SMU Data Science Review

Volume 2 | Number 2 Article 5

2019

Longitudinal Analysis with Modes of Operation for AES

Dana Geislinger
Southern Methodist University, dgeislinger@smu.edu

Cory Thigpen
Southern Methodist University, cthigpen@smu.edu

Daniel W. Engels
Southern Methodist University, dwe@smu.edu

Longitudinal Analysis with Modes of Operation for AES

Dana Geislinger, Cory Thigpen, and Daniel Engels

Master of Science in Data Science
Southern Methodist University
Dallas, Texas USA
{dgeislinger, cthigpen, dwe}@smu.edu

Abstract. In this paper, we present an empirical evaluation of the randomness of the ciphertext blocks generated by the Advanced Encryption Standard (AES) cipher in Counter (CTR) mode and in Cipher Block Chaining (CBC) mode. Vulnerabilities have been found in the AES cipher that may lead to a reduction in the randomness of the generated ciphertext blocks that can result in a practical attack on the cipher. We evaluate the randomness of the AES ciphertext using the standard key length and NIST randomness tests. We evaluate the randomness through a longitudinal analysis on 200 billion ciphertext blocks using logistic regression and a dense neural network. We also trained a dense neural network on thirty billion sequential CBC blocks. Each of these models targeted a single bit location in ciphertext but none resulted in a prediction accuracy greater than 50%. We were unable to find evidence that AES ciphertext is not random, and we conclude that AES remains a safe cipher to use in modern cryptography applications.

1 Introduction

Encryption ciphers will always be susceptible to attackers looking to compromise their security with methods such as brute force and cryptanalysis in a reasonable amount of time. It is crucial that these ciphers remain secure in the face of these types of attacks. Modern computing power is strong enough to render previously secure methods obsolete and deciphering AES ciphertext is very quick, if the encryption key is known. AES works by using a secret encryption key to generate an encrypted message (called ciphertext) from a plaintext message. The ciphertext can only be decrypted back to the original message if the encryption key is known. The current recommendation for AES implementations uses a 128-bit encryption key and all AES implementations generate ciphertext in blocks of 128 bits at a time.

Deciphering AES ciphertext by brute force requires correctly guessing all 128 bits of the encryption key, and modern computers are still not strong enough to brute force Advanced Encryption Standard (AES) ciphertext within a reasonable amount of time. On average, 2^{127} guesses are necessary to correctly guess the 128 bits in the encryption key. As a result, researchers and hackers have sought methods to reduce the computational complexity of compromising AES.

One hallmark of AES ciphertext is that the bits in a ciphertext block are random. This means that AES ciphertext should be free of any identifiable patterns, should have a

roughly uniform distribution of bits as either 0 or 1, and should not repeat any sequence as new ciphertext blocks are generated from the same key among other randomness requirements [1]. This randomness of ciphertext bits is crucial to the security of the AES cipher because it means that an attacker should not be able to glean any information about an encryption key or the underlying plaintext message by analyzing the random ciphertext alone.

Since bit randomness is such an important attribute of AES ciphertext, if any patterns or trends were to be discovered within AES ciphertext, they could be leveraged to compromise the AES cipher. The assumption of random ciphertext means that there should be approximately a 50% chance that any algorithm will guess the correct state of any single bit in an AES ciphertext block. A theoretical method to reliably predict a ciphertext bit assignment with a prediction rate greater than random chance (greater than 50%) would provide evidence that some non-random pattern exists within AES ciphertext and the pattern could potentially be exploited to compromise information about the encryption key or underlying plaintext message.

Randomness in AES is analyzed by fifteen tests for randomness designed by the National Institute for Standards and Technology (NIST). These tests ensure that AES-generated ciphertext is free from certain repeating patterns and has a uniform distribution of bits among other things; however, the test suite was designed as a series of hypothesis tests for the demonstration of randomness when tested against a relatively small sample of ciphertext [1]. These tests were not designed for implementation against a large data set of AES ciphertext and it is possible that vulnerabilities exist within the cipher that are only apparent after millions of iterations of ciphertext generation.

While AES ciphertext is designed to be random, another crucial aspect of the AES cipher is the mode of operation under which ciphertext is generated. AES was designed with five potential modes of operation and these modes dictate the method that the cipher uses to generate a ciphertext block from a plaintext message and an encryption key. For example, Counter (CTR) mode encrypts blocks of plaintext data by incorporating an incremental number into the encryption process and Cipher Block Chaining (CBC) mode incorporates the previously encrypted ciphertext block as each subsequent block of ciphertext is generated [2].

While all five modes of operation were designed to be secure for use in different applications, it has been shown that Electronic Code Book (ECB), Cipher Feedback (CFB), and Output Feedback (OFB) modes can be susceptible to vulnerabilities under certain conditions or in specific implementations. Therefore, of the original five modes of operation only CTR and CBC modes are generally recommended for use today [3].

These two modes are both still used in modern applications of AES. CTR mode is widely used as a way to use AES as a stream cipher and is considered by many to be the most secure of the original five modes of operation [1] [4]. CBC mode is also commonly used as a block cipher for applications including internet security and whole filesystem encryption [5] [6]. Since CTR and CBC modes are still used today and believed to be secure, non-random patterns discovered in ciphertext generated using these two modes would be the most impactful to the security of modern AES implementations.

