Evaluation of MULTI-S01

January 17, 2001

Information Security Research Centre
Queensland University of Technology
Ed Dawson, Andrew Clark, Helen Gustafson,
Bill Millan, Gary Carter

TABLE OF CONTENTS

Evaluation of MULTI-S01 Algorithm

1 Executive Summary

This is a report on the stream cipher MULTI-S01. The MULTI-S01 cipher uses Panama output as the keystream for a block-based chaining operation which provides a message integrity check. Thus the MULTI-S01 cipher provides an extra service beyond that normally associated with stream ciphers. The proposed mode of operation adds two integrity check blocks to the message length.

In this report the evaluators discuss all relevant aspects of MULTI-S01. In particular we discuss the properties of the operations involved and their suitability for the stated design goals. Critical aspects of the design are examined and the requirements for its correct operation are noted. For this algorithm the evaluators have

(i) analysed structural aspects;
(ii) evaluated the basic cryptographic properties;
(iii) evaluated the security from attack;
(iv) evaluated the statistical properties;
(v) surveyed the speed.

There are several specific claims that are made for MULTI-S01 in the abstract of [MULTI-S01]. The security claims are that MULTI-S01..

1. achieves high security of data confidentiality, assuming a secure PRNG.
2. high security of data integrity
3. an attacker cannot determine any part of PRNG output just from known plaintexts.

The performance claims are that MULTI-S01

1. operates faster than encryption algorithms based on block ciphers.
2. pre-computation of the random sequence is easy to do.

In this report we examine the accuracy of these claims. Our results may be summarized by noting that the following two basic flaws have been found in MULTI-S01.

FLAW 1 Lacks Robustness

The security claims, while strictly true, are *not robust* in that they fail to be true under a minor violation of the key management rules. It is vitally important that implementations of MULTI-S01 adhere strictly to the key management rules that require a new key to be used for every encryption.

FLAW 2 Attacks on integrity check

With knowledge of the key, it is a simple task for an insider to find two messages which produce the same integrity check. This is a serious flaw in integrity check mechanism.

Due to these two flaws, especially Flaw 2, the evaluators would not recommend the adoption of MULTI-S01 Algorithm as a standard.

We finally note that the lack of a theoretical proof for the security of the PRNG Panama is clearly a potential weakness in this system. It is interesting that a scheme that relies so heavily on the security of the underlying PRNG has been defined to use Panama, which is certainly very fast but unfortunately has no proof of security. Given the existence of design methods and criteria for provably secure stream ciphers, it is clear that the design of MULTI-S01 is more strongly influenced by performance efficiency than by the need for future security.

2 Description of MULTI-S01

The MULTI-S01 cipher is an entire message encryption algorithm. A new key is used for every new message. The encryption process can be outlined as follows.

1. Create the key material (A, all B_i and S) by running Panama on the combination of key K and diversification parameter Q.
2. Pad the message to a multiple of 64 bits using standard means. Then append two blocks with key and redundancy data.
3. Encrypt block "i" by XOR with 64-bits of Panama output: B_i. Store this as F_i
4. Multiply each block by the 64-bit value A, using the designated finite field.
5. XOR F_{i-1} by the result of (b) to make ciphertext block "i".

The decryption operation is the reverse of encryption, with the provision that if the redundancy values do not match, then the ciphertext is deemed invalid and the recovered plaintext is not output. Decryption is as follows.

1. Execute Panama(K,Q) to produce the key material {A, B_i, S}
2. Decrypt each block by XOR with F_{i-1} data, multiply by A^{-1}, and XOR with B_i.
3. Check that the calculated redundancy values agree with what was expected.
4. Either output the recovered plaintext block "i", or declare it to be invalid.

The Panama stream cipher is constructed as a large buffer that is slowly updated, together with a state that is quickly changed by a nonlinear operation and XOR with buffer material. The internal data of the Panama cipher is initially set to zero. Then the key and diversification data (each a block of 256 bits) are PUSHed into the cipher. Then 32 blank PULLs are performed to diffuse the input data around the buffer and state. After this the output bits are taken 256 bits per subsequent PULL operation. These PUSH and PULL operations are described in detail in [DAEM 98].

3 Structural Aspects

3.1 The MULTI-S01 Keystream Combining Operation

The typical method for combining keystream and plaintext for stream ciphers is the bitwise exclusive-or (XOR). This provides perfect secrecy for each bit but does still allow ciphertexts to be altered during transmission so that altered plaintext is recovered. In the worst case this may allow an attacker to subtly alter the semantic meaning of a message, for example changing the account number of a bank transaction.

To foil this attack, the MULTI-S01 algorithm uses a block chaining method to prevent ciphertext alteration. Firstly, finite field multiplication allows changes within a block to alter many plaintext bits. Secondly, the XOR of internal data from the previous block, together with the encoding of redundancy information, provides a mechanism for any alterations to result in the wrong value of redundancy being recovered. Any decryption that results in an invalid redundancy value is declared invalid, and the recovered "plaintext" is not revealed.

The features we would wish from an integrity check is that it will reject any altered ciphertext, and that passing the test establishes the validity of the recovered plaintext. Attack scenarios against the integrity check are discussed in Section 5.6. These attacks demonstrate tat the integrity check mechanism is seriously flawed.

3.2 Panama Structural Aspects

Panama has several thousand internal memory bits. Without obvious structural weaknesses, this is probably sufficient to provide resistance to conventional attacks. These aspects are discussed in Section 5.

The cycle structure of Panama is unknown. This relates to the lack of proof for the minimum period. There may be many separate cycles for each initialisation by K,Q, or they may all be on the same large cycle, merely out of phase. There may even be cycles that exist for the operation of iterated PULLS in Panama, but are never achieved in practice since there are no values of K,Q that create initial data that exist on those cycles. We contrast these uncertainties with the theoretical proof that exists for the period and cycle structure of traditional LFSR based stream ciphers: where all keys produce the same (albeit shifted) keystream that has maximal period. It may be that different K,Q values cause Panama output sequences that have different periods, so that not all keys would offer equal security. It is possible (but certainly improbable) that there exists small cycles for Panama, so that there would exist weak keys in the sense that the induced sequences have period smaller than the message length.

3.3 Equivalent Keys

The existence of equivalent keys was noted in [SELF], as corollary 1 in Section 2.3, where it was used to demonstrate the property of perfect secrecy. While we agree with this, there are also disadvantages of equivalent keys which the evaluators point out here.

In MULTI-S01, there are equivalent keys that reduce the effective keyspace by a factor of 2^{64}. This reduces the size of the keyspace from 2^{256} to 2^{192}. For any given triple of plaintext message, key and associated ciphertext, there is a set of 2^{64} keys that satisfy the triple. To see this, consider changing the value of A to any other value, then the values of all F_i change to new values, and so there will be some different value of B_i that allows the plaintext blocks to remain unchanged. Thus there exists 2^{64} -1 other equivalent sets of keys $\{A,B_i,S\}$. Whether these could have been generated by any original key K and diversification parameter Q is unknown. In implementations where there is no method of "authenticating" the key material $\{A,B_i,S\}$ there is no barrier to a key falsification attack, and hence a third party can be convinced that a different key $\{A',B'_i,S'\}$ was used in an encryption. This demonstrates that the key management is vital to the security of MULTI-S01.

3.4 Mode of Operation and Key Management

MULTI-S01 is a stream cipher. The confidentiality of the plaintext is guaranteed by the fact that every message is encrypted with distinct key material. Contrast this style of key management with that for block ciphers: many messages can be encrypted with the same key. It is somewhat dangerous for this stream cipher to be presented as a block encryption mechanism, since the differing key management requirements of these schemes result in the opportunity for a devastating attack. In particular, if MULTI-S01 is ever used in a manner that allows two (or more) messages to be encrypted under the same key, then a powerful differential-style attack can recover the value of A. Then, following the comments above, the entire key becomes known. Subsequently, any additional message that is encrypted under the

same key will be recoverable from the ciphertext alone. The attack can be undertaken as a chosen text or a known text attack, with very little data requirements. Either way the attack can be done in real time. Details of this attack appear in Section 5.5.

The significance of this attack will depend on the application context. For example in a situation where MULTI-S01 is chosen for use as the in-house encryption method for a company, then distributing a Trojaned version of the encryption software that does not change the key for every new message would allow the attack whenever two known plaintexts are available. This condition might be easily met if a standard header were encrypted at the start of every message. Then the attack would allow real time decryption of any ciphertext. Of course, this attack is true for any stream cipher, but we feel that MULTI-S01 is more likely to become vulnerable to the attack since it adopts a block mode that may encourage it to be used like a block cipher. Moreover, the consequences of such a mistake are catastrophic for the security of the cryptosystem.

This example shows that the keys do need to be changed for every message. The security of MULTI-S01 depends critically upon the correct key management techniques.

3.5 Bit Dependencies of Panama Algorithm in Stream Cipher Mode.

We have written a computer program that traces exactly those sets of K,Q bits upon which state bits depend. This allowed the diffusion properties of Panama to be studied. It was found that the key initialisation process is a very thorough mixing.

Summary: all state bits depend on all input K,Q bits after 5 of the 32 blank PULLS during the set-up phase. Thus there is over six times the effort required to achieve completeness: this is thorough mixing, when compared to DES [FIPS DES] that achieved completeness after 5/16 of its operation.

Each bit of the state depends on exactly the same number of K,Q bits as every other bit in the same word. This greatly simplifies the presentation of this data. We now present the range taken by the number of K,Q bits that influence the state bits, over all of the 17 words in the state. The table below shows the influence after a small number of PULLS, to illustrate the rate at which the influence increases.

Number of PULL operations	Number of influencing K,Q bits (512 total)
0	3 - 7
1	31 - 36
2	162 - 187
3	389 - 413
4	510 - 512
5	512

This data shows that all bits in a word have the same level of dependence on the initial K and Q values, so that we have proved there do not exist “weak” bits positions, in the sense of bits that depend on relatively few of the initial K,Q bits. We find the diffusion of Panama to be satisfactory, and we find no exploitable weaknesses in this aspect of the cipher.

