Security analysis and new models on the intelligent symmetric key encryption

Lu Zhou, Jiageng Chen, Yidan Zhang, Chunhua Su,
Marino Anthony James

PII: S0167-4048(18)30964-7
DOI: https://doi.org/10.1016/j.cose.2018.07.018
Reference: COSE 1387

To appear in: *Computers & Security*

Received date: 25 February 2018
Revised date: 16 June 2018
Accepted date: 7 July 2018

Please cite this article as: Lu Zhou, Jiageng Chen, Yidan Zhang, Chunhua Su, Marino Anthony James, Security analysis and new models on the intelligent symmetric key encryption, *Computers & Security* (2018), doi: https://doi.org/10.1016/j.cose.2018.07.018

Highlights

- Reinvestigate the neural network based data protection solution and improve the model.
- Investigate its security based on the designated statistical methods.
- Study the learned function shape of the underlined neural network.
- Several models with stronger adversaries are proposed, and the system efficiency is investigated.

Security analysis and new models on the intelligent symmetric key encryption

Lu Zhou[a], Jiageng Chen[b,*], Yidan Zhang[b], Chunhua Su[a], Marino Anthony James[b]

[a] *University of Aizu, Japan*
[b] *Central China Normal University, China*

Abstract

Data protection is achieved in modern cryptography by using encryption. Symmetric key cryptography is mainly responsible for the actual user data protection in various network protocols such as SSL/TLS and so on. The design of such encryption algorithms have always been one of the most important research targets, where heavy cryptanalysis works have been performed to evaluate the security margin. As a result, the research community is busy with fixing the security flaws based on the cryptanalysis results. Recently, the idea of building the automatic security protection scheme based on the neural network has been proposed. The encryption algorithm, which is a neural network is instead constructed by machine during the learning stage in an adversarial environment. This is a totally different approach compared with our current design principle, and could potentially change our understanding about how the (symmetric key) encryption works and what is the security requirement for the scheme. In this paper, we investigate the security of the underlined scheme which remains unexploited based on several statistical models. And furthermore, we strengthen the automatic encryption schemes by introducing much stronger adversaries. Our results showed that the security solutions based on the advanced deep learning techniques may start to play an important role in the future related directions.

☆This is the full version of the conference paper which was accepted by ISPEC2017.
*Corresponding author
Email address: chinkako@gmail.com (Jiageng Chen)

Keywords: neural network, generative adversarial network, statistical analysis, security models, tensorflow

1. Introduction

Security solutions are widely deployed in our daily digital communication through different types of networks. Complicated security protocols such as SSL/TLS, HTTPS and WPA family in the wireless network have already been used heavily to protect our data transmission. However, the security patches are continuously being applied to these security solutions due to the security flaws being discovered. Furthermore, the security protections of the relatively less mature areas such as IoT which requires lightweight security solutions are far from being satisfied. Provable security while has been studied thoroughly in recent years can be applied to only limited small primitive protocols. Complicated real world security solutions usually do not enjoy the benefit from the provable security, and many serious flaws have been identified such as [1], [2] and [3]. As a result, this motivates the new protocol standard discussion and its analysis [4]. Recently, a new security solution to facilitate the secure tunnel techniques was proposed in NDSS [5], and the later analysis showed that the original scheme cannot be proven secure unless modified [6]. This example very well demonstrated the various options the researchers have to make when designing the security algorithms. In order to address the security model, and to make the scheme practical, usually only computational bounded adversaries are considered in the design stage. Under this big assumption, lots of security models are proposed such as the semantic security [7], chosen plaintext attack (CPA), chosen ciphertext attack (CCA) and so on in theory. Each cryptographic schemes are analyzed under these different models, which are linked to some mathematical hard problems.

In 2016, Google Brain team proposed the idea [8] to build encryption scheme automatically without manually designing any concrete algorithms. In this new direction, all parties joining the computation are well trained neural networks.

In order to achieve the security goal, the core idea of the paper is to introduce the adversarial neural networks, and let it compete with the legal users. This however is not a brand new idea, which has appeared in several previous works [9], [10], [11]. It shares the similarity to the well known model called generative adversarial network (GAN), which was originally proposed to determine whether a sample value was generated by a model or drawn from a given data distribution. As we will see in Section 2 that the introducing of the adversarial neural network is the key part to make the communication secure from the eavesdropper. Before [8], there are several other works which took advantage of the machine learning techniques to protect communication such as [12], [13], [14], which focused more on the other fundamental issues such as how the secret keys can be established and so on.

In the modern symmetric key cryptography, heavy cryptanalysis efforts such as differential attack [15], linear attack [16] and their various extended versions such as impossible differential attack [17], zero correlation attack [18], boomerang attack [19] and so on have been applied to evaluate the security margin of the cipher. For example, based on the boomerang attack, [20] proposed the full round AES attack which relies on the ladder switch technique that is specifically related to the design of AES as well as its key schedule algorithm. Thus most of the attacks when performed in detail are limited to the specific algorithms, or at least the general structures of the cipher (iterative ciphers) should be specified. The work [8] provided a whole new insights on how to construct security primitives in an intelligent way. The intelligence here especially denotes that the as long as the adversary's general goal to attack the encryption scheme is defined, then the scheme will be constructed automatically after the training stage to defend against such attacker. As a result, we do not need the cryptanalysis work anymore in theory, since the neural network has learned to achieve the goal in the competitive training phase.