While CTR and CBC modes are believed to be secure implementations of AES, it is possible that vulnerabilities exist within these modes of operation and that ciphertext

generated using these modes may not be random. We address this problem by performing a longitudinal analysis by generating ciphertext blocks in CTR and CBC modes of operation to predict whether a bit of AES ciphertext is a `0` or a `1`. This analysis empirically tests the randomness of generated ciphertext by quantifying the prediction accuracy we achieve targeting each ciphertext bit location.

We generated 100 billion blocks of AES ciphertext in both CTR and CBC modes of operation for further analysis. Blocks in each mode of operation were generated using the same random encryption key and encrypting the same sixteen character (128-bit) message: " The Plain Text ". CBC-generated blocks were generated in sequence with each ciphertext block being incorporated in the encryption of the next block. CTR mode requires a counter to be assigned that is incremented with each block generation step and this counter was initialized with a `1`.

Blocks were saved to disk on SMU's ManeFrame II high performance computing resource for further analysis. Each block of ciphertext was treated as a 1×128 matrix of `0` and `1` values corresponding to the value of each individual binary bit. Binary classification models were developed using logistic regression and dense neural networks. The models were each trained on samples of the generated blocks with one bit treated as the target variable. These models were then used to predict whether the target bit was a `0` or a `1`.

Models were trained on the first ten million blocks generated in both CTR and CBC modes to predict each of the 128 bit locations in turn for a total of 256 logistic regression models and 256 dense neural networks. When predictions were carried out on a test data set, no models were able to a achieve a prediction accuracy greater than 50%. A three-layer dense neural network was also trained on thirty billion sequentially generated CBC blocks. This model was trained by sequentially batching in ciphertext blocks from hundreds of files saved to disk on SMU's ManeFrame II. This dense neural network was only able to achieve a training set accuracy of 50% when predicting the first ciphertext bit location.

Our work shows that there is not sufficient evidence that AES ciphertext is non-random based on a longitudinal study of billions of AES CTR and CBC-generated blocks. All the models we tested resulted in a prediction accuracy no greater than 50%, which is the same prediction rate that would be expected if each bit were guessed randomly. Therefore, we conclude that AES ciphertext appears to be random to the extent that we have analyzed it and have found no evidence to conclude that AES is insecure for use in modern cryptography.

2 Advanced Encryption Standard (AES)

2.1 Overview

AES is the Rijndael cipher, selected by NIST in October 2000 via an open platform out of four other finalist ciphers, which included MARS, RC6, Serpent, and Twofish [7]. AES is a symmetric key block cipher that has been used for many applications in cryptography since it was adopted. AES is a symmetric key algorithm because the same hidden encryption key is used for both the encryption of a message into illegible ciphertext

and the decryption from ciphertext back to the original message. This is relevant to the security of AES because it means that a compromised key can be used to read encrypted messages and to generate false messages using the encryption key. AES is a block cipher because all encryption is performed on chunks (or blocks) of a fixed length. As a block cipher, AES utilizes a fixed 128-bit block size, meaning plaintext messages are split up into separate blocks of this size to perform encryption. Though Rijndael tested multiple key sizes, block sizes, and number of rounds [8], AES standardized the implementations displayed in Table 1 [9]. All generated ciphertext is available as blocks of a fixed length (128-bits in the case of AES) [4]. This contrasts a stream cipher, which encrypts data as part of a continuous stream of information.

AES was adopted by NIST because it showed strong performance in a wide range of environments, whether hardware or software-based. The implementation also had high agility especially in low-memory use cases. It was also the easiest implementation to guard against well-known side-channel attacks, such as power and timing attacks. Lastly, the internal round infrastructure benefited from parallelism. AES is an iterative block cipher implementation that transforms blocks of plaintext of a fixed size to ciphertext blocks of the same size by performing iterative rounds of Boolean transformations using the cipher key, as well as substitutions [7].

Bit length security of a cipher refers to the expected number of encryption key guesses required to successfully guess the encryption key used to encrypt a message. For AES-128 there are 2^{128} possible keys and ciphertext would on average be determined in half of the key guesses. This corresponds to 2^{127} (1.7×10^{38}) possibilities.

Table 1. Variants of AES

Name	Key Size (bits)	Block size (bits)	Rounds of Iterations
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

2.2 Properties

The general application for the algorithm is implemented by any of the variants of AES in order to apply the concepts of diffusion (obfuscate the statistical relationship between the plaintext and ciphertext) and confusion (obfuscate the statistical relationship between the ciphertext and key) [9]. AES does this by performing the repeated transformation functions illustrated in Table 2.

AES is designed to satisfy all of the National Institute for Standards and Technology (NIST) tests for randomness [10] and provide secure encryption when implemented properly; however, vulnerabilities that render some modes of AES insecure weren't discovered until years after AES was first introduced. For example, all five modes were recommended for use when AES was created in 2001 [1], but weaknesses within the modes of operation [3] were published by researchers concluding that the Counter

(CTR) mode of operation was the only mode secure enough for implementation, while Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB) modes should not be used due to their vulnerabilities to certain types of attack.

