preimage and Second Preimage Attacks- (a Conjectured Observation)

In section 2) we demonstrated how the dual track orthogonal feedback streams preclude Message Modification. [bard] has shown that the cryptocomplexity is such that with many millions of monomials preclude algebraic artifacts. We have shown there are no differentials, no impossible differentials, and no less important- no correlation between state variable words and no correlations between bits in any state variable word. This should seem obvious to anyone who examines deterministic graphs of first order diffusion of a single Message bit into more than 200 binary state variable equations, and realizes that there are no end effects in the 32 bit Word Manipulator.

The ZK-Crypt algorithm is an expansion of 32 bits to at least 200 bits which expand to all bits (including the Random Controller) in just a few rounds. The chaining value is never truncated. The only compression consists of

¹ Calculating the added average diffusion caused in a single round of the January 2009 revision is extremely difficult. We can safely assume a new lower bound of $144 + 64$ bits

generating the hash value.

In the following section we conjecture that any "tinkering" with false messaging precludes valid short hash values, with lower probabilities than previously suspected. We realize that the fundamental increase of security is of no practical value for hash value lengths of 224 or more bits, but if the conjecture is true, it will be an interesting observation.

In [zk-secure app 1 & 2] we prove with examples, that because of the orthogonality of the two feedback streams, at the first instance of a false Message Word input, the ZK-Crypt engine is corrupted. The remainder of the message is a new start in what is now actually a new preimage search with inconsistent values in the Result Store and the Lower Feedback Store. We conjecture in this section, that the preimage resistance is greater than 02^n ; where n is the number of bits in the hash value.

The HAIFA LFSR Counters are incremented at the clock tick activation of each hash Message Word digest computation. The counter outputs [zk-ccc fig H03A] are dispersed and linearly combined into unique "marks" in the 64 bits of the two feedback streams. Colliding chaining values can only occur at the valid Mersenne prime LFSRs' HAIFA Message Word count and the binary bit count in the tail Words. The authenticator assembles the valid predefined Tail Words, as shown in section 6) and in [zk-secure ch 8]. We present a conjecture, based on observations of Repeated Word test which showed that on the average, the number of Repeated Words in the state variables was lower than the optimum. Recent tests have contributed limited support to the conjecture, which may show that smaller hash values may be unique reflections of single larger chaining values which have "Similar Initial Values"..

In the next paragraphs, we deal with the cases where the attacker replaces a valid message with an invalid message of exactly the same bit length.

If the attack was completely successful; i.e., a 2nd pre-image generated the valid final chaining value; achieved with a probability of about 2^{-s} where s is the number of significant bits in the final chaining value.

The final hashing sequence is generated from the second Tail Word chaining value. We have assumed that both the invalid and the valid message have the same bit length, i.e., the same last Tail Message Word in the chaining value and the same HAIFA count are the same in both the valid and invalid messages. We call these same 2×96 bits a starting partial IV point for a false message to generate a valid Hash Value generating sequence. Our exhaustive statistics have lead us to believe it is virtually impossible for such large Tail Word chaining values in the ZK-Crypt to generate more than one smaller valid Hash Value. Typically, there are 554 bits of chaining value and only 256 bit length hash values; however, we conjecture that because of innate qualities of the PRF, the Hash Value is unique though one might suspect that 554 bits of chaining value could theoretically generate 2^{300} same Hash Values; e.g., in the ZK-Crypt in each of the two final chaining values of two same length messages, there are, 96 bits of identical value.

If the valid and invalid chaining values are different, but both include six same words in the final tail chaining values, we may call these new sequence "similar" starting point initial values (SIVs) for generating both the valid and invalid Hash Value generation sequence. These are SIV chaining values each of which generates a sequence of Cipher Mask words which ends with $n/32$ bit Hash Value clocked word outputs; which may or may not be preceded by Scramble steps. (A Scramble step is a hash computation wherein the Message is locked to zero, and the output of the Cipher Mask is typically not read.) [zk-ccc B06] We assume small domains may be as small as 20, or as large as 11,000.