3.6 Long Message Techniques

The MULTI-S01 algorithm has a specific method to deal with long messages: they are broken up into portions of 2^{38} bits each, indexed from 0, which are each then encrypted as 2^{32} blocks under the same key K, but different diversification parameters Q. The standard method is to let Q equal the index of the sub-message. The redundancy value R is stated to be the same as Q for each encryption. As R is 64 bits and Q is 256 bits it seems that R must be set to the 64 least significant bits of Q. We note that it is not clear from [MULTI-S01] how R is determined for standard, or short, messages.

This splitting of large messages has the effect of placing an upper bound of 2^{-32} on the probability that a random ciphertext will pass the integrity check. While this is a fine ambition, we note that the *overall* complexity of the random forgery attack is constant at 2^{64}, as set by the block size in bits (see Section 5.6).

4 Keystream Properties

The evaluators analysed the basic keystream properties of the Panama algorithm which provide cryptographic security. These include a large period, large linear complexity and white-noise statistics. It should be noted that for the Panama algorithm there are no theoretical results which can be applied to measure these properties, unlike LFSR based keystream generators where these properties are known.

Experimental results, which are included below for linear complexity and statistical analysis, were conducted by the evaluators using the CRYPT-X package. This is a statistical package which was previously designed by the evaluators for analysing encryption algorithms. The relevant pages from the CRYPT-X manual have been included in Appendix A.

4.1 Period

The actual period of the Panama algorithm is difficult to determine. All together there are 2^{512} keystream sequences that can be generated from Panama by selecting values for key, K, and diversification parameter value, Q. As mentioned in Section 3.3 there is a small probability that some of these sequences are equivalent. In order to determine the expected period of one of these sequences the evaluators are of the opinion that a suitable model is a block cipher in output feedback mode (OFB). As shown in [DAVI 82] this period can vary greatly. However it is possible to determine an expected value for the period by applying the birthday paradox. For block ciphers in OFB mode the expected period depends on the length of the input block to generator, n, and the number of bits which are feedback, L. If L = n then expected period is approximately 2^{n-1} and for L < n the expected period is approximately $\mathbf{2^{n/2-1}}$. In the case of the Panama cipher we have n = 544 and L = 256. This would mean that the expected period is approximately 2^{273}.

We should emphasise that the actual periods of different keystreams produced by Panama most likely will have a large variance. However the expected results above indicate that for any given sequence there is a high probability that this period is extremely large.

4.2 Linear Complexity

Since there is no theoretical support for the linear complexity of Panama, empirical tests on the value of linear complexity and the linear complexity profile have been investigated using the CRYPT-X package. The linear complexity tests were applied to the ten Panama output streams of length 10^6 bits, and the results gave linear complexity values extremely close to that expected for random data (i.e. half the bit-stream length). For more detailed results see Appendix B. The results for the linear complexity profile indicate that, as the bit-stream increases in length, the changes in linear complexity maintain the expected value of half the stream length. These results support the randomness of the output from Panama, based on linear complexity. These results support the fact that the whole bit-stream is required to re-construct the stream itself - thus giving an attacker no advantage in being able to create the bit-stream with a smaller number of output bits.

4.3 Statistical Analysis

The statistical randomness applied to Panama are the tests explained in the CRYPT-X package (see attached documents), namely the frequency, binary derivative, change point, subblock, runs distribution, sequence complexity, and linear complexity tests. The tests are based on the hypothesis that the measure obtained from the output stream supports randomness. The p-values obtained for the tests represent the probability that such a sample result would be obtained if the algorithm produces a random stream. Very small p-values would support non-randomness.

The first five tests were applied to binary output streams of 10^7 bits, and the two complexity tests were applied to binary output streams of 10^6 bits (due to the amount of time required for the tests), using ten different keys.

The subblock tests were applied to the output stream by dividing the bit-stream into non-overlapping subblocks of length ranging from 2 to 30 bits. The maximum subblock length of 30 was determined from the length of the file and the limitations of the test applied.

Results of Statistical Analysis

The results give the lowest p-value for any one test as 0.001, with 13 of the 360 p-values obtained falling below 0.05. This represents a proportion of 0.036 of the tests applied, which is below the 0.05 level, and so supports the randomness of the output from Panama. For more detailed results see Appendix C.

In the results of the statistical analysis of Panama presented in [SELF] there is recorded a run of length 32 in a sample of 2^{21}. We agree that this is a rare event with a small probability. In order to investigate this in more detail the length of the longest run was recorded for the samples of 10^7 bits. The length of the longest run in these larger samples that the evaluators analysed was 29 (Key 5), whereas the remaining keys gave the longest runs as 28. Hence, together with the results of the runs test applied from CRYPT-X package, this indicates there is no problem with the distribution of runs in samples generated by Panama.

The sequence complexity test provides an effective method of detecting periodicity or periodic patterns in the bit-stream. In the bit-streams tested all sequence complexity

values exceeded both the threshold value and the average value of sequence complexity for bit streams of length 10^6. These results support the conclusions on the period of Panama in Section 4.1 above.

5 Possible Attacks

The majority of the literature discusses attacks on stream ciphers that adhere to the LFSR combiner/filter model, in order to achieve provable properties with regard to period and linear complexity. The Panama cipher avoids these models and uses large internal memory, with the result the standard attack strategies seem to be ineffective. On the other hand, there are no proofs for even the basic security properties. Despite the lack of proof, however, we feel that the memory size and structure of Panama is sufficiently complex to ensure that the security of the cipher cannot be compromised. The possible attacks are categorised and examined individually in the next few paragraphs.

In this section we also consider attacks against the integrity check. We have established the complexity of random forgery attacks made by a wiretapper, and also assessed the ability of an insider to cheat the system. These integrity attacks are discussed in more detail in Section 5.6.

5.1 Simple Attacks

Simple attacks are those based on exhaustive search, period, linear complexity, and statistical attacks on non-uniform output. The memory size of Panama is clearly so large that exhaustive search is infeasible. The period of Panama (see also Section 4.1) is (very probably) not less than 2^{273}. This is obtained by considering the Panama structure to be similar to a block cipher in OFB mode, with less then the full amount of feedback [DAVI 82]. The linear complexity is also likely to be very large. Note that so long as the period and linear complexity are larger than the amount of data being encrypted under a single key, then the scheme can be said to be operationally secure, ie., it is secure so long as the correct operation of the key management is maintained. It seems likely that the period and linear complexity of Panama are both many orders of magnitude larger than any feasible message length, and so Panama appears to be operationally secure against the basic attacks. However, we re-state that there is *no proof* for the minimum period or linear complexity of Panama.

5.2 Divide and Conquer Attacks

Divide and conquer attacks are applicable where the structure of a keystream generator can be exploited by assuming some known data, and the remaining bits can be recovered faster than exhaustive search. In the case of Panama the total memory size is several thousand bits, and there is no clear way to conduct a divide and conquer attack so that it results in a break faster than exhaustive search.

There is a two bit rotation in the buffer update process, and this value is not relatively prime to the size of buffer data, so that the buffer may be considered to be two separate buffers of half the size, which do not interact directly. This observation might lead to a divide and conquer attack , except for the fact that these two half buffers do interact

indirectly via the effect of the state data. This interaction serves to prevent the attack being effective.

5.3 Correlation Attacks

Correlation attacks can be performed where the model of a noisy LFSR sequence is used to exploit correlations between state bits and output bits, and in particular correlations with single bits are preferred. Our analysis of the Boolean function that underlies the nonlinear state update transformation indicates that no biases exist with fewer than 3 input bits, or equivalently that the function is second order correlation immune [SIEG 85]. This, combined with the vast internal memory, make correlation attacks on Panama appear to be infeasible. All non-zero correlations have probability 0.5 ± 2^{-3}. These correlations are not great enough to make the attack feasible, given the order of correlation immunity and the large size of the internal memory. We conclude that correlation attacks against Panama are infeasible.

5.4 Linear Cryptanalysis

The basic approach for linear cryptanalysis of block ciphers, as pioneered by Matsui [MATS 93] is the collection of a sufficiently large number of data samples, so that any available input/output correlations can be used to correctly determine the values of key bits. The bias provided by a set of linearly independent modulo 2 equations is given by the piling-up lemma, which relies on the equations being independent, and all internal values cancelling out leaving a linear expression for key bits in terms of known plaintext and ciphertext bits.

Linear cryptanalysis of stream ciphers using a linear sequential circuit approximation was investigated by Golic [GOLI 94], where a general attack was suggested that exploits a general statistical weakness of output blocks which exceed the size of the internal memory. These attacks are especially effective against stream ciphers based on LFSR combination/filtering. However, structures such as Panama are not particularly susceptible to this attack. Given a suitable nonlinear Boolean function (which Panama has) then the dominant factor affecting the complexity of the linear attack is the size of the internal memory. According to [GOLI 94], the complexity of finding useful linear correlations, is at least $O(2^m)$, where m is the size of the internal memory in bits. As Panama has 8736 internal memory bits, the attack is infeasible.

The evaluators suggest that the wrong model for linear cryptanalysis of stream ciphers was employed in [SELF], and that the approach described in [GOLI 94] should be used. We have several criticisms of the analysis in [SELF]:

1) The approach in [SELF] ignores the 256 bits of additional state data that is not output at each PULL operation, but is fed back into the buffer.
2) In addition, [SELF] contains a claim that linear correlations involving one bit exist for the nonlinear operation in Panama, which have bias of 2^{-3}. We strongly disagree with these claims, as an exhaustive search of the appropriate Boolean function properties reveals that there are exactly 64 linear correlations with a bias of 2^{-3}, but these biases exist only for input masks of weight 3 or more. In fact, all other linear approximations have no bias at all, so that the Panama state update transformation is immune to correlations with linear functions of 2 inputs or fewer. This means that the Boolean function in Panama satisfies second order correlation immunity, which is a very desirable property in stream ciphers as it places a lower bound on the complexity of a

correlation attack. The details of this Boolean function, and its analysis, appear in Appendix D.

3) The number of independent equations required to complete the cryptanalysis is stated without evidence in [SELF] as 63. However, our analysis suggests that *at least 256* independent equations for single buffer/key bits are required. This would greatly increase the data requirements for an attack.

We conclude that, due to the very effective and complex nonlinear mixing operation of Panama, conducting any linear cryptanalytic attack would require infeasible amounts of memory and computation.

