International Journal of Cryptology Research 11(2): 1 - 16 (2021)

THE AA_β PSEUDO-RANDOM BIT GENERATOR AND ITS STATISTICAL TESTS

Aniza Abd.Ghani[*1] and **Muhammad Rezal Kamel Ariffin**[1,2]

[1]*Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia*
[2]*Institute for Mathematical Research, Universiti Putra Malaysia*

E-mail: aniza@upm.edu.my
[*]*Corresponding author*

ABSTRACT

There are many ways to generate random bit sequences. Each generator's quality should be evaluated in the context of cryptography using various statistical tests. Passing statistical tests is a must since any statistical flaw can be exploited by an attacker to learn more about the output's future. We propose a new pseudo random bit generator (PRBG), whose fundamental component is the AA_β - function. The NIST Statistical Test Suite, which was regarded as the most rigorous statistical test suites to detect the precise properties anticipated of truly random sequences, was used to perform statistical testing on the generated bit sequences. The results are presented in detail as well.

Keywords: Pseudorandom number generator, statistical test

1 INTRODUCTION

In many situations such as cryptography, modeiling or simulation application, it is essential to generate sequences of random numbers or bits for various pur-

poses. For cryptographic purposes, Random Number Generators (RNG) are paramount for algorithms and protocols. For instance, they are required in the generation of unpredictable values to be used in key generation applications and also in other cryptographic algorithm parameters. As a result, the use of appropriate random number (bit) generators becomes vital and those generated sequences must be random in the sense that their behaviour should not be easily predicted. Even though there is no standard formal definition of it, randomness refers to the outcome of a probabilistic process that produces independent, uniformly distributed and unpredictable values that cannot be reliably produced (Schneier, 2007). Intuitively, one of the most important properties of a good random number generator is unpredictability. Since randomness is related to the unpredictable property, it can be described in probabilistic terms, and the randomness of a sequence can be studied by means of a hypothesis test. (Alcover, 2013).

Apparently from a direct observation of an output, a generator is said to be unpredictable if it is impossible to foresee any information about the future progress of a sequence. For example, we can observe a long sequence of numbers, by looking for some patterns. Then if a pattern is identified, the possibility to predict the future progress of the sequence is very high. On the other hand, if no pattern has being identified, we can't simply conclude that no pattern is present. Just that we failed in the effort of finding it. A good interpretation of what is random all about is by observing the result of flipping a fair coin. A random bit sequence could be interpreted as the result of flipping an unbiased 'fair' coin with sides that are labelled '0' and '1'. Each of flip having a probability of exactly $1/2$ of producing either '0' or '1'. Furthermore, the flips are independent of each other as the result of any previous coin flip does not affect future coin flips. All elements of the sequence are generated independently of each other, and the value of the next element in the sequence cannot be predicted, regardless of how many elements have already been produced. The output of such an idealized generator of a true random sequence serves as a yardstick for the evaluation of random and pseudo- random number generators.

Nonetheless there are many techniques to generate random bit sequences,

in the context of cryptography the quality of each generator should be measured by means of different statistical tests. "Passing statistical tests is a necessary condition, since any statistical defect can be used by an attacker to gain information about the future output", as it is stated in (Koeune, 2005). In this section, we propose a novel pseudo- random bit generator (PRBG) in which the AA_β - function is the basic building block. We also present the detailed result of the statistical testings on the generated bit sequences, done by using the NIST Statistical Test Suite which was considered the most stringent statistical test suites to detect the specific characteristics expected of truly random sequences (Patidar and Sud, 2009).

2 THE PROPOSED GENERATOR

2.1 The AA_β- function

Initially, the AA_β- function was employed in generating their public keys and the keys in the encryption and decryption procedure (Ariffin and Abu, 2009). Although their public key scheme was later found to be insecure, the recurrence function defined was a promising PRNG to be studied. We begin with the following definitions.