While CBC mode was found to be at risk against certain padding attacks and chosen ciphertext attacks the risk of these attacks is minimized under proper implementations. Specifically, it is crucial to implement a truly random initial value (IV), use a secure method for padding ciphertext blocks, and refresh encryption keys at least once every 2^{64} rounds of encryption [3]. While CTR mode may be considered a more secure mode of operation, CBC-based encryption schemes remain secure when implemented and maintained properly.

Today CTR mode has been shown to be the most secure mode of operation as it does not rely on chaining blocks for encryption and therefore is not susceptible to the many attacks that exploit the propagation of errors between subsequent blocks during encryption [1, 4]. CBC mode is commonly used for block ciphers and is no exception with applications in IPSec such as in TLS and in whole filesystem encryption schemes such as Apple's FileVault [5, 6, 11]. CBC has recently been supplanted by other encryption schemes such as AES-XTS for filesystem-level encryption and schemes directly incorporating authentication checks such as AES-CCM in TLS 1.3 [6, 12]. Nevertheless, CBC is still frequently used for general purpose block cipher applications and therefore it is still crucial that the algorithm be secure. Therefore, this research focuses on the generation of blocks using CTR and CBC modes of operation.

Table 2. Repeated Transformation Functions in AES

Function	Description
AddRoundKey	Generate a set of expanded keys as required by the number of rounds based on parts of secret key; each expanded key part is then applied as an XOR to the incoming text from previous step
SubBytes (and InvSubBytes)	Byte by byte substitution of the block using an S-box
ShiftRows (and InvShiftRows)	Permutation
MixColumns (and InvMixColumns)	Additional substitution

2.3 Modes of Operation

Encryption engines do not inherently have methods for ensuring each occurrence of ciphertext is different from a previous iteration of ciphertext on the same plaintext. Various methods were implemented in 2001 to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block [2] and are displayed in Table 3. These methods are known as *Modes of Operation*. Along with Cipher Block

Chaining, Counter mode of operation is one of two block cipher modes recommended by cryptography experts Niels Ferguson and Bruce Schneier [13].

Table 3. Modes of Operation

Name	Abbrev.	Proven Vulnerable to Attack
Electronic Code Book	ECB	Yes
Cipher Feedback	CFB	Yes
Output Feedback	OFB	Yes
Cipher Block Chaining	CBC	No
Counter	CTR	No

XOR, short for *exclusive or*, is a binary operation where plaintext bits (Column A from Table 4) are XORed with another value (Column B from Table 4) to return *False* or `0` if both operators are equal and *True* or `1` if both operators are not equal. The logic behind this operation is that it is impossible to reverse the operation without knowing the initial value of one of the two arguments [14].

Table 4. XOR Operation

A	B	A XOR B
1	1	0
1	0	1
0	1	1
0	0	0

In Cipher Block Chaining the first block of plaintext is XORed with an Initialization Vector (IV) prior to encryption (Figure 1). The ensuing block of ciphertext is then XORed with the next block of plaintext and so on until all blocks of plaintext have been encrypted.

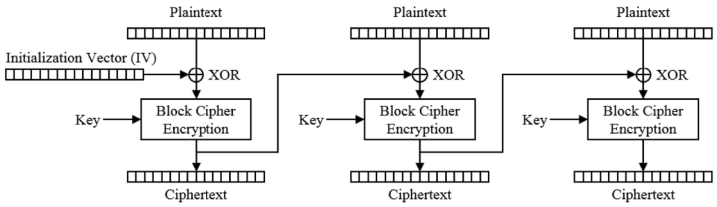

Fig. 1. Cipher Block Chaining (CBC) Mode of Operation

Counter mode encrypts a nonce and counter before XORing it with each block of plaintext (Figure 2). A nonce is a random, single use value and it need not be kept secret. So long as the key is secure and the nonce is of single use then encryption method is said to be secure.

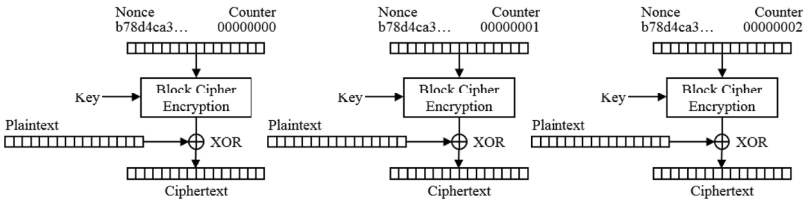

Fig. 2. Counter (CTR) Mode of Operation

2.4 Randomness

Should an AES encryption engine be found to be nonrandom, then a cryptanalysis attack could reduce computation complexity, potentially making a brute force attack viable. If the encryption process is indeed random, then the likelihood of guessing the key for a brute force attack becomes on average 2^{n-1} for n-bit encryption.

NIST outlined a statistical test suite for random and pseudorandom number generators for cryptographic applications [7]. The tests in Table 5 were satisfied for AES in 1998 when NIST declared AES met the criteria for security, cost, and algorithm and implementation characteristics [15].