5.5 Differential Cryptanalysis

Firstly, we note that differential attacks are not generally applicable to stream ciphers. This is primarily because the attacker has no ability to change internal data of a stream cipher operation. We recall that differential cryptanalysis was originally developed in relation to block ciphers (see [BS90]) where it exploited the attackers ability to generate the encryptions/decryptions of specially chosen pairs of text, under the same key. As the keystream generator of a stream cipher is a deterministic finite state machine initialised by the key, the concept of a differential attack is not meaningful in the context of stream ciphers, where every message is supposed to be encrypted under a separate key.

However, in the case that two MULTI-S01 messages are encrypted with the same key, then we note the possibility for a powerful attack that recovers the value of the A key. Once that is found the other B_i and S keys can be found easily. The attack works as follows.

Given two different messages (which may be no more than a single block each), encrypted under the same (but unknown) key, we have two different ciphertexts. Alternatively, in a chosen ciphertext attack, we may select the exact XOR difference in the ciphertext block and observe the effect on the recovered plaintext. The attack can proceed either way, and we note that only the key A affects the XOR differences, since all other keys are XORed, and so cancel out.

Analysis of the finite field multiplication (FFM) operation reveals that all output bits are second order, or *quadratic*, Boolean functions of the bits in the fixed multiplication input, which is the key A. It follows from results in [LAI 94], [MILL 95] that the Boolean function of the derivative of FFM is linear in these input bits. Hence the XOR difference in the recovered plaintext gives 64 bits of information that can be used as the right hand side data for a set of 64 simultaneous equations in the 64 unknown A bits. The matrix is specified by the polynomial used to determine the finite field representation, so this matrix is public information, and the same for any key being used. If this matrix is non-singular, then a unique solution exists and can be found by inverting the matrix. If, on the other hand the matrix does not have 64 linearly independent equations, then extra equations can be obtained by repeating the attack on messages of length greater than one block. With high probability a set if linearly independent equations will be obtained from the encryption of very few blocks. The complexity of matrix inversion is polynomial in the size of the matrix. Here the matrix is 64*64, so no more than a couple of thousand fast operations are required to invert it. We conclude that the complexity of this attack is

so very low that it is quite feasible to perform in real time. Of course it is only applicable if the strict key management requirements of stream ciphers is violated.

5.6 Attacks on the Integrity Check

The security of MULTI-S01 has been investigated in relation to its capability of providing an integrity check.. The complexity of creating a wrong ciphertext that does satisfy the integrity check is 2^{64} block encryptions. This value depends entirely on the block size. There is a message length/time/probability tradeoff in the attack, so, for example, there exists a probability of 2^{-32} that a single message of size 2^{32} blocks will satisfy the integrity check. This probability might be too high for some secure applications.

Our analysis has shown that the integrity check performs correctly in one aspect: given a fixed ciphertext and the corresponding key there is only one possible plaintext that could have been sent. To see this, consider the decryption of the last two blocks of a message, which are the integrity check blocks. If the last two blocks are confirmed as equalling S and R then the chaining data F_{n-2} must be correct. Tracing the effects backwards in decryption for each block reveals that the recovered plaintext blocks are correct.

However, we note from [SELF] that the possibility to alter the ciphertext and yet pass the integrity check is non-zero. In fact it has been shown that the probability for a random *n*-block ciphertext to pass the decryption integrity check is given by the proportion of the A key values which are the solution to a polynomial of degree *n-2* in A. As there are at most *n-2* different solutions, the maximum probability of passing the integrity check with each try is $(n-2)/(2^{64}-1)$. As the number of blocks of a message in never more than 2^{32}, we see that the probability for a randomly selected ciphertext to pass the integrity check is no more than 2^{-32}. The complexity of making the attempt at this maximum probability is the encryption of 2^{32} blocks. The overall complexity of the attack is $2^{32} * 2^{32}=2^{64}$. Hence the complexity of the random forgery attack is given by the number of bits per block: 64. Increasing the block size will directly increase resistance to the forgery attack and make the integrity check more reliable.

We visualise two possible attacks on the integrity check of MULTI-S01. The first involves an eavesdropper(wiretapper) who randomly alters the ciphertext hoping the resulting ciphertext passes the receiver's integrity check. The second involves an insider who knows the fixed value *A*, message *M*, and ciphertext *C* enabling such a person to make changes to the ciphertext which are guaranteed to pass the receiver's integrity check. Note that message *M* consists of the first *n*-2 blocks of the plaintext *P*.

We refer to the following equation that appears towards the end of section 2.4.1 in [SELF].

$$0 = \oplus_{i=0\ldots n'-2} \delta_{n'-i-1} A^{-(i+1)} \qquad (1)$$

This is the equation that must be satisfied for an altered ciphertext to pass the receiver's integrity check. Note that if C_i is the original ciphertext block then $C_i \oplus \delta_i$ represents the altered ciphertext C^*_i.

5.6.1 Wiretapper Attack

In this attack, the wiretapper selects some δ_i's at random and hopes that they satisfy equation (1). As per [SELF], the probability that the randomly chosen δ_i's satisfy equation (1) is the probability that the actual value of *A* used is one of the *n*-2 possible values of *A* that satisfy equation (1) for the chosen δ_i's. This probability is $(n\text{-}2)/(2^{64}\text{-}1)$ as appears in [SELF]. Given that the largest value of *n* is 2^{32} this probability is upper bounded by approximately $1/2^{32}$. Thus, the complexity of this attack is approximately 2^{32}. Even if the chosen δ_i's result in a ciphertext that passes the receiver's integrity check, it is highly unlikely that the plaintext, resulting from the altered ciphertext, will be meaningful, so in some sense this attack represents a denial of service. Note that this attack is a ciphertext only attack, since the wiretapper need only have access to the ciphertext. Note further that such an attacker cannot verify the success or otherwise of the attack.

In the light of the note at the end of the preceding paragraph and the complexity of the attack, we do not believe that this attack is a serious threat to MULTI-S01 which is in agreement with the analysis in [SELF].

5.6.2 Insider Attack

In the second attack, we are assuming the insider has access to the value *A*, message *M* and ciphertext *C*. While in [SELF] knowledge of *A* is specifically denied, it is plausible that either the sender or the receiver could act maliciously and deny sending or receiving a particular message, and be able to demonstrate as proof, a different message that satisfies the integrity check. This attack is clearly more menacing in that it is possible for the attacker to select δ_i's guaranteed to pass the integrity check as follows.

Since the attacker has knowledge of the value *A*, such a person can substitute this value into equation (1) and get a linear relationship among the *n-1*, δ_i's. The attacker can then choose $\delta_i = 0$ for $i = 0$ to *n-3* and $i = n$, then make a judicious selection for δ_{n-2} and finally set δ_{n-1} to be a certain predetermined value which will guarantee the receiver will have an altered message that will pass the integrity check.

The attacker carries out the following steps.

1. Change C_{n-2} to C^*_{n-2}.
2. Determine $P^*_{n-2} = (C_{n-2} \oplus C^*_{n-2}) \otimes A^{-1} \oplus P_{n-2}$.
3. Determine $C^*_{n-1} = P_{n-2} \oplus P^*_{n-2} \oplus C_{n-1}$.
4. Send ciphertext C_i for $i = 1$ to n-3 and for $i = n$, and C^*_{n-2} and C^*_{n-1}.

The proof is as follows.

By the decryption algorithm

$$P^*_{n-2} = (C^*_{n-2} \oplus F_{n-3}) \otimes A^{-1} \oplus B_{n-2} \qquad (2)$$

and

$$P_{n-2} = (C_{n-2} \oplus F_{n-3}) \otimes A^{-1} \oplus B_{n-2} \quad (3)$$

(2) ⊕ (3) gives

$$P^*_{n-2} \oplus P_{n-2} = (C^*_{n-2} \oplus F_{n-3} \oplus C_{n-2} \oplus F_{n-3}) \otimes A^{-1} \quad (4)$$

Thus, $$P^*_{n-2} = (C^*_{n-2} \oplus C_{n-2}) \otimes A^{-1} \oplus P_{n-2} \quad (5)$$

Now $$S = (C^*_{n-1} \oplus F^*_{n-2}) \otimes A^{-1} \oplus B_{n-1} \quad (6)$$

and $$S = (C_{n-1} \oplus F_{n-2}) \otimes A^{-1} \oplus B_{n-1} \quad (7)$$

(6) ⊕ (7) gives

$$C^*_{n-1} = C_{n-1} \oplus F_{n-2} \oplus F^*_{n-2}$$
$$= C_{n-1} \oplus P_{n-2} \oplus P^*_{n-2} \quad (8)$$

A simple example with $n = 4$ has been implemented and is described below.

```
Panama parameters (K & Q) as per supplied test vectors:

K: bcfb8d1629964f571bea19eacdd1a9a8870622392094af2bc18e6942eb017b0f
Q: 70b91d006387740d0397095a33de361f4a631b3023f06c6a2e3c0cb2647f26bb

R: 0000000000000000

Output from Panama:
A:  0D8B8EF7C5814F2F    ( A-1: 5AA6FCBEC904B0CA )
B1: 0431A2CEFB23CBA4
B2: BFF041318C4F63E8
B3: 65F97EC547EC7B63
B4: 82CFE82374CD5CC0
S:  19AE68546A9DCDAF


Message
P1: FE61A5EE9147C270
P2: A944B97E03269A33

Ciphertext
C1: 84A1CB62375A8DD2
C2: B4905ED917F05E15
C3: 0F918ACD813EEBB0
C4: 33EEA7357B9BAFDA

Let
C2*: 7AD3EFA784A51367   ( selected at random )

Then
P2*: 5E3D8D13E061F8BB   ( = ( ( C2 ⊕ C2* ) x A-1 ) ⊕ P2 )

and
C3*: F8E8BEA062798938   ( = P2 ⊕ P2* ⊕ C3 )

Decrypting ( C1 | C2* | C3* | C4 ) gives
P1': FE61A5EE9147C270   ( = P1 )
```

```
P2′: 5E3D8D13E061F8BB   ( = P2* )
P3′: 19AE68546A9DCDAF   ( = S )
P4′: 0000000000000000   ( = R )
```

The significance of the above result is the ability of this second type of attacker to forge a meaningful message. The attack can be applied to any message, of any length. Given two blocks of changed ciphertext, say blocks i and i+q, then the recovered plaintext is altered for consecutive blocks {i, i+1, …, i+q-1}, and is subsequently identical to the original plaintext, including satisfying the integrity check. We note that this simple example has only two blocks of ciphertext being altered. A real attack may change any number of ciphertext blocks, so long as the last alteration restores the plaintext to the original, thus allowing the integrity check to succeed.