Definition 2.1. : *The set of binary strings with a length of k bits is defined by* $S_k^* = \left\{ s = \{b_i\}_{i=0}^{k-1} : b_i \in \{0,1\} \right\}$ *where* $k \in Z^+$

Definition 2.2. : *Let* $\alpha, \beta > 0$ *and* $\alpha < \beta$ *. The function* $AA_\beta(x_i)$ *is defined as*

$$AA_\beta^s(x_i) = x_{i+1} = \begin{cases} \alpha x_{i-1} + x_i, \ b_i = 0 \\ x_{i-1} + \beta x_i, \ b_i = 1 \end{cases} \tag{1}$$

where $i = 0, 1, 2, \ldots, (k-1)$, $x_{-1} = 0$, $x_0 \in \mathbb{Z}^+$ *and* $s \in S_k^*$.

2.2 The proposed Pseudo- Random Bit Generator (PRBG)

We explain our proposed generator as follows: Suppose k and l are positive integers where $k+1 \leq l \leq 2^k - 1$. Let $\alpha, \beta > 0$. A seed s_0 is an integer or $s_0 = \{b_i\}_{i=0}^{k}, b_i \in 0, 1$ where $1 \leq s_0 \leq 2^k - 1$. For $0 \leq i \leq l-1$, we define

$$s_{i+1} = \begin{cases} \alpha s_i + s_{i-1}, \ b_i = 0 \\ s_i + \beta s_{i-1}, \ b_i = 1 \end{cases} \tag{2}$$

and define

$$f(s_0) = (z_1, z_2, \cdots, z_i), \tag{3}$$

where

$$z_i = s_i \pmod 2, \quad 1 \leq i \leq l \tag{4}$$

Then f is a (k, l) - PRBG, called the AA_β pseudo- random bit generator. The proposed algorithm of the generator is as in Figure 1.

INPUT:	α , β, $x[0]$, n
OUTPUT:	Stream of bits

(1) **procedure** AA_β (α, β, x, y, b)
(2) For i from 2^{n-1} to $2^n - 1$ do $K \leftarrow i$
(3) For j from n by -1 to 1 do $B[j] \leftarrow K \pmod 2$
End do
(4) Set initial values $X \leftarrow 0, Y \leftarrow 1$
(5) For j from 1 to n do **procedure** in (1) by setting $Z \leftarrow fn(\alpha, \beta, X, Y, B[j])$;
$X \leftarrow Y; Y \leftarrow Z$
End do
(6) Convert Z into binary
(7) Choose the middle bit of the string
(8) End do
(9) Return generated bits.

Figure 1: Algorithm : TheAA_β pseudo random bit generator

3 THE STATISTICAL TESTING

In computer security, suitable metrics are needed to investigate the degree of randomness for binary sequences produced by cryptographic random number generators (RNGs). With the intention of gaining the assurance that newly developed pseudo random bit generators are cryptographically secure, a generator should be subjected to a variety of statistical tests. The tests were designed to detect the explicit characteristics expected of truly random sequences. Prior to be applied in any cryptographic applications, the statistical tests were designed to detect the specific characteristics expected of truly random sequences. Soto in (Soto, 1999) highlighted the four most popular options: NIST Test Suite (Elaine and Allen, 2011), the DIEHARD suite of statistical tests (Marsaglia, 1985), The Crypt-XS suite of statistical tests and the Donald Knuth's statistical tests set. There are different number of statistical tests in each of the above- mentioned test suites to detect distinct type of non-randomness in the binary sequences. Various efforts based on the principal component analysis show that not all the above- mentioned suites are needed to implement at a time as there are redundancy in the statistical tests.

Practically, statistical testing is employed to gather evidence that a generator indeed produces numbers that appear to be random (Soto, 1999). In other words, those tests help only to detect certain kinds of weaknesses a tested generator may have. Even if a sequence passes a finite number of statistical tests, there is no concrete evidence that the sequence was indeed generated by a (truly) random number generator (Kocarev and Jakimoski, 2003). For analysing the randomness of the proposed pseudo- random bit generator, we use the most stringent tests of randomness: the NIST suite tests. There are many papers using NIST statistical tests in analysing their PRBG, see (Alcover, 2013),(Cristina et al., 2012), (Patidar and Sud, 2009), (Jessa, 2010), (Ilyas et al., 2013), among others. Referring to (Soto, 1999) and (Pareschi et al., 2012), one of the advantages of NIST tests suite is the approach used does not require any assumption on the generator under test, since it only looks for evidence of particular statistical recurrences in a generated stream.