3 Types of Attacks

The simplest form of attack on a cryptographic cipher is the brute force attack, which involves systematically guessing every possible combination of bits that could comprise the encryption key until the right bit combinations is found. In the case of AES a brute force attack is carried out by performing key generation, decryption, and checking for a readable plaintext in a loop until plaintext is found, meaning that the correct encryption key has been identified. Python libraries provide enough tools to set up this loop, including the Cryptography Toolkit (library name *pycrypto*) to generate blocks and check for plaintext, a threading library for multi-threading (library name *threading*), and an encoding library (library name *base64*).

To brute force attack AES-128 encryption, it would be expected that on average the 2^{n-1} value, or half, of all possible values would lead to the correct private key. This equals 2^{127} (1.7×10^{38}) possibilities. The AES encryption algorithm was used to generate 128-bit blocks on a 0.6 teraflop cluster (Intel Xeon E5-2680 v4 CPU) from

Table 5. NIST Randomness Tests

Number	Test Name
1	Frequency (Monobit) Test
2	Frequency Test within a Block
3	Runs Test
4	Tests for the Longest-Run-of-Ones in a Block
5	Binary Matrix Rank Test
6	Discrete Fourier Transform (Spectral) Test
7	Non-overlapping Template Matching Test
8	Overlapping Template Matching Test
9	Maurer's "Universal Statistical" Test
10	Linear Complexity Test
11	Serial Test
12	Approximate Entropy Test
13	Cumulative Sums (Cusums) Test
14	Random Excursions Test
15	Random Excursions Variant Test

SMU's Maneframe II. Average generated blocks per second were 4.7 million with a standard deviation of 257 thousand, assuming 85% efficiency.

At this rate a data center with one septillion (1.1×10^{24}) CPUs would be needed to brute force attack half of the AES-128 blocks within twelve months in the HPC. With manufacturing costs of $1,700 each, this data center would take 25 trillion years of the current gross world product to fund. At a size of $2.1 \times 2 \times 1.5$ inches the entire surface of the earth would be covered with CPUs to the height of an 11-story building.

Individual brute force attacks are unrealistic but enhanced computational power from nation-states or other creative means for attack must be considered, including leveraging supercomputers, botnets, and quantum computers.

Despite formal acknowledgement that Moore's Law is near its end [16], consider this false for the sake of example. Keeping Moore's Law constant, AES-128 would remain secure for 43 years from a single supercomputer. A nation-state leveraging the power of one thousand supercomputers would only cut their key recovery time down by ten years (from 43 to 33). Even ignoring the warnings about Moore's Law ending, neither option is promising.

AES-128 is not realistically susceptible to standard brute-force attacks as computation times to determine AES encryption keys are extremely prohibitive, even with modern hardware. In fact, NIST recommended in 2016 that the AES-128 variant still be used for purposes in which confidentiality is desired [15].

4 Generating Data for Longitudinal Analysis

Longitudinal analysis involves taking repeated measures over time to identify trends or changes in the data being sampled. We performed longitudinal analysis of AES ciphertext

by repeatedly encrypting the same message and analyzing the resulting ciphertext over billions of iterations.

The primary goal of this research project was to identify any patterns in AES generated blocks that might compromise the security of AES and allow an attacker to gain privileged information about the data being encrypted. Such a leak of information represents a significant security risk because AES security revolves around the idea that encrypted ciphertext blocks should not have any meaning for a user not in possession of the key that was used to encrypt the blocks. While AES is widely believed to be secure if implemented properly today, not all of the five AES modes of operation are currently recommended for use as there are vulnerabilities and limitations present in implementations of some modes.

Analysis of AES ciphertext blocks requires an enormous data set of encrypted blocks of ciphertext. To obtain a data set of adequate size, blocks were generated using SMU's ManeFrame II. All blocks were encrypted with the same plaintext message: " The Plain Text " which is 128 bits in length. This is the necessary size of a CBC block, so no padding was necessary. For each encryption session, a pseudo-random key was generated, and the same key was used for the generation of each block. Initially, an initialization vector (IV) of 1 (padded out to the necessary number of bits) was used, but in future generation sessions a random IV can be specified as well. Each block was encrypted using standard CBC parameters so each ciphertext block was XORed into the plaintext when the subsequent block was encrypted. Block generation was performed with SLURM Workload Manager, an open-source job scheduler for Linux, on SMU's ManeFrame II batch queuing system. Scripting was performed in Python 3 using the *pycrypto* library.

For each session, ciphertext blocks were numbered in the order in which they were generated and stored as hexadecimal values in comma separated values (.csv) files split every 100 million blocks. Ciphertext bits were encoded as hexadecimal values and stored on each row of the output file with the associated number of the block stored as the first column. Each .csv file of 100 million blocks took approximately seven gigabytes to store and ten billion CBC blocks were generated in 18.3 hours using SMU's ManeFrame II.

Additionally, ten billion blocks were generated in CTR mode. These blocks were generated in the same manner as the CBC blocks, except that CTR mode requires an incremental counter instead of an IV, so an incremental counter was implemented beginning at 1 for CTR block generation. This process completed in 18.4 hours on SMU's ManeFrame II (Table 6).

Table 6. Computation Run Time to Generate 100 Billion 128-Bit Blocks in CTR Mode of Operation

Blocks Generated	Computation Run Time
10 Billion	18.4 hours *(average)*
100 Billion	7 days 16 hours *(actual)*
1 Trillion	76 days 16 hours *(estimated)*

In the end, one hundred billion blocks of ciphertext were generated in both CTR and CBC modes for use in further analysis.