The above attack is possible because the integrity check is really only a function of the last 3 plaintext blocks. What is really needed is for the integrity check to be a function of a greater number of plaintext blocks. The check should mimic a block cipher in Cipher Block Chaining (CBC) mode where the last output is a function of all the input blocks. As an example, for a similar attack to be performed on the Data Encryption Standard (DES) [FIPS DES] in CBC mode, the birthday paradox implies that the complexity would be 2^{32} since the block length of DES is 64. For the Advanced Encryption Standard (AES) this figure is 2^{64} since the block length is 128. Complexities of 2^{32} and 2^{64} are acceptable from a designer's point of view but the simplicity of the above attack (especially the determinism plus also the flexibility to choose the position of altered blocks) indicates a serious flaw in the design of MULTI-S01.

5.7 Other attacks

The Inversion attack [GOLI 96] is prevented by the large memory. The requirements are exponential in the total memory size (which is 8736 for Panama), so it is straightforward to conclude that the inversion attack against Panama is infeasible.

A reconstruction attack was found to be very effective against the mobile telecommunications algorithms ORYX [WAGN98] and CAVE [MILL98]. These attacks exploited a fundamental weakness of those algorithms. Well designed ciphers will be resistant to reconstruction attacks due to the increased time and data requirements. In particular the Panama cipher is far too complex to attack by reconstruction: there is no fast way of rejecting a bad guess for any set of data. We conclude that Panama is secure against the reconstruction attack.

6 Implementation

In this section we investigate the implementation issues for the MULTI-S01 algorithm. Section 4 of the self evaluation report [SELF] gives details of both software and hardware implementations of the MULTI-S01 algorithm.

6.1 Software

Panama (the MULTI-S01 "engine") is well suited to 32-bit processors because of its heavy use of 32-bit word operations. The Panama algorithm can also be optimised for 64-bit processors. The MULTI-S01 algorithm (which "wraps" Panama) performs a 64-

bit finite field multiplication which is better suited to 64-bit processors. The figures quoted in the self evaluation report (Section 4.1 of [SELF]) of 270Mbps on a 64-bit processor running at 600MHz and 55Mbps on a 32-bit processor running at 350Mhz highlight the difference in performance between 32-bit and 64-bit processors.

The evaluators were able to confirm that the Panama implementation is extremely efficient running at approximately 730Mbps on a Pentium III 650MHz processor. The overhead of the MULTI-S01 enhancements (especially the finite field multiplication) significantly reduces the performance (especially on computers with 32-bit processors).

The code size and memory stack utilisation figures quoted (Table 8, Section 4.1 of [SELF]) indicate that a software implementation of MULTI-S01 on a memory-poor device, such as a smart card, would be feasible.

6.2 Hardware

Section 4.2 of [SELF] describes logic circuit designs and gate size and throughput evaluations for MULTI-S01. The hardware implementation described is actually a simulation and the document states that the quoted figures may actually be slower in practice. The throughput rates of 8.3-9.1Gbps for a speed-optimised implementation, and 620-730Mbps for a gate-count-optimised implementation, if correct, are impressive and indicate suitability for all but the most bandwidth-intensive applications.

7 Conclusion

The evaluators have conducted a thorough analysis of the properties of the Panama and MULTI-S01 algorithms. There are some observed flaws in MULTI-S01, relating to the integrity check mechanism and also the key management. We also point out that there are several problems with the analysis of [SELF].

The prospect of attacks on the integrity check mechanism are significant. For this reason MULTI-S01 should not be used for high-security applications. Also the catastrophic results of violating the key management imply that only fully trusted implementations of MULTI-S01 should be used in any case. The evaluators suggest that MULTI-S01 be re-designed to address these issues before it is considered as a future standard.

We note that the lack of proof for the security properties of Panama remains a concern, as other schemes do have proof of basic requirements. On the other hand, there seems to be no pressing reason to doubt the security of Panama, and all of our analyses on that algorithm suggest that, with high probability, it has no exploitable weaknesses. An advantage of this scheme is the very high rate of data throughput.

8 References

[BS 90]
E. Biham and A. Shamir, Differential Cryptanalysis of DES-like cryptosystems, In *Advances in Cryptology – Crypto'90*, Volume 537 of *Lecture Notes in Computer Science*, pages 2-21, Springer-Verlag, 1991.

[DAEM 98]
J. Daemen and C. Clapp, Fast hashing and Stream Encryption with Panama, Proceedings of *Fast Software Encryption Conference*, Volume 1372 of *Lecture Notes in Computer Science*, pages 60-74, Springer-Verlag, 1998.

[DAVI 82]
D.W. Davies and G.I.P. Parkin, The average cycle size of Key stream in Output feedback Encipherment. Cryptography, Proceedings of Burg Feuerstein Conference, Volume 149 of *Lecture Notes in Computer Science*, pages 263-280, 1982.

[FIPS DES]
Federal Information Processing Standard: The Data Encryption Standard, US National Bureau of Standards, 1977

[GOLI 94]
J. Dj Golic, Linear Cryptanalysis of Stream Ciphers, Proceedings of *Fast Software Encryption – Leuven '94*, Volume 1008 of *Lecture Notes in Computer Science*, pages 154-169, Springer-Verlag, 1994.

[GOLI 96]
J. Dj. Golic, On the Security of Nonlinear Filter Generators, In Proceedings of *Fast Software Encryption – Cambridge '96*, Volume 1039 of *Lecture Notes in Computer Science*, pages 173-188, Springer-Verlag, 1996.

[LAI 94]
X. Lai, Higher Order Derivatives and Differential Cryptanalysis, in *Communications and Cryptography: Two sides of one tapestry*, pages 227-233, Kleuwer Academic Publishers, 1994

[MASS 69]
J. L. Massey, Shift Register Synthesis and B.C.H. Decoding, *IEEE Trans. Inform. Theory*, IT-15:122-127, January 1969.

[MATS 93]
M. Matsui, Linear Cryptanalysis method for DES cipher, In *Advances in Cryptology - Eurocrypt'93*, Volume 765 of *Lecture Notes in Computer Science*, pages 386-397, Springer-Verlag, 1994.

[MILL 95]
W.L. Millan, Low order approximation of cipher functions, In *Proceedings of CPAC 1995*, LNCS Vol. 1029, pages 144-155, Springer-Verlag, 1995.

[MILL 98]
W.L. Millan, Cryptanalysis of the Alleged CAVE Algorithm, In *Proceedings of ISISC'98*, Seoul, December 1998.

[MULTI-S01]
A Symmetric Key Encryption Algorithm MULTI-S01, Technical Document, October 2000.

[SELF]
Self Evaluation Report MULTI-S01. Technical Document, October 2000.

[WAGN 98]
D. Wagner, L. Simpson, E. Dawson, J. Kelsey, W. Millan and B. Schneier, Cryptanalysis of ORYX, In *Workshop on Selected Areas in Cryptography (SAC'98)*, Volume 1556 of *Lecture Notes in Computer Science*, pages 296-305, Springer-Verlag, 1998.

9 Appendices

Appendix A. Description of Statistical Tests

This appendix gives a mathematical description of the statistical tests used.

A.1 Mathematical Description of Stream Cipher Tests

This section contains a mathematical description of each of the tests. In each case an example is given to illustrate a particular test. The first five tests examine the hypothesis that the bit stream was based on Bernoulli trials where the proportion of ones and zeros is $\frac{1}{2}$. The two complexity tests examine the knowledge that a small subsection of the bit stream can be used to produce the remainder of the stream. If this is possible the string would not be considered to be random, especially in relation to its use in a stream cipher. The recommended size of a sample stream to test depends on the size of the average message which is being encrypted using the keystream. i.e. If an average cryptogram has size five million bits then one should use test samples of this length. It should be noted that not all of the tests can be applied to a string of this length due to computational limitations. For example, in the linear complexity test one would need to examine a smaller substring of the keystream. It is recommended that strings of length at least 100000 bits be used for testing.

A.1.1 Frequency Test

The *frequency test* checks that there is an equal proportion of ones and zeros in the bit stream. For randomness the proportion of ones and zeros in the bit stream should be approximately equal, since any substantial deviation from equality could result in a successful cryptanalytic attack on the cipher. For example, assume that a cryptanalyst attacking the stream cipher knows the type of plaintext being used, e.g. standard English text coded in 8-bit ASCII, and the keystream has $\frac{3}{4}$ of the bits zero. Under this assumption the cryptanalyst knows the frequency distribution of the plaintext in terms of single bits, digraphs and trigraphs. With this knowledge the cryptanalyst could recover a substantial amount of the plaintext, using ciphertext alone.

The number of ones in a random binary sequence follows a binomial distribution, with mean $\frac{n}{2}$ and variance $\frac{n}{4}$. This may be approximated using a normal distribution. The following notation is used:

n = total number of bits;

n_0 = number of zeros;

n_1 = number of ones;

$\hat{p} = \frac{n_1}{n}$ = proportion of ones in the sequence.

The aim of the frequency test is to determine how the proportion of ones, $\hat{p}$, in the sample stream of length n bits, fits into the hypothesised distribution where the proportion of ones, $\pi = 0.5$ and the variance, $\sigma^2 = \frac{1}{4n}$. This is a two-tailed test [BHAT 77]. The standardised normal test statistic is : $z = 2\sqrt{n}(\hat{p} - 0.5)$. The significance probability value, p, of the normal distribution is calculated for this statistic. This

measures the probability of obtaining a number of ones equal to or further from the mean of $\frac{n}{2}$ than this sample gives for the hypothesised (where $\pi = 0.5$ and $\sigma^2 = \frac{1}{4n}$).
A small significance probability indicates a significant result (i.e., the stream is considered to be non-random). For large values of n ($n > 100000$) a highly significant result (significance probability < 0.001) indicates a possible weakness in the cipher and it is recommended that no further tests be carried out on this sample as the imbalance of ones and zeros may effect their results.
It should be noted that passing the frequency test does not mean the stream is not patterned. The following highly patterned streams, where the number of ones and zeros are equal, will pass the frequency test:
11111111.....00000000.....
10101010101010.....
Hence further testing is required to obtain knowledge of any patterns in the stream.