The small domain in this section, we conjecture, show why the starting point invalid chaining value may generate the valid Hash Value with a probability that is exponentially smaller than 02^{-s} virtually impossible. A Hash Value generation is a deterministic hashing procedure where the Message input is locked on zero.

We conjecture that from the above "similar" starting point SIVs (locations), there is a lower bound of the number of hash computation steps (a distance) that a non-valid Message sequence can practically "navigate" to a valid "candidate" chaining value location. A valid candidate chaining value can be a valid Cipher Mask output located at the valid location. Our conjecture from recent some results led us to estimate 12,600 hash computation steps. The lower bound distance is virtually the same for a part of the valid chaining value, the Cipher Mask output, and for the total chaining value. Later tests have not given full support to this larger distance observation.

Because of massive diffusion, all significant chaining value variables in one machine cycle affect most of the variables in the following cycles' chaining values. The full chaining value includes all binary state variables in the ZK-Crypt engine. The new 64 bit Mersenne Prime LFSR/binary counter 64 bit output and the two tail words with the bit counts are part of the two last digest chaining values.

The significant number of chaining value bits s, does not vary, as we assume that the attacker searches for a

preimage of a message of exactly the same bit length. We define message size by m bits. For small m domains (small messages) we conjecture that the probability is exponentially reduced to $O2^{-s}$ which is a function of m ;

i.e., $s' (m \text{ small domain}) > s$.

We will generate stronger statistic tests, which, we estimate, will support our suppositions and extrapolations to Hash domains in the ZK-Crypt.

The expected occurrence of a repeated 32 bit word with a "perfect" distribution, with n trials should be: the number of possible tested pairs, divided by the probability that the pair will be "identical twins" - $(n(n-d) \cdot 2^{-1}) 2^{-32}$ where $d = 1$. If d can be equal to 1 the estimate assumes that the twin of a repeated word can be juxtaposed to its sibling. In our individual trials n is 10M [32 bit Cipher Mask output words], and the number of repeated words (RW) is the average of one hundred 10 million hash compilation tests with a low STD.

For $d=1$, the naïve expected number of repeated words, #RW = 11641, for $n = 10M$ as:

$$(n(n-1) \cdot 2^{-1}) 2^{-32} = 11641.$$

Intuitively we can estimate that there are internal equations that could reduce the small domain probability, and would virtually preclude many more small d 's. We call this an innate attribute of the ZK-Crypt. All tests show that we have close to perfect distribution of Cipher Mask outputs, so that the small d domain, if it exists, does not appreciably affect small statistics.

However, with trivial (all zero) Message input, a trillion 32 bit word tests averaged #RW = 11633 with a small variance, on the ZK-Crypt, on the RD5 and on Linux RW tests. (We found that if we removed inter word correlations with 7 successive correlation immunizers, we lowered the #RW to 11613. This was costly, and abandoned.)

#RW = 11633 suggests that because of the hidden PRF idiosyncrasies, RWs with Messages equal to 0 virtually never occur in a domain of 3000 Message Words, but are ideally distributed for larger domains, as:

$$(n(n-3000) \cdot 2^{-1}) 2^{-32} = 11633.$$

However, recent exhaustive testing, with non-trivial Messages, (and the above mentioned previous tests with carefully decorrelated Results) again showed #RW \approx 11613. This suggests that the occurrence of a repeated 32 bit Cipher Mask, will only occur for distances larger than $d_{32} = 12,600$ as:

$$(n(n-d_{32}) \cdot 2^{-1}) 2^{-32} = (n(n-12,600) \cdot 2^{-1}) 2^{-32} = 11613 \text{ (where } n \text{ again is equal to } 10M).$$

The repeated Cipher Mask is part of the chaining value. Assume we extend the size of the words in our search from 32 bits, we will probably affect the size of the small d_{32} domain. As if we increase the word size by one bit, previous RW pairs would continue to be RWs with a probability of one half. Said differently d distance would increase by 0.5. An increment of 2 bits to the word size would increase the small domain size by 1. For the full chaining bit equivalent of about 420 active unbiased bits, we would achieve an increment of about 178 bits $(420/2 - 32)$, or an estimated small domain of 12,778 Message Words. The small domain is virtually the same for candidate Cipher Mask words and for the full chaining value.

