Breaking Small Variants of AES using Deep Learning Based Attack

Max Pilipovic-Kljajic , Zarif J Nafee, Zachary McCloud
Mathematics and Computer Science
Eastern Illinois University
{mppilipovickljajic, zjnafee, zamccloud}@eiu.edu

Abstract. This paper presents a new variation of Advanced Encryption Standard (AES). This variant is a smaller and simpler version which was created to replicate the 128-bit system. The smaller version uses an input block size and key size of 16 bits and 24 bits, using this model the 16-bit AES and 24-bit AES was attacked using a Neural Network aimed at targeting the secret key. Our goal is to show that deep learning based attacks on a smaller variant of AES, such as the 16-bit AES and 24-bit AES will have an improved computational complexity over a brute force method. Our results show…

Keywords: Advanced Encryption Standard, Smaller, Replicate, Attacked, Neural Network, Key

1 Introduction

https://nvlpubs.nist.gov/nistpubs/jres/126/jres.126.024.pdf

https://www.engr.mun.ca/~howard/PAPERS/ccece_2005a.pdf

In the modern world, security is the most important factor in transmitting and storing personal information across the internet. Therefore, the modern solution is Encryption which helps individuals secure and transfer their data safely between parties. There are many different algorithms that support this, one of which is the Advanced Encryption Standard which was started development in 1997 and was an alternative form of the Data Encryption Standard (DES), which was starting to become vulnerable to Brute Force Attacks. Two Belgian cryptographers, Joan Damen and Vincent Rijmen submitted a proposal to the National Institute of Standards and Technology during the AES selection process and in 2001 AES was announced by NIST and was adopted by the U.S government.

The original Advanced Encryption Standard allowed for block and key sizes of 128, 192 and 256 bits. The AES is based on rounds of byte-oriented substitution and linear transforms with a fixed block length of 128 bits. There are five main functions for the encryption process, namely byte substitution, rotation, linear transformation, key addition, and key expansion.

In this paper, we will propose a smaller variation of the AES and train a Neural Network to target our key. From this, we can gain insight into the algorithm and see if artificial intelligence can break the AES and uncover the key.

The algorithm proposed is a new variation of the original AES, the AES 32-BIT. This variation has a similar build to the original AES with smaller plaintext and key size (32-bit instead of 128, 196, 256-bit). By implementing a smaller 32-bit

version, compacting the algorithm and lowering the matrix size we have created a simpler version of the AES that still replicates the original version. Using this new variation, we can assume that all results found can apply to the original AES.

2 Literature Review and Critique

We take a look at previous attempts to recover the key from AES using various concepts and also look at attacks on other block ciphers using deep learning based attacks.

2.1 Improving attacks on round-reduced SPECK 32/64 Using Deep Learning

https://eprint.iacr.org/2019/037.pdf

In this paper Gohr shows that machine learning can be very useful cryptographic distinguishers. Machine learning has been useful in a number of areas in computer science, however, it is said to be not very effective in cryptanalysis because of the randomness it deals with. Machine Learning algorithms such as a neural network thrives in structured environments where it has something to learn. So far in cryptanalysis machine learning has only been used with side channel attacks. The main issue with this is for a side channel attack there needs to be some information leaked about the inside of the algorithm that will be attacked. Gohr claims that this paper is the first to show that neural networks can be used to produce attacks against a round-reduced version of a modern block cipher. He will do this by teaching the neural network to exploit the differential property of a round reduced SPECK.

SPECK is a block cipher very similar to how the Advanced Encryption Standard is also a block cipher. Block Ciphers are algorithms that encrypt groups of data called blocks to produce a ciphertext which is later decrypted to produce the data. This encryption and decryption is done using a cryptographic key. The difference between speck and aes is that speck does not use lookup tables to encrypt the block into ciphertext. The AES however uses the S-box to encrypt and decrypt.

The author of this paper, Gohr uses input differences as distinguishers for the outcome of his experiment. He trains a neural network to distinguish the output of the SPECK with a given input difference from random data. The attack will be based around this known input difference. This attack is effective because it doesn't require key search . The differential distinguisher attack of neural networks is also compared to a pure cryptanalytic differential distinguisher to show the effectiveness of attacking using machine learning. He launches multiple differential attacks on SPECK32/64 and gives a detailed description of the attacks, talking about how he set up the attack, why he used it, errors if any, the results and observation from the attack. He launched attacks with pure differential distinguishers and differential distinguishers using the full distribution of ciphertext pairs. After these he used neural networks to launch distinguishing attacks on the SPECK and showed results based on the best neuron models he had. The Neural Networks were trained with real and random classifiers. To improve the distinguishers that were obtained, Gohr used key search in a way so it acted as a teacher to work better with random data. It was noted that this training method was not successful as it only worked as a 8-round distinguisher.