Example:
Test stream:
10100010000101110001011000111010101010101010000001
Calculations and results:

$$n = 50$$
$$\sigma^2 = \frac{1}{4\times 50}$$
$$n_1 = 21$$
$$\hat{p} = 0.42$$
$$z = \sqrt{50}(0.42 - 0.5) = -1.13137$$
$$p = 0.2579$$

Interpretation:

> 25.79 % of bit streams of length 50 will have a number of ones equal to or further from the mean of 25, for the hypothesised distribution, than this sample. This sample satisfies the frequency test.

A.1.2 Binary Derivative Test

The binary derivative is a new stream formed by the exclusive-or operation on successive bits in the stream. Successive binary derivative streams may be obtained from each new binary derivative, each one being of length one less than its predecessor [CARR 88].
The proportion of ones in the i-th binary derivative gives the proportion of overlapping (i+1)-tuples from the original stream in one of two known groupings of these (i+1)-tuples. This will be explained for $i = 1$ and $i = 2$.
When $i = 1$ (first binary derivative) we are looking at the overlapping two-tuples: 00, 01, 10, 11 (in the original stream).
The proportion of ones in the first binary derivative, $\hat{p}(1)$, gives the proportion of the total number of 01 and 10 patterns in the original stream.
$\hat{p}(1) > ½$ means there is a larger proportion of the group of 01 and 10 two-tuples (in the original stream).
$\hat{p}(1) < ½$ means there is a larger proportion of the group of 00 and 11 two-tuples (in the original stream).

A combination of the frequency test on the original stream and its first binary derivative is equivalent to testing that there is an equal number of these four overlapping two-tuples in the original stream. This replaces the well-known Serial Test [DAWS 91].
When $i = 2$ (second binary derivative) we are looking at overlapping three-tuples: 000, 001, 010, 011, 100, 101, 110, 111 (in the original stream). The proportion of ones in the second binary derivative, $\hat{p}(2)$, gives the proportion of the total number of 001, 011, 100, 110 patterns in the original stream.
$\hat{p}(2) > ½$ means there is a larger proportion of the group of 001, 100, 110, and 011 three-tuples.
$\hat{p}(2) < ½$ means there is a larger proportion of the group of 000, 010, 101, and 111 three-tuples.
A combination of the frequency test on the original stream and a similar test on the first and second binary derivatives, tests that there is an equal number of the eight overlapping three-tuples in the original stream, for practically all cases. If a cipher gives a satisfactory result to these tests AND also the change point test, then it can be considered to generate equal numbers of the overlapping three-tuples.

Notation:

$n_1(i)$ = number of ones in the i - th derivative

$\hat{p}(i) = \dfrac{n_1(i)}{n-i}$ = proportion of ones in the i - th derivative

The frequency test is applied to each stream and the standardised normal variable is found for the proportion of ones in each of the first two binary derivatives:
$z(i) = 2\sqrt{n-i}\,(\hat{p}(i) - 0.5)$, for $i = 1,2$.
The significance probability value, p_i, of the normal distribution is calculated for each statistic. A small significance probability indicates a significant result. For large n ($n > 100000$) a highly significant result (significance probability < 0.001) indicates a possible weakness in the cipher.

Example:
Test stream:

101000100001011100010110001110101010101010000001

Calculations and results:

D_1 : 111001100011100100111010010011111111111111000001

D_2 : 001010100100101101001110110100000000000000100001

Frequency test on first binary derivative (D_1) :

$n_1(1) = 30$

$\frac{n-1}{2} = 24.5$

$\hat{p}(1) = \frac{n_1(1)}{n-1} = \frac{30}{50-1} = 0.61224$

$z(1) = 2\sqrt{49}(0.61224 - 0.5) = 1.57143$

$p_1 = 0.1161$

Interpretation:

11.61 % of bit streams of length 49 will have a number of ones equal to or further from the mean of 24.5, for the hypothesised distribution, than this sample. This sample satisfies the frequency test on the first binary derivative.

Since the frequency test is satisfied for the original stream and the first binary derivative then the cipher can be regarded as producing an equal number of overlapping two-tuples.

Frequency test on second binary derivative (D_2) :

$$n_1(2) = 16$$

$$\frac{n-2}{2} = 24$$

$$\hat{p}(2) = \frac{n_1(2)}{n-2} = \frac{16}{50-2} = 0.333$$

$$z(2) = 2\sqrt{48}(0.333 - 0.5) = -2.3094$$

$$p_2 = 0.0209$$

Interpretation:

2.09 % of bit streams of length 48 will have a number of ones equal to or further from the mean of 24, for the hypothesised distribution, than this sample. This sample satisfies the frequency test on the second binary derivative.

Even though the frequency tests on the original stream and the first and second binary derivatives were all satisfied, the cipher will still have to satisfy the change point test before regarding it as producing an equal number of overlapping three-tuples.

A.1.3 Change Point Test

At each bit position, t, in the stream the proportion of ones to that point is compared to the proportion of ones in the remaining stream.

The difference or *change* in these proportions is compared for all positions in the bit stream. The bit where the maximum change occurs is called the *change point*. The test applied determines whether this *change* is significant for a binomial distribution where the proportion of ones in the stream is expected to be 0.5.

This test is very useful for detecting patterned streams which have passed the frequency test on the stream and the first two binary derivatives. Even if $\pi = \frac{1}{2}$ and the stream has passed the frequency test it could be, for $n = 10^6$, that $\pi = \frac{1}{4}$ for the first 500000 bits and $\pi = \frac{3}{4}$ for the second 500000 bits. This is not considered to be a good pseudorandom sequence to be used as a keystream, and the change point test would detect such cases.

This test is also useful for checking that there is an equal number of overlapping three-tuples for streams which have passed the frequency test on the original stream and also on the first two binary derivatives.

The hypothesis to be tested is that there is no change in the proportion of ones throughout the whole stream. The statistic [PETT 79] used is $U[t] = n \times S[t] - t \times S[n]$ where

n = total bits in stream

$S[n]$ = total ones in stream

$S[t]$ = number of ones to bit t

The maximum absolute value of this statistic is found:

Max = Maximum of ABS($U[t]$), for $t = 1 \ldots n$

The significance probability, p, associated with this statistic is approximated by:

$$p = e^{-\frac{2Max^2}{nS[n](n-S[n])}}.$$

For small values of p the actual significance probability is smaller than that calculated. The smaller the value of p then the more significant the result. For large streams a highly significant result, p < 0.001, indicates a possible weakness in the algorithm.

Example:
Test stream:

s = 10100010000101110001011000111010101010101010000001

Calculations and results:

$n = 50$

$S[n] = 21$

$t = 43$

$S[t] = 20$

$\text{Max} = |50 \times 20 - 43 \times 21| = 97$

$p = e^{-\frac{2 \times 97^2}{50 \times 21 (50-21)}} = 0.5390$

Interpretation:

The actual significance probability of the change in the proportion of ones is less than 53.9%. This result indicates there is no significant change in the proportion of ones in the bit stream. This sample satisfies the change point test.

A.1.4 Subblock Test

The stream is divided into S non-overlapping subblocks, each of length b. Any fractional subblocks remaining are ignored. For a stream of length *n*, the number of subblocks is the integral part of $\lfloor \frac{n}{b} \rfloor$, i.e. S = $\lfloor \frac{n}{b} \rfloor$.

For a subblock size of b ≤ 16 a test of uniformity is applied – i.e., there should be an equal number of each b bit pattern. The test compares the observed number of each b bit pattern with $S/2^b$.

The test statistic used is $\chi^2 = \frac{2^b}{S} \sum_{i=0}^{2^b-1} f_i^2 - S$ [BEKE 82], where f_i is the frequency of subblock pattern whose equivalent decimal value is i. This statistic is compared with a chi-square distribution with degrees of freedom equal to $2^b - 1$. For values of b > 6 the normal distribution may be used to approximate the chi-square distribution. Limitations: The minimum length required for the stream to test for randomness using b-bit subblocks is $5b \times 2^b$ bits.

For a subblock size of b > 16 the repetition test is applied. The repetition test measures the number of repeated patterns in a sample of S subblocks, each containing b bits. Given the binary stream is divided into S b-bit subblocks then, for a random stream, each of the $N = 2^b$ possible binary b-bit patterns is equally likely to occur. As the block length increases and $N \to \infty$, with a sample of size $S \to \infty$ where $\frac{S}{N} \to 0$, then the distribution of the number of subblock repetitions in the sample approaches a Poisson distribution with a mean of $\lambda = S - N(1 - e^{-\frac{S}{N}})$. When $S = 8\sqrt{N}$ the mean converges to 32, for large

values of b (say b > 16). The Poisson distribution is well approximated by the normal distribution for $\lambda = 32$.

The test requires a count of the number of subblock repetitions, r. (Note that if a particular pattern occurs three times, then this would add two to the number of repetitions).

The number of b-bit subblocks required for the test is $S = 8\sqrt{N}$, and gives $\lambda \approx 32$.

The procedure is to sort the subblocks and then determine the number of repetitions, *r*.

The test statistic is $z = \frac{r - \lambda}{\sqrt{\lambda}}$ (standard normal statistic for a Poisson distribution with a mean equal to λ), and is compared with the standard normal distribution. A two-tailed test applies since both too few or too many repetitions may indicate non-randomness of the stream.

The required stream length to apply the repetition test using b-bit subblocks is $b \times 2^{\frac{b}{2}+3}$ bits. This is considerably less than the length of stream required to apply the uniformity test for subblocks of the same size. Since the stream lengths required are very large, no sample stream will be shown. Instead, the following data will be used to illustrate a test calculation for the uniformity test:

$b = 8$ (hence the uniformity test is applied)

$n = 100000$

$S = \left\lfloor \frac{100000}{8} \right\rfloor = 12500$

Number of 8 bit patterns = $2^8 = 256$

Assume $f_0 = 45,\ f_1 = 50, \ldots,\ f_{255} = 44$, to give :

$\chi^2 = \frac{2^8}{12500} \sum_{i=0}^{255} f_i^{\,2} - 12500 = 260$ (say)

Degrees of freedom = 255

$z = \sqrt{2 \times 260} - \sqrt{2 \times 255 - 1} = 0.24248$

$p = 0.4042$

Interpretation:

40.42% of all possible streams of length 100000 will have a distribution of 8-bit subblocks less uniform than this sample shows. This sample satisfies the subblock test for subblocks of length 8.