5 Testing Randomness Among Blocks

The suite of NIST Tests for Randomness was applied against one million blocks generated in both the CTR and CBC modes on SMU's ManeFrame II. All tests were performed in Python (library name *sp800_22_tests*) [17].

Table 7. NIST SP800-22-rev1a Test Suite Results on One Million Generated AES-128 Blocks

Test Name	CBC Mode		CTR Mode	
	Raw Score	Result	Raw Score	Result
Frequency (Monobit) Test	0.045571	PASS	0.524403	PASS
Frequency Test within a Block	0.462516	PASS	0.042495	PASS
Runs Test	0.218229	PASS	0.271389	PASS
Tests for the Longest-Run-of-Ones in a Block	0.616947	PASS	0.666535	PASS
Binary Matrix Rank Test	0.994579	PASS	0.583539	PASS
Discrete Fourier Transform (Spectral) Test	0.557586	PASS	0.831076	PASS
Non-overlapping Template Matching Test	1.566922	PASS	0.031982	PASS
Overlapping Template Matching Test	0.220586	PASS	0.151612	PASS
Maurer's "Universal Statistical" Test	0.999915	PASS	0.999778	PASS
Linear Complexity Test	0.478255	PASS	0.518329	PASS
Serial Test	0.141819	PASS	0.659190	PASS
Approximate Entropy Test	0.141832	PASS	0.837311	PASS
Cumulative Sums (Cusums) Test	0.064543	PASS	0.414218	PASS
Random Excursions Test	0.047738	PASS	0.042544	PASS
Random Excursions Variant Test	0.068640	PASS	0.035013	PASS

One million blocks generated using both modes of operation pass all of the tests in the NIST suite. These tests were designed with much smaller sample sizes in mind, so processing time and available memory are both factors that significantly limit the number of blocks that can be run through the NIST tests. Processing time to run the NIST tests on these blocks took more than four hours on SMU's ManeFrame II, and we were not successful in analyzing a larger sample of blocks without encountering memory limitations on a node with 768 GB of memory. Nevertheless, since all fifteen tests passed for the one million blocks of each mode of operation, we will continue under the assumption that the AES-generated blocks are indeed random enough for use in encryption and proceed with longitudinal analysis of the generated ciphertext.

The generated blocks passed all fifteen tests and seem to meet the general qualifications for randomness. Of particular note, it was important that the blocks passed the monobit test because we would expect the generated blocks to have an approximately even distribution of `0` or `1`. Similarly, passing the runs test was crucial for any encryption cipher because long strings of `0` or `1` could be representative of a pattern exploitable by cryptanalysis against the cipher [10].

The results of these tests illustrate some issues with current cipher testing. With the high volume of internet-connected devices present in the world today it is not unlikely that encryption implementations involving large data sets will exist, but the current NIST Tests for Randomness is not tailored to identify issues in large sample sizes. This is why it was important to test for shortcomings in these algorithms in novel ways such as large scale classification using tools including logistic regression and dense neural networks. These might allow for identification of patterns of ciphertext non-randomness that the legacy NIST Tests for Randomness are simply unable to identify.

6 Repeated Encryption for Odds Greater than 50%

Blocks of ciphertext were analyzed in order to ascertain whether or not any individual bit in the ciphertext could be predicted with a probability greater than chance (greater than 50% prediction accuracy).

With two hundred billion AES-128 blocks generated using the same key and encrypting the same plaintext message, we believe the data set was sufficiently large enough to perform this type of analysis. The purpose of building models was to see if any non-random patterns could be found for any bits in the ciphertext blocks. If any patterns were found in the data this could provide valuable insight into how AES could potentially be compromised since AES was designed to produce blocks of ciphertext in which the designation of each bit has no probabilistic relationship with the bits in the rest of the encoded message.

To facilitate the analysis, each block of ciphertext was encoded as a vector of binary values 128 bits in length, with each point in the vector corresponding to either a `0` or a `1` in the encrypted ciphertext. Statistical analysis was performed on these bits and models were created to predict whether any given bit in a ciphertext sequence would be a `0` or a `1`. Standard statistical analysis such as hypothesis testing with p-values from a test statistic were not employed for these analyses; since we were working with very large data sets, we have found that standard statistical measures (such as significance tests in logistic regression) are unreliable and many results have low p-values but no *practical* significance.

For example, a result with 50% prediction accuracy may result in a low p-value; however, this would still be the same prediction rate that random chance would provide so the result would not be practically significant. Therefore, all classification techniques employed in testing each bit location in the generated ciphertext blocks were evaluated by prediction accuracy alone, with any results above 50% being subjected to further analysis to determine the true practical significance of the result. The classification models trained were logistic regression models as well as dense neural networks to leverage machine learning principles to categorize the ciphertext bits.

AES-generated ciphertext should ideally have an even distribution of `0` and `1` bits, so predictive models were evaluated on the basis of their ability to classify bits correctly at a probability greater than 50%. Prediction accuracy was measured based on a randomly selected test portion of the ciphertext blocks that were held back when training the model.