We have temporarily ceased testing the conjectured statistics as a result of a miniscule increase of #RWs in tests with the enhanced non-linear functions introduced in the last implementation. This is typical of any introduction of an unbalanced non-linear element. The statistics are very good but the average d is not consistent with the conjecture. However the observation shows promise for $d < 50$; allowing accelerated password validation, with commensurately smaller hash values.

9) Evaluating Algebraic Complexity

The ZK-Crypt was purposefully designed to be a highly diffused multipermutation device. Presently known analyses are designed to catch onto "hooks" in sparse equations, which a fine toothed comb has not found in the ZK-Crypt. [biham] [courtois]

In [bard] a supporting document in this NIST submission, the author painstakingly modeled the simplest component in the 32 bit Word Manipulator. In the Data Churn (without 64 bits of Lower Feedback) he found over fifty-five thousand monomials. He conservatively estimated the number of monomials in the Register Bank, the Result/Feedback Processor and the Random Controller. In the aggregate, even if we take into account accidental cancellations, creative attackers and reducing algorithms, the results were beyond the realm of feasibility. Present technology cannot handle matrices with many millions of monomials.

The author adds, "This might lead to a staggering problem with 10^5 to 10^{15} monomials. We estimate that this is well out of reach for all computers in the decades to come."

10) HMAC_K or Authenticating Encrypted Messages

The FIPS 180-1 HMAC_K can efficiently be implemented on the ZK-Crypt where the input m can be clear or encrypted, with little overhead as is obvious from the process:

$$\text{HMAC}_K = h \left((K \oplus \text{opad}) \parallel h \left((K \oplus \text{ipad}) \parallel m \right) \right),$$

where we assume m includes the tail and "opad" and "ipad" are constants. $K \oplus \text{ipad}$ and $K \oplus \text{opad}$ are typically precomputed, and stored in secure memory.

The HMAC_K procedure is considered to be a solution to strengthen a weak hash function and for providing proof of origin of data.

If encrypted data is stored in insecure memory, we claim the ZK-ENMAC $_{K_1, K_2}$ procedure in hardware is a preferred solution, as seen in the following:

$$\text{ZK-ENMAC}_{K_1, K_2} = h \left(K_1 \parallel 8 \cdot 0000\ 0000x \parallel HF \left(E_{K_2} (m) \right) \parallel HF \left(16 \cdot 0000\ 0000x \right) \right),$$

where (for a 384 bit digest) operational overhead is 36 clock cycles. Encrypted m , $E_{K_2} (m)$ significantly taunts the hacker with two distinct problems.

HF is a unique 64 bit dispersion of the 64 bit HAIFA counter XORed into the two orthogonal 32 bit Feedback Streams; h is the ZK-Crypt hash function; $j \cdot 0000\ 0000x$ signifies a string of j all zero 32 bit Message Words; " \parallel " signifies concatenation; and $(E_{K_2} (m))$ is the ZK-Crypt encrypted message with the stream cipher K_2 key. In this instance, with two concatenated 9.5K gate engines, the encrypted data can simultaneously be processed by a second ZK-Crypt engine receiving the same input m data. This is especially valuable for transparently inputting encrypted data from unprotected memory; e.g. for safe booting, see [zk-secure ch 7].

We claim that the ZK-ENMAC $_{K_1, K_2}$ superiority is conclusive if an IV or Key is hashed into the final Scramble, $(HF (16 \cdot 0000\ 0000x))$, with no additional timing overhead see [zk-ccc B06].

If data is stored in the clear, the input into the right hand ZK-Crypt is the output from the left hand engine. If data is stored enciphered, the input data into both R/H and L/H ZK-Crypts is the same. In both instances, hashing is performed on encrypted data with little overhead.