The NIST Statistical Test Suite is a statistical package based on hypothesis testing provided by The National Institute of Standards and Technology (NIST), USA. The package was developed, purposely to test the randomness of arbitrarily long binary sequences produced by pseudo- random bit generators. These hypothesis-based testing focus on a various type of non-randomness that could exist in a binary sequence. The NIST statistical package consists of 15 statistical tests which can be classified into two categories: (i) non-parameterized tests : Frequency (monobit) test, Runs test, Test for longest run of ones in a block, Binary matrix rank test, Cumulative sums test, Discrete Fourier transform (spectral) test (ii) parameterized tests: Frequency test within a block, Approximate entropy test, Linear complexity test, Maurel's universal statistical test, Serial test, Overlapping template matching test and Non-overlapping template matching test. For the detailed description of all 15 tests of NIST suite, we refer the readers to NIST document (Rukhin et al., 2010).

3.1 Testing Strategy

A hypothesis test is a formal procedure used in the statistics to accept or reject an assumption. The NIST structure, similar to other statistical tests, is based on hypothesis testing. The statistical test is formulated to test a specific null hypothesis denoted as H_0. The null hypothesis is determining whether or not a selected sequence of zeroes and ones is random. On the other hand, the associate alternative hypothesis denoted by H_a is that the tested sequence is not random. A decision or conclusion is derived from each particular test, as whether to accept or reject the null hypothesis, based on the tested sequence that was produced.

Referring to the NIST document (Rukhin et al., 2010), for each test a relevant randomness statistic must be chosen and used to determine the acceptance or rejection of the null hypothesis. Under an assumption of randomness, such a statistic has a distribution of possible values. A theoretical reference distribution of this statistic under the null hypothesis is determined by mathematical methods and corresponding probability value (*P*-value) is extracted. The *P*-value summarizes the strength of the evidence against the null hypothesis. If a *P*-value for a test is determined to be equal to 1, then the sequence appears to have perfect randomness. On the other hand, if a *P*-value equal to zero indicates that the sequence appears to be completely non-random.

A significant level (denoted as α) be chosen for the tests and if *P*-value $\geq \alpha$, then the null hypothesis is accepted that is the sequence appears to be random. If $P < \alpha$, then the null hypothesis is rejected that is the sequence appears to be non-random. Typically, the significant level α is chosen in the interval $[0.001, 0.01]$. For example, the $\alpha = 0.01$ indicates that one would expect 1 sequence out of 100 sequences to be rejected. A *P*-value ≥ 0.01 would mean that the sequence would be considered to be random with a confidence of 99%. An α of 0.001 indicates that one would expect one sequence in 1000 sequences to be rejected by the test if the sequence was random. For a regarding the quality of the sequence can be made on conclusion *P*-value ≥ 0.001, a sequence would be considered to be random with a confidence of 99.9%. For a *P*-value < 0.001 would mean that the conclusion was that the sequence is

non-random with a confidence of 99.9%.

For our numerical implementations on the proposed pseudo- random bit generator, we have divided our empirical testing into two parts. For the first part of it, we have generated 5 blocks of sequences. Each block consists of 1000 bit streams (sample size $m = 1000$), each of length 1024 bits , generated by 5 different parameters and randomly chosen seed, (α, β,$x[0] \in N$). For our testing purposes, we refer to (Abu et al., 2009) for using short size of sequence. Furthermore, as suggested by the NIST, there are only 8 tests which are particularly suitable for practical cryptographic keys size. The selected NIST Tests for short keys are listed in table 1 . In addition, for the second part of the testing, we have generated 5- bit streams, each of length 10^6, which was also generated via 5 different parameters and randomly chosen seed.

Item	Statistical Test	Min Practical bits *n*
1	Frequency Monobit	128 bits
2	Block Frequency	128 bits
3	Cumulative Sums (Forward and Backward)	128 bits
4	Runs	128 bits
5	Longest Runs of Ones	128 bits
6	Spectral DFT	1024 bits
7	Approximate Entropy	128 bits
8	Serial	128 bits

Table 1: The list of suitable random tests for short keys (Abu et al., 2009)

3.2 Interpretation of the empirical results

In this section we present the interpretation of our empirical results obtained from testing the randomness of the proposed generator using the NIST Statistical Tests. The empirical result can be interpreted in many ways. NIST adopted

the two approaches as follows: (1) the examination of the passing ratio of sequences that pass a statistical test (we denote as *PR*) and (2) the distribution of *P*-values (*P*-values uniformity) computed by the software itself. For every test that has been done to each block, final analysis report is generated when statistical testing is complete.