7 Analysis with Logistic Regression

To assess whether the bits in a 128-bit block of AES ciphertext impacted the probability of another bit location having predictable odds greater than 50%, 128 logistic regressions were performed for each bit location using both CTR and CBC modes of operation. The bit location of interest was the dependent binary variable and the logistic regression was used to help explain the relationship — if any — between the target bit and the independent variable bits in the remaining 127 locations.

Each logistic regression model was fit using ten million CTR or CBC ciphertext blocks generated on SMU's ManeFrame II. Before model fit, the ten million blocks were partitioned into a training and testing data set with the testing data being held back for validation purposes after the regression models were fit. Two million blocks representing 20% of the total data were randomly held back as test data. This data partitioning was random (using pseudorandom number generation in the Python module *scikit-learn*), but the same random seed was used to partition each time so that blocks chosen for the test set were the same for each target bit location.

After train and test data were split, logistic regression models were fit targeting each of the 128 bit locations in sequence with the remaining 127 known bits being used as model input for each validation step. Models were built using the *scikit-learn LogisticRegression* class with L1 and L2 variable regularization via elastic net penalization. The optimization chosen during model building was SAGA in order to facilitate binary classification of the target variable while also supporting elastic net penalty.

Models targeting each bit location were evaluated by prediction accuracy on the test set in both modes. Results are shown in Figure 3. When evaluated on the two million block test set, each model scored no greater than 50%, indicating that logistic regression models trained in this way resulted in no increase in predictive power over what would be expected by random selection.

8 Dense Neural Network Analysis to Reduce Computational Complexity

Dense neural networks were trained to predict the bit assignment in each position of a ciphertext block generated in both CTR and CBC modes of operation. The Keras API with a Tensorflow backend was used for the creation of these models. Models were trained on ten million blocks for each mode of operation. Each ciphertext block was treated as a vector of 128 integer values with each sequential bit in the block represented in the vector. 128 total models were trained for each mode of operation with one bit location held out as the target variable and the other 127 other bit positions serving as input to the dense neural network.

The target bit position was incremented in each subsequent model with the result being the creation of 256 dense neural networks for the prediction of each bit location given the other 127 bits for CTR and CBC-generated blocks. For model learning an 80:20 training:testing split was randomly assigned for each model using the *scikit-learn* Python module. Additionally, 10% of the training data was held for validation during network learning.

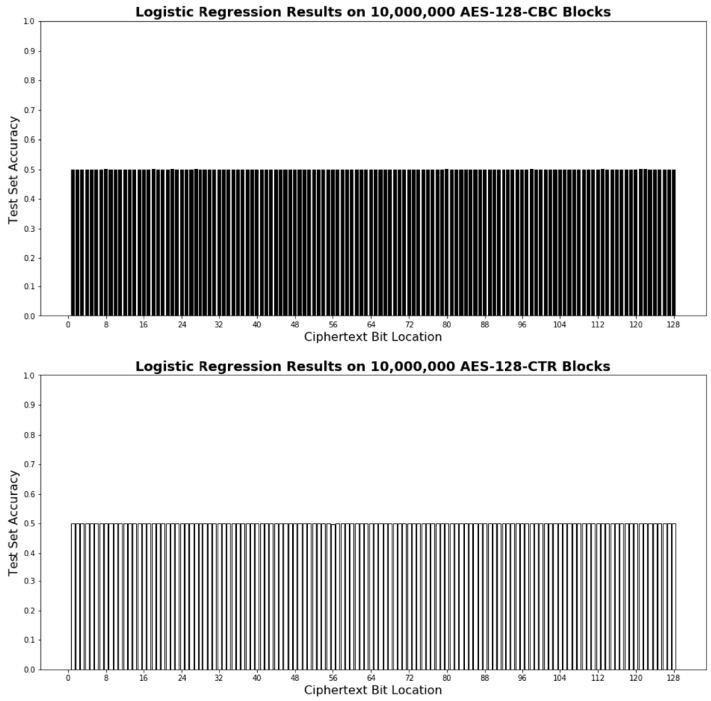

Fig. 3. Test Accuracy of Logistic Regression for Bit Prediction of AES-128 Generated Ciphertext

All 256 dense neural networks share the same sequential structure and hyperparameters. The input layer consists of 127 inputs corresponding to each ciphertext bit position aside from the current target bit position. The hidden layers of the network consist of three fully connected layers of 96, 64, and 32 nodes respectively each with ReLU activation function. Additionally, dropout layers with keep probabilities of 0.5 were introduced after each of the three hidden layers to prevent overfitting of individual weights during learning. Finally, the output layer was a single node with sigmoid activation to provide the final prediction (whether the class of the target bit was a `0` or a `1`).

Binary cross-entropy was chosen as the loss function because these networks are designed to perform binary classification. The *rmsprop* optimizer was used for all networks with default parameterization. Learning for each network was carried out over twenty epochs with a batch size of ten thousand.

The dense neural networks did not successfully predict bits in any location. Prediction accuracy on the test set for each model (corresponding to predictions on a single bit location) was very close to 50% for each model (see Fig. 4). Since this was a binary classification problem (whether a bit was a `0` or a `1`) and AES ciphertext bits were meant to be truly random and we expected random chance to produce a prediction accuracy of approximately 50%, so these results have not shown any improved prediction accuracy.

