Design and Implementation of a Statistical Testing Framework for a Lightweight Stream Cipher

Ahmed Alamer
Department of Computer Science and Information Technology, School of Engineering and Mathematical Sciences, La Trobe University, Victoria, Australia and Department of Mathematics, Tabuk University, Saudi Arabia
a.alamer@latrobe.edu.au

Ben Soh
Department of Computer Science and Information Technology, School of Engineering and Mathematical Sciences, La Trobe University, Victoria, Australia
b.soh@latrobe.edu.au

***Abstract*—The Shrinking Generator (SG) is a popular synchronous, lightweight stream cipher that uses minimal computing power. However, its strengths and weaknesses have not been studied in detail. This paper proposes a statistical testing framework to assess attacks on the SG. The framework consists of a d-monomial test that is adapted to SG by applying the algebraic normal form (ANF) representation of Boolean functions, a test that uses the maximal degree monomial test to determine whether the ANF follows the proper mixing of bit values, and a proposed unique window size (UWS) scheme to test the randomness properties of the keystream. The proposed framework shows significant weaknesses in the SG output in terms of dependence between the controlling linear-feedback shift register (LFSR) and non-linearity of the resulting keystream. The maximal degree monomial test provides a better understanding of the optimal points of SG, demonstrating when it is at its best and worst according to the first couple of results. This paper uses UWS to illustrate the effect of the LFSR choice on possibly distinguishing attacks on the SG. The results confirm that the proposed UWS scheme is a viable measure of the cryptographic strength of a stream cipher. Due to the importance of predictability and effective tools, we used neural network models to simulate the input data for the pseudo-random binary sequences. Through the calculation of UWS, we obtained solid results for the predictions.**

Keywords-stream ciphers; randomness testing; shrinking generator; cryptanalysis

I. INTRODUCTION

Cryptography is used to transform information from plain text to cipher text and vice versa in order to prevent unauthorized access to information [1, 2]. This paper focuses on symmetric encryption, specifically synchronous stream ciphers, of which the shrinking generator (SG) is an example. The SG functions use two linear-feedback shift registers (LFSRs). $LFSR_A$ generates the output bits and $LFSR_B$ generates the controlling bits. The bit from $LFSR_A$ is output as part of the keystream whenever the bit from $LFSR_B$ is 1, otherwise the output is not selected by the cipher. Although some might argue that the SG is an old form of cipher technology, it is significant as a foundation for other variants of ciphers, such as the self-shrinking generator (SSG) [3] and its variations. Consequently, the SG is still studied, and its practical advantages continue to be appreciated. For example, researchers recently examined the SG from a cellular automaton point of view [4]. While the study was simplistic in its approach, the SG was nonetheless useful. Provided the key is unknown, the only method that can attack the SG is an exhaustive search. The SG continues to attract interest because of its reputation as a standard and a model for enlightening cryptanalysis techniques. Lessons learned from the SG can be transferred to the cryptanalysis of other cipher techniques.

Corresponding author: Ahmed Alamer

II. BACKGROUND

The SG is a lightweight stream cipher that uses minimal computing power [5]. It has been applied in various practical settings, such as radio frequency-based identification systems (e.g. in Bluetooth protocol) and in Microsoft Word, Excel, etc. This study follows the summarized presentation of the SG in accordance with [6]. The SG was first introduced in [7]. It is composed of two shift registers, namely shift register B (which is used as S in their paper), LFSRs and $LFSR_A$. $LFSR_B$ is designated as a control register that orchestrates what is produced by $LFSR_A$. To illustrate this, $\{s_i\}$ represents the bits produced by $LFSR_B$, $\{a_i\}$ represents the bits produced by $LFSR_A$ and $\{z_i\}$ is the final output produced by the sequence. Thus, the final bit sequence will follow this rule: if s_i=1, then $z_i = a_i$, else if s_i=0, then a_i is discarded (i.e. the bit is skipped). Prior attacks on the SG were studied in [6, 8–10] among others. This paper presents the lessons learned in subjecting the SG to testing methods that have not been previously applied to it. It relays the newly discovered weaknesses of the SG which may be used to strengthen its succeeding variations. Lightweight encryption is important for small devices with limited computation power and is more effective than using the more popular Advanced Encryption Standard (AES), which consumes more computation power [11]. Thus, the lightweight encryption algorithm SG and its security flaws are the focus of this study.

III. USING THE D-MONOMIAL TEST

As mentioned above, the SG produces a sequence of ciphertexts in bits dictated by two LFSR registers. For this reason, the SG falls under the category of ciphers defined by Boolean functions. These Boolean functions may be examined

using their algebraic normal form (ANF). In 2002, the authors of [12] introduced a d-monomial test that examines the randomness of the ANF of Boolean functions. Many researchers have considered this test as the most appropriate for examining ANFs. It has been used to detect the bias found in gate complexities, which can be represented by Boolean functions, and it has been extended to stream ciphers using initialisation vectors (IVs) [13]. Further, this test has been applied using polynomial description to find monomials of degree d, where $d = 1,2, \dots, n$ per polynomial, to find the weaknesses in several other stream ciphers not dealt with by the SG, such as the ciphers Trivium and Grain [14]. Because the SG is a Boolean function that can be expressed in ANF, the first hurdle that needs to be cleared by the SG is the d-monomial test, which we have specifically adapted for our purpose of examining the SG.

A. Our Approach to the d-Monomial Test

The treatment of this subject in [13] explains the work of [12] and their own adoptions of the tests. The fundamental idea behind our approach is that each bit of the SG output is viewed as a Boolean function of all different key variables (i.e. the initial loading of the LFSR). Given a Boolean function $\mathbb{F}_2^n \longrightarrow \mathbb{F}_2$, and vector x with x_i as an element, we have:

$$f(X^T) = a_0 \oplus a_1 x_1 \oplus \dots \oplus a_n x_n \oplus a_{2^n-1} x_1 x_2 x_3, \dots, x_n. \quad (1)$$

This function can be transformed as:

$$\hat{f}(x^T) = \sum_{a \in \mathbb{F}_2^n} f(a) \prod_{i=1}^{n} x_i^{a_i} \quad (2)$$

where $\hat{f}$ is a multivariate polynomial representation of f. The expression of $\hat{f}$ is used for convenience in finding the algebraic form of f, as this is generally given in a truth table. This expression of $\hat{f}$ is comprised of permutations of monomials in x_i. This is also based on the original Boolean function *f*. The subjects of this test are these monomials. The d in this test pertains to the number of non-zero bits in the Boolean string. It is the Hamming weight (i.e. the length of the longest monomial in the ANF). Practically speaking, this test involves counting the number of $\hat{f}(x^T) = 1$ with Hamming weight d. Moreover, the keystream length of the SG is $L = (2^c - 1)(2^{t-1})$, where c is the degree of $LFSR_S$, and t is the degree of $LFSR_A$. This length is not a power of 2, so the largest Boolean function we can use is obtained by taking the largest power of 2 less than L as the length of the truth table. Thus, we use:

$$N_{max} = \lfloor log_2(2^c - 1).2^{t-1} \rfloor \quad (3)$$

and choose $n \leq N_{max}$ for the Boolean representations of the relevant cryptographic bits.

In general, $f_i(x_1, \dots, x_n)$ is the Boolean function representing the i^{th} output bit $z(i)$ of the keystream $(z_1, z_2, \dots)$ of an SG. As the key $K = (x_1, \dots, x_n)$ ranges over $\mathbb{F}_2^n$, the i^{th} bit of the keystream produces the ANF of the Boolean function f_i. We can then use fast Möbius transform to compute it for each f that we choose to represent. As shown in [12], the number of weight d monomials that appear in the ANF of a random function has an approximately normal distribution, as follows:

$$\mathcal{N}\left(\frac{1}{2}\binom{n}{d}, \frac{1}{2}\sqrt{\binom{n}{d}}\right) \quad (4)$$

The author of [12] proposed χ^2 testing to examine this property based on the null hypothesis that the original Boolean function f is not random (i.e. comes from a known distribution). In the context of this study, when we describe the test as passing, this means we can reject H0. For Example 1, $LFSR_A$ corresponds to $(x^3 + x + 1)$ and $LFSR_S$ to $(x^4 + x + 1)$ as characteristic polynomials. We initialized both LFSRs with all possible values and then used (3), which yielded keystream length 56, which means we had a truth table of length 32. We then used Möbius transform to obtain the ANF. Next, we examined every monomial weight to calculate how many exist per function, and we applied the χ^2 tests. For the algorithm for this test and other monomial tests, see [14], in which these tests to IV-based ciphers using IV were applied. In our case, we used the initial LFSRs' loading, while authors in [15] performed this test on the SSG, which is a variant of the SG. However, the test had to be adapted because there are two LFSRs in the SG to generate the keystream but only one in the SSG. For the monomial distribution test, we needed to count the number of monomials across all functions. We performed this calculation using the ANF representation of the functions. For a particular monomial (say x1x3), we counted how many functions contain that monomial. Finally, we compared the monomial count with the expected monomial distribution [14]. In short, the difference between the two tests is that in the d-monomial test, we counted the number of monomials with a certain weight for each function.

B. Our Experiments for the d-Monomial Test

To perform this test, we set up all combinations of pairs of primitive $LFSR_S$ with relatively prime lengths and applied three different scenarios:

- We initialized both $LFSR_S$ with all possible random values.
- We fixed the control $LFSR_B$ with chosen values and varied $LFSR_A$ with all possible random values.
- We fixed the input $LFSR_A$ with chosen values and varied $LFSR_B$ with all possible random values.

For each LFSR combination, in each scenario, we created a truth table of the maximal length possible given the applicable keyspace and the limitation form from (3). We performed exhaustive computations for all admissible pairs of primitive $LFSR_S$, and we tested the SG outputs corresponding to $LFSR_S$ of combined length up to n=19. As Tables II and III show, certain bit functions failed the randomness test at 0.01 and 0.05 levels of significance, therefore they have an ANF that is far from that expected of a random function. Note that for $LFSR_S$ of combined length 16–19, we took half of the maximum keystream length possible and therefore half as many f_{is} because of memory constraints in the computations. The examples in Tables II and III illustrate the d-monomial test. For the results of this test for SG degrees 7–15 with full keystream output applying (3), we used the SSG results from [15] for

comparison (see Table I, last column). For degrees 16–19, we used half of the keystream outputs and applied (3) (see Table II). By applying the monomial distribution test to the first example, we found that the keystream fails in 16 out of 24 pairs for SG with degree 7, therefore the passing percentage is approximately 33.33%. More extensive results are reported later in this paper for higher degrees.

TABLE I. D-MONOMIAL TEST FOR SG DEGREES 7–15, WITH FULL KEYSTREAM OUTPUT USING χ^2 TEST, DISPLAYED WITH SSG RESULTS FROM [15] FOR COMPARISON

SG degree	Number of $LFSR_S$ pairs	Number of fi	Fails at α=0.01	Total passes	Passing percentage	SSG passing percentage
7	24	768	251	505	67310%	**98.6%**
8	72	4608	1561	2255	66124%	
9	72	9216	3121	4329	66135%	
10	216	55296	27847	27359	49640%	
11	624	319488	55203	198895	82721%	
12	648	663552	339627	206071	48817%	**99.8%**
13	2520	5160960	2831437	2083684	45137%	
14	3840	15728640	7938000	6542587	49531%	**99.9%**
15	3840	31457280	18213724	13183968	42100%	

TABLE II. D-MONOMIAL TEST FOR SG DEGREES 16–19, WITH HALF OF KEYSTREAM OUTPUT USING χ^2 TEST

SG degree	Number of $LFSR_S$ pairs	Number of fi	Fails at α=0.01	Passing percentage
16	50	409600	198895	51441%
17	50	819200	510001	37744%
18	50	1638400	932719	43071%
19	50	1114112	655568	41158%

TABLE III. EXHAUSTIVE TESTING RESULTS FOR MAXIMAL MONOMIAL TEST FOR COMBINED LFSR LENGTHS 7-9

$LFSR_A$	$LFSR_B$	Observed with fixed $LFSR_B$	Observed with fixed $LFSR_A$	Number of functions
x^3+x^2+1	x^4+x^3+1	0	12	32
x^4+x^3+1	x^3+x^2+1	0	14	32
x^4+x^3+1	x^3+x+1	0	14	32
x^4+x+1	x^3+x^2+1	0	16	32
x^4+x^3+1	x^5+x^2+1	0	66	128
x^4+x^3+1	x^5+x^3+1	0	66	128
x^5+x^2+1	x^4+x+1	0	70	128
x^5+x^3+1	x^4+x^3+1	0	68	128

C. Maximal Degree Monomial Test

The maximal degree monomial is the monomial where all variables are factors in the term. Thus, when taking a binomial function's ANF for variables x_1, x_2, x_3, the maximal degree monomial will be the term $a_7 x_1 x_2 x_3$ (a_7 is the coefficient of the term with maximum weight). This monomial may not be present in an arbitrary ANF. It is unlikely to exist if lower degree monomials do not exist, but its existence is an indication of how the ANF follows proper mixing of bit values. It is therefore worth testing. Here, we followed [14], which demonstrated that the maximal degree monomial may be detected using the Reed-Muller transform by conducting an XOR of all entries in the truth table. The existence of this monomial may be checked by XORing the first keystream bit from the initialisation. We then determined whether this exists in the output of the cipher. See [14] for a full description of this test. The results are presented in Table III and discussed below.

D. Discussion of d-Monomial and Maximal Degree Monomial Tests

For the higher degrees of 16–19, we chose 50 LFSR pairs and used the first half of the keystream output $(2^c - 1)(2^{t-2})$ bits to derive the maximal Boolean function possible, which was then subjected to the d-monomial test. This was done for computational efficiency. Table I shows that varying percentages of keystream bits z_i, from 18–58%, failed the d-monomial test at significance level α=0.01. We also carried out a limited d-monomial test on the SSG for comparison (Table I). This exhaustive testing provided further evidence that the SG is much weaker than the SSG. Although the data in Table II are not exhaustive, they show that the same kind of failure rates continued for the d-monomial test applied to 16–19 SG degrees at α=0.01 significance level. We performed a maximal degree monomial test for SG degrees 7–15 and presented a chosen sample for LFSR pairs of degrees 7–9. The maximal degree monomial occurred roughly between 37% (12 of 32, indicated by dividing the number in Column 4 with Column 5) to 54.6% (70 of 128). We observed that if $LFSR_s$ (the controlling LFSR) is fixed, the output is linear, and thus no monomial of degree ≥1 appears, which directly rules out the maximal degree monomial. In summary, it is ideal for an SG to have a high degree of monomials because this makes more difficult to guess the Boolean function behind it. However, as shown in Table I, the percentage of passing samples decreases with higher degrees, meaning that the failure to be random becomes high. Conversely, we can keep the degree of the monomials low, which the data show to have a high pass rate, however this has the disadvantage of being computationally easy to break.

IV. TESTING FOR THE UNIQUE WINDOW SIZE

The unique window size (UWS) of a keystream is the minimal length w where every observed window of length w in the keystream is unique. An m-sequence (pseudorandom binary sequence) of period 2^n-1 is generated by an LFSR of length n. Thus, $a = (a(0), \dots, a(k-1)) \in \mathbb{F}_2^n$ is an arbitrary k-tuple over $\mathbb{F}_2^n$, where $1 \le k \le n$ Then we used the function:

$$N(a) = \begin{cases} 2^{n-k} & if\ \ a \neq (0,0,\dots,0) \\ 2^{n-k} - 1 & else \end{cases} \quad (5)$$

where $N(a)$ denotes the number of times the k-tuple a appears as a 'window' $(s(i), s(i+1), \dots . s(i+k-1))$ of the m-sequence $(s(t))_{t \ge 0}$. Therefore, it is a count of repetitions of the tuple that appear in the keystream.

For an LFSR, the UWS is trivially n+1 where n is the length of the LFSR. Further, if we let the $LFSR_s$ in the SG be run for a full period, that is, for $(2^c - 1)(2^{t-1})$ output bits, this corresponded to about twice as many clock cycles, namely $(2^c - 1)(2^t - 1)$ cycles, since c and t were chosen to be relatively prime. This means that the state space is the product of the two state spaces. In the following example, we investigated the UWS distribution for the SG. For Example 2, $LFSR_A$ was $x^3 + x + 1$ and $LFSR_B$ was $x^4 + x + 1$ with initial states of 001 and 0001, respectively. The $LFSR_A$ output was

0011101 and the $LFSR_B$ output was 000111101011001. Therefore, the SG output (keystream) was 11010110101011000100101110011101001100010111101011 100111. Next we found the UWS, which in this case was 11. We examined the UWS for a certain degree (i.e. the total length of the two LFSRs) for SG. For example, for degree 8, we could see that the UWS was not well-distributed. There was a large variation in size (Table IV). Table VI shows the UWS computations for the different LFSRs combinations for SG with degree 8.

TABLE IV. UWS COUNT FOR SG KEYSTREAM OF DEGREE 8 WHERE THE AVERAGE OF UWS IS 13.25

UWS	11	12	13	14	15	16
Count	4	6	4	2	6	2

A. UWS Tests: Results, Statistical Tests and Observations

We performed exhaustive testing of the UWS distribution for the SG keystream using Algorithm 1:

Algorithm 1 UWS Algorithm
Given a periodic bit sequence B[i] with period P, the UWS algorithm calculates the UWS.
Given: B = Periodic bit sequence, P = Period of B.
Calculate: L = Minimum subsequence length such that all L-bit subsequences are unique.
Initialize L with 1.
REPEAT
for each bit index, i of B do
Test
for each bit index, j of B greater than i do
if L-bit subsequence of B starting at i = L-bit subsequence of B starting at j then
INCREMENT L
CONTINUE the next REPEAT loop
end if
end for
end for
UNTIL all L-bit subsequence of B are unique.
Return L

Authors in [15] introduced general observations, whereas we conducted deeper analysis with more tests and investigations to produce a greater understanding of the SG's weaknesses based on tests and predictions applied on the UWS. Figure 1 shows the UWS, also called the minimal window size, for degree 20 of the SG. We ran the computations for 70,416 $LFSR_S$ pairs. As shown in Figure 1, the UWS is concentrated between 37 and 41. This means that it is highly likely to be concentrated below approximately $2n$, where n is the combined degree of the SG, indicating that a distinguishing attack on SG may be feasible. As shown in [16], UWS is a very good indicator of the randomness of a keystream. This relationship is discussed below. To explain the maximum order complexity of a given sequence S, it is necessary to find the possible shortest FSR that can produce S. This concept was introduced in [17] in 1990. Thus, for S, the maximum order complexity is equal to $1+l$, where l represents the longest $s \subset S$ length that can be found twice within S, and that is our UWS. The authors of [17] found an approximation of maximum order complexity distribution of a chosen pseudorandom binary sequence. More importantly for our purposes, the UWS dependence on the choice of polynomials is an indicator that there might be a possible distinguishing attack based on this variability. Of course, the LFSR polynomials are public knowledge, by Kerchoff's principle, and this can be used as a guide in choosing good polynomials from the vast number of choices $\varphi\frac{(2^n-1)}{n}$ for degree n. If the SG user broke Kerchoff's principles and kept the polynomials secret, as is sometimes ill-advisedly suggested, the investigation of the UWS would enable an attacker to mount an attack more efficient than the brute force attack on the polynomials. Recall that when the polynomials are kept secret, their degree sequences can be considered a part of the key, squaring the search space for brute force attacks. The UWS enables one to search a much more limited set of polynomials. See Table V for UWS distribution in the SG keystreams of degree 15.

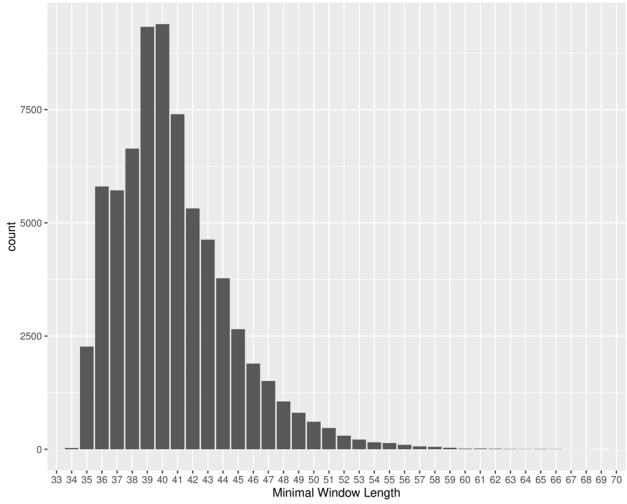

Fig. 1. UWS distribution for degree 20

B. Statistical Distribution and Prediction Modeling

Using Easyfit and R software, we confirmed that the UWS of degree 20 (UWS20) was skewed (see Figure 1). In fact, it approximately follows a lognormal distribution, as presented in Figure 2. The distribution function of lognormal distribution is given by:

$$\mathcal{N}(\ln x;\mu,\sigma) = \frac{\exp\left[-\frac{(\ln x-\mu)^2}{2\sigma^2}\right]}{\sigma\sqrt{2\pi}} \quad (6)$$

Given that the SG keystream's UWS exhibits an approximate lognormal distribution, we can follow standard confidence interval information, such as the 68-95-99.7 rule, in determining the range of probable UWS values in the keystream by taking the log of UWS. Using the Bartel rank test, Cox Stuart test, rank test and runs test, we found that the sequence of data (UWS20) is a non-random sequence (p<0.001). This indicates that this sequence of data could be predicted using statistical models. We then used several methods to learn about the pattern and predict the sequence,

namely: 1. Generating a theoretical lognormal sequence from the mean and standard deviation of observed data of the same length and comparing the accuracy of prediction, 2. Using a linear regression method to predict and check the accuracy of the predicted sequence, and 3. Using a non-linear (non-parametric regression) method to predict and check the accuracy of the predicted sequence (including validation and calibration).

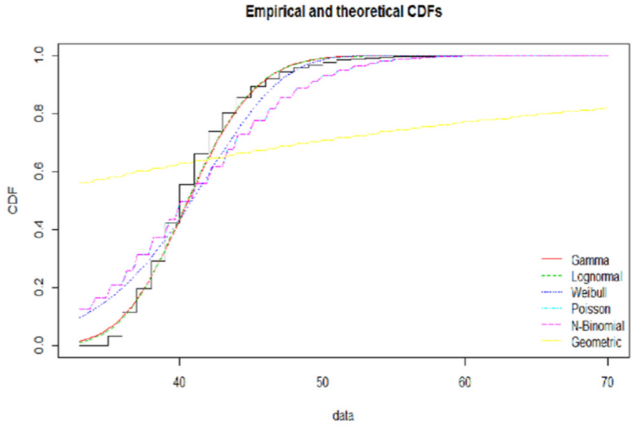

Fig. 2. Comparison of CDFs of observed and theoretical distributions, UWS20

TABLE V. UWS COUNT FOR SG KEYSTREAM OF DEGREE 15

UWS	Count	UWS	Count
24	4	34	48
25	52	35	32
26	112	36	12
27	166	37	20
28	232	38	4
29	168	39	6
30	146	40	4
31	116	42	10
32	92	45	2
33	54		

1) Method 1

Here, we used R software to generate an equal number of observations (70,416) from lognormal distribution using the mean and standard deviation of observed UWS20. We then differentiated between observed UWS20 and simulated UWS20 and compared the accuracy (the difference between the observed and simulated predicted UWS20 being 0). The accuracy was 7.8%, which was expected, as UWS20 was found to be non-random. Hence, a random sequence may not entirely predict the sequence.

2) Method 2

We used the linear regression model, where the outcome was UWS20 and the explanatory variables were input degree, input weight, control degree, control weight, input polynomial, and control polynomial. Each input and control polynomial variables had polynomials of up to 17 degrees and hence produced 17 separate binary variables based on the possible terms in a primitive polynomial of degree 17 (for an SG with degree 20, the highest LFSR degree is 17 and the lowest is 3 for a primitive polynomials combination).

Step 1 of Method 2 was to extract 34 variables from the input and control polynomials and add them to four other independent variables to form a pool of independent variables.

In Step 2, we ran a simple linear regression model on each of suitable candidates for the multi-variable linear regression model. The input degree was found to be the strongest predictor because it had the highest R-square value and accuracy rate. Interestingly, it was highly correlated with the control degree and, as such, the two cannot be put together in the same model. This means that control degree and input degree as variables are correlated, or input degree $=f$(control degree), so just one of them is necessary.

TABLE VI. UWS COMPUTATIONS FOR DEGREE 8 WITH $LFRP_S$ PAIRS

Input $LFSR_A$	Control $LFSR_B$	UWS
x^3+x+1	x^5+x^2+1	13
x^3+x+1	x^5+x^3+1	15
x^3+x+1	$x^5+x^3+x^2+x+1$	11
x^3+x+1	$x^5+x^4+x^2+x+1$	14
x^3+x+1	$x^5+x^4+x^3+x+1$	12
x^3+x+1	$x^5+x^4+x^3+x^2+1$	15
x^3+x^2+1	x^5+x^2+1	15
x^3+x^2+1	x^5+x^3+1	13
x^3+x^2+1	$x^5+x^3+x^2+x+1$	15
x^3+x^2+1	$x^5+x^4+x^2+x+1$	12
x^3+x^2+1	$x^5+x^4+x^3+x+1$	14
x^3+x^2+1	$x^5+x^4+x^3+x^2+1$	11
x^5+x^2+1	x^3+x+1	16
x^5+x^2+1	x^3+x^2+1	12
x^5+x^3+1	x^3+x+1	12
x^5+x^3+1	x^3+x^2+1	16
$x^5+x^3+x^2+x+1$	x^3+x+1	15
$x^5+x^3+x^2+x+1$	x^3+x^2+1	13
$x^5+x^4+x^2+x+1$	x^3+x+1	12
$x^5+x^4+x^2+x+1$	x^3+x^2+1	11
$x^5+x^4+x^3+x+1$	x^3+x+1	11
$x^5+x^4+x^3+x+1$	x^3+x^2+1	12
$x^5+x^4+x^3+x^2+1$	x^3+x+1	13
$x^5+x^4+x^3+x^2+1$	x^3+x^2+1	15

In Step 3, we calculated and analyzed the multicollinearity among the univariable significant variables to determine which variables had to be discarded from the multi-variable regression model. Multicollinearity is defined here as high correlation among explanatory variables, which can be assessed from a matrix of correlation coefficients of the explanatory variables along with p values. We discarded the control degree, input polynomial of degree 14 and control polynomial of degree 17 from the model due to multicollinearity with other covariates. We also checked first order interaction.

In Step 4, a backward stepwise multi-variable regression model was run with the suitable variables for the prediction of UWS20. We could then retain 24 variables (in the model), which were used in our final predictive model.

Predicted:log(UWS20)=3.9541011-InputDegree*0.0121524+ InputWeight*0.0005625-ControlWeight*0.1042675 - input1 * 0.0023569 + input11 * 0.0051394 -input12 * 0.0057957 -

input13 * 0.0122590 - input17 *0.0106996 + control1 * 0.1060109 + control2 * 0.1035639 +control3 * 0.1034507 + control4 * 0.1062923 + control5 *0.1051447 + control6 * 0.1061192 + control7 * 0.1001495 +control8 * 0.1053034 + control9 * 0.1104407 + control10 *0.1036258 + control11 * 0.1058865 + control12 *0.0999843 + control13 * 0.0953821 + control14 * 0.1055764 + control15 * 0.1042613 + control16 * 0.1049406

Input 17 refers to the term of power 17 in the input LFSR and similarly in control 17 (the number indicates the power of term in the given polynomial). Using the above model, UWS20 can be predicted by inserting the values of the variables in the model. A numerical value must be inserted for numerical variables, with 0 = absence and 1 = presence. This model can be used as a selection tool for the LFSRs pairs by choosing two primitive polynomials and inserting their parameters in the model to see if the UWS is high enough for the chosen pairs.

In Step 5, bootstrap validated bias was used to correct the adjusted R-square and assess the discrimination. Calibration graphs were produced to assess the calibration of the model. These were done using the rms package in R. Afterwards, we measured the accuracy with respect to the difference of observed and predicted UWS20. Sensitivity analysis was performed to ensure that the effect of a relatively smaller sample size on prediction capabilities was not detrimental. Hence, we randomly selected 50%, 25%, and 10% of the data and performed the above calculations again to check the accuracy and performance of the prediction model.

In Method 2, the accuracy was approximately 12.1%. However, the accuracy increased from 12.1% to 37.8%, 61.8%, 78.6%, and 87.1% if we allowed the prediction to be within 1 (e.g. observed UWS20 was 40, prediction is 39), 2, 3, and 4 degrees of tolerance with observed UWS20 respectively. Although the bootstrap validated adjusted R-square (0.262) indicated a weak to moderate prediction performance, the calibration graph showed good calibration between the observed and predicted UWS20 (see Figure 3).

3) Method 3

Using R software (e.g. np package), we implemented non-parametric regression analysis with the same modeling structure as the multi-variable linear regression model but without any assumptions about the inherent distribution of the observed UWS20. Unfortunately, Method 3 failed to produce better accuracy than the parametric multi-variable linear regression model. Thus, we used the multi-variable linear regression model as the final model because it was more powerful than nonparametric regression.

C. Implications of the UWS

When coupled with statistical techniques, the UWS can be used as a method of sampling to help better comprehend the behavior of a cipher under testing. As discussed, we used regression tools as a support for detecting cipher properties. Thus, sophisticated pattern recognition methods may be used for improved prediction. Lastly, we considered the given sequence *s* and what the linear complexity (LC) of an LFSR produces upon it. Thus, we analyzed *s* against UWS. Both LC and UWS must be higher for good security. Thus, like LC, UWS can be used as a measure of security strength.

V. PROPOSED NEURAL NETWORK PREDICTION MODEL

Neural Networks (NNs) have many everyday applications, but they are principally focused on simulating the ability of neurons in the human brain to process information and data. The human brain consists of nerve cells and neurotransmitters to handle orders and inputs, and a NN follows this structure, processing information and learning from it to predict the outcomes based on inputs (data). An NN functions by employing a flexible algorithm that learns from the available data and produces expectations in response to them [18]. NNs have many applications, including climate forecasting [19], stock market prediction [20], music production [21], or even cancer prediction [22]. They may also be applied in the fields of confidentiality and security of information, such as in the selection of keys used for encryption or in pattern recognition [23]. It is important to note that deep NN modeling is still a valid method and is evolving to be used in new areas, as shown in [24] which is an interesting study demonstrating the effectiveness of an NN-based model for investigating a complex system.

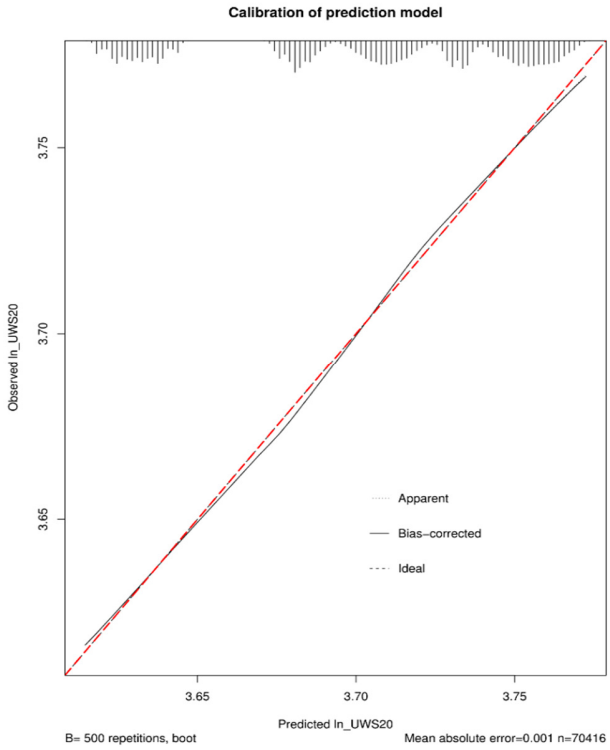

Fig. 3. Calibration graph for predicting UWS of degree 20 SG

A. Neural Networks and Security

Using the NN model provided by authors in [30] in 1976, it was possible to emulate the method of exchanging secret keys as a means of communicating through an insecure network. Especially in cases where a large number of keys are required, the application of NN models allows the study of the most secure ways to handle the exchange of keys using an unsafe

communication channel [25]. NNs have also been implemented to produce a secret key on a public network [26]. It is important to select a design and the appropriate weights for the NN to test and choose the best encryption system and to ensure that the messages are encrypted correctly [27]. To follow up the applied studies of NN models in terms of design and testing of encryption systems as well as measuring their effectiveness and potential problems, see [28]. Because there are limited studies on the use of NN models to test the randomization of the binary sequence required for use in an effective cryptosystem, it is important to study the NN as a measurement tool. Therefore, an extra step must be added in the implementation of the NN to enhance the desired cryptosystem.

B. Neural Network Model Implementations

Given that the pseudorandom binary sequence plays an important role in encryption, the selection of the best binary sequence should be as random as possible to produce a powerful cipher (algorithm) that will generate an attack-resistant sequence. Therefore, in this study we employed NN models to measure the prediction based on UWS to ascertain whether they are able to predict UWS. The study succeeded, and the results were very promising, enabling us to measure the strength of the pseudorandom binary sequence, which is predicated on the strength of the algorithm (cipher) produced by it. These results may be applied to other algorithms, thus contributing to the analysis and selection of the best encryption systems. We calculated UWS using EC2 for SG and SSG to obtain data for a large number of these algorithms at different degrees (SG and SSG). We sought to verify the validity of the application of NN at the different SG and SSG degrees along with the accuracy of our expectations. To confirm the effectiveness of the models, a high prediction rate exceeding 97% for some degrees and of no less than 90% for the others with negligible mean square errors (MSEs) is expected. Tables VII–IX detail the anticipated results. From this, we conclude that the NN is a more effective tool for prediction than the other statistical methods previously introduced in this study. The MSE is a good indicator of the validity of the method used [29] which protects the messages and shows the accuracy of the results by measuring MSE.

C. NN Models and Results for SG

Table VII shows that UWS20 comprises three layers with ten, five and two nodes each, exhibiting a learning rate of 0.001 and an MSE of 0.0088, which is very low and indicates the effectiveness of the model when applied to UWS prediction. There were 6,791 overall training parameters. 56,333 were used as exercise data, representing 80% of the total data. The remaining 20% were used for testing and validating. The effectiveness of the forecast pertains to 94.30% of the total data. By way of another example relating to UWS24, we found the effectiveness of the prediction to be 95.42% and the learning rate to be 0.0001 with an MSE of 0.0019 and the same number of training parameters. The training dataset was 192,750.

D. Comparison with Self-Shrinking Generator

We found that with a range from UWS4 to UWS20 in a single model, the number of layers was four with 100, 50, 20, and 10 nodes on each layer, exhibiting a learning rate of 0.0001 with an MSE of 0.0012. The training set consisted of 58,232 samples, the validation set of 14,558, total training parameters were 6,591 and the prediction rate was 96.05% (Table VIII). For UWS24, we located the same number of layers, nodes and training parameters and the same learning rate with an MSE of 0.0098, with training set equal to 22,1184, with validation set equal to 55,296, and with a prediction rate of 97.01% confirming the efficiency of the models for SG and SSG (Tables VII–IX).

TABLE VII. SG MODEL RESULTS FOR THE NEW NN MODELS INCLUDING RESULTS FOR DEGREES 20, 21, 23, AND 24

	UWS20 model	**UWS21 model**	**UWS23 model**	**UWS24 model**
Layers	3	4	4	4
Number of nodes	10, 5, 2	50, 20, 10, 5	50, 20, 10, 5	50, 20, 10, 5
Learning rate	0.0001	0.0010	0.0001	0.0001
MSE	0.0088	0.0138	0.0033	0.0019
Training set	56333	196502	713395	770997
Validation sample	14084	49126	178349	192750
Total parameters	6791	6791	6791	6791
Prediction percentage	96.01	94.07	95.39	95.42

TABLE VIII. SSG MODEL RESULTS FOR THE NEW NN MODELS INCLUDING RESULTS FOR DEGREES 4–20, 21 AND 22

	UWS Degrees 4-20	**UWS21 model**	**UWS22 model**
Layers	4	4	4
Number of nodes	100, 50, 20,10	100, 50, 20,10	100, 50, 20,10
Learning rate	0.0001	0.0001	0.0001
MSE	0.0012	0.0014	0.0160
Training set	58232	67737	96025
Validation sample	14558	16935	24007
Total parameters	6591	6591	6591
Prediction percentage	96.05	96.66	89.61

E. The Difference between SG and SSG

SSG is another type of cipher derived from SG. However, in SSG, the process of selection and input is conducted using a single primitive polynomial working as an LFSR. In general, this has been found to be more effective in the field of confidentiality. Using the same principles as SG, the SSG keystream was used to calculate UWS. When applying the NN model, the dependent variable was UWS, as it was in SG, and the independent variables were the SSG polynomial degree and the polynomial weight. The results of this prediction can be seen in Tables VIII and IX.

F. NN Model Design

We used a sequential model to define the model layer by layer, and we used a rectified linear unit (ReLu) for an activation function. Our model had four hidden layers, with 100, 50, 10, and 5 nodes and the output layer had 1 node. The loss function was defined as the MSE, and the learning rate was 0.0001. The optimization technique was regulated by the 'Adagrad' optimizer, and the data were fitted to train the model. Most of the data refer to SG with UWS20 under slight modifications.

1) Cost Function

To ascertain whether the model was learning as required, we needed to calculate the MSE, which needed to be small to demonstrate that the prediction was close to the actual output data. The smaller the MSE, the better the model and the more accurate the prediction. The mathematical equation for the MSE is:

$$\frac{1}{n}\sum_1^n\left(X_i - \hat{X_i}\right) \qquad (7)$$

where n is the number of data inputs (simulations), X_i is the observed data and $\hat{X_i}$ is the predicted data.

TABLE IX. SSG MODEL RESULTS FOR THE NEW NN MODELS INCLUDING RESULTS FOR DEGREES 23, 24, AND 25

	UWS23 model	**UWS24 model**	**UWS25 model**
Layers	4	4	4
Number of nodes	100, 50, 20, 10	100, 50, 20, 10	100, 50, 20, 10
Learning rate	0.0001	0.0001	0.0001
MSE	0.0046	0.0098	0.0052
Training set	285568	221184	1036800
Validation sample	71392	55296	259200
Total parameters	6591	6591	6591
Prediction percentage	90.14	97.01	97.14

2) The Rectified Linear Unit Activation Function

This function is nonlinear and has become popular for use in deep learning. Since the data we entered were positive and nonlinear, we achieved the best results using the ReLu activation function. Its advantage is that it is a fast learner and thus reduces the cost of use. We used ReLu in the inner layers of our models with the following formula:

$$(x) = \max(x, 0) \qquad (8)$$

Here, the learning rate must be adjusted to reduce the MSE as much as possible. We found a learning rate of 0.0001 to be the optimal setting (see Tables VII–IX). The importance of NN for our study lies in its ability to accurately predict UWS, which makes it an optimal instrument for measuring randomness in a given binary sequence, which is important in applying effective and safe encryption. This study and its application of NN models paves the way for future applications to measure the validity of the cryptosystems used, evaluate their effectiveness and further develop them. We found that by comparing NN models to study the behavior of the randomized binary sequence, more effective and more efficient predictions could be made than those of the model in the previous section, which used multivariable linear regression (a non-NN-based model). Further, the extent and efficiency of the predictions remained high, even though SSG appeared to be more random than SG. This can be seen in the results of the prediction in Tables VII–IX, which confirm that NN-based models are effective even in cases of random and discrete data, as in our case.

VI. OUR FINDINGS

This paper analyzed new weaknesses in the SG, a popular synchronous stream cipher. The first test was the d-monomial test adapted to SG (applied to the ANF representation of Boolean functions), which showed significant weaknesses in the SG output. We also investigated the dependence between the controlling LFSR and the non-linearity of the resulting keystream. After this, we performed a side test, called the maximal degree monomial test, which is related to the d-monomial test. In the result analysis, we reported a better understanding of the optimal points of the SG, showing when it was at its best and worst. Finally, we analyzed the cipher technique using the UWS and demonstrated dependencies between the UWS and the two LFSR polynomials. The results were assisted by statistical techniques provided by the UWS. Using the UWS, we showed the effect of the LFSR choice on possible distinguishing attacks on the SG. The results confirmed that the UWS is a viable measure of cryptographic strength. Lastly, the statistical model for the UWS model can be used to test the SG pairs, which can help to measure their strength based on their UWS so the user can choose the pairs with a higher UWS. The d-monomial test showed that the controlling LFSR causes more non-linearity if it has a higher degree than the input LFSR. Conversely, using the UWS test and modeling, we found that the input LFSR helps the prediction, which confirms that controlling LFSR with higher degree than input LFSR makes better choices. Additionally, the most effective elements (in decreasing order) in the model to help the prediction are presented in Table X.

TABLE X. MOST EFFECTIVE ELEMENTS IN THE PREDICTION MODEL FOR UWS20

Variable	**R square**
Input polynomial term of order 17	0.18
Input degree	0.127
Control degree	0.127
Control weight	0.10
Input weight	0.09
Input polynomial term of order 12	0.09
Input polynomial term of order 13	0.08
Input polynomial term of order 14	0.08
Input polynomial term of order 15	0.08
Input polynomial term of order 16	0.08

As another measure of validity analysis, we modeled UWS17, UWS18 and UWS19 and found similar prediction performance (adjusted R-square) and accuracy. By using 50%, we modeled the data with similar accuracy with an R-square of approximately 26.6%. This means that we can model with less data, which reduces computational complexity. To ensure that every state was unique, we applied the UWS to find the minimal length of the SG keystream. Doing so, we found that identifying the UWS can lead to discovering the LFSR pairs' weakness, which, in forming the keystream, can in turn lead us to understand which LFSR pairs should be avoided. Our research also showed that by using statistical tools along with UWS, we can experimentally predict the SG's behavior. This again illustrates the possible weaknesses in the SG keystream against attacks that may use this measure. Finally, we found the possibility of using NN models to predict UWS with better effectiveness and promising results compared with the non NN-based models. NN models are an effective measuring instrument for non-LC tests and should be adopted in the field of cryptanalysis more often.

VII. CONCLUSIONS

The goal of this research was conducting exhaustive experimental testing of some cryptographic security measures applied to the SG. By running complex computations for different SG degrees, it was found that the SG keystream output failed the *d*-monomial test. Thus, it is possible to establish distinguishing attacks on this kind of stream cipher. This test can be applied to other ciphers in different ways, depending on whether it is an IV-based or IV-less cipher. Additionally, the controlling LFSR had a greater effect than the input LFSR on the ANF non-linearity. This was confirmed by applying the *d*-monomial test. UWS test results can identify the LFSR pairs' weakness. A weakness can result in vulnerability to attacks, thus the cipher user must pay attention to the LFSR pairs selection.

We found that a model of the NN predictor offered highly precise results that contribute to its efficacy as a tool for measuring the extent of pseudorandom binary sequences and, thereby, the strength of the produced encryption systems. The results show that care must be taken when using the SG as a cryptographic technique. First, with a clearer understanding of the SG, analysts can design new attack techniques to mount on the SG. Second, the UWS can be used to analyze any keystream generated by any cipher. We intend to examine the UWS on selected ciphers in the future. We also plan to dig deeper into the mathematical relationship of the UWS and LC for specific ciphers. Finally, we found that applying the models by implementing the NN gave us superior results compared with a multi-nonlinear regression model (non-NN-based model).

ACKNOWLEDGEMENT

The authors would like to thank Molla Huq for the comments on the Statistical Modeling, as well the Victorian Partnership for Advanced Computing for the use of their supercomputing facilities. In addition, the authors would like to show their appreciation to Serdar Bostaz for his valuable comments and feedback.

REFERENCES

[1] C. Paar, J. Pelzl, Understanding Cryptography: a textbook for students and practitioners, Springer Science & Business Media, 2009

[2] A. J. Menezes, P. C. van Oorschot, S. A. Vanstone, Handbook of applied cryptography, CRC Press, 1996

[3] W. Meier, O. Staffelbach, "The self-shrinking generator", in: Communications and Cryptography, Springer, 1994

[4] S. D. Cardell, A. Fuster-Sabater, "Cryptanalysing the shrinking generator", International Conference on Computational Science, Reykjavík, Iceland, June 1-3, 2015

[5] D. Maimut, K. Ouafi, "Lightweight cryptography for RFID tags", IEEE Security & Privacy, Vol. 10, No. 2, pp. 76-79, 2012

[6] P. Caballero-Gil, A. Fuster-Sabater, M. E. Pazo-Robles, "New attack strategy for the shrinking generator", Journal of Research and Practice in Information Technology, Vol. 41, No. 2, 2009

[7] D. Coppersmith, H. Krawczyk, Y. Mansour, "The shrinking generator", 13th Annual International Cryptology Conference on Advances in Cryptology, Berlin, Germany, August 22-26, 1993

[8] J. D. Golic, "Towards fast correlation attacks on irregularly clocked shift registers", International Conference on the Theory and Applications of Cryptographic Techniques, Saint-Malo, France, May 21-25, 1995

[9] L. Simpson, J. D. Golic, E. Dawson, "A probabilistic correlation attack on the shrinking generator", Information Security and Privacy, 3rd Australasian Conference, Brisbane, Queensland, Australia, June 21, 1998

[10] B. Zhang, H. Wu, D. Feng, F. Bao, "A fast correlation attack on the shrinking generator", Topics in Cryptology-CT-RSA 2005, The Cryptographers' Track at the RSA Conference, San Francisco, CA, USA, February 14-18, 2005

[11] A. H. Al-Omari, "Lightweight dynamic crypto algorithm for next internet generation", Engineering, Technology & Applied Science Research, Vol. 9, No. 3, pp. 4203-4208, 2019

[12] E. Filiol, "A new statistical testing for symmetric ciphers and hash functions", 4th International Conference on Information and Communications Security, London, UK, December 9-12, 2002

[13] M. J. O. Saarinen, Chosen-IV statistical attacks on eSTREAM stream ciphers, eSTREAM, ECRYPT Stream Cipher Project, Report 2006/013, 2006

[14] H. Englund, T. Johansson, M. S. Turan, "A framework for chosen IV statistical analysis of stream ciphers", 8th International Conference on Progress in Cryptology, Berlin, Germany, December 9-13, 2007

[15] S. Boztas, A. Alamer, "Statistical dependencies in the self-shrinking generator", 7th International Workshop on Signal Design and its Applications in Communications, Piscataway, USA, September 14-18, 2015

[16] D. Erdmann, S. Murphy, "An approximate distribution for the maximum order complexity", Designs, Codes and Cryptography, Vol. 10, No. 3, pp. 325-339, 1997

[17] C. J. A. Jansen, D. E. Boekee, "Modes of blockcipher algorithms and their protection against active eavesdropping", Workshop on the Theory and Application of Cryptographic Techniques EUROCRYPT 1987, Amsterdam, The Netherlands, April 13-15, 1987

[18] K. Gurney, An Introduction to Neural Networks, CRC Press, 2014

[19] S. S. Baboo, I. K. Shereef, "An efficient weather forecasting system using artificial neural network", International Journal of Environmental Science and Development, Vol. 1, No. 4, pp. 321-326, 2010

[20] E. Guresen, G. Kayakutlu, T. U. Daim, "Using artificial neural network models in stock market index prediction", Expert Systems with Applications, Vol. 38, No. 8, pp. 10389-10397, 2011

[21] B. Gold, N. Morgan, D. Ellis, Speech and audio signal processing: Processing and perception of speech and music, John Wiley & Sons, 2011

[22] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks", Nature, Vol. 542, No. 7639, pp. 115-118, 2017

[23] A. Klimov, A. Mityagin, A. Shamir, "Analysis of neural cryptography", International Conference on the Theory and Application of Cryptology and Information Security, Queenstown, New Zealand, December 1-5, 2002

[24] L. B. Salah, F. Fourati, "Systems modeling using deep Elman neural network", Engineering, Technology & Applied Science Research, Vol. 9, No. 2, pp. 3881-3886, 2019

[25] W. Kinzel, I. Kanter, "Interacting neural networks and cryptography", in: Advances in solid state physics, Springer, 2002

[26] T. Godhavari, N. Alamelu, R. Soundararajan, "Cryptography using neural network", 2005 Annual IEEE India Conference - Indicon, Chennai, India, December 11-13, 2005

[27] E. Volna, M. Kotyrba, V. Kocian, M. Janosek, "Cryptography based on neural network", ECMS 2012, Koblenz, Germany, May 29-June 1, 2012

[28] A. El-Zoghabi, A. H. Yassin, H. H. Hussien, "Survey report on cryptography based on neural network", International Journal of Emerging Technology and Advanced Engineering, Vol. 3, No. 12, pp. 456-462, 2013

[29] R. J. Rasras, Z. A. AlQadi, M. R. A. Sara, "A methodology based on steganography and cryptography to protect highly secure messages", Engineering, Technology & Applied Science Research, Vol. 9, No. 1, pp. 3681-3684, 2019

[30] W. Diffie, M. E. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, Vol. 22, No. 6, pp. 644-654, 1976

AUTHOR PROFILES

Ahmed Alamer conducted his postgraduate studies at Adelaide University (BSc) and QUT University (MSc Degree), majoring in Mathematical Sciences with a research focus in cryptography. He also graduated with a BSc Degree in mathematical sciences from King Khalid University. He currently focuses on the applications of mathematics in the field of cryptanalysis and security. He is currently a PhD candidate at the Department of Computer Science and Computer Engineering of La Trobe University, where he has published and under-review papers in cryptology, neural networks applications in security, statistical analysis and randomness testing, lightweight cryptosystem design, and cryptanalysis. He has also held a lecturer position at the University of Tabuk since 2012 and has organized several workshops in mathematics applications there.

Ben Soh (S'89–M'92–SM'03) received his PhD degree in computer science and engineering from La Trobe University, Melbourne, Australia, in 1995. He is currently an Associate Professor with the Department of Computer Science and Computer Engineering, La Trobe University. He has taught numerous successful PhD graduates. He has authored more than 180 peer-reviewed research papers and made significant contributions in various research areas, including fault-tolerant and secure computing and web services.