Gohr ran further experiments where he did a key recovery attack to show how useful neural distinguishers can be by showing how the distinguishers have less complexity than a brute force attack does. He ran a real differences experiment where random and real difference distribution was the same. This allowed him to run differential cryptanalysis with the distribution and show that neural distinguishers are successful while also providing an efficiency comparison with a key search algorithm.

In the end Gohr succeeded in showing that neural networks can be useful tools in doing cryptographic research. He provided results and believes that his work can be further improved upon to find more interesting tools and techniques to do cryptanalysis with. An assessment which would be correct as his deep learning based attack would be broken down and simplified to get a better understanding of how machine learning can be used in cryptanalysis.

2.2 A Deeper Look at CryptoAnalysis

https://eprint.iacr.org/2021/287.pdf

In this paper, a deeper look is taken into Gohr's Deep Learning based attack using neural networks. They aim to show that machine learning can be a useful tool in cryptanalysis. Especifically by pre-checking a cipher and recognizing possible weaknesses, or patterns to be exploited. It could also check a round reduced version or a small scale variant of a cipher instead, in the same way. This will be very helpful to cryptanalysts. Machine learning has previously been used in side-channel analysis but the machine-learning for balck box analysis was not talked about until Gohr presented his article. As innovative as Gohr's paper was, it raised many questions, which this paper tries to answer. The most important of those questions was the interpretability of the distinguisher, how does the distinguisher work, what aspects of the cipher does it exploit that has not been exploited by previous cryptanalysts. In this paper, they try to improve upon his attack by making it more interpretable using simpler machine learning tools and also try to see what information is being used during the attack.

Gohr uses his trained neural networks to see if they can accurately distinguish the difference between real and random ciphertext pairs. The neural networks are trained with data from ciphering plaintext pairs with a fixed input input difference and from random values. After adding a key recovery process to this structure he was able to have the best known key recovery attack on a SPECK32/64 for a non-negligible number of rounds. IN this paper they carefully analyze Gorh's work and see if they can figure out what kind of cryptanalysis is his neural distinguishers learning. This is to see what causes his distinguishers to perform better than other attacks. They show that given a 5-round SPECK-32/64 ciphertext pair, N5 is able to determine the difference of certain bits at rounds 3 and 4 with high accuracy. Then they use a key rank distinguisher independent of any machine learning which almost matches the accuracy of Gohr's neural network for not only 5-round SPECK-32/64 but also 6 and 7 rounds. They compare the degree of closeness, how similar the classifications for the ciphertext pairs are for both, and they also compare their complexities. The results they received supported their hypothesis that "neural networks may

learn differential-linear cryptanalysis" which would mean "they have the ability to learn short but strong differential, linear or differential-linear characteristics for small block ciphers for a small number of rounds."

Then they try to tackle the problem of interpretability of the neural network. Knowing how the neural network is working in the black box, such as what aspects of the cipher it is exploiting would allow cryptanalysts to extensively study those weaknesses and further exploit them and improve the ciphers. To do this, Gohr's model for the neural network which contains three blocks, will each be replaced by a more readable block either by machine learning or pure cryptanalysis, and use the new distinguisher to see if they achieve the same accuracy. They run their experiments on SPECK and SIMON ciphers.

In their results, it is found out that in the attack not only is the ciphertex differential being looked at by the neural networks but so are the differentials in the penultimate and antepenultimate states in the block cipher.

By using a machine learning distinguisher that uses simpler tools and a pure cryptanalysis distinguisher this paper has attempted to provide answers to the mystery of what happens inside of th ebalck box, what information the neural network is using after the input to attack SPECK.