However, a lot of questions still remain untouched such as the security margin of the underlined scheme as well as how the model will behave under other much more stronger adversaries. In this paper, we follow the work of [8] by

first providing a thorough security analysis on the scheme to gain a better understanding of the security margin. Some designated statistical models have been built for our analysis task, and then we try to recognize the structure of the learned algorithm. Also other encryption protocols are proposed in this paper, which are evaluated under the different and stronger adversarial models. We discussed the key recovery models as well as the key or plaintext leakage models. Finally, we try to enhance the encryption scheme by introducing two adversary models simultaneously.

This paper is organized as follows. In Section 2, we briefly introduce the work [8], reproduce and improve their results along with other preliminaries. In Section 3, we investigate the security margin by using different statistical models, and study the shape of the automatically learned function. In Section 4, other models are discussed including the key recovery model, leakage model and the dual adversary model. And finally we conclude our paper in Section 5.

2. Preliminary

Martin Abadi and David G.Andersen's paper [8] presented a multi-agent system, where the communication between the two parties needs to be protected. The setting is very similar to the symmetric key encryption, where there are two legal parties Alice and Bob who want to communicate with each other. And there is an adversary Eve who wants to understand the content of the communication without knowing the secret key that is shared between Alice and Bob. In their paper, the ability of the adversary Eve is limited to knowing the ciphertext generated by Alice only. So it is a ciphertext only attack (COA) from the point of view of the traditional security models, which is the among the weakest models regarding the attacker's ability. Different from the traditional symmetric key cryptography, all the participants in this protocol are neural networks including the adversary Eve. Each neural network has its own purpose, which is expected to be achieved after the training stage. For example, the goal of Alice and Bob is to minimize the distance between P and P_{Bob}, which are

the input and output plaintexts by Alice and Bob respectively. In order to train the model to be useful, an adversarial neural network has to be introduced, which is Eve in our setting. In the training stage, Eve's goal is to minimize the distance between P and P_{Eve}. As a legal party, however, Alice and Bob would like to maximize the plaintext distance, which is obviously against Eve's goal. The competition mechanism introduced here borrows the idea from the generative adversarial network (GAN) [9]. Notice that no specific algorithms are specified beforehand, so our protection goal is rather flexible which depends on the adversarial neural network. The model is shown in Figure 1. By using Tensorflow [21], we rebuilt their encryption systems and the results are shown in Figure 2.

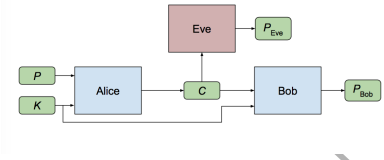

Figure 1: Original model in paper [8]

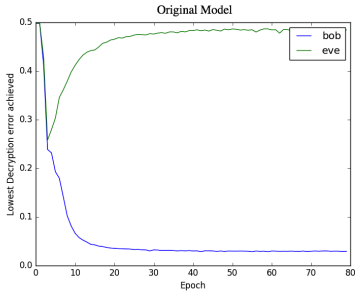

Figure 2: Batch size: 1024, epochs: 80

The result we derived is similar to [8]. The x-axis denotes the training epoch with batch size 1024. The y-axis denotes the decryption error which is a little bit different from the work [8]. The decryption error is calculated by using the mean of the differences in every output bit. Therefore we would expect the decryption error of Eve to be 0.5, which indicates that she only guesses the result randomly. On the other hand, we would like the decryption error of Bob to be as close as possible to 0 to guarantee the correctness. As we can see from the experimental result, as the number of training steps increases, Bob can successfully decrypt messages. Eve has some advantages in decrypting the ciphertext at the beginning, but soon Alice and Bob changed their strategies to

better defeat Eve such that she does not have any advantages in the end.

This competitive model is designed to let Alice and Bob achieve the confidentiality function while it can defeat the adversary Eve who has the ability of eavesdropping the information channel. While the original model managed to achieve this goal, it remains unknown how well is the encryption scheme when facing other kind of adversaries. Furthermore, in [8], the adversary Eve's ability is only passively eavesdropping the ciphertext, and try to recover the original plaintext without knowing the secret key. However, there is no guarantee that it can resist against other attacks such as the statistical analysis or even more sophisticated analysis. In other words, no security proof is provided and we are more curious on how the automatically generated encryption scheme based on the neural network behave against various analysis. This paper will investigate these aspects in depth.

One thing needs to be clarified is that the security margin is evaluated based on the bit level instead of the whole block size like in the modern cryptography, which is partly due to the implementation issue. Recall that during the training stage, Bob calculates the distance between the derived plaintext and the correct one, and try to minimize it. Here the distance is computed based on bit level. If we apply to the whole block size, then it will become computationally difficult to train the model to become stabilized, and meanwhile, it is not easy for us to observe the loss error evolving result during the training phase.

3. Security analysis of the google's model

In this Section, we put our main focus on the statistical behavior of the scheme. We have to point out that in the original paper [8], the authors did not put their effort in making the cipher indistinguishable from a random chosen one, which is required by the modern block ciphers. So the fail of the following analysis did not violate the original motivation of the paper.

First we would like to know the randomness of the output generated by the neural network. In order to be consistent with the binary format in the tra-

ditional symmetric key encryption, outputs are limited to the range between $[-1, 1]$, but the value in between is permitted. Before diving into the security analysis, we first discuss the scheme's communication efficiency. Since the ciphertext is encoded between $[-1, 1]$, the ciphertext domain is potentially larger than the traditional ciphers. When we encode using 32-bit floating point number for the neural network output, we are actually mapping one-bit message to 32-bit floating point number, which adds to the communication burden, and how to maintain the efficiency of the communication while maintain the functionality is a future work. We could base our analysis on the straightforward encoding: $x \in [-1, 0]$ to be 0 and $x \in (0, 1]$ to be 1, but unless the related neural network is supported for the corresponding decryption operation, massive information will be lost, which is not suitable here. Since we are analyzing floating-point number, we would like to take the following two approaches: a). χ^2 approach and b). Kolmogorov-Smirnov (KS) approach.