3.2.1 Empirical Results for the five Blocks

i) Passing ratio of each test

To determine the passing ratio of each test, the significance level for each test is set to 0.01 meaning that 99% test samples should pass the test. We considered a significant level of 1%. By estimation theory and resuming 1000 times each test, the range of acceptable proportions for each of the individual test can be determined by using the confidence interval defined as $\hat{p} \pm 3\sqrt{\hat{p}\left(1-\hat{p}\right)/m}$, where $\hat{p} = 1 - \alpha = 1 - 0.01 = 0.99$ and $m = 1000$. The acceptance region in our case will be $0.99 \pm 3\sqrt{0.99(1-0.99)/1000}$, namely $[0.9806, 0.9994]$.

We summarize the result for 8 tests from the NIST suite for each of the blocks tested in Table 2 : If the passing ratio belongs to the acceptance region (0.980561), the decision is SUCCESS or the test is passed. From the table, all 8 tests in the 5 different blocks passes the NIST test.

	Block 1		Block 2		Block 3		Block 4		Block 5	
Statistical Test	Passing Ratio	Decision	Passing Ratio	Decision	Passing Ratio	Decision	Passing Ratio	Decision	Passing Ratio	Decision
Frequency Test	0.9920	Success	0.9870	Success	0.9880	Success	0.9900	Success	0.9850	Success
Frequency Test Within a Block	0.9870	Success	0.9930	Success	0.9970	Success	0.9910	Success	0.9880	Success
Cumulative Sums (Forward)	0.9880	Success	0.9880	Success	0.9900	Success	0.9890	Success	0.9870	Success
Cumulative Sums (Reverse)	0.9900	Success	0.9900	Success	0.9910	Success	0.9920	Success	0.9830	Success
Runs Test	0.9920	Success	0.9930	Success	0.9870	Success	0.9930	Success	0.9890	Success
Longest Run	1.0000	Success	1.0000	Success	1.0000	Success	1.0000	Success	1.0000	Success
FFT	0.9990	Success	1.0000	Success	1.0000	Success	1.0000	Success	1.0000	Success
Approximate Entropy Test	0.9840		0.9890		0.9880		0.9870		0.9830	
Serial	0.9840	Success	0.9910	Success	0.9910	Success	0.9820	Success	0.9840	Success
Serial	0.9870	Success	0.9910	Success	0.9890	Success	0.9860	Success	0.9920	Success

Table 2: Proportion of the sequences passing against each test

ii) *P*-values uniformity of each test

The second approach adopted by NIST measures the distribution of *P*-values in interval [0,1] divided into ten equal-sized sub-intervals. It is generally suggested the uniformity of *P*-values for each of the individual test using χ^2 test. The χ^2 test value is defined in formula 5 where F_i is the number of *P*-values in the *i*-class (the *P*-values are put into 10 classes between 0 and 1, thus the degree of freedom in this case for χ^2 is 9; *m* is number of sampels (here $m = 1000$). As a result, we have obtained a new *P*-value (P_T) for each statistical tests. If $P_T \geq 0.0001$, then the sequences can be considered to be uniformly distributed. We refer to (Rukhin et al., 2010) for the details of computing (P_T).

$$\chi^2 = \sum_{i=1}^{10} \left(F_i - m/10\right)^2 / \left(m/10\right) \tag{5}$$

We present the results of the (P_T) for each statistical tests in Table 3. Only FFT and Approximate Entropy tests are not uniformly distributed in their *P*-values .