- 11) Fundamental Increase of Security; Parallelization for 64 bit Computers; 2 x Throughput;
The single 9.5 gate ZK-Crypt engine is 32 bit oriented, and inefficient when coprocessing with 64 bit computers. This observation is true for two side by side engines, where one engine decrypts and the second engine authenticates, as both the cipher and the data authenticator simultaneously process identical 32 bit words.

Concatenated ZK-Crypt engines where the Lower Feedback streams are swapped, process 64 bits at each clock, with the same energy per processed bit as in a single engine configuration, and with more than an exponential increase of security.

- 12) Flexibility for a Wide Variety of Applications

All ZK-Crypt engines are identical and configurable in concatenations, in standalone, and in side by side processes, with low power per bit processed in all modes and virtually in all clocked frequencies.

Smart Cards with one or two engines can communicate with ZK-Crypt engines in servers configured to 32 or 64 bit speeds.

2 or 4 ZK-Crypts engines on a satellite communicator can encrypt and authenticate up to 10 Gbits per second at 160 MHz.

Two ZK-Crypt engines on a PC CPU, can transparently decrypt a boot from unsecured memory and authenticate the data, transparently, far faster than a DMA controller can feed and unload Messages and Results.

The most secured applications will have an on-board finite state machine, to freeze protocol.

- 13) Tunable Implementations – Wait & Read; Pairing Engines; Locking Matrices; No IV/Scrambles
Standard and special protocols may be necessary, or desired for higher security, for maximum throughput at a given speed, or for increasing throughput with a reduced overhead procedure for communicating with PC software.

Wait & Read, essentially increases the number of rounds in a hash computation. This is accomplished by inserting one or more Scrambles between each read. Each added Scramble multiplies the energy per processed bit, reduces throughput and exponentially increases security.

Concatenating n ZK-Crypt engines multiplies throughput n times (assuming the concatenation isn't data starved) and fundamentally increases security, while maintaining the same energy per bit.

The Splash Matrices can be locked on the D Rule (Identity) Vector, eliminating a costly software operation. Diffusion is reduced by about 8%, probably with little loss of security.

We know of no real loss of security if the initialization process is simplified, wherein the IV is the natural "power on" state, instead of a NIST prescribed IV for each hash value.

We know of no loss of security, if the initial and final Scramble steps (hash computation with Message locked on zero) are eliminated. Eliminating hashed in IVs and all Scrambles reduces standard overhead by 32 clocked cycles. This is of little consequence in hardware but immensely important for software hashing of small messages.

14) Long Chaining Value Security Despite Smaller Hash Values – Conjecture not fully supported.

We conjecture that there is a lower bound of the number of hash computation steps (a distance) that a non-valid Message sequence can practically "navigate" to a valid "candidate" chaining value location. We conjecture, on the basis of massive statistics, that this is a minimum of up to 3000 words for trivial Message Words (locked on zero). We conjecture from experience and massive testing that the navigated short distance (number of clocks) from the final Tail Word chaining value to the last hash value word precludes false hash value generation from false chaining values; if we know that the HAIFA counter output and the significant bit count in the false chaining value are valid. This would mean that the strength of the hash computation is a function of the significant bits in the final chaining value.

15) Provenance of the origins and attributes of Tables & Constants in the ZK-Crypt

In the hardware implementation [undst] [zk-ccc H03] the "power on" Global (Re)set, the bijective S Box transformations and the Splash Matrices are hard-wired constants:

- 1) The fixed "power on" IV Global (Re)set is a constant IV designed to assure immediate wake up of all permutations. This assures immediate wake-up of all permutations, and precludes weak IVs. A weak key study in [zk-secure app 3] led to a sparse wake-up diffusing Global (Re)set assuring that the hash load of the IV/Key/Message would "naturally" exercise every subsection of the engine in 4 clocks.
- 2) The 16 bijective S-Boxes were designed by Biham for the Serpent, a short listed NIST AES candidate. We know that each box transformation can be programmed as a one-way logic function. The S Box transformations delinearize the Super Mix component in the Super Tier Feedback. In the hardware version the boxes are implemented in minimal random "table like" logic. In the software version, the boxes are implemented in LUTs.
- 3) The two Splash Matrices efficiently assure diffusion of a single Message bit into a minimum of 144 binary equations in the 32 bit Word Manipulator, ruled by 4 vector constant rules. Implementations in three 32 by 32 bit addressed tables would have demanded an outlandish amount of memory, and would be extremely time consuming (lengthened the critical path). The hardware displacement matrices were manually derived from 3 32 bit extended 5 bit LFSR sequences; we assured that no two vectors would have same indexed input output pairs; e.g. Vector A input i and Vector B input i would cause displacement output of Vector A to be $j \bmod 32$ whereas Vector B output would be $j+x \bmod 32$.
- 4) All shift register tap (constant) configurations were chosen from standard spread spectrum listings [spread] and were tested before implementations.
- 5) Each hash length has a unique 10 32 bit IV or 20 32 bit IV, for single or dual engines, respectively. The values were drawn, in an orderly fashion from the 64 x 32 bit FIPS-180 SHA-256 section 2.2 IV value table. The specific values can be found in [zk-algo].

16) Innate Advantages, Limitations and Demerits

Advantages:

The hardware implementations confer the following attributes- exceptionally high diffusion, strong defenses against attacks, 32 bit per clock throughput, low gate count, extraordinary algebraic complexity, simultaneous decryption and authentication, low current consumption per processed bit, concatenation for low current consumption highest speed and strongest keys, easily understood tenets, simple primitives and rationales to be easily implemented on FPGAs and laid out on silicon [eth].

The ZK-Crypt has exhaustive proven statistical absence of differentials and impossible differentials and distinguishing features.

The optional sparse all digital random FM Oscillator feeding the ZK-Crypt core when in TRNG configuration is smaller than popular mixed signal RNGs. Every good cipher device needs true random numbers for security procedures. We assume that a sponsor will offer the noise generation option royalty free. Silicon vendors will then have an advantageous multipurpose PRF and TRNG device, cost free for constrained single engine implementations. Two, three or four additional 9.5 K gate engines would have miniscule effect on the gate count of CPUs with millions of transistors.

Implementations in protected hardware with frozen protocols are innately more secure against insider and



remote attacks than unsecured software options.

Smart Card implementations with reduced overhead, operating at low frequencies hash/decrypt 1000 bit messages in just one or two milliseconds.

Limitation:

The fastest accelerated, most efficient implementations of the ZK-Crypt are only possible in hardware. See attached SW/HW comparisons.

Demerits:

Many practitioners do not appreciate multipermutation high diffusion implementations based on simple tenets supported by clear rationales and most exhaustive testing. They prefer simpler "provable" security of mathematical equations, which may be efficient in both software and hardware.

The ZK-Crypt is inefficient in software, even with tunable acceleration options designed for legacy applications.

Non-linear functions, used alone degraded statistics, causing the addition of a plurality of additional disparate linear combiners.

17) Throughput, Size, Current Consumption, Memory Constraints, Cost

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

A single engine ZK-Crypt HW block consists of an estimated less than 9500 two input NAND gate equivalents for full implementation including an FSM, for simple or direct memory access activation.

Concatenating a second engine will double the estimated size. The ZK-Crypt engine has over 438 flip-flops, used in hashing and ciphering.



Deterministic timing – ZK-Crypt engines operating at 100 MHz

IV/Hash Value Bit Size	IV/Key Initialization Includes- Config 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 Platform			
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
			256 bit total time hash	3.05 μsec
Standard 64 bit Concatenation	64 Bit Platform			
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
			512 bit total time hash	1.78 μsec
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
			256 bit total time hash	1047 μsec

Estimated current consumption for a single or double ZK-Crypt Engine at 0.09 micron technology 941 Mbits per milliwatt second. Note, the double engine 64 bit concatenation doubles speed and more than exponentially increases strength, with the same estimated current consumption per processed bit. The single engine ZK-Crypt processes 32 bits/clock compared to popular [m-sys] SHA-1 5.6 bits/clock.