2.3 Output Prediction Attacks on Block Ciphers using Deep Learning

https://eprint.iacr.org/2021/401.pdf

The ability to use machine learning in attacks against modern day symmetric-key ciphers is very important. Linear/differential cryptanalysis requires some sort of knowledge on the architecture of the cipher it will be attacking. Deep-Learning on the other hand requires only the knowledge of algorithm interfaces such as the key and block sizes. This means that deep-learning base attacks in a blackbox setting is very strong and thus raises the need to make symmetric-key ciphers that are secure to these attacks The issue that is raised is the lack of information on what causes the attack to be so successful. Previously we saw the possibility of deep-learning base cryptanalysis and then the interpretability of such an attack to see how it works. In this paper we see what affects the probability of success of such an attack. According to this paper, previous work has not identified any exclusive deep learning characteristics that affect the success probabilities of the deep-learning base attacks. This paper provides such characteristics so they can be used to make deep-learning resistant symmetric-key ciphers in the future.

In this paper they launch a deep-learning based attacks on two SPN block ciphers (16-bit block variants of PRESENT and an AES-like cipher) and one toy Feistel block cipher (a type-II generalized Feistel structure with 4 branches called small TWINE) in a blacbox setting. Then they perform whitebox analysis using deep learning models to examine the connections between deep-learning based attacks and classical attacks such as linear/differential attacks. Then they change the block sizes of target block ciphers to 32 and 64 bits and do the same attack in a blackbox and apply whitebox analysis.

After running their experiments they were able to examine the relationship

between deep-learning based attack and classical attacks. The results they got were:

- For PRESENT, the maximum number of rounds that the proposed attack can be successful is at least equal to that of classical linear/differential attacks.
- For AES-like and TWINE-like ciphers, we conjecture that the maximum number of rounds 19 that the proposed attacks can be successful also becomes equal to that of classical linear/differential attacks when the amount of training data is increased more.

They also obtained a deep-learning specific characteristic that affects its success probabilities but not the success probabilities of a classical linear/differential attack. The characteristic being the swapping and replacing internal components of the cipher such as the S-box and bit permutation. This can now be used to make deep-learning resistant symmetric key ciphers for future work.

2.4 Small Scale variants of the AES

https://www.iacr.org/archive/fse2005/35570143/35570143.pdf

In this paper an algebraic attack is done on a small scale variant of AES. It is important to note that compared to other block ciphers where attacks were done on a round reduced version, the attack is not being done on a round reduced version of AES. Algebraic cryptanalysis is a two step process. The first step is converting the cipher system into a system of equations. Then the second step is to solve that system of equations to get a key or a plaintext. In the case of AES, the system of equations in a round reduced version differs from the system of equations that one would encounter in a typical 128-bit, 192-bit, 256-bit AES. To ensure that the system of equations are similar to those we expect to see in AES they use a small scale variant instead of a round reduced version. The paper analyzes these small scale variants and gives us insight on how algebraic cryptanalysis performs on the variants.

They define two variants with the main difference being in the final round. The variants are denoted as SR(n, r, c, e) and SR* (n, r, c, e). The parameters are defined similar to our AES.

They both had n rounds. Each block was a data block of "words" which are e bits. The data block was an array of size rxc. The big difference between these two variants would be in their final round where SR* (n, r, c, e) did not use a MixColumns in their final round and SR(n, r, c, e) did still use MixColumns. With these similarities and differences the solution from the system of equations of one cipher would give the solution for the system of equations for the other cipher. Using this generalization they use the variant SR(n, r, c, e) for the rest of the paper assuming finding a solution for one would mean finding a solution for both.

This uses [11] as inspiration to derive sparse multivariate quadratic equation systems over GF(2^e) for the small scale variants of the AES. They then ran experiments with simple variants SR(n, 1, 1, 4) and SR(n, 1, 1, 8) for different numbers of rounds. For the first they get better timings with GF(2), however for the second variant they get better timings with GF(2^e). They also ran timing experiments with a system of

equations from SR(n, 2, 1, 4) and SR(n, 2, 2, 4). Using the results from these they were able to claim that inter-word diffusion played an important role in the complexity of the computations.

The paper shows how they made the small scale variant of the AES and how they ran experiments to get a better understanding of the behavior of algebraic attacks on AES and allow for future work on these AES variants.

2.5 New Key Recovery Attacks on Round Reduced AES

https://eprint.iacr.org/2022/487.pdf

This paper has the first attack on a 7-round AES by exploiting the zero difference property. The zero difference property is "a new deterministic 4-round property in AES, which states that sets of pairs of plaintexts that are equivalent by exchange of any subset of diagonals encrypts to a set of pairs of ciphertexts after four rounds that all have a difference of zero in exactly the same columns before the final linear layer". They give a s impeller definition of the zero differenced property using the concepts of related differences and related differentials.