The χ^2 approach. Suppose we divide the $[-1, 1]$ into k equal intervals, and denote p_i be the probability that the observed ciphertext in one bit location will be in the i-th interval. Denote θ_i^0 to be the theoretical probability in the i-th interval that we would like to compare with. In our setting, $\theta_i^0 = 1/k$ because of the ideal uniform distribution that we expect from the output ciphertext. And we would like to study the following hypotheses:

$$H_0 : p_i = \theta_i^0, \text{ for } i = 0, 1, ..., k-1$$
$$H_1 : p_i \neq \theta_i^0, \text{ for } i = 0, 1, ..., k-1$$

Assume N_i is the ciphertext counter that locates inside the i-th interval, and $\sum_{i=0}^{k-1} N_i = n$, and it is known that the following statistic follows the χ^2 distribution with degree $k-1$.

$$Q = \sum_{i=0}^{k-1} \frac{(N_i - n\theta_i^0)^2}{n\theta_i^0} \sim \chi^2(k-1)$$

Notice that in order for the χ^2 distribution to work well, we will need $n\theta_i^0 > 5$. This can be easily achieved since the number of ciphertexts we collected is linked to our computational bound which will be around 2^{20}, while the number of intervals will be no more than 2^{10}. The experiment will be carried out by setting a significant level α, then we can derive the constant c which is the $1-\alpha$ quantile of the χ^2 distribution. If the test statistic is greater than c, then H_0 should be rejected, which indicates that the ciphertext output by the neural network is not uniform distributed.

In our experiment, the neural network model is first trained as described before until it is stabilized. And then we randomly generate 25600 16-bit messages and keys and feed them to the Alice model to derive the outputs. We do not distinguish between different bit locations, namely, bit order is not taken into consideration. Figure 3 shows the distribution of the bit value in the range $[-1, 1]$. Roughly a bell shape curve is demonstrated which shows the failure of the random distribution. In Figure 4, this phenomena is further supported by the χ^2 test data, where the quantile given $\alpha = 0.05$ equals 123.2, and $Q = 37105.72$ which is far right to the rejection region.

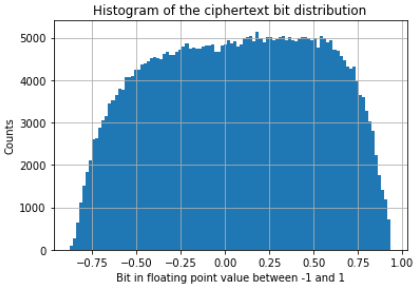

Figure 3: bit value distribution

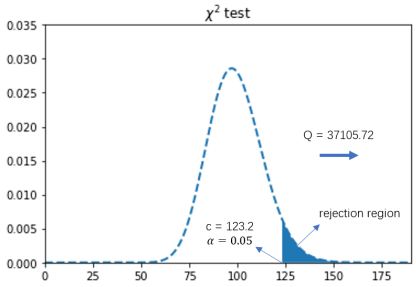

Figure 4: χ^2 test for Alice's output

The Kolmogorov-Smirnov approach. The χ^2 approach requires to divide the range ($[-1, 1]$) into k intervals and thus we need to fix the parameter k beforehand. The Kolmogorov-Smirnov approach (KS approach) on the other hand does not require such setting. The goal of this approach is to analyze whether the observed ciphertext distribution follows the uniform distribution

or not. In the KS approach, we will compute the sampled c.d.f $F(x)$ of the ciphertext and test if it is from the uniform distribution $F^*(x)$. The hypothesis testing is arranged as follows:

$$H_0 : F(x) = F^*(x), \text{ for } -\infty < x < \infty$$
$$H_1 : F(x) \text{ follows some distribution other than } F^*(x)$$

Denote D_n^* the maximum difference between $F(x)$ and $F^*(x)$ as follows:

$$D_n^* = sup_{-\infty<x<\infty}|F_n(x) - F^*(x)|$$

It is easy to see that D_n^* will tend to be small if the null hypothesis is true. The important theorem given by A. N. Kolmogorov and N. V. Smirnov back in the 1930s provides us with the criteria on how to reject the H_0. It states that if H_0 is true, then for each given value $t > 0$, we have the following:

$$lim_{n\to\infty} Pr(n^{1/2} D_n^* \leq t) = 1 - 2\sum_{i=1}^{\infty}(-1)^{i-1} e^{-2i^2t^2}$$

We will reject H_0 if $n^{1/2}D_n^* \geq c$. By checking the corresponding table, we can easily derive that $Pr(n^{1/2}D_n^* \geq 1.36) = 0.05$. As as result, if we choose the significance level α to be 5%, then the reject region should be $[1.36, +\infty]$.

We collect 409600 ciphertext bits in floating point value between -1 and 1, sorted in ascending order. Also we compute the CDF of uniform distribution in the range $[-1, 1]$, which can be derived by computing $\frac{x+1}{2}$. The largest distance D is found when x is moving from index 33916 to 33917, and the distance value is 0.118, which however does not locate in the rejection region. According to the Glivenko-Cantelli Lemma, the distance D will move towards the sampled CDF if n is large, which is the case in our experiment. However this method has its own limitation since discrete distribution addressed by n discrete points is not an accurate way to address the corresponding continuous distribution.