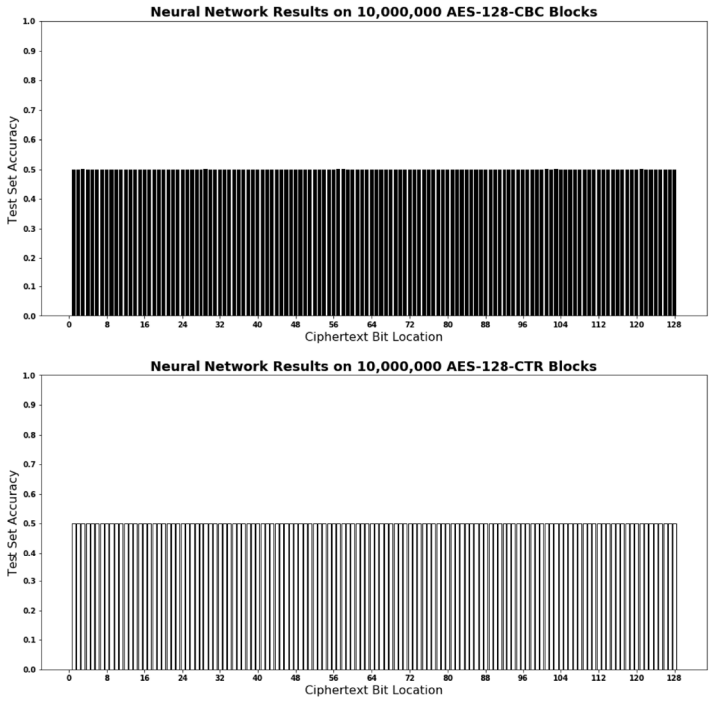

Fig. 4. Test Accuracy of Dense Neural Networks for Bit Prediction of AES-128 Generated Ciphertext

From the results of the current analysis, we cannot conclude that the dense neural networks described here are useful tools for the prediction of AES ciphertext bits.

In addition to these 256 dense neural networks targeting a different bit location in each mode, a dense neural network was trained using thirty billion CBCs sequentially generated ciphertext blocks with no shuffling or testing data kept back (all thirty billion supplied to the model in the sequence in which they were generated). The purpose of training this model is to better determine if a non-random pattern can be learned by a dense neural network with a large body of sequentially generated blocks.

Only CBC-generated blocks were evaluated with this dense neural network because the ciphertext from each block is incorporated into the encryption process of the next block in CBC mode so it is possible that a pattern may emerge when CBC blocks are evaluated in sequence. In CTR mode, each block is encrypted using only the encryption key and an incrementing counter variable so it was not evaluated in the same way.

The network architecture trained with the thirty billion blocks was similar to the previously-trained networks; a 127 (each of the known bits in a ciphertext block) unit input layer was linked to three fully-connected layers of 96, 64, and 32 nodes each with a single node output layer at the end of the network. There were no dropout

layers included in this network as the risk of overfitting on sequential blocks was not a concern; previous smaller tests that included validation data had not shown signs of overfitting when sequential blocks were trained and we only trained this model for a single epoch which limits the opportunity for overfitting provided by multiple runs through the training data.

The activation function of each of the three hidden layers was ReLU and the output layer used a sigmoid activation function. The optimizer chosen for this model was rmsprop and binary cross-entropy was again used as the loss function. No validation data was provided in this case and no testing data was held back from the thirty billion blocks used for training.

In order to train the network on such a large quantity of data, a Keras Sequence generator class was defined that read in multiple files from disk sequentially as needed to provide the model with the next batch of training data. Files on disk containing *numpy* arrays of ten million $\times$ 128 storing subsequently generated ciphertext were loaded in as needed by the Sequence class and the model was trained on this data in batches of ten thousand blocks. This was necessary as models trained with more than ten million blocks at a time resulted in an out of memory error when run on SMU's ManeFrame II GPU instance with 256 GB of memory and a graphics processor with 16GB of video memory.

Due to the high computational cost, limited ManeFrame II run time, and disk access throughput necessary to train this network on thirty billion blocks, only one dense neural network was evaluated. This dense neural network was trained on thirty billion blocks with the first bit position of the ciphertext as the target bit position to predict.

The network was trained for one epoch at which time the accuracy on the training data set was evaluated. The dense neural network completed training on the thirty billion blocks after 20.293 hours with a final loss (binary cross-entropy) of 0.6931 and a training prediction accuracy of 50.00%. This result shows that prediction accuracy when training the dense neural network on thirty billion sequentially generated CBC ciphertext blocks was not greater than what would be expected by random selection.

9 Ethical Considerations

This research aimed to promote the expansion of knowledge, collaboration between researchers, honesty, integrity, and openness in order to minimize harm and maintain confidentiality and privacy. It also applied to various ethical principles from the Association for Computing Machinery (ACM) Code of Ethics and Professional Conduct, specifically "respect privacy and honor confidentiality" and "be honest and trustworthy" [18].

To "respect privacy and honor confidentiality" only self-generated data was used for this research (the same 128-bit block) and no research participants were used. To "be honest and trustworthy" all findings associated with this research — longitudinal analysis of CTR and CBC modes of operation for AES — were fully disclosed for all pertinent system capabilities, limitations, and potential problems.