Statistical Test	Block 1		Block 2		Block 3		Block 4		Block 5	
	Test Value	Decision	Test Value	Decision	Test Value	Decision	Test Value	Decision	Test Value	Decision
Frequency Test	0.021849	Success	0.039329	Success	0.019056	Success	0.001923	Success	0.000054	Success
Frequency Test Within a Block	0.000000	Fail	0.119508	Success	0.152902	Success	0.209948	Success	0.064418	Success
Cumulative Sums (Forward)	0.034031	Success	0.0012219	Success	0.001475	Success	0.006379	Success	0.008266	Success
Cumulative Sums (Reverse)	0.099513	Success	0.020973	Success	0.023386	Success	0.000127	Success	0.108150	Success
Runs Test	0.620465	Success	0.699313	Success	0.208837	Success	0.296834	Success	0.223648	Success
Longest Run	0.000000	Fail	0.000000	Fail	0.000000	Fail	0.000000	Fail	0.000000	Fail
FFT	0.000000	Fail	0.000000	Fail	0.000000	Fail	0.000000	Fail	0.000000	Fail
Approximate Entropy	0.047478	Success	0.368587	Success	0.026948	Success	0.008446	Success	0.000030	Success
Serial	0.009333	Success	0.784927	Success	0.630872	Success	0.564639	Success	0.078086	Success
Serial	0.165340	Success	0.908760	Success	0.597620	Success	0.116065	Success	0.558502	Success

Table 3: P-values uniformity of each test

Next, we present the result for the bit streams. Each of the stream of length 10^6 were tested using all 15 statistical tests in the NIST Suite and the *P*-values obtained were recorded in the table. in Table 4. The P-values of of each single stream is required to be larger that $\alpha = 0.01$ in order to pass the statistical tests. From the table, it is clearly the tested stream passes all the 15 tests.

Item	Statistical Test	P-Values					Conclusion
		Stream1	Stream2	Stream3	Stream4	Stream5	
1	The Frequency (Monobit) Test	0.645153	0.736877	0.943277	0.604582	0.753330	SUCCESS
2	Frequency Test within a Block	0.817803	0.783079	0.563716	0.300432	0.223636	SUCCESS
3	Tests or the Longest-Run-of-Ones in a Block	0.556928	0.022036	0.256329	0.343877	0.954251	SUCCESS
4	The Binary Matrix Rank Test	0.065506	0.608969	0.650978	0.156065	0.275808	SUCCESS
5	The Discrete Fourier Transform (Spectral) Test	0.943767	0.832413	0.125385	0.022067	0.150897	SUCCESS
6	The Non-overlapping template Matching Test	0.296955	0.516541	0.111179	0.297834	0.364104	SUCCESS
7	The Runs Test	0.354874	0.672247	0.741669	0.512815	0.857190	SUCCESS
8	The Overlapping Template Matching Test	0.398553	0.987944	0.964777	0.583231	0.887990	SUCCESS
9	Maurer's Universal Statistical" Test	0.054254	0.321236	0.901046	0.190352	0.173285	SUCCESS
10	The Linear Complexity Test	0.509183	0.603357	0.179382	0.008396	0.706155	SUCCESS
11	The Serial Test	0.073787,0.172616	0.658946,0.714130	0.741669,0.633958	0.830727,0.861989	0.342947,0.813848	SUCCESS
12	The Approximate Entropy Test	0.331277	0.396291	0.607344	0.227891	0.083729	SUCCESS
13(a)	The Cumulative Sums (Cusums) (Forward)Test	0.938312	0.932395	0.972843	0.568970	0.647327	SUCCESS
13 (b)	The Cumulative Sums (Cusums) (Reverse)Test	0.541674	0.729567	0.991700	0.711926	0.719355	SUCCESS
14	The Random Excursions Test	0.398553	*	0.195385	*	0.711453	SUCCESS
15	The Random Excursions Variant Test	(x=-9) , 0.564418	*	(x=-9) , 0.207851	*	(x=-9) , 0.669155	SUCCESS
		(x=-8) , 0.630006	*	(x=-8) , 0.122673	*	(x=-8) , 0.887905	SUCCESS
		(x=-7) , 0.538501	*	(x=-7) , 0.139783	*	(x=-7), 1.000000	SUCCESS
		(x=-6) , 0.401760	*	(x=-6) , 0.229468	*	(x=-6) , 0.799199	SUCCESS
		(x=-5) , 0.217927	*	(x=-5) , 0.214500	*	(x=-5) , 0.608067	SUCCESS
		(x=-4) , 0.160416	*	(x=-4) , 0.115852	*	(x=-4) , 0.843857	SUCCESS
		(x=-3) , 0.260335	*	(x=-3) , 0.041085	*	(x=-3) , 0.781446	SUCCESS
		(x=-2) , 0.259335	*	(x=-2) , 0.018278	*	(x=-2) , 0.988569	SUCCESS
		(x=-1) , 0.139282	*	(x=-1) , 0.019183	*	(x=-1) , 0.518811	SUCCESS
		(x=1) , 0.388445	*	(x=1) , 0.190477	*	(x=1) , 0.585120	SUCCESS
		(x=2) , 0.894902	*	(x=2) , 0.253383	*	(x=2) , 0.337111	SUCCESS
		(x=3) , 0.776895	*	(x=3) , 0.195473	*	(x=3) , 0.498442	SUCCESS
		(x=4) , 0.915233	*	(x=4) , 0.317004	*	(x=4) , 0.645823	SUCCESS
		(x=5) , 0.698595	*	(x=5) , 0.752632	*	(x=5) , 0.655119	SUCCESS
		(x=6) , 0.632923	*	(x=6) , 0.772223	*	(x=6) , 0.793425	SUCCESS
		(x=7) , 0.922225	*	(x=7) , 0.534401	*	(x=7) , 0.994509	SUCCESS
		(x=8) , 0.855755	*	(x=8) , 0.601627	*	(x=8) , 0.898037	SUCCESS
		(x=9) , 0.847666	*	(x=9) , 0.697957	*	(x=9) , 0.621652	SUCCESS