The NIST statistical test approach. Besides from the above two specific methods designated for the underlined neural network model, we would like to

backup with the NIST statistical tools to our further investigation. However, we have to notice that testing the randomness of the floating point number is far more complicated than integers, and the accuracy may suffer due to the methods we apply. Since the NIST statistical test suit only deals with integers, we need to transfer the floating point number to integer first and then we can apply the NIST statistical test suit [22] . To reserve the randomness of the double float number maximum, we try to transfer the 32-bit floating point numbers to integers by capturing the significand field. The transformed value C is in the integer format and thus are ready to be tested by the NIST test suit. After the neural network is stabilized after the training stage, we randomly feed the neural network Alice with 16-bit key and 16-bit plaintext and record the corresponding output ciphertext. By applying the integer transformation, we collect 10000 bits ciphertext for the statistical test. The report is shown in Figure 5.

```
------------------------------------------------------------------------------
RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES
------------------------------------------------------------------------------
   generator is <data5051143.dat>
------------------------------------------------------------------------------
 C1  C2  C3  C4  C5  C6  C7  C8  C9 C10  P-VALUE  PROPORTION  STATISTICAL TEST
------------------------------------------------------------------------------
 10   0   0   0   0   0   0   0   0   0  0.000000 *    5/10   *  Frequency
  3   3   0   1   1   0   1   0   1   0  0.213309     10/10      BlockFrequency
 10   0   0   0   0   0   0   0   0   0  0.000000 *    5/10   *  CumulativeSums
  9   1   0   0   0   0   0   0   0   0  0.000000 *    6/10   *  CumulativeSums
 10   0   0   0   0   0   0   0   0   0  0.000000 *    0/10   *  Runs
 10   0   0   0   0   0   0   0   0   0  0.000000 *    0/10   *  LongestRun
 10   0   0   0   0   0   0   0   0   0  0.000000 *    1/10   *  Rank
  5   1   1   0   1   2   0   0   0   0  0.008879      7/10   *  FFT
  5   1   1   1   0   0   0   0   2   0  0.008879      9/10
NonOverlappingTemplate
  3   1   0   1   2   0   1   1   1   0  0.534146     10/10
NonOverlappingTemplate
  0   2   0   3   1   0   1   2   1   0  0.350485     10/10
NonOverlappingTemplate
  9   0   0   0   0   0   1   0   0   0  0.000000 *    1/10   *
NonOverlappingTemplate
 10   0   0   0   0   0   0   0   0   0  0.000000 *    0/10   *
NonOverlappingTemplate
  0   0   1   0   0   2   3   2   1   1  0.350485     10/10
NonOverlappingTemplate
  0   1   1   2   1   1   2   1   1   0  0.911413     10/10
NonOverlappingTemplate
 10   0   0   0   0   0   0   0   0   0  0.000000 *    0/10   *
```

Figure 5: NIST's statistical result on the original model

From the test result shown in Figure 5, we find that the proportion of passing for some of the tests is very low. For example, in our experiment with 50% chance we failed to pass the frequency test, which tests the ratio between the number of 0 and 1. In order to confirm our test, we exploit the number of 0

and 1 in the sample data, and surprisingly, we found out that the ratio of 1 and 0 is a fixed number 0.942. As long as the hyper-parameter of the neural network remains unchanged, the ratio is not changed no matter how many times we trained the neural networks using various random inputs.

We try to figure out the reasons that caused the phenomenon by adjusting the hyper-parameter of the convolutional neural network (CNN) that is being used. We set the concrete layers to be [3, 1, 2] (filter size is 3×1, generate 2 outputs), [3, 2, 3], [2, 3, 4], [1, 4, 1], with the same strides as the original one, which is 1, 2, 1, 1. Connected with a full connection layer, the first CNN layer was slided by a 3×1 size filters window and generated 2 outputs. Due to our new setting, the passing rate of the statistical frequency test improved to around 80% while Bob can still remain a high decryption successful probability. But still we are not able to pass all the statistical test like we can achieve in testing the block ciphers. Thus the trained model is far from being statistically indistinguishable, which is required by modern security framework. How to improve the model to close the gap is worth further investigation. The observation also shows that the statistical behavior of the output largely depends on the structure of the underlined neural network. To achieve better statistical randomness, we need to evaluate the relationship between the network and the statistical behavior in depth.

The function shape. In the competitive learning process, the neural network of the legal parties learned how to protect from an eavesdropping attacker Eve. However, the training process is very intelligent and highly automatic, thus we do not control what kind of functions to learn. Compare to the modern security framework where concrete algorithms are specified, we are also interested in knowing the learned function. The first approach is to investigate the structure of the neural network, figure out all the relationship between all the neurons among different layers. Another approach is to match the encryption pattern with the currently well known encryption strategies to study the compatibility. The first approach is complicated requiring to figure out the logic connection in the neural network. Here we prefer the second approach because it is a rela-

tively easier to test, and most importantly, we would like to know if our human invented strong security solutions are also preferred ones chosen by machines, since we have already seen that the strategies that being used by our humans in the Go game are obviously not the best solution from AlphaGo's point of view.

We design the following framework for the further investigation. We randomly choose n-bit message and n-bit key as the input to two models. **Model 1** is our automatic encryption approach which is processed by the neural networks. And **Model 2** is the human invented security approach which also accepts message and the key as input. Both models output equal length n-bit ciphertext. We encode the output as bit-oriented format and we would like to observe the distance between the two patterns. One of the straightforward way is to use the XOR operation since the same bit value will be cancelled out. Here we use X_i to denote the random variable of i-th bit of the output by **Model 1**, and use Y_i to denote the random variable of i-th bit of the output by **Model 2**. If **Model 1** and **Model 2** are unrelated to each other and random, then we can reasonably assume that $X_i, Y_i \sim Bernoulli(0.5)$. Let $Z_i = X_i \oplus Y_i$, and it is easy to see that $Z_i \sim Bernoulli(0.5)$ as well. We investigate the statistic $T = \sum Z_i$, which follows the binomial distribution $T \sim B(n, 0.5)$. The statistic T is actually the number of ones in the final xored result. And it is also easy to see that as n becomes large, T can be approximated by using normal distribution $T \sim N(n/2, n/4)$. On the other hand, if the two models are related, it is highly unlikely that the statistic T will follow the same normal distribution. Thus we can summarize into the following hypothesis testing:

$H_0(unrelated) : T \sim N(n/2, n/4)$

$H_1(related) : T$ does not follow the normal distribution $N(n/2, n/4)$

Actually the distribution can be quite easy to identify if the two models are related. In this paper, we mainly discuss the one-time pad encryption model and assume it to be the **Model 2** since it is one of the most secure and widely used encryption solutions. Under this scenario, if **Model 1** is indeed the XOR logic, then the distribution of T will be highly skewed towards the left-hand side, since

0 will dominate the distribution. This framework can be easily adapted to test other existing encryption solutions by replacing **Model 2** to the corresponding algorithms. Figure 6 shows the structure of our test model.

By following the above experiment procedure, we collected the experimental data as well as the theoretical distribution. As shown in Figure 7, the distribution of the number of ones in the output is demonstrated by histogram as well as the corresponding fitting norm curve. We also add the theoretical distribution under Hypothesis H_0, and it turns out to be that the two curves are very close. From Figure 7, we can see that Model 1 follows the Normal distribution $N(8.02, 1.98^2)$, which is very close to the theoretical distribution $N(8, 2^2)$. As a result, we have to reject the hypothesis that the learned function is an XOR logic operation.

XOR logic is the core operation of the most secure " one-time pad" encryption scheme, and due to its balanced statistical property and simplicity, it is chosen here for the deep investigation. As the future work, we could study other popular and more sophisticated structure such as SPN or Feistel to see how the two are correlated. However, we are pessimistic to see any correlation with the current known designs, since in the training phase, we only define the ability of the adversary without feeding any information on how it should achieve the goal, as a result machine is smart to come out with many options. Also we should not feed the adversary with any information regarding the current design principle so that the encryption scheme is guided in the specific way to learn. Otherwise we are recreating another known encryption algorithm in the neural network form only. To understand the underlined scheme, further research in the neural network theory is required.

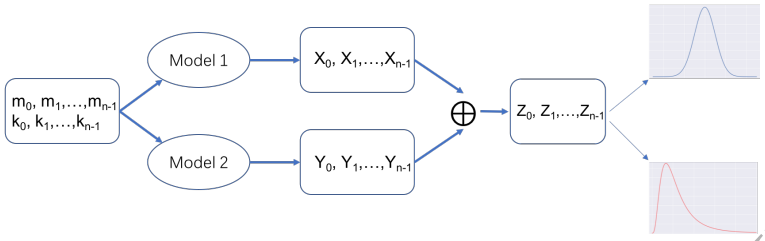

Figure 6: Function shape test framework

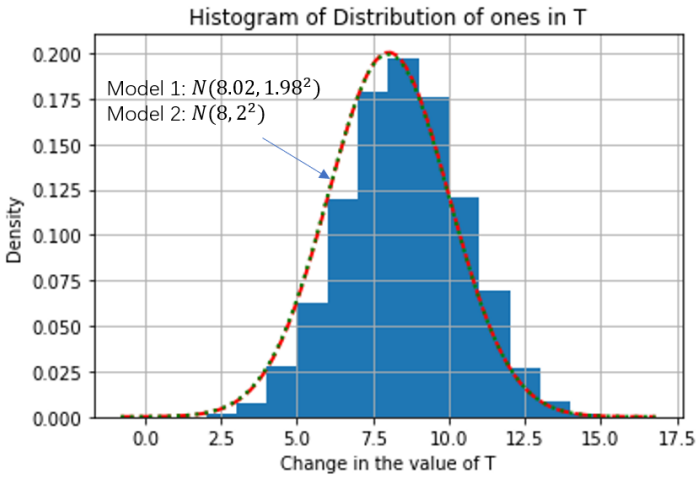

Figure 7: Result on testing the function shape

The χ^2 and the Kolmogorov-Smirnov methods are suitable for testing the random distribution of the floating point numbers which corresponds to the output of the neural networks given the experiment above. However, for the NIST statistical tools, it is originally proposed for the integer numbers. Although we transform the floating point number to integers, it is far from perfect. One floating point number is originally denoted as one ciphertext bit, but the ciphertext space is extended significantly. Simply shrinking to one bit will lose much information, and testing the floating point number in its raw form will suffer from the precision problem. In summary, we believe the χ^2 test is the most suitable test regarding this scenario.

4. Discussions on various models

We enhance the original encryption scheme by introducing different attackers with various abilities. Since the strength of the scheme depends on the ability of the attacker during the training stage, some of the models showed in this Section will be more stronger and more flexible in resisting against various attacks. A first straightforward idea is to feed the attacker the plaintext and ciphertext pair, or the ciphertext only and to see if he or she can recover the secret key. This model resembles the known plaintext attack (KPA) and the ciphertext only attack models (COA). In the experiment which is omitted here, we found that the adversary Eve does not have any advantage since the beginning of the training. As a result, the competitive model does not have any effect in improving the quality of the encryption scheme since Alice and Bob could just exclude the involvement of the secret key in the encryption scheme.