There are possible adverse effects to consider from any form of cryptanalysis. Information asymmetry could lead to societal consequences; specifically for those with the

ability to acquire or access a means to cryptanalysis tools. The adverse effects associated with code breaking are also tied with war and diplomacy — see Turing's work on the enigma cipher during World War II.

AES is used for encryption applications in many domains including high level branches of government and industries with strict privacy requirements such as medicine and law. Since the use of AES encryption is so ubiquitous, the primary ethical concern regarding research into the security of AES is the ramifications the disclosure of any vulnerabilities in the standard could have. Successfully finding a non-random ciphertext pattern in AES could lead to the exploitation of that vulnerability to compromise AES encryption. Depending on the severity and ease of exploitation of the discovered vulnerability, this could potentially expose the underlying message from any encrypted AES ciphertext.

For example, AES is used in internet protocol security (IPSec) applications such as transport layer protocol (TLS) 1.2 [11]. TLS is a protocol designed to facilitate secure avenues of communication for web applications. Since TLS relies on AES encryption for part of its underlying encryption measures, compromising AES also has the potential to compromise the security of any TLS security. TLS is used in many web applications including hypertext transport protocol secure (HTTPS) which is used to provide secure access to web pages and is crucial for many online transactions such as processing credit card payments or securely transmitting sensitive personal data.

It is important to consider that finding any vulnerability in the AES cipher has the potential to cripple secure transactions over the internet. This includes the risk that millions of personal or confidential documents could be made publicly available to attackers which could cause severe harm to people of the world both at a personal level and at a higher level as government or institutional databases are compromised. It is for these reasons that the publication of vulnerabilities within a cipher such as AES must be carefully considered in order to minimize the damage the discovery of these vulnerabilities might cause.

10 Conclusions

We performed the only known longitudinal study on over 200 billion of AES CTR and CBC blocks and found significant evidence that AES produces random values. We implemented classification models, based on either logistic regression or dense neural networks, with each model targeting specific bit locations in AES ciphertext blocks generated in CTR and CBC modes of operation. Models trained on ten million blocks predicted the bit assignment within an accuracy of 50% ± .01%. The dense neural network on thirty billion sequential CBC blocks predicted the bit assignment within an accuracy of 50% ± .00061%, which is to say they predict no better than random.

Repeating this analysis with a broken cipher would need to be done to verify the appropriateness of the logistic regression or dense neural network.

References

1. Dworkin, M.: 2001 Edition Cipher Modes of Operation Methods and Techniques. NIST Special Publication **800** 38A

2. Katz, J., Menezes, A.J., Van Oorschot, P.C., Vanstone, S.A.: Handbook of applied cryptography. CRC press (1996)
3. Blazhevski, D., Bozhinovski, A., Stojchevska, B., Pachovski, V.: Modes of Operation of the AES Algorithm. (2013)
4. Rogaway, P.: Evaluation of some Blockcipher Modes of Operation. Cryptography Research and Evaluation Committees (CRYPTREC) for the Government of Japan (2011)
5. Dierks, T., Rescorla, E.: The Transport Layer Security (TLS) Protocol Version 1.2. Technical report (2008)
6. Choudary, O., Gröbert, F., Metz, J.: Infiltrate the Vault: Security Analysis and Decryption of Lion Full Disk Encryption. IACR Cryptology ePrint Archive **2012** (2012) 374
7. Daemen, J., Rijmen, V.: The Design of Rijndael: AES-The Advanced Encryption Standard. Springer Science & Business Media (2013)
8. U.S. Department of Commerce/National Institute of Standards and Technology: FIPS PUB 197, Advanced Encryption Standard (AES). (2001)
9. Stallings, W.: Cryptography and Network Security: Principles and Practice. Pearson Upper Saddle River (2017)
10. Bassham, L.E., Rukhin, A.L., Soto, J., Nechvatal, J.R., Smid, M.E., Leigh, S.D., Levenson, M., Vangel, M., Heckert, N.A., Banks, D.L.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications | NIST. Technical report (2010)
11. Salowey, J., Choudhury, A., McGrew, D.: AES Galois Counter Mode (GCM) cipher suites for TLS. Technical report (2008)
12. Rescorla, E.: The transport layer security (TLS) protocol version 1.3. Technical report (2018)
13. Ferguson, N., Schneier, B., Kohno, T.: Cryptography engineering: design principles and practical applications. John Wiley & Sons (2011)
14. Buhler, P., Dykeman, H.D., Eirich, T., Kaiserswerth, M., Kramp, T.: Secure PIN Management of a User Trusted Device (June 2 2011) US Patent App. 12/783,210.
15. Barker, E., Dang, Q.: NIST Special Publication 800-57 Part 1, Revision 4. NIST, Tech. Rep (2016)
16. Waldrop, M.M.: The chips are down for Moore's Law. Nature News **530**(7589) (2016) 144
17. Johnston, D.: A Python Implementation of the SP800-22 Rev 1a PRNG Test Suite (2019)
18. Gotterbarn, D., Brinkman, B., Flick, C., Kirkpatrick, M.S., Miller, K., Vazansky, K., Wolf, M.J.: ACM Code of Ethics and Professional Conduct. (2018)