According to Damen and Rijmen related differences are a pair of differences $\Delta x , \Delta x' \in \mathbb{F}_q^n$ if and only if :

$$\Delta x \, \Delta x' \, (\Delta x \oplus \Delta x') = 0, \; for \; i = 0, \ldots, n - 1$$

Related differentiated can be combined into related differentials such as Two differentials $(\Delta x, \Delta y)$ and $(\Delta x' , \Delta y')$ for a linear map M are related differentials if and only if, $\Delta y = M(\Delta x)$, $\Delta y' = M(\Delta x')$, the differences Δx,$\Delta x'$ are related differences and the differencesΔy, $\Delta y'$ are related differences.

They redefine the zero-difference iproperty with related differences because there to now be related differentials in the zero- difference property. This will allow cryptanalysis for more rounds, whereas before the max rounds an attack was for 5-rounds now with the reduction using the concepts of related differences an drelated differentials in the zero-difference property the new generalization can be used to extend the attack to 6- and 8- round AES. Using this they were able to come up with a new key recovery attack for 7-round of AES- 128. The way the attack works is they start off with $2^{218.4}$ quartets which are two pairs of plaintexts generated by exchanging two diagonals ,in the block, between plaintexts.Then they collect data as a set of quartets that meets a certain condition. MEeting the condition would mean that they meet the differential characteristics they are looking for. The expected number of quartets is 2^{93} , Which they get from the original $2^{218.4}$ quartets by using hash tables to find them. Once we have the quartets that meet the condition they can then be used to recover key candidates.

This attack would be independent of any key scheduling, the key that is recovered during this attack will be the key only from the last round, when the MixColumn operation doesn't happen. The impossible-differential and the meet-in-the-middle attack, both attacks require the key schedule so they can exploit some of the relation between the round keys. This is good because the new key recovery attack proposed doesn't need any knowledge of the key scheduling in the attack compared to the two best attacks on 7-round AES-128.

2.6 Improved Key Recovery Attacks on Reduced-Round AES in the Single-Key Setting

https://eprint.iacr.org/2012/477.pdf

In this paper the authors present a new attack on 7-round AES of all versions, using ideas from Dunkelman, Keller and Shamir's Attack and improving upon it. Dunkelman, Keller and Shamir's attack is a differential attack. In the online phase first, they propose to look at the first and last subkeys randomly, and then find pairs of plaintext that follow the first and last rounds of the differential characteristics using a rebound technique. Using the characteristics they encrypt the pair for the intermediate rounds and then decrypt the last rounds. Then the encryption is checked to see if it belongs in a table where functions with no differential characteristics are stored. For the description of the functions, differential enumeration is used to have 16 parameters. This is a lookup table that takes a large amount of memory that is constructed in the pre computation phase. If the encryption does belong, then the subkeys are correct and the whole key is found by searching the other parts of the key in-depth.

The authors of this paper use much of the same idea but improve on aspects of it. An enumeration technique to enumerate the set of solutions for the table more efficiently and also reduce the parameters to 10. Once the table is constructed in the precomputation phase they move to the online phase where they first find pairs of messages that match the new differential characteristic, then make a δ-set with the pairs to test against the table and finally retrieve the whale key. The authors believe this to be the most efficient result from a key recovery attack on a singley key model of the AES-128

They also extended their key recovery attack to 8-round attacks on AES 192 and AES-256. The big difference here is that for each pair there is a key so the pair follows the differential characteristic. They tackle the problem by now enumerating the key bytes needed to identify a δ-set for each pair and summing the pair is correct, build a related multiset. The 8 round attack on AES- is also extended to a 9-round attack by adding a round in the middle.

2.7 Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds

https://eprint.iacr.org/2009/374.pdf

In this paper the authors describe a key recovery attack on AES-256 variants with different rounds. PRevious attacks on these variants have required a lot of related keys and time to break, however the attack the authors describe will do it in less time and require less related keys. The total runtime of an attack, the resources an attack takes can be used to measure the usefulness of a cryptanalytic attack. The best previously published attacks on a 9-round AES-256 variant required 4 related keys and 2^{120} time to break. The previous best attack on a 10-round variant of AES-256 took 64 related keys and 2^{172} times to break. The authors of this paper detail an attack in 2^{39} time by using only the simplest type of related keys for a 9-round AES-256 variant and only two keys and 2^{45} time, using a stronger type of related subkey attack, for a 10-round AES-256.