The following data is used to illustrate a test calculation for the repetitions tests:

$b = 18$ (hence the repetition test is applied)

$n = 100000$

$N = 2^{18} = 262144$

$S = 2^{\frac{18}{2}+3} = 4096$

Stream length tested $= 4096 \times 18 = 36864$ bits

$r = 38$ (say)

$z = \dfrac{38 - 32}{\sqrt{32}} = 1.06066$

$p = 0.1444$

Interpretation:

14.44% of all possible streams of length 36864 will have a 18-bit subblock repetition count further from the mean (32) than this sample shows. This sample satisfies the subblock test for subblocks of length 18.

A.1.5 Runs Test

The runs distribution test compares the distribution of the number of runs of ones (blocks) and zeros (gaps) with that expected under randomness. For a random binary stream where $\Pr(1) = \Pr(0) = \frac{1}{2}$ there should be an equal number of number of blocks and gaps of the same length. Based on Golomb's postulates, the expected number of runs of length i for a random binary stream should be $\frac{1}{2^i}$ of the number of runs, and for each length there should be an equal number of runs of ones and zeros, i.e. $E(r_{0i}) = E(r_{1i}) = \frac{Runs}{2^{i+1}}$, where *Runs* indicates the number of runs in the binary stream. The hypothesis to be tested is that the distribution of runs in the stream fits a binomial population for which $\Pr(1) = \Pr(0) = \frac{1}{2}$. The test applied is adapted from [MOOD40]. The long runs are added together to form new variables s_{0k} and s_{1k} corresponding to the number of gaps and blocks of length k or more, where $s_{0k} = \sum_{i=k}^{n_0} r_{0i}$ and n_0 is the number of zeros in the stream.

By adding the long runs together a certain amount of information will be lost. In order to minimise the amount of information lost, it is recommended here that $k = \lfloor \log_2 \frac{n+1}{5} - 1 \rfloor$. For a stream of length $n = 10^6$ this would give a maximum value of k = 16, and hence the number of gaps of length 16 or more would be added together to give $s_{0,16}$ and the number of blocks of length 16 or more would be added together to give $s_{1,16}$.

Explanation of terms:

n = number of bits in stream

n_1 = number of ones in the bit stream

r_{0i} = number of runs of 0 of length i

s_{0i} = number of runs of 0 of length i for $i < k$

s_{0k} = number of runs of 0 for lengths $\geq k$

r_{1i} = number of runs of 1 of length i

s_{1i} = number of runs of 1 of length i for $i < k$

s_{1k} = number of runs of 1 for lengths ≥ k

The variables:

$$u_i = \frac{r_{1i} - n(\frac{1}{2})^{2+i}}{\sqrt{n}} \quad i = 1,..., k-1$$

$$u_k = x_k = \frac{s_{1k} - n(\frac{1}{2})^{k+1}}{\sqrt{n}}$$

$$u_{k+i} = y_i = \frac{r_{0i} - n(\frac{1}{2})^{2+i}}{\sqrt{n}} \quad i = 1,..., k-1$$

$$u_{2k} = z = \frac{n_1 - \frac{1}{2}n}{\sqrt{n}}$$

are asymptotically normally distributed with zero means and variances and covariances:

$$\sigma(x_i, x_i) = \sigma(y_i, y_i) = (\tfrac{1}{2})^{i+2} - (2i-1)(\tfrac{1}{2})^{2i+4}$$

$$\sigma(x_i, x_j) = \sigma(y_i, y_j) = (1-i-j)(\tfrac{1}{2})^{i+j+4}$$

$$\sigma(x_i, x_k) = -(i+k)(\tfrac{1}{2})^{i+k+3}$$

$$\sigma(x_k, x_k) = (\tfrac{1}{2})^{k+1} - (2k+1)(\tfrac{1}{2})^{2k+2}$$

$$\sigma(x_i, y_j) = (5-i-j)(\tfrac{1}{2})^{i+j+4}$$

$$\sigma(x_k, y_j) = (4-k-j)(\tfrac{1}{2})^{k+j+3}$$

$$\sigma(x_i, z) = \sigma(y_i, z) = (i-2)(\tfrac{1}{2})^{i+3}$$

$$\sigma(x_k, z) = (k-5)(\tfrac{1}{2})^{k+2}$$

$$\sigma(z, z) = \sigma_{zz} = \tfrac{1}{4}$$

Test procedure:

1. Determine k.
2. Take a sample stream of n bits from a stream cipher. Determine the number of runs of each length to give s_{1i} and s_{0i} for $i = 1,..., k$.
3. Calculate u_j for $j = 1,...,2k$ using above formulae.
4. Determine $S = [\sigma_{ij}]$ which is a $2k \times 2k$ matrix. Calculate $S^{-1} = [\sigma^{ij}] = [\sigma_{ij}]^{-1}$. This will require obtaining the inverse of a matrix of up to 32^2 (1024) elements for $n \le 10^6$ bits. Calculate $Q = \mathbf{u}^T S^{-1} \mathbf{u} = \sum \sigma^{ij} u_i u_j$ which follows a χ^2_{2k} distribution (chi-squared distribution with 2k degrees of freedom). There are $(2k)^2$ terms in this sum.

$$Q = \sum_{i=1}^{2k} \sigma^{ii} u_i^{\,2} + 2 \sum_{i<j} \sigma^{ij} u_i u_j$$

The significance probability value, p, of the chi-squared distribution is calculated for this statistic. A small value of p indicates a significant result. For large streams a highly significant result, $p < 0.1\%$, indicates a possible weakness in the algorithm.

The runs test can be used to support results from the previous tests. Failure of the runs test indicates that there is a bad distribution of run lengths or that there are no runs recorded above a certain length that are expected to occur for streams of the sample size. The zero frequencies recorded will result in a higher chi-squared statistic thus giving a smaller significance probability.

Example:

Test stream:

10100010000101110001011000111010101010101010000001

Calculations and results:

$n = 50$

$\text{total runs} = 31$

$n_1 = 21$

$k = \lfloor \log_2 \frac{n+1}{5} - 1 \rfloor = \lfloor \log_2 \frac{51}{5} - 1 \rfloor = 2$

$s_{01} = 10, s_{02} = 5, s_{11} = 13, s_{12} = 3$

$u_1 = x_1 = \frac{13 - 50(\frac{1}{2})^3}{\sqrt{50}} = 0.9545941546018$

$u_2 = x_2 = \frac{3 - 50(\frac{1}{2})^3}{\sqrt{50}} = -0.4596194077713$

$u_3 = y_1 = \frac{10 - 50(\frac{1}{2})^3}{\sqrt{50}} = 0.5303300858899$

$u_4 = z = \frac{21 - \frac{1}{2}(50)}{\sqrt{50}} = -0.5656854249492$

$\sigma(u_1, u_1) = \sigma(u_3, u_3) = (\frac{1}{2})^3 - (2-1)(\frac{1}{2})^6 = 0.109375$

$\sigma(u_1, u_2) = -3(\frac{1}{2})^6 = -0.046875$

$\sigma(u_2, u_2) = (\frac{1}{2})^3 - 5(\frac{1}{2})^6 = 0.046875$

$\sigma(u_1, u_3) = 3(\frac{1}{2})^6 = 0.046875$

$\sigma(u_2, u_3) = 1(\frac{1}{2})^6 = 0.015625$

$\sigma(u_1, u_4) = \sigma(u_3, u_4) = -1(\frac{1}{2})^4 = -0.0625$

$\sigma(u_2, u_4) = -3(\frac{1}{2})^4 = -0.1875$

$\sigma(u_4, u_4) = \sigma_{44} = 0.25$

Elements of the inverse matrix, S^{-1} :

$\sigma^{11} = 6\frac{2}{3}, \sigma^{12} = -5\frac{1}{3}, \sigma^{13} = -4, \sigma^{14} = -3\frac{1}{3}, \sigma^{21} = -5\frac{1}{3}, \sigma^{22} = -5\frac{1}{3}, \sigma^{23} = 0, \sigma^{24} = -5\frac{1}{3},$

$\sigma^{31} = -4, \sigma^{32} = 0, \sigma^{33} = 12, \sigma^{34} = 2, \sigma^{41} = -3\frac{1}{3}, \sigma^{42} = -5\frac{1}{3}, \sigma^{43} = 2, \sigma^{44} = -\frac{1}{3}.$

Q = 8.4733.. follows a χ^2_4 distribution.

p = 0.076

Interpretation:

> *7.6% of bit streams of length 50 will have a distribution of run lengths further from the expected distribution than this sample gives. This sample satisfies the runs distribution test.*

A.1.6 Sequence Complexity Test

The sequence complexity, c(s), is the number of different substrings encountered as the stream, s, is viewed from beginning to end [LEMP 76].

Example (n = 16) :

s = 1 0 0 1 1 1 1 0 1 1 0 0 0 0 1 0

Marking in different substrings :

s = 1/0/0 1/1 1 1 0/1 1 0 0/0 0 1 0/

Here the sequence complexity c(s) = 6

A *threshold value* of sequence complexity is used to measure the randomness of a sequence. This *threshold value* is $\frac{n}{\log_2 n}$ where n is the total bits in the stream. A stream with a sequence complexity measure below this *threshold value* would be considered to be patterned, ie not random. For the example given, the *threshold value* $= \frac{16}{4} = 4$. Hence the stream is not considered patterned.

An expected value for the sequence complexity of a random stream of the same length is calculated using the following algorithm [GUST 96]:

$i = 2;$

$c = 2;$

while $(i < n)$ do

begin

$i = i + \left\lfloor \frac{\log(i-1)}{\log(2)} \right\rfloor + 2;$

$c = c + 1;$

end;

if $(n < i)$ then $c = c - 1;$

It is expected that a good pseudo-random number sequence has a sequence complexity which is close to this value. It should be noted that the expected value of sequence complexity is always greater than the threshold value. However, a bit stream will only be considered to not satisfy the sequence complexity test if the value of c(s) is less than the threshold value.