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

The timing tests were executed on a Dell-Latitude D620, with a T2400 Intel CPU (1.83GHz) and 0.99GB of RAM. In all tests the set up time was measured separately from Message digestion and Hash Value/Tag generation. We timed 1000 byte random messages. Each test was repeated 10 times for each bit length. We see no advantage of a 64 bit platform, as all computations are on 32 bit operands. An alternate solution may be found with 2 CPUs.

Average Performance Times for PC Computing of the ZK-Crypt Algorithm on 4 Byte & 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 at least 80 to 150 times faster than software PC speeds. Hardware constants and tables are implemented in random logic with efficient critical propagation timing.

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.

18) 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.

19) Hardware ZK-Crypt on 8 Bit CPUs, Satellites, Small Memory & Limited Real Estate

8 bit CPU hardware implementations will typically be constrained by the speed of the specific CPU's Load and Store functions.

Typically, commands for configuration changes are single clock commands, reducing the need for an onboard finite state machine, in constrained circumstances.

Loading and unloading in serial communication implementations, e.g., TV on satellites, can be independent of CPU size, so that a high speed 8 bit CPU, in many instances, may provide ideal performance.

DMA application speed is a function of memory access timing, and in simplest smart card applications, by the overall efficiency of the CPU.

References

- [bard] G.V. Bard, A. Hecht, C. Gressel, Security Analysis of the ZK-Crypt Data Authenticator & Stream Cipher Against Algebraic Cryptanalysis, Differential & Correlation Attacks, www.fortressgb.com, January 2009
- [courtois] N.T. Courtois, Algebraic Attacks on Combiners with Memory and Several Outputs, Courtois@minrank.com, 2005.
- [diehrd] DieHard Tests, to be found at [ftp://ftp.csis.hku.hk/pub/random/source](http://ftp.csis.hku.hk/pub/random/source), 2004.
- [diffrent] E. Biham & O. Dunkelman, Differential Cryptanalysis in Stream Ciphers, CS-2007-10, Technion, Haifa.
- [eth] F.K. Gürkaynak, et al., Hardware Evaluation of eSTREAM Candidates. SASC 2006, Leuven.
- [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.
- [m-sys] SuperMAP SHA-1 Coprocessor, Tech Highlights, 01-SR-001-03-8L, Kfar Saba, 2002
- [reword] 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.
- [spread] J.K. Holmes, Coherent Spread Spectrum Systems, Wiley, New York, 1982.
- [trichina] E. Trichina, Personal Conversations, 2005/6.
- [vaudenay] S.Vaudenay, A Classical Introduction to Cryptography, Pg. 75, Springer, NY
- [zk-algo] N.T. Courtois, A. Hecht, C. Gressel, G.V. Bard, R. Granot, ZK-Crypt Algorithmic Specification, www.fortressgb.com, London & Omer, January 2009.
- [zk-a-z glos] C. Gressel, G.V. Bard, A. Hecht, O. Dunkelman, The A-Z Guide to the ZK-Crypt, An annotated glossary, www.fortressgb.com, vers 4, January 2009.
- [zk-ccc] C. Gressel, N.T. Courtois, G.V. Bard, A. Hecht, R. Granot, ZK-Crypt Dual Track FB 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-infrac] "What the Host Sees in the ZK-Crypt", Interfacing the ZK-Crypt functions, www.fortressgb.com, vers January 2007.
- [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, January 2009
- [zk-undst] C. Gressel, O. Dunkelman, A. Hecht, "Understanding the ZK-Crypts- Ciphers for (Almost) All Reasons", www.fortressgb.com, London & Omer, August 2008.
- [zk-vs AES] "Comparing the ZK-Crypt to the NIST-AES Block Cipher", www.ecrypt.eu.org, February 2007.