The attack they do on the 9-round AES-256 variant encrypts plaintexts with 2 unknown keys. The attack chooses to

exploit a relation between the keys. The relation in most cases comes down to a simple XOR difference. For the attack they use a differential characteristic which is a combination of a key and a state characteristic to generate the desired subkey differences that can be used to recover the whole key. They generate 2^{38} input pairs in 2^{39} time which is the most time consuming part of the attack. Then they filter the pairs to get the subkeys and lastly get the remaining parts of the subkeys.

The attack on the 10-round variant of AES-256 is very similar to the attack on 9-round with the big difference being the addition of a stronger related key attack. This is to exploit the key relation from the fixed difference on two consecutive subkeys. The authors also run a Related-Key Attack, Related-Subkey Attack on 8 rounds and Related-Subkey Attacks on 11 Rounds where the attack is possible but the time complexity is not the desired. All the attacks they do on the variants of AES-256 get subkeys but not the whole key, they measure the time by using the number of AES encryptions. They end the paper with the cryptanalysis that the AES-256 had a problem with their key schedule which can be exploited to retrieve keys or at least big parts of the key.

2.8 Automatic Search of Attacks on round-reduced AES and Applications

https://eprint.iacr.org/2012/069.pdf

This paper takes a look at round reduced AES variants in a practical security model and describes an algorithm that can search for the attack necessary based on some byte-oriented symmetric primitives. The attacks they search from are guess-and-determine and meet-in-the-middle. This itself is a sort of algebraic attack as it takes in a system of equations that describes the cipher and then solves the system of equations. Here is when the search is implemented to find an attack that will solve the system of equations after which the attack takes place and the solution to the system of equations is computed.

2.9 Deep Learning-based Side-channel Analysis against AES Inner Rounds

https://eprint.iacr.org/2021/981.pdf

This paper looks at side channel attacks done on AES in their inner rounds. The inner rounds are expected to be the least protected which is why they will be targeted by authors in this attack. They show that a deep-learning based side channel attack will be unaffected by the complexity of the non-linear AES structure and is an improvement against the limitations other attacks might face. A neural network base profile attack can work through the countermeasures in the cipher and get the key. QA non profiling attack can be hindered by the limitations causing the complexity to get worse.

2.10 Biclique Cryptanalysis of the Full AES

https://eprint.iacr.org/2011/449.pdf

In this paper, the authors introduce block cipher cryptanalysis using bicliques. Compared to other key recovery attacks this doesn;t need to assume any related keys. The attack they describe here is a meet-in-the-middle attack with the use of bicliques. They claim the attack to be computationally infeasible since it had to carry ou tth eattack in full. They propose

the concept of bicliques with meet-in-the-middle attacks. Meet-in-the-middle attacks are not as popular as differential, linear, impossible differential and integral attacks due to the limitations that the attack has. But the attack is useful because of its data complexity which allows the attacker to know information-theoretical minimum of plaintext-ciphertext pairs. When paired with bicliques this attack was used as a key recovery attack on all versions of full AES with faster times than a brute force method.

It is important to see how a meet-in-the-middle attack would work to understand why combining the concepts of bicliques makes it so strong.The attacker first chooses a partition of the key space into groups of keys and indexes each key in the group. The attack works with an internal variable in the data transform of the cipher. They use a plaintext-ciphertext pair to compute values for the internal variable and if the values match then that yields a key candidate which is one of the previously indexed keys. The number of candidates depends on the absolute value of the internal variable. The issue is that this internal variable with the properties to do what the attacker wants is only possible in a small number of rounds only.

This is where the concept of biclique comes in where instead of internal variables the authors use internal states. The preparation is the same with partition key space into groups of keys and then builds a ciphertext for each group, so when decrypting said ciphertext with a key candidate it yields the internal state. The same ciphertext is then decrypted with the actual key to get the plaintext. Then lastly the key candidates are used with the internal state to see if it yields the plaintext, if so then that is the actual key.

The authors propose two approaches for constructing the biclique, the first approach using Independent Related-Key Differentials. This makes the partitioning of the key space straightforward and high dimensional, although limited in length, bicliques. The second approach uses Interleaving Related-Key Differential Trails, which should be used in single-key models. They also give a full description on long bicliques and independent bicliques to compare and contrast the strength and weakness of both.