The sequence complexity is used to replace the autocorrelation test which is commonly used to determine any periodicity in the pseudorandom number generator. Periodicity would greatly reduce the number of "different" substrings encountered. Hence c(s) would be low and fall below the threshold value. [DAWS 91]

Example

Test stream:

10100010000101110001011000111010101010101010000001

Calculations and results:

n = 50

c(s) = 10

Threshold value = 8.859191

Expected value = 13

Interpretation:

This sample stream is considered random based on the sequence complexity test.

A.1.7 Linear Complexity Test

A.1.7.1 Linear Complexity

The linear complexity test checks for the minimum amount of knowledge (bits) needed to reconstruct the whole stream. Every finite stream, s, can be produced by a linear

feedback shift register (LFSR). The length of the shortest LFSR which will produce the stream is said to be the linear complexity of the stream, which will be denoted by L(s). If the value of L(s) is L then 2L consecutive terms can be used to reconstruct the whole sequence using the Berlekamp Massey algorithm. [MASS 69] Hence, in order to avoid stream reconstruction, the value of L should be large.

Example:

0101100101010010011110000011011100110001110101111110110 1

This shortest recurrence relation which will create this sequence is:

$$u(t+6) = u(t+5) \oplus u(t+4) \oplus u(t+1) \oplus u(t)$$

where $\oplus$ is addition mod 2, and the first bit is $u(0)$.

For example:

If $t = 0$ then $u(6) = u(5) \oplus u(4) \oplus u(1) \oplus u(0)$

$0 = 0 \oplus 1 \oplus 1 \oplus 0$.

If $t = 1$ then $u(7) = u(6) \oplus u(5) \oplus u(2) \oplus u(1)$

$1 = 0 \oplus 0 \oplus 0 \oplus 1$.

If $t = 2$ then $u(8) = u(7) \oplus u(6) \oplus u(3) \oplus u(2)$

$0 = 1 \oplus 0 \oplus 1 \oplus 0$.

This means that the linear complexity, L(s), of this sequence is six. If any twelve consecutive bits are known then the whole sequence can be reconstructed. [MASS 69]

It should be noted that some keystreams can pass all the previous tests yet possess a very small linear complexity. An example of this is an m-sequence (see [RUEP 84]). An m-sequence has a period of length $2^L - 1$ and a linear complexity of L. An m-sequence has the *best* possible distribution of zeros and ones for a sequence of period $2^L - 1$. In this fashion an m-sequence appears to be *statistically* random in terms of tests A.1.1 to A.1.6. In fact m-sequences are commonly used as *white noise* generators. However, in terms of their use in a stream cipher an m-sequence offers very low security. Knowledge of only 2L consecutive bits of the keystream is needed to derive the defining LFSR and hence determine the whole keystream.

For large n, L(s) is approximately normally distributed with $\mu = \frac{n}{2}, \sigma^2 = \frac{86}{81}$ [RUEP 84], [KREY 81]. Using the standardised normal statistic $z = \sqrt{\frac{81}{86}}(L(s) - \frac{n}{2})$ the significance probability value, *p*, of the normal distribution is calculated.

Since only low values of L(s) signify a possible weakness to the cipher, only a one-tailed test (lower tail) need apply. A small value of p indicates a significant result. For large streams a highly significant result ($p < 0.1\%$) indicates a possible weakness in the algorithm.

The linear complexity test by itself can classify as random, streams which may be highly patterned, or contain large substrings which are highly patterned. Some of the previous test results should support this. e.g. a stream of $\frac{n}{2} - 1$ zeros followed by a one, and then followed by a repetition of these $\frac{n}{2}$ terms, has a linear complexity of $\frac{n}{2}$. This stream would be classified as being random using the linear complexity test. Clearly, such a stream is highly patterned and would not satisfy the previous tests. However, it is possible to construct a stream of length n which would pass all the previous statistical tests, and have a linear complexity of approximately $\frac{n}{2}$ yet would contain a large highly patterned substring. Hence the following linear complexity profile tests are carried out.

A.1.7.2 Linear Complexity Profile

Since some highly patterned streams can give a linear complexity measure close to $\frac{n}{2}$ a second test measures the change in the linear complexity profile of the stream as each bit is added. Let s(i) be the substring formed by taking the first i bits of s. If L(s(i)) for i = 1,...,n denotes the linear complexity of s(i) then the values of s(i) are defined to be the linear complexity profile of s and should follow approximately the $\frac{i}{2}$ line [MASS 69]. A failure in this test would highlight any large deviations from the $\frac{i}{2}$ line, which would appear for strings passing the linear complexity test and containing any large highly patterned substrings. A change in linear complexity signifies a *jump*.
There are two tests relating to the Linear Complexity Profile:

A.1.7.3 Linear Complexity Profile – Number of Jumps

Let the total number of jumps be F. For large n, F is approximately normally distributed with $\mu = \frac{n}{4}$ and $\sigma^2 = \frac{n}{8}$ [CART]. The standardised statistic for the number of jumps is $z = \sqrt{\frac{8}{n}}(F - \frac{n}{4})$. The significance probability, *p*, for this standardised statistic is calculated. Since a small number of jumps would indicate a sequence within which patterns may exist, a one-tailed test (lower tail) is applied. A small value of p ($p < 0.1\%$) indicates that the number of jumps in linear complexity is low, and there may be patterns in the stream which would indicate a possible weakness in the cipher.

A.1.7.4 Linear Complexity Profile – Jump Size

If a stream passes the test on the number of jumps in linear complexity, then the distribution of jump heights may be investigated. The height of a jump is the difference in linear complexity when a change occurs. Let the total number of *jumps* in linear complexity be F, where f_i is the number of jumps of *height* i. For a random string based on Bernoulli trials where the probability of a one on each trial is one half, the probability, p_i that a given jump has height i is given by $p_i = (\frac{1}{2})^i$. Hence the expected number of jumps of height i, e_i, is given by $e_i = p_i \times F$.

The chi-squared statistic used is $\chi^2 = \sum_{i=1}^{m} \frac{(f_i - e_i)^2}{e_i}$ [CART 87]. The maximum value of $i = m$ is determined from the condition for the chi-squared test, that $e_i > 5$. The number of degrees of freedom, $m - 1$, is determined from the sample taken.
The significance probability value, p, of the chi-squared distribution is calculated for this statistic. A small significance probability indicates a significant result – i.e., the stream is considered to be non-random. For large samples a highly significant result, $p < 0.1\%$, indicates a possible weakness in the algorithm.

Example

Test stream:

10100010000101110001011000111010101010101010000001

Calculations and results:

Linear Complexity Test

$n = 50$

$\mu = \frac{n}{2}$

$\sigma^2 = \frac{86}{81}$

$L(s) = 25$

$z = \sqrt{\frac{81}{86}}(25 - \frac{50}{2}) = 0$

$p = 0.5$

Interpretation:

50 % of bit streams of length 50 will have a linear complexity less than this sample. This sample satisfies the linear complexity test.

Hence $2 \times L(s) = 50$ bits (the whole stream) is needed to reconstruct the stream using the Berlekamp-Massey algorithm.

Linear Complexity Profile - Number of jumps

$F = 15$

$z = \sqrt{\frac{8}{50}}(15 - \frac{50}{4}) = 1$

$p = 0.8413$

Interpretation:

84.13% of streams of length 50 will have a number of jumps in linear complexity less than this sample. This sample satisfies the test on the number of jumps in linear complexity.

Linear Complexity Profile – Jumps size

$f_1 = 11,\ f_2 = 2,\ f_3 = 0,\ f_4 = 1,\ f_5 = 1.$

$e_1 = 7.5$ Since $e_i < 5$ for $i > 2$, then these values are combined to give $e_{2+} = 7.5$. The corresponding values of f_i are combined to give $f_{2+} = 4$.

$\chi^2 = \frac{(11-7.5)^2}{7.5} + \frac{(4-7.5)^2}{7.5} = 3.27$

Degrees of freedom $= m - 1 = 2 - 1 = 1$

$p = 0.0707$

Interpretation:

Approximately 7.07% of bit streams of length 50 will have a sump size distribution further from the expected distribution than this sample gives. The sample satisfies the test on the distribution of the linear complexity jump size.

A.2 Bibliography

[BEKE 82] H. Beker and F.Piper, **Cipher Systems: The Protection of Communications**, Northwood Books, London, 1982.

[BHAT 77] G. Bhattacharyya and R. Johnson, **Statistical Concepts and Methods**, John Wiley & Sons, 1977.

[CARR 88] J.M. Carroll and L.E. Robbins, “Using binary derivatives to test an enhancement of DES”, **Cryptologia**, Vo1 12 number 4, 1988, pp 193-208.

[CART 87] G. Carter, “A statistical test for randomness based on the linear complexity profile of a binary sequence”, **Technical Report** for Racal Comsec Ltd., 1987.

[DAWS 91] E.P. Dawson, **Design and Cryptanalysis of Symmetric Ciphers**, PhD Thesis, Queensland University of Technology, 1991.

[DAWS 98] E.P. Dawson and H.M. Gustafson, “A Method for measuring Entropy of Symmetric Cipher Key Generators”, **Computers and Security**, Vol. 17 No. 2, pp 177 - 184, 1998.

[DIFF 79] W. Diffie and M.E. Hellman, “Privacy and Authentication: An Introduction to Cryptography”, **Proceedings of the IEEE**, Vol. 67, No. 3, March 1979, pp 397-427.

[FOLK 84] L.J. Folks, “Combination of Independent Tests”, **Handbook of Statistics**, Vol. 4, Elsevier Science Publishers, 1984, pp 113-121.

[GUST 96] H.M. Gustafson, **Statistical Analysis of Symmetric Ciphers**, PhD Thesis, Queensland University of Technology, Brisbane Australia, 1996.

[KOLC 66] V.F. Kolchin, “The Speed of Convergence to Limit Distributions in The Classical Ball Problem”, **Theory of Probability and its Applications**, 11, 1966, 128-140.