Sometimes we need to consider what advantage can the adversary gain when he or she has already obtained part of the information such as the secret key or the plaintext. Especially, the strength of our encryption scheme heavily depends on the adversarial network, thus it is necessary to adjust its ability to tune our encryption network. To give the adversarial network the extra information about part of the plaintext and the secret key shares some similarities with the leakage resilient cryptography. In the leakage resilient cryptography, the provable secure mathematical algorithms face the challenges from the various side channel attacks that may leak information about the key/plaintext or could be any functions of it [23] [24] [25], while in this paper we only consider the straightforward information leakage where the information leaked is linearly related. We exploit several models and use "Info1(L) + Info2 $\rightarrow$ Info3" to denote the corresponding types. Here the information on the left hand side of $\rightarrow$ is given to the attacker, and the information on the other side is the target of the attacker. If "(L)" is appended, it means that this information is partially leaked to the attacker. The filters' sizes of CNN and the batch size remain the same as the previous models.

Among the models that will be introduced, some can be seen as a more general one than the other, for example model Key(L) + Ciphertext → plaintext can be seen as a more general version of model Key + Ciphertext + Plaintext(L) → plaintext. Still we make separate discussion in this paper since the partial plaintext input to the adversary will greatly affect the behavior of our encryption scheme.

4.1. One-Eve model

This section provides various models assuming that there is only one attacker Eve. The attacker has only one target, for example, to either recover the plaintext or the secret key.

Key(L) + Ciphertext → Plaintext

In this model as shown in Figure 8, Eve has access to the ciphertext as well as part of the secret key. We change the input bits of the key from 0 to 16. The model is shown in Figure 8 . When the number of leaked key bits increases, the loss-rate of Eve and Bob could not go convergence. As shown in the experiment, when the amount of bits leaked to Eve is below 5, the curve can converge at a certain point. Figure 9 shows the situation that Bob finally defeats the attacker, while Figure 10 demonstrates the situation that the curve cannot be convergent even we trained it with 500 epochs.

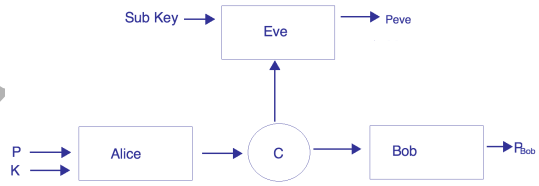

Figure 8: Key(L) + Ciphertext → Plaintext

When we increase the leaked key bits to Eve, Bob gradually loses the ability decoding the ciphertext that he received. The advantage that Eve hold can not directly help her to generate the correct plaintext from the ciphertext. However, the advantage can help her prevent Bob from decoding the information from the

ciphertext. It is evident that Eve can successfully damage the communication between Alice and Bob as shown in Figure 10.

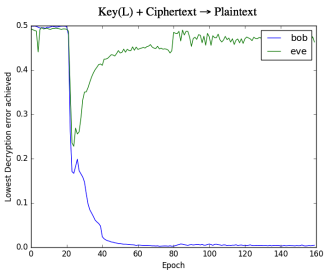

Figure 9: Eve gets 4 bits key. Trained with 80 epochs while test with 80 epochs.

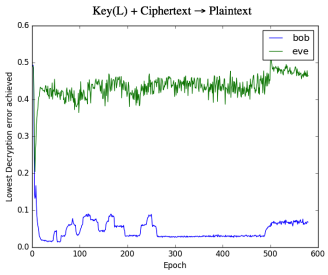

Figure 10: Eve gets 8 bits key. The model is trained with 500 epochs

Key(L) + Ciphertext → Key

In this model, Eve's goal is to recover the secret key instead of the plaintext. This model has the same input as the previous model as shown in Figure 11. We show one of the experiment results in Figure 12 and the trend between the leakage bits and the loss rate are shown in Figure 13.

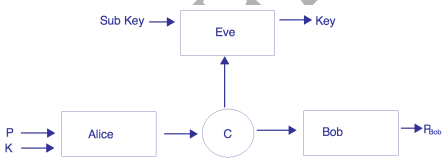

Figure 11: Key(L) + Ciphertext → Key

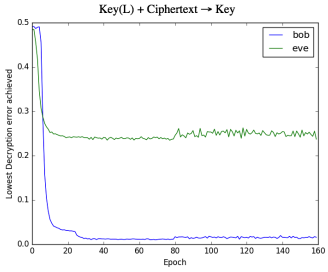

Figure 12: 8 bits key is leaked. Epoch is 80 for both training and testing.

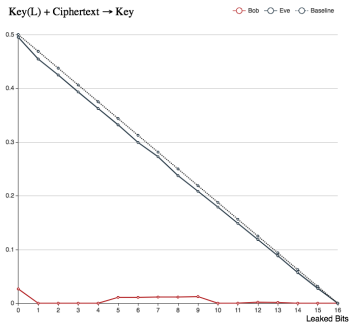

Figure 13: The trend between the leakage bit and the decryption error

The loss rate of Eve in Figure 13 is very close to the baseline which is the straight line in the figure. The baseline shows the information about key that Eve has known before we trained this system. Obviously, Eve do not know more information about the key than we told her before the training. From this result, we can make a hypothesis that the Alice will reduce the involvement of the secret key when Alice find that Eve's goal is to recover the whole secret key instead of the plaintext. If Alice do not use the key to generate the output, Eve would not find out the key for sure. Therefore, the loss rate of Eve should be a straight line, which means the amount of the key that we told Eve is all what she knows about key. Therefore, we could consider using new models forcing Alice to add the key material when generating the ciphertext. And this inspires us designing the "Two-Eve model" that would be described later.

Also we can make Eve more stronger by designing the model "Key(L) + Plaintext + Ciphertext → Key". In this model, Alice is smart enough to reduce the influence of the secret key during the encryption process. Even Eve knows all the information about the plaintext, she can not derive the key used in the communication because Alice do not use the information of the key to derive the ciphertext in the beginning. The experiment which is omitted here shows that the chance for Eve to derive the key information is very small.

