Array-Based Statistical Analysis of the MK-3 Authenticated Encryption Scheme

Peter Bajorski[1], Alan Kaminsky[2], Michael Kurdziel[3], Marcin Łukowiak[4], and Stanisław Radziszowski[2]

[1]School of Mathematical Sciences, Rochester Institute of Technology
[2]Department of Computer Science, Rochester Institute of Technology
[3]Harris Corporation, Rochester, NY
[4]Department of Computer Engineering, Rochester Institute of Technology

***Abstract*—Authenticated encryption (AE) schemes are symmetric key cryptographic methods that support confidentiality, integrity and source authentication. There are many AE algorithms in existence today, in part thanks to the CAESAR competition for authenticated encryption, which is in its final stage. In our previous work we introduced a novel AE algorithm MK-3 (not part of the CAESAR competition), which is based on the duplex sponge construction and it is using novel large 16×16 AES-like S-boxes. Unlike most AE schemes, MK-3 scheme provides additional customization features for users who desire unique solutions. This makes it well suited for government and military applications. In this paper, we develop a new array-based statistical analysis approach to evaluate randomness of cryptographic primitives and show its effectiveness in the analysis of MK-3. One of the strengths of this method is that it focuses on the randomness of cryptographic primitive function rather than only on the randomness of the output.**

I. Introduction

The overarching goal of symmetric key cryptography is to enable private communication over an insecure channel in the presence of adversaries. Two fundamental requirements for achieving this goal are encryption and authentication. Encryption provides *confidentiality* while authentication provides data *integrity* and assurance of message origin [1].

Many authenticated encryption algorithms are in existence today, but they are often unsatisfactory in terms of performance, security, or ease of use. Some algorithms require two passes per block of plaintext to encrypt and authenticate. This is generally undesirable because it often means a slower algorithm. Some algorithms have been shown to be insecure or difficult to use properly. Many algorithms, such as the ones based on generic composition, require two unique and unrelated keys. This should be avoided whenever possible because key management is a difficult problem [2].

Furthermore, new authenticated encryption algorithms are needed that meet the stringent requirements of government and military applications. Such algorithms are typically not in the public domain. The goal of such restriction is to reduce or eliminate academic interest in cryptanalyzing the algorithm and publishing results about it [3]. Our previous work consisted of designing the MK-3 authenticated encryption scheme satisfying all of the above [4], [5].

In this paper, we study randomness of the mapping between the key, plaintext and cipertext as defined by the MK-3 scheme using a new array-based statistical method described in the following.

In the process of checking randomness of cryptographic primitive functions, various inputs are selected, and then randomness of the resulting outputs is checked. This task is computationally intensive, since we want to check a large number of possible inputs. Hence, it is important to use efficient techniques, which, in turn, increases chances of detecting non-random behavior. The main purpose of the array-based approach is to improve the efficiency of this process. This approach is a follow-up to some ideas presented in [6], [7]

We start with a simple scenario, where a cryptographic primitive function can be represented as a function of two inputs, the key and the plaintext, with the ciphertext being the output of that function. The two inputs will be denoted as $\mathbf{A}$ and $\mathbf{B}$, and the cryptographic function output will be written as $\mathbf{Y} = f(\mathbf{A}, \mathbf{B})$. Since $\mathbf{Y}$ consists of N bits, we will treat it as an N-dimensional vector of bits. The inputs $\mathbf{A}, \mathbf{B}$ consist of K bits each and will be treated as K-dimensional vectors. In the array-based approach, we use sequences of vector inputs $\mathbf{A}_i, i = 1, \ldots, I$ and $\mathbf{B}_j, j = 1, \ldots, J$, and produce the outputs $Y_{n,i,j}$, for all possible pairs (i, j) and $n = 1, \ldots, N$, where $Y_{n,i,j}$ represents the n-th bit of the output for $\mathbf{A}_i$ and $\mathbf{B}_j$. The values of $Y_{n,i,j}$ can be arranged into a 3-dimensional array, and the statistical testing is done along various dimensions of that array. For example, we can fix the value of j and test randomness of the bit-array consisting of $N \cdot I$ bit positions

$$[Y_{n,i,j_0}]_{n=1,\ldots,N;i=1,\ldots,I}. \tag{1}$$

This testing would be repeated J times for all possible values of j_0. Due to performing many statistical tests here, the critical values for randomness detection need to be adjusted for the multiple inference effect, which we do here using the Bonferroni approach. When D instances of hypothesis testing are performed, the probability of type I error of each individual test needs to be set to α/D in order to obtain the joint probability of one or more type I errors (in all D instances of hypothesis testing) that is not larger than α. See [8] for more details.

In a similar fashion, other dimensions can have a fixed value, and the corresponding bit-array is tested. Two dimen-

sions can also be fixed, so that the set of bits for testing is selected based on changing the third dimension. For example, we could fix i at i_0, j at j_0 and the bit positions for testing randomness would become

$$[Y_{n,i_0,j_0}]_{n=1,\ldots,N}. \quad (2)$$

One advantage of this array-based approach is that we have a potential to detect non-randomness in a more selective fashion. For example, if the non-randomness occurs only for a certain value of input $\mathbf{B}$ being tested, this would be detected. On the other hand, if the sequence of all bits $Y_{n,i,j}$ is tested, this non-randomness might not be detected, since it could potentially be masked by the remaining random data.

In our testing, we focused on the inputs that were non-random. Using a random selection of inputs is detrimental to an efficient process of detection of non-randomness. It is not surprising to see random output from random inputs, even if the function is non-random. An extreme example is that of the identity transformation, where the output would be taken as a concatenation of the two inputs. In this case, the outputs from random inputs would also be random, even though the function is highly non-random.

Algorithm 1 MK3 Permutation f

1: $N_r \leftarrow 10$ (or 16) ▷ number of rounds
2: $N_b \leftarrow 512$ ▷ number of bits in state
3: $N_w \leftarrow 32$ ▷ number of 16-bit words in state
4: **function** $f(S)$ ▷ permute state S in place
5: **for** $r \leftarrow 1, N_r$ **do**
6: **for** $i \leftarrow 0, N_w - 1$ **do** ▷ substitution step S
7: $S.word[i] \leftarrow sbox(S.word[i])$
8: **end for**
9: $S' \leftarrow S.copy()$
10: **for** $b \leftarrow 0, N_b - 1$ **do** ▷ permutation step π
11: $b' \leftarrow (31b + 15) \mod 512$
12: $S_b \leftarrow S'_{b'}$
13: **end for**
14: **for** $j \leftarrow 0, N_w/2 - 1$ **do** ▷ mix step M
15: $i \leftarrow 2j$
16: $(S.word[i], S.word[i+1])$
17: $\leftarrow mix(S.word[i], S.word[i+1])$
18: **end for**
19: $S \leftarrow S \oplus RC[r]$ ▷ add round constant
20: **end for**
21: **end function**

II. MK-3 AE Algorithm

The MK-3 algorithm is a single pass authenticated encryption algorithm designed with hardware implementation in mind. The core of MK-3 is a permutation function f comprised of 4 transformations, which are computed on a 512-bit state split into 16-bit words. This permutation function f is used as the core of a duplex sponge construction [9]. The sponge is a relatively new primitive made popular by Keccak, the winner of the SHA-3 competition [10]. The sponge is unique in that it can be configured to allow for a variety of different cryptographic uses. It has an internal state S comprised of b bits split into a rate r and capacity c, such that $b = r+c$. Basic sponge construction consist of two stages. Data is "absorbed" into the sponge by passing it through the underlying function f in r-bit length blocks. It is then "squeezed" out, generating output of arbitrary length specified at run time. The duplex sponge construction is a slight modification of the sponge which maintains state between calls while absorbing and squeezing the data during each iteration [9]. Figure 1 illustrates this approach in the context of the MK-3 authenticated encryption scheme, where AE functionality can be achieved through successive duplexing calls with the key, initialization vector (IV), additional authenticated data (AAD) and blocks of the plaintext (M_i). Each absorbed r-bit plaintext block M_i is used to produce one ciphertext block C_i, and the authentication tag T is outputed at the end.

Pseudocode of the permutation f is given in Algorithm 1 and an illustration of a round operation is presented in Figure 2. The first transformation of Algorithm 1 is non-linear. It passes each 16-bit state word through an S-box (line 7). The S-box returns the modular inverse of the input multiplied by a transformation matrix and added to an offset vector. After substitution, the data is permuted by passing it to the affine function $\pi(x) = 31x + 15 \mod 512$ (lines 11–12). The data is then mixed, providing local diffusion. Neighboring pairs of words are mixed together to produce two new words (lines 15–17) which are passed onto the next step. Finally, a round constant is added to the state (line 19).

The security level of a keyed sponge was shown by Jovanovic et. al [11] to be

$$\min(2^{(r+c)/2}, 2^c, 2^{|K|}).$$

The duplex sponge construction we use is the same as keyed sponge for the first application of permutation, and thus this security claim holds for one-block message. For general case of duplex more thorough argument is needed as in [9].

The MK-3 scheme allows for keys K of length 128 and 256 bits. In both cases the rate r is kept at 128 bits and $c = 384$, which permits for simple transitioning between the two key lengths. The operational difference in MK-3 between key sizes is the number of rounds N_r which are iterated while computing f, namely using a 128-bit key requires 10 rounds while 256-bit key requires 16 rounds. The number of rounds was determined by calculating the linear and differential complexity over a number of rounds until it exceeds the generic security of the algorithm [4]. Differential and linear cryptanalysis of MK-3 showed that 6 and 12 rounds are required to obtain a level of security above 2^{128} and 2^{256}, respectively. The additional 4 rounds for each key size are used to further increase the security margin.

1) Substitution Step: A substitution box aims to provide a non-linear operation to a cryptographic function. Generally, an S-box takes in k bits of input and produces l bits. Although k and l need not be equal, they are equal in both the AES and

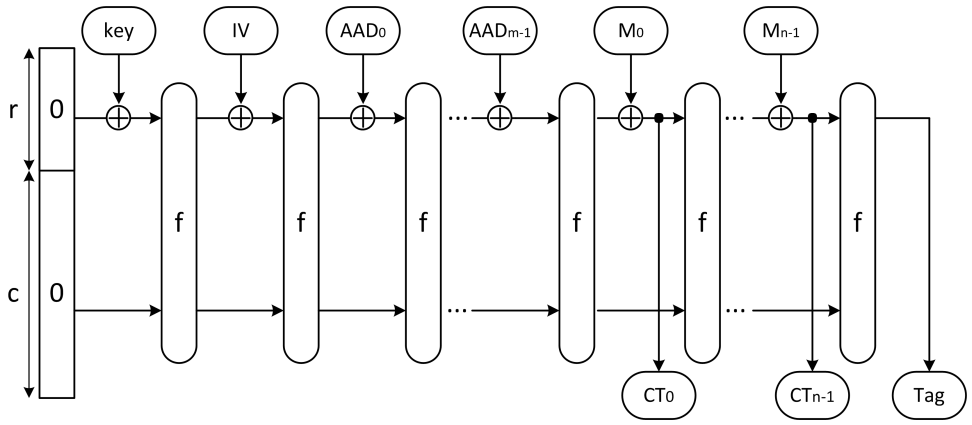

Figure 1: The duplex construction

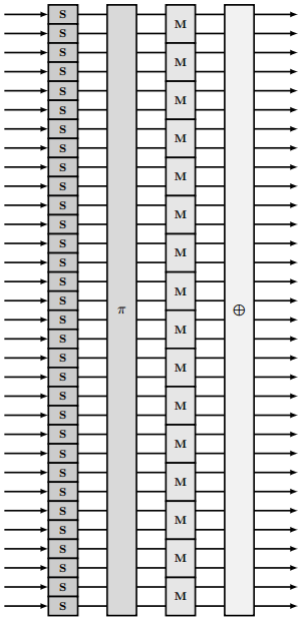

Figure 2: MK-3 round operation

MK-3, with $k = l = 8$ and $k = l = 16$, respectively. MK-3 is the first cryptographic algorithm to our knowledge to employ larger than 8-bit S-boxes [4]. The construction of 16-bit S-boxes for MK-3 was presented by Wood et al. in [12]. These S-boxes, similarly as the AES S-box, find the inverse of the input and pass it through an affine transformation. The default case for MK-3 is:

$$\begin{pmatrix} 0&0&1&0&0&0&0&1&0&0&1&1&1&1&1&0\\ 1&1&0&0&0&0&0&1&0&1&1&0&1&0&1&0\\ 1&1&0&0&1&0&1&1&0&1&0&1&0&0&1&1\\ 1&1&1&0&0&0&1&0&0&1&1&0&0&0&0&0\\ 1&1&0&0&0&1&1&0&0&1&1&1&1&0&1&1\\ 0&1&0&0&0&0&1&1&0&1&1&1&1&1&0&1\\ 0&0&1&0&1&0&1&0&1&1&0&0&1&1&0&0\\ 1&0&1&1&1&0&1&1&0&0&0&1&0&1&1&1\\ 0&1&0&0&0&0&0&0&1&0&0&1&1&1&0&1\\ 1&0&1&1&0&0&0&1&0&0&1&0&1&0&0&0\\ 1&0&1&0&0&1&1&1&0&0&1&1&0&1&0&0\\ 1&0&1&1&1&0&1&1&1&1&0&1&1&0&0&1\\ 1&0&1&0&0&1&0&1&1&0&0&1&0&0&0&1\\ 0&1&0&0&0&1&1&1&1&0&0&0&0&0&0&1\\ 1&0&0&0&1&1&0&1&0&1&1&1&1&0&0&0\\ 1&1&0&1&0&1&1&0&1&0&0&1&1&0&0&0 \end{pmatrix} \times \begin{pmatrix} x_{15}\\ x_{14}\\ x_{13}\\ x_{12}\\ x_{11}\\ x_{10}\\ x_9\\ x_8\\ x_7\\ x_6\\ x_5\\ x_4\\ x_3\\ x_2\\ x_1\\ x_0 \end{pmatrix}^{-1} \oplus \begin{pmatrix} 0\\1\\0\\0\\0\\1\\0\\1\\1\\0\\1\\1\\0\\1\\1\\1 \end{pmatrix}.$$

A common way of finding the multiplicative inverse is to use the Extended Euclid Algorithm. This is a fine solution for software, however it is very complex to implement in hardware, especially over 16-bit elements. To reduce complexity, a non-standard composite field construction can be used to directly solve for the inverse [4].

2) Permutation Step: After passing each word of the state through an S-box, the state bits are permuted by $\pi(x) = 31x + 15 \mod 512$. In hardware this is a reordering of wires according to $\pi(x)$.

3) Mixing Step: Mixing functions operate over pairs of 16-bit state words. The mixer's operation consists of the multiplication of a 2×2 matrix of elements in the Galois field $GF(2^{16})$ by the vector containing two consecutive state words (3). The Galois field $GF(2^{16})$ is represented by using irreducible polynomial $q(x) = x^{16} + x^5 + x^3 + x^2 + 1$. The two output words A', B' of the mixing step are given by

$$\begin{bmatrix} A' \\ B' \end{bmatrix} = \begin{bmatrix} 1 & x \\ x & x+1 \end{bmatrix} \times \begin{bmatrix} A \\ B \end{bmatrix}. \tag{3}$$

After the mixing step, the entire state has a 512-bit constant added to it with bitwise $\oplus$ (XOR). The constants are derived using Keccak, the SHA-3 competition winner. Each round constant RC_i is the output of Keccak calculated for the ASCII value of the round number [4].

III. MK-3 Statistical Analysis

A. Data Generation Process

For the purpose of checking randomness of MK-3, we used an approach depicted in Figure 3. The inputs $\mathbf{A}$ and $\mathbf{B}$ are as described in the introduction section, and the whole state $\mathbf{C}$ consisting of 512 bits is used as the $\mathbf{Y}$ output.

In order to generate a sequence of inputs $\mathbf{A}_i$, $i = 1, \ldots, I$, we started with a sequence of 128 zero bits as the $\mathbf{A}_1$ vector. Then we defined $\mathbf{A}_2$ vector by flipping the first bit of $\mathbf{A}_1$ to 1, defined $\mathbf{A}_3$ vector by flipping the second bit of $\mathbf{A}_1$ to 1, and so on, for all 128 bits of $\mathbf{A}_1$. This resulted in 129 vectors $\mathbf{A}_i$, for $i = 1, \ldots, 129$. Note that each $\mathbf{A}_i$ for $i > 1$ has exactly one bit equal to 1. The inputs B were defined in the same way, so $\mathbf{B}_i = \mathbf{A}_i$, for $i = 1, \ldots, 129$. The f function (see its use in Figure 3) was calculated using k rounds, for k ranging from

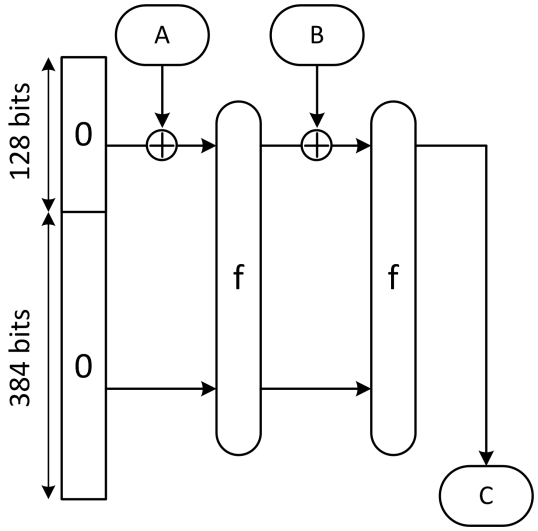

Figure 3: Test setup

1 to 10. To summarize, for each fixed number of rounds k, our data can be written as an array of bits $Y_{n,i,j} = f_n(\mathbf{A}_i, \mathbf{B}_j)$, where $n = 1, \ldots, 512; i, j = 1, \ldots, 129$ (for simplicity our notation ignores the dependence on the number of rounds, k).

B. Randomness Checking

The first step in our randomness checking was to test the frequency of 1's against the expected statistical variability. The top panel of Figure 4 shows the frequency of 1's among $129 \cdot 129$ bits of the array

$$[Y_{n_0,i,j}]_{i=1,\ldots,129,j=1,\ldots,129},$$

where n_0 is the index shown on the horizontal axis. The horizontal dashed lines mark the limits of the acceptable statistical variability, which were calculated as

$$0.5 \pm s \cdot z(1 - 0.05/2D) \tag{4}$$

for $s = \sqrt{0.25/m}$, where m is the number of bits being tested for the frequency of 1's (here $m = 129 \cdot 129$), D is the number of multiple comparisons being performed (here $D = 512$), and $z(\alpha)$ is the α level percentile from the standard normal distribution. The limits in (4) are calculated based on the normal approximation of the thresholds for rejection of the null hypothesis that the probability of observing the value 1 is equal to 0.5. The Bonferroni adjustment was explained in Section I and more details are given in [8]. Observed frequencies of 1's out side of the limits defined in (4) point to nonrandom behavior.

We can see in the top panel of Figure 4 that the majority of the frequencies are outside of the acceptable limits for the case of only one round. This points to some non-randomness when only one round of the algorithm is being used. The second panel of Figure 4 shows analogous results when two rounds are being used. Again, we can see a fair amount of non-randomness in the data. Figure 5 shows analogous results for cases when three and four rounds are being used, respectively.

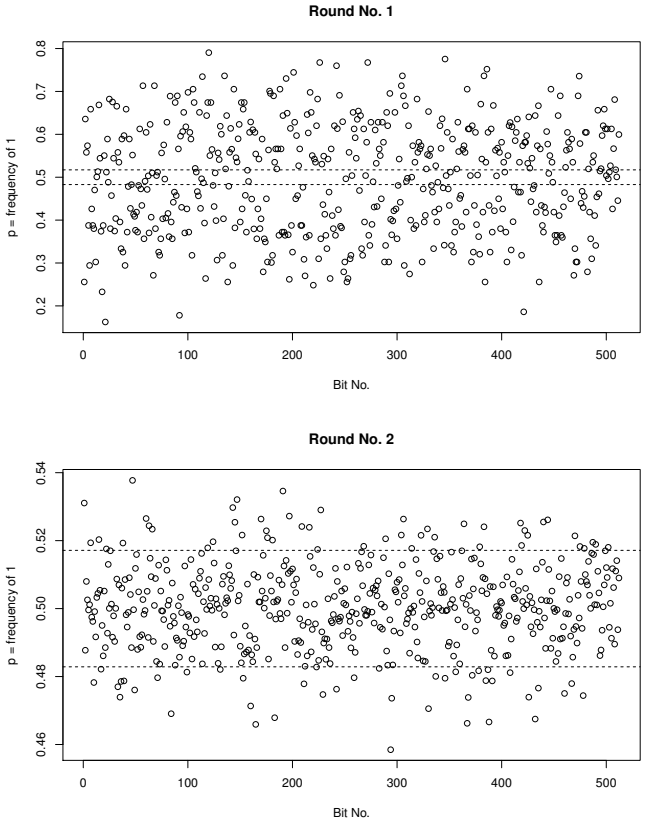

Figure 4: Frequency of 1's accross all bit positions for 1 and 2 rounds

This time, all frequencies are within the acceptable limits of variability. We don't show the results with more rounds, since they look similar to those in Figure 5, and again no non-randomness was detected in those cases.

Figure 6 shows the frequency of 1's among $512 \cdot 129$ bits of $[Y_{n,i,j_0}]_{n=1,\ldots,512,i=1,\ldots,129}$, where j_0 is the column index (representing the index of the $\mathbf{B}$ input) shown on the horizontal axis. The limits of the acceptable variability are again calculated based on (4), however here $m = 512 \cdot 129$ and $D = 129$. The top panel of Figure 6 shows some non-random patterns, when only one round is being used. On the other hand, two rounds are sufficient, in this case, in order to generate frequencies within the acceptable range. We also observe it for 3 and 4 rounds (see Figure 7) and for more rounds up to 10.

Figures analogous to 6 and 7 were also created (but not shown here) in order to show the frequency of 1's among $512 \cdot 129$ bits of $[Y_{n,i_0,j}]_{n=1,\ldots,512,j=1,\ldots,129}$, where i_0 is the row index (representing the index of the $\mathbf{A}$ input) on the horizontal axis. Those figures are not shown here, since they show results very similar to those shown in Figures 6 and 7.

We also considered patterns in several consecutive bits. For example, in random data, we should observe two-bit pattern 00 in approximately 25 percent of cases. The same is true

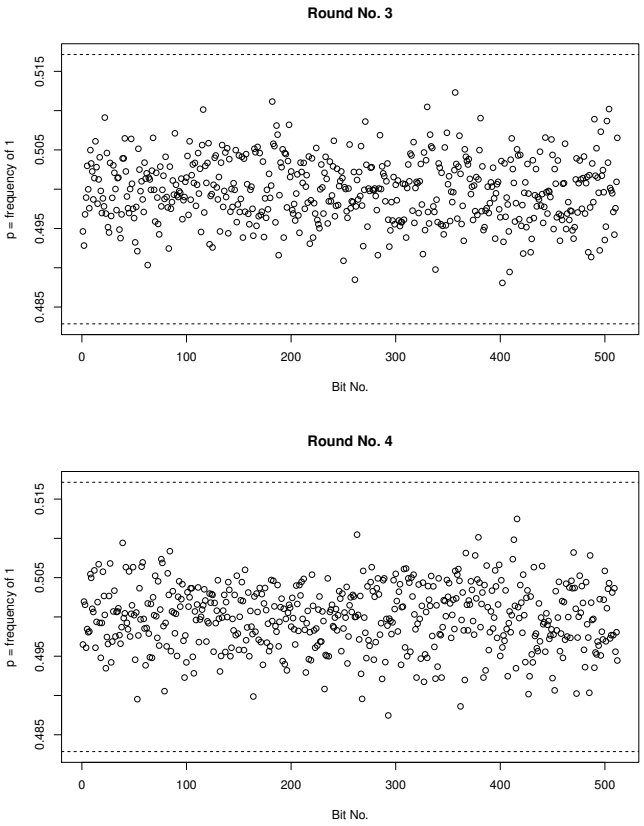

Figure 5: Frequency of 1's accross all bit positions for 3 and 4 rounds

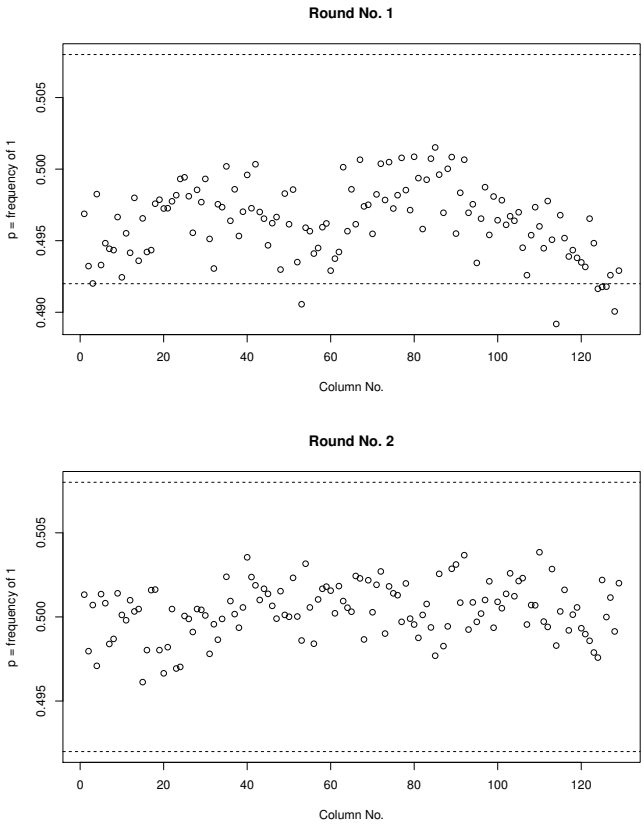

Figure 6: Frequency of 1's accross $\mathbf{B}$ inputs for 1 and 2 rounds

	number of rounds		
number of bits H	1	2	3-10
2	2006	1804	0
3	4096	2158	0
4	7815	931	0
5	1025	919	0
6	1029	907	0
7	1026	891	0
8	1034	883	0
9	1025	843	0
10	1032	828	0

Table I: Number of non-random cases across a fixed bit position of the output

	number of rounds		
number of bits H	1	2	3-10
2	11	0	0
3	1	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0

Table II: Number of non-random cases across $\mathbf{B}$ inputs

for the remaining patterns 01, 10, and 11. We explore here H consecutive bits, where $H = 2, \ldots, 10$, with 2^H possible patterns in each case. The randomness is assessed based on staying within the acceptable limits defined in a way similar to formula (4), which now takes the form

$$p_0 \pm s \cdot z(1 - 0.05/2D), \tag{5}$$

for $p_0 = 2^{-H}$ and $s = \sqrt{p_0 \cdot (1 - p_0)/m}$, where m is the number of tested patterns, and the remaining notation is the same as that for (4). The number of non-random cases is summarized in Table I for testing across a fixed bit position of the output. The total number of cases is given by $D = 512 \cdot 2^H$ and $m = \lfloor 129 \cdot 129/H \rfloor$. We observe non-random cases for up to two rounds only. Table II shows analogous counts of non-random cases across columns ($\mathbf{B}$ inputs). Here $D = 129 \cdot 2^H$ and $m = \lfloor 512 \cdot 129/H \rfloor$. We observe non-randomness only for some cases when using one round.

Table III shows analogous counts of non-random cases across rows ($\mathbf{A}$ inputs). Here the values of D and m are the same as those for the case of Table II. This time, non-randomness is observed only for up to two rounds.

IV. Conclusions and Future Work

The new array-based statistical analysis approach developed in this paper confirms the desired randomness properties of the MK-3 scheme, when at least three rounds are used. This

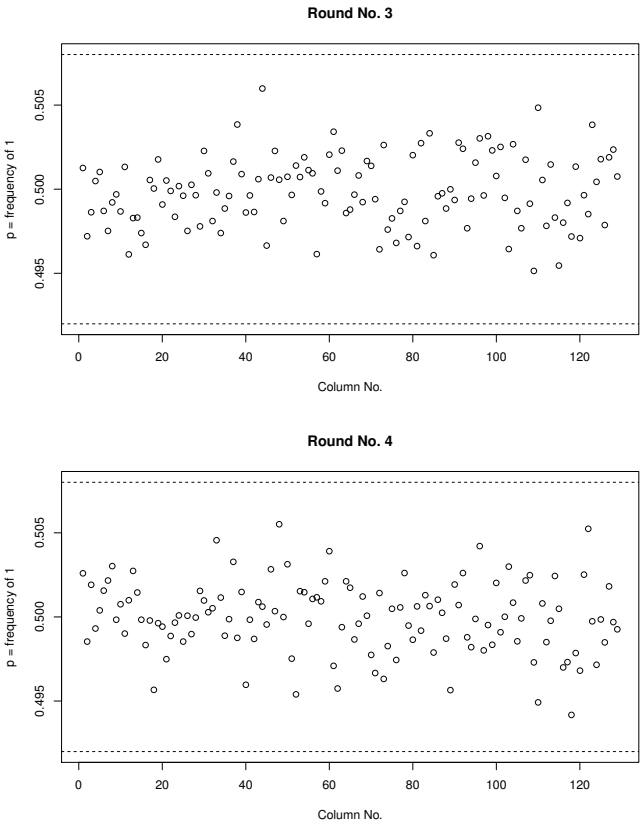

Figure 7: Frequency of 1's accross $\mathbf{B}$ inputs for 3 and 4 rounds

	number of rounds		
number of bits H	1	2	3-10
2	322	61	0
3	564	20	0
4	783	0	0
5	910	0	0
6	487	0	0
7	369	0	0
8	312	0	0
9	183	0	0
10	78	0	0

Table III: Number of non-random cases across $\mathbf{A}$ inputs

new array-based approach is very general and can be used for checking randomness of other cryptographic primitives in a variety of contexts. However, depending on the type of the cryptographic primitive and the statistical tests used, this approach may require some customization. This will be the subject of our future research.

Independently, further cryptanalysis of the MK-3 scheme should be performed on its customized versions, including statistical analysis as in this paper.

REFERENCES

[1] D. Stinson, *Cryptography: Theory and Practice, Third Edition*. Chapman & Hall/CRC, 3rd ed., 2006.

[2] N. Ferguson, B. Schneier, and T. Kohno, *Cryptography Engineering*. Indianapolis, IN: Wiley Publishing, 2010.

[3] M. T. Kurdziel and J. Fitton, "Baseline Requirements for Government and Military Encryption Algorithms," in *MILCOM 2002. Proceedings*, vol. 2, pp. 1491–1497, IEEE, 2002.

[4] M. Kelly, A. Kaminsky, M. T. Kurdziel, M. Lukowiak, and S. Radziszowski, "Customizable Sponge-Based Authenticated Encryption Using 16-bit S-boxes," in *MILCOM 2015. Proceedings*, pp. 43–48, IEEE, 2015.

[5] G. Werner, S. Farris, A. Kaminsky, M. T. Kurdziel, M. Lukowiak, and S. Radziszowski, "Implementing authenticated encryption algorithm MK-3 on FPGA," in *MILCOM 2016. Proceedings*, pp. 17–25, IEEE, 2016.

[6] A. Kaminsky, "The Coincidence Test: a Bayesian Statistical Test for Block Ciphers and MACs." http://www.cs.rit.edu/~ark/parallelcrypto/cryptostat/coincidence.pdf, September 2013.

[7] K. Hoyt, "Structuring Statistical Tests for Validating Encryption: An Array-based Approach," Master's thesis, Rochester Institute of Technology, 2016. http://scholarworks.rit.edu/theses/9073.

[8] J. Devore, *Probability and Statistics for Engineering and the Sciences*. Cengage, 2012.

[9] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Duplexing the sponge: single-pass authenticated encryption and other applications," in *Selected Areas in Cryptography*, pp. 320–337, Springer, 2012.

[10] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "The KECCAK reference," *NIST SHA-3 Submission Document*, January 2011. http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

[11] P. Jovanovic, A. Luykx, and B. Mennink, "Beyond $2^{c/2}$ Security in Sponge-Based Authenticated Encryption Modes." Cryptology ePrint, Report 373, 2014. http://eprint.iacr.org.

[12] C. Wood, S. Radziszowski, and M. Lukowiak, "Constructing large S-boxes with area minimized implementations," in *MILCOM 2015. Proceedings*, pp. 49–54, IEEE, 2015.