Using this cryptanalysis on all versions of full AES has proved to be fruitful, as they were the first key recovery method with the following low complexities,

- Full AES-128 with computational complexity $2^{126.1}$.
- Full AES-192 with computational complexity $2^{189.7}$.
- Full AES-256 with computational complexity $2^{254.4}$.

And with faster preimage search for compression functions than brute force

- Full AES-128 with computational complexity $2^{125.8}$ with success rate 0.632.
- Full AES-192 with computational complexity $2^{125.7}$ with success rate 0.632.
- Full AES-256 with computational complexity $2^{126.3}$ with success rate 0.632.

2.11 Yoyo Tricks with AES

https://www.iacr.org/archive/asiacrypt2017/106240276/106240276.pdf

This paper presents key-independent yoyo-distinguishers for 3- to 5-rounds of AES to show the usefulness of SPN properties. The authors apply yoyo based cryptanalysis to generic SPNs and come up with a SPN distinguisher that according to the authors is purely structural and immediate. This distinguisher is a new key independent secret key distinguisher. They try this distinguisher with AES for 4 and 6 rounds, then 3 and 5 rounds and then use it for a secret key recovery attack for 5 rounds.

SPNs are formed by a round function that has concatenated layers of S-boxes and generic affine linear transformations. The authors use a yoyo distinguisher and generic zero differential property for 3-round SPN, to be able to differentiate between a pair of plaintext or ciphertext and use it to make more pairs with the same zero difference. The idea is then applied to an AES where the round function is a 4x4 matrices over $\mathbb{F}_q$ where $q = 2^8$ and two rounds of AES means one S-box layer, that consists of 4 parallel superboxes, and a large linear super-mix layer. Observe that this is similar to one generic SPN round. This would mean that every one generic SPN round would correspond to two rounds of AES. They work around this with 3 rounds of AES by not counting the linear layer before and after the last s-box layer and use the distinguisher. They also use the yoyo distinguisher for 4 and 5 rounds of AES. When using a yoyo distinguisher for 6 rounds of AES they also have to use an impossible differential concept combined with the yoyo distinguisher.

In this paper the authors do a 5-Round Key Recovery Yoyo on AES, where determining the full round key depends mainly on guessing the first subkey. It is important to use 5 pairs of plaintext for this attack for input and since recovering the first subkey, which is the dominating portion, costs roughly 2^{31} ^{5}-round AES encryptions and $2^{11.32}$ adaptively chosen ciphertexts and plaintexts, the attack in its entirety requires only $2^{11.3}$ plaintexts/ciphertexts and 2^{31} computations.

3 Architecture

The AES 32-BIT supports block and key sizes up to 32-bits, this allows the algorithm to break up plaintext into a 2x2 state matrix with slices [2, 2, -1]. Our AES is a SPN so we will substitute, permutate and then add in a key every round. Following this, comes four main byte substitutions; first is byte substitution which is a lookup table of an array of 64-bytes, each value corresponding with a different value. Second is shift rows, which each row number corresponds with the amount to rotate to the left. For example, row 0 has no left rotation, row 1 rotates left once and row 2 rotates left twice. Third is mix columns, where each value is multiplied by another value using the S-Box and Inverse S-Box. The final round is XORING the output with the round key, this is referred to as adding the round key. The AES 32-BIT only processes one full iteration for each encryption. This allows for simple implementation, which results in a 32-bit ciphertext. We define the AES parameters as (n,r,c,e).

n - number of encryption rounds.

r - number of rows in the states 2 text input.
c - number of columns in the states 2 text input.
e - key size in bits

The Advanced Encryption Standard has n rounds and a block size of rce bits. Where, (r x c) is the data block size of states2text and e is donated as the key size in bits.

N - Numbers of Rounds
The number of encryption rounds for the default AES is 11. This consists of subbytes, shiftrows, mixcolumn and addroundkey. For small scale variants, the number of rounds can range from $1 \leq n \leq 10$. In this project, we will be using 1, 3 and 5 rounds.

R & C- Number of Rows and Columns (Data Block Size)
The (r x c) refers to the data block size, the array itself has r rows x c columns. For the small scale variants we used 2 x 2 and 3 x 3. The default AES has 32 x 32 which is a total of 128 bits.