Key + Ciphertext + Plaintext(L) → Plaintext

In this model, we give Eve part of the plaintext and the key along with the ciphertext output by Alice, Eve's goal is to recover the plaintext here. Figure 14 shows the structure of the model.

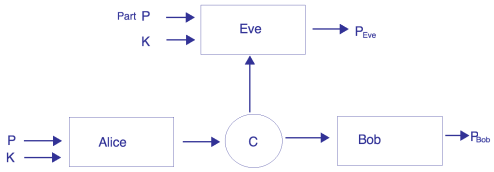

Figure 14: Key + Ciphertext + Plaintext(L) → Plaintext

The experiment shows an opposite performance compared with the previous model. While the amount of leaked plaintext to Eve is small, both Bob and Eve can not easily converge to a certain point as shown in Figure 17. However when the leaked bits is 16, Eve and Bob can get a good performance as shown in Figure 18. When the amount of bits leaked is larger than 9, Eve and Bob can gradually get the convergence as shown in Figure 15 Figure 16.

In this model, Alice wants to prevent Eve from recovering the correct plaintext. However, she cannot make the communication happen without using the plaintext as the input because Bob needs to decrypt the ciphertext correctly. In this situation, Alice and Bob can also prevent the attack from Eve even she knows most part of the plaintext information. But Bob also suffers from the low decryption accuracy. Therefore both Bob and Eve cannot get a convergence easily as shown in Figure 17. When Eve obtains rich information about the plaintext, then Alice and Bob cannot efficiently prevent the attack from Eve, and finally Alice and Bob give up protection. System will reach an agreement to ensure that Bob can get the correct information without defeating Eve. In this situation, both Bob and Eve can get a convergence as shown in Figure 18.

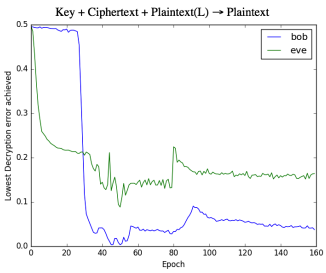

Figure 15: Leaked plaintext bits is 9. Epoch is set to 80.

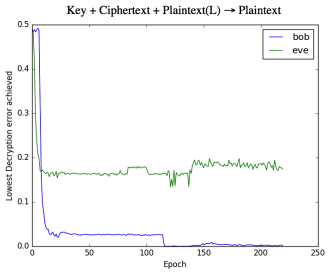

Figure 16: Leaked plaintext bits is 10. Epoch is set to 140.

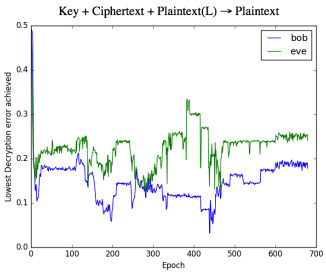

Figure 17: Leaked plaintext bits is 8. Epoch is set to 600.

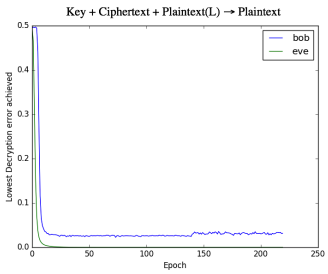

Figure 18: Leaked plaintext bits is 16. Epoch is set to 140.

4.2. Two-Eve Model

From the discussion in the previous models, we can see that the strength of the encryption scheme depends largely on the ability of the adversary. Inspiring from the fact that recovering the key or the plaintext only cannot make the encryption protocol stronger (Alice can reduce the involvement of the key), we introduce a two-eve adversarial model to further enhance the encryption protocol. In this new model, we have two attackers Eve1 and Eve2 with different attacking goals. Eve1 focuses on recovering the plaintext and Eve2's goal is to recover the secret key. In this model, Alice cannot simply ignore the involvement of the secret key due to the existence of Eve1. We further divide into the following two categories.

Key(L) + Ciphertext → Plaintext(Eve1) + Key(Eve2)

In this model, Eve1 and Eve2 have two inputs, which are part of the leaked key and the ciphertext. Eve1 outputs the plaintext and Eve2 outputs the secret key. The other structure and parameters are same as the previous models. Figure 19 shows the model and the experiment results including the decryption error and the trend of the loss rate are shown in Figure 20 and Figure 21. 8 bits key is leaked to Eve, and we train the model using 80 epochs data, and 80 for testing. The batch size of each epoch is 256.

The curve of Eve is entirely below the baseline in the Figure 21, which reflects that Eve2 is helping Eve1 to gain more advantages. Eve2 forces Alice to use the key when she generates the ciphertext, and meanwhile, Eve1 has an advantage when she decrypt the ciphertext. Also the attack from Eve1 and Eve2 makes Bob impossible to recover the plaintext accurately when Eve1 and Eve2 have already obtained most of the key information. In Figure 21, after Eve1 and Eve2 gained more than 7 bits of key, the loss-rate of Bob begins to rise. For testing, the batch size of each epoch of data is set to 256.

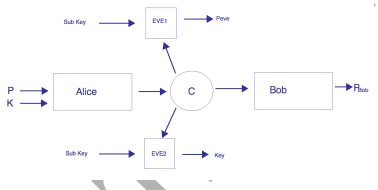

Figure 19: Key(L)+Ciphertext Plaintext(Eve1)+Key(Eve2)

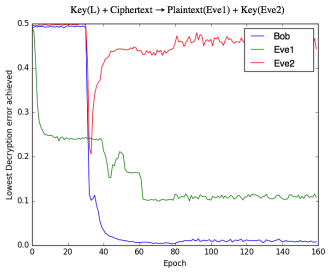

Figure 20: 8 bits key is leaked. Epoch is set to 80.

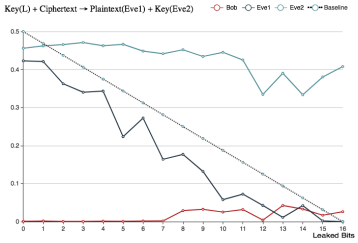

Figure 21: The trend between the leakage bit and the decryption error

Key(L) + Ciphertext + Plaintext → Plaintext(Eve1) + Key(Eve2)
In this model, Eve1 and Eve2 have three inputs, which are the plaintext, ciphertext and part of the key. Other settings are the same as the previous models. We give Eve 8 bits of the secret key and use 80 epochs for training the model and 80 for testing. Eve in this model behaves very similar to the model "Key(L) + Ciphertext → Key". It is reasonable to assume that Eve2 has already known what she want, which makes the adversary too strong, thus Alice gives up fighting back finally. We omit the experiment details for this model.

5. Conclusion and Discussion

In this paper, we investigate the techniques of building an automatic encryption scheme based on the neural networks. Starting from the basic symmetric key model, we first investigated the security of the scheme based on several dedicated statistical models to show that the original proposed scheme is not secure in the sense of distinguishability. The learned function remains unclear and through the experiment we eliminate the possibility of being an one-time pad encryption scheme. Then we extend the original model by investigating other more powerful adversaries. Except for a few models where the attackers are too strong, most of the models can be trained to be stabilized after the training stage. The new proposed encryption schemes are more strong and flexible in resisting against various attacks. Future works include how to further optimize the neural networks to make the legal party communication more efficient given less training steps, and how to choose the appropriate hyper-parameters

to improve the statistical randomness to resist against distinguishing attack is worth further investigating. Also how to make other security solutions with rich functionalities by using neural network is worth investigation. For example, data integrity is not protected in the current neural network, and it would be interesting to implement the concepts such as the authenticated encryption [26] by using the neural network.

Acknowledgment

This work has been partly supported by the National Natural Science Foundation of China under Grant No. 61702212 and the research funds of CCNU from colleges basic research and operation of MOE under Grand No. CCNU16A05040.

References

References

[1] Y. Liu, Z. Jin, Y. Wang, Survey on security scheme and attacking methods of wpa/wpa2, in: 6th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM 2010), IEEE, 2010, pp. 1–4.

[2] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, E. Tews, Revisiting ssl/tls implementations: New bleichenbacher side channels and attacks., in: USENIX Security Symposium, 2014, pp. 733–748.

[3] Y. Sheffer, R. Holz, P. Saint-Andre, Summarizing known attacks on transport layer security (tls) and datagram tls (dtls), Tech. rep. (2015).

[4] B. Dowling, M. Fischlin, F. Günther, D. Stebila, A cryptographic analysis of the tls 1.3 handshake protocol candidates, in: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, ACM, 2015, pp. 1197–1210.

[5] J. A. Donenfeld, Wireguard: Next generation kernel network tunnel, in: Proceedings of the 2017 Network and Distributed System Security Symposium, NDSS 2017, Vol. 17, 2017.

[6] B. Dowling, K. G. Paterson, A cryptographic analysis of the wireguard protocol, Cryptology ePrint Archive, Report 2018/080, https://eprint.iacr.org/2018/080 (2018).

[7] S. Goldwasser, S. Micali, Probabilistic encryption, Journal of computer and system sciences 28 (2) (1984) 270–299.

[8] M. Abadi, D. G. Andersen, Learning to protect communications with adversarial neural cryptography, arXiv preprint arXiv:1610.06918.

[9] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in neural information processing systems, 2014, pp. 2672–2680.

[10] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, arXiv preprint arXiv:1412.6572.

[11] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, R. Fergus, Intriguing properties of neural networks, CoRR abs/1312.6199.

[12] R. Mislovaty, E. Klein, I. Kanter, W. Kinzel, Security of neural cryptography, in: Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, ICECS 2004, Tel Aviv, Israel, December 13-15, 2004, 2004, pp. 219–221. doi:10.1109/ICECS.2004.1399654.

[13] A. Ruttor, Neural synchronization and cryptography, Ph.D. thesis, Julius Maximilians University Würzburg, Germany (2006).

[14] A. Klimov, A. Mityagin, A. Shamir, Analysis of neural cryptography, in: International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2002), Springer, 2002, pp. 288–298.

[15] E. Biham, A. Shamir, Differential cryptanalysis of des-like cryptosystems, Journal of CRYPTOLOGY 4 (1) (1991) 3–72.

[16] M. Matsui, Linear cryptanalysis method for des cipher, in: Workshop on the Theory and Application of of Cryptographic Techniques (Asiacrypt 1992), Springer, 1993, pp. 386–397.

[17] R. C.-W. Phan, Impossible differential cryptanalysis of 7-round advanced encryption standard (aes), Information processing letters 91 (1) (2004) 33–38.

[18] A. Bogdanov, V. Rijmen, Linear hulls with correlation zero and linear cryptanalysis of block ciphers, Designs, codes and cryptography 70 (3) (2014) 369–383.

[19] D. Wagner, The boomerang attack, in: International Workshop on Fast Software Encryption (FSE 1999), Springer, 1999, pp. 156–170.

[20] A. Biryukov, D. Khovratovich, Related-key cryptanalysis of the full aes-192 and aes-256, in: International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt 2009), Springer, 2009, pp. 1–18.

[21] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems, arXiv preprint arXiv:1603.04467.

[22] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A statistical test suite for random and pseudorandom number generators for cryptographic applications, Tech. rep., Booz-Allen and Hamilton Inc Mclean Va (2001).

[