[KREY 81] E. Kreysig, **Introductory Mathematical Statistics**, John Wiley and Sons, 1981.

[LEMP 76] A. Lempel and J. Ziv, “On the complexity of finite sequences”, **IEEE Trans. on Information Theory**, Vol.IT-22, Jan.1976,pp 75-81.

[MASS 69] J.L. Massey, “Shift register sequences and BCH decoding”, **IEEE Transactions on Information Theory**, Vol. IT-15, Jan. 1969, pp 122-127.

[MAUR 92] U.M. Maurer, “A Universal Statistical Test for Random Bit Generators”, **Journal of Cryptology**, 1992, 5, pp 89-105.

[MARS 84] G. Marsaglia, “A Current View of random Number Generators”, Proceedings of the Sixteenth Symposium on the Interface, **Computer Science and Statistics**, Editor L. Billard, Elsevier Science Publishers, 1984. pp3 - 10.

[MASS 69] J.L. Massey, “Shift Register Sequences and BGH Decoding”, **IEEE Trans. on Information Theory**, Vol. IT-15, Jan. 1969, pp 122-127.

[MOOD 40] A.M. Mood, “The distribution theory of runs”, **Ann. Math. Statist.**, Vol 11, 1940, pp 367-392.

[PETT 79] A.N. Pettitt, “A non-parametric approach to the change - point problem”, **Appl. Statist.**, Vol. 28 No. 2, 1979, pp 126-135.

[RUEP1 84] R.A. Rueppel, **New Approaches to Stream Ciphers**, PhD Thesis, Swiss Federal Institute of Technology, 1984.

[RUEP2 84] R.A. Reuppel, “**Analysis and Design of Stream Ciphers**”, Springer-Verlag, 1986.

[STEP 86] M.A. Stephens and R.B. D'Agostino, “Tests based on EDF Statistics”, **Goodness of Fit Techniques**, **in Statistics, Textbooks and Monographs**; Vol. 68, Marcel Dekker Inc., 1986, pp 97-193.

[WEBS 86] A.E. Webster and S.E. Tavares, “On the Design of S-Boxes”, **Advances in Cryptology: Crypto' 85**, Springer-Verlag, 1986, pp. 523-530.

Appendix B.

Results of Linear Complexity Tests on Panama

Length of output stream = 1,000,000 bits.
Number of streams = 10

Test	***Key 1***	***Key 2***	***Key 3***	***Key 4***	***Key 5***	***Key 6***	***Key 7***	***Key 8***	***Key 9***	***Key a***
Linear Complexity	500000	500000	500001	500000	499999	500000	500000	500000	500002	500000
LC p-value	.500	.500	.834	.500	.166	.500	.500	.500	.974	.500
LC profile – Jumps	.814	.357	.791	.801	.074	.037	.467	.715	.498	.090
LC Profile – Jump Size	.419	.750	.186	.908	.921	.198	.077	.015	.745	.584

Secret Keys Used (64 HEX):
Key1: bcfb8d1629964f571bea19eacdd1a9a8870622392094af2bc18e6942eb017b0f
Key 2: 0000000000000000000000000000000000000000000000000000000000000001
Key 3: 0000000000000000000000000000000000000000000000000000000000000002
Key 4: 0000000000000000000000000000000000000000000000000000000000000004
Key 5: 0000000000000000000000000000000000000000000000000000000000000008
Key 6. 70b91d006387740d0397095a33de361f4a631b3023f06c6a2e3c0cb2647f26bb
Key7: 008cf9209a7df04d2d2a990b08b8fec3 bad772b3f0c401fce5cf1db8cfebd599
Key8: bad772b3f0c401fce5cf1db8cfebd599008cf9209a7df04d2d2a990b08b8fec3
Key9: f878de38af18a9a1098bd8dc9f9c5e015d5ae4d5004fcd1bbf37a779bb1fce2b
Keya: 5d5ae4d5004fcd1bbf37a779bb1fce2b f878de38af18a9a1098bd8dc9f9c5e01

Appendix C.

Results of Statistical Randomness Tests on Panama

Length of output stream = 10,000,000 bits.
Length of output stream = 1,000,000 bits. (Sequence Complexity Test)
Number of streams = 10

Test	***Key 1***	***Key 2***	***Key 3***	***Key 4***	***Key 5***	***Key 6***	***Key 7***	***Key 8***	***Key 9***	***Key a***
Frequency	.073	.705	.059	.580	.894	.071	.247	.009	.212	.134
Binary Derivative (1)	.181	.375	.058	.599	.192	.272	.622	.855	.460	.559
Binary Derivative (2)	.915	.487	.318	.625	.626	.956	.551	.597	.585	.700
Change Point	.964	.964	.964	.964	.964	.964	.964	.964	.964	.964
Subblock b = 2	.351	.895	.153	.855	.993	.089	.104	.028	.364	.212
Subblock b = 3	.248	.088	.153	.716	.302	.100	.412	.101	.077	.468
Subblock b = 4	.194	.015	.290	.675	.593	.339	.173	.257	.347	.034
Subblock b = 5	.621	.079	.681	.953	.623	.595	.411	.234	.610	.838
Subblock b = 6	.913	.124	.848	.787	.502	.491	.172	.265	.452	.411
Subblock b = 7	.590	.344	.001	.994	.030	.166	.474	.464	.706	.688
Subblock b = 8	.787	.022	.961	.459	.048	.468	.495	.392	.300	.198
Subblock b = 9	.776	.202	.406	.793	.587	.463	.812	.842	.626	.029
Subblock b = 10	.452	.805	.764	.963	.505	.484	.902	.403	.555	.629
Subblock b = 11	.533	.796	.791	.586	.546	.211	.654	.059	.577	.067
Subblock b = 12	.052	.117	.246	.689	.418	.589	.382	.389	.506	.188
Subblock b = 13	.654	.351	.447	.434	.861	.624	.955	.195	.461	.299
Subblock b = 14	.392	.028	.852	.908	.294	.215	.684	.447	.856	.001
Subblock b = 15	.554	.424	.253	.347	.580	.513	.140	.328	.676	.502
Subblock b = 16	.362	.554	.079	.353	.939	.606	.340	.128	.181	.764
Subblock b = 17	.596	.860	.216	.724	1.00	.724	.860	.112	.596	.596
Subblock b = 18	.216	.596	.216	.724	.289	.379	.724	.377	.724	.860
Subblock b = 19	.860	1.00	.289	.052	.860	.216	.377	.596	.480	.724
Subblock b = 20	.860	.377	.596	.596	.860	.860	.157	.596	1.00	.480
Subblock b = 21	.112	.860	.596	.596	.289	1.00	.058	.289	.724	.289
Subblock b = 22	.724	.480	.480	.880	.596	.480	.724	.724	.860	1.00
Subblock b = 23	.596	.596	.596	.724	.860	.480	.377	.596	.052	.596
Subblock b = 24	1.00	.377	.596	1.00	.860	.724	.724	.377	.289	.596
Subblock b = 25	.480	.480	.480	.860	.112	.216	.289	.724	.289	.860
Subblock b = 26	.377	.860	1.00	.596	.860	.724	.724	.596	.480	.289
Subblock b = 27	1.00	.480	.860	.216	.724	.052	.860	.077	.860	.724
Subblock b = 28	.377	.157	.596	.112	.289	.377	.724	.377	.289	.377
Subblock b = 29	.724	.077	.112	.377	.596	.860	.724	.860	.860	.724
Subblock b = 30	.724	.860	.216	1.00	.052	.596	.860	.860	.480	.480
Runs Distribution	.773	.145	.547	.574	.509	.229	.994	.702	.615	.493
Longest Run	28	28	28	28	29	28	28	28	28	28

Sequence Complexity	50787	50783	50774	50753	50776	50744	50773	50775	50789	50811
Linear Complexity	.500	.500	.834	.500	.166	.500	.500	.500	.974	.500
LC profile – Jumps	.814	.357	.791	.801	.074	.037	.467	.715	.498	.090
LC Profile – Jump Size	.419	.750	.186	.908	.921	.198	.077	.015	.745	.584

Sequence Complexity on 10^6 bits: Threshold value = 50,441 Mean value = 50172

Secret Keys Used (64 HEX):

Key1: bcfb8d1629964f571bea19eacdd1a9a8870622392094af2bc18e6942eb017b0f
Key 2: 0000000000000000000000000000000000000000000000000000000000000001
Key 3: 0000000000000000000000000000000000000000000000000000000000000002
Key 4: 0000000000000000000000000000000000000000000000000000000000000004
Key 5: 0000000000000000000000000000000000000000000000000000000000000008
Key 6. 70b91d006387740d0397095a33de361f4a631b3023f06c6a2e3c0cb2647f26bb
Key7: 008cf9209a7df04d2d2a990b08b8fec3 bad772b3f0c401fce5cf1db8cfebd599
Key8: bad772b3f0c401fce5cf1db8cfebd599008cf9209a7df04d2d2a990b08b8fec3
Key9: f878de38af18a9a1098bd8dc9f9c5e015d5ae4d5004fcd1bbf37a779bb1fce2b
Keya: 5d5ae4d5004fcd1bbf37a779bb1fce2b f878de38af18a9a1098bd8dc9f9c5e01

Appendix D.

Details of the Nonlinear Boolean Function in Panama

The nonlinear operation of Panama consists of three parts: a nonlinear combination of words, word rotations, and the XOR combination of these values. This results in every bit of the state being a function of 9 bits of the previous state. (Here we ignore the extra XOR with buffer data). Due to the regular design, these Boolean functions all have the same structure, which we may represent as follows (where addition is mod 2)

$f(x)=x_1 + x_2x_3 + x_3 + x_4 + x_5x_6 + x_6 + x_7 + x_8x_9 + x_9 + 1$

This Boolean function has been analysed by Walsh-Hadamard Transform (WHT) and all of the possible linear correlations have been found. There are 64 linear functions with which f(x) is uncorrelated. These 64 linear functions are exactly all those which include the terms x1,x4, and x7. These linear functions all have weight 3 or more. All other linear functions have zero correlation with f(x). Hence, f(x) is second order correlation immune, an important fact not previously reported. The only non-zero WHT values are 64. The available linear bias is then given by $64/512=2^{-3}$.