E - Key size in bits
The key size is defined by the size of the array (r x c). For our small scale variants, the 2x2 and 3x3 input state text. We use a key size of 4 and 3 respectively, these sizes are straightforward and allow us to perform experiments on reduced AES while still receiving accurate results.
The AES consists of the following operations.

1. SubBytes
2. ShiftRows
3. MixColumns
4. AddRoundkey

SubBytes - Byte Substitution uses a S-Box which is a lookup table of an array of 64-bytes, each value corresponding with a different value. The Inverse S-Box is used for decryption which is the reverse of the original S-Box.

ShiftRows - Shift Rows, uses an operation where each row number corresponds with a simultaneous left rotation. For example, row 0 has no left rotation, row 1 rotates left once and row 2 rotates left twice. This is an iterative process where we begin to permute each byte.

MixColumns - The MixColumns is the most expensive operation and provides the most security. The operation multiples each column with a constant column which is a constant 2x2 or 3x3 matrix. We get this matrix from the Galois Field or $GF(2^n)$. The operations that take place in the finite field are not simply addition and multiplication. The addition is a XOR operation and the multiplication is a Polynomial multiplication modulo polynomial on this field. This happens in all the rounds except the last round.

AddRoundKey - The AES begins with an initial addroundkey, the transformation in the Cipher and Inverse Cipher in which a Round Key is added to the State using an XOR operation.

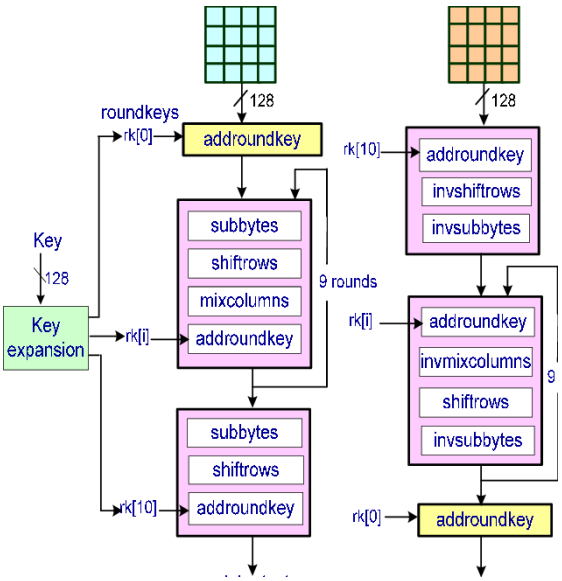

Figure 1: Architecture of 128-bit AES

The architecture of the original 128-bit AES is shown in Figure 1. The difference between the original AES and the 32-BIT AES is block size (32 and 128) and the amount of iterations processed, the original AES has 10, 12 and 14 iterations depending on the block size; the 32-bit only has one iteration. Therefore, its main difference is key exhaustion (trying every possible combination until you find one that works).

4 Neural Network

The neural network requires input from the 3x3 or 2x2 AES implementations, which give a bin file with the key (in binary) being the file name. The code in the notebook separates the 32-bit plaintexts and ciphertexts into two columns, and then combines the data frames representing the binary data. Currently the code only combines two bin files, but more can easily be added individually by calling the bin_to_dataframe function, with the file name as the argument. This results in a single data frame with three columns; plaintext, ciphertext, and key. The plaintext and ciphertext are stored as their decimal integer equivalents, while the keys are kept in binary as strings. The data is then split into training and testing data.

The network itself currently consists of an input layer of 100 neurons, hidden layers of 50 and 20 neurons, and an output layer of 2 neurons. Dropout is currently not used, as the produced training data is very large. The optimization function used is Adam, while the possible learning rate, activation function, epochs, and batch size are stored in params, which are tuned using RandomizedSearchCV. Once this is finished (it is very time-consuming) it will report the best performing combination of parameters, which can then be placed for the final model training.

Our goal with training the neural network is to use it to recover the whole secret key, or bits of the secret key with less computational complexity than the brute force method. This would show that deep learning cryptanalysis can be a very useful tool.

5 Conclusion

In this paper we have implemented a 2x2/ 16-bit AEs and a 3x3/24 bit AES. Then we trained our neural network using input from the implementations, which will then be used to retrieve a secret key from those implementations of AES. As of right now we were not able to finish the training and get quantitative results. We hope that future work can be done on this and it can be shown that deep learning based cryptanalysis is more efficient than brute force attacks when it comes to AES.