Table 4: The result of P-values for Stream Bits.

4 CONCLUSION

We have put forth a design of a pseudo random bit generator (PRBG) based on AA_β function iterated independently from independent initial conditions. We have tested rigorously the generated sequence using the NIST suite, which are the most rigorous statistical tests suites to identify the specific characteristics expected of truly random sequences. The results of the statistical testing are encouraging and show that the proposed PRBG has some properties and hence can be suggested in the design of a new stream cipher.

REFERENCES

Abu, N., Lang, W., and Sahib, S. (2009). Cryptographic key from webcam image. *International Journal of Cryptology Research*, 1(1):115–127.

Alcover, P. (2013). A new randomness test for bit sequences. *INFORMATICA*, 24:339–356.

Ariffin, M. and Abu, N. (2009). aa_β cryptosystem: A chaos-based public key cryptosystem. *Int. J. Cryptology Research*, 1(2):149–163.

Cristina, D., Radu, B., and Ciprian, R. (2012). A new pseudorandom bit generator using compounded chaotic tent maps. In *Communications (COMM), 2012 9th International Conference on*, pages 339–342. IEEE.

Elaine, B. and Allen, R. (2011). Recommendation for cryptographic key generation. NIST special publication 800-133.

Ilyas, A., Vlad, A., and Luca, A. (2013). Statistical analysis of pseudorandom binary sequences generated by using tent map. *University Politehnica of Bucharest Scientific Bulletin (series A) -Applied Mathematics and Physics*, 75(3):113–122.

Jessa, M. (2010). Improving statistical properties of number sequence generated by multiplicative congruential psudorandom generator. *Advances in Electronics and Telecommunication*, 1(2):51–54.

Kocarev, L. and Jakimoski, G. (2003). Pseudorandom bits generated by chaotic maps. *IEEE Transactions on Circuits and Systems-1: Fundamental Theory and Applications*, 50(1):123–126.

Koeune, F. (2005). *Encyclopedia of Cryptography and Security*. Springer, Berlin.

Marsaglia, G. (1985). A current view of random number generators. In *Computer Science and Statistics, Sixteenth Symposium on the Interface. Elsevier Science Publishers, North-Holland, Amsterdam*, pages 3–10.

Pareschi, F., Rovatti, R., and Setti, G. (2012). On statistical tests for randomness included in the nist sp800-22 test suite and based on the binomial distribution. *Information Forensics and Security, IEEE Transactions on*, 7(2):491–505.

Patidar, V. and Sud, K. (2009). A novel pseudo random bit generator based on chaotic standard map and its testing. *Electronic Journal of Theoretical Physics*, 20:327–344.

Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., and Leigh, S. (2010). A statistical test suite for random and seudorandom number generators for cryptographic applications. NIST special publication 800-22 Revision 1a.

Schneier, P. (2007). *Applied Cryptography:Protocols, Algorithms, and Source Code in C*. John Wiley and Sons, Reading, MA.

Soto, J. (1999). Statistical testing of random number generators. In *Proceedings of the 22nd National Information Systems Security Conference*, volume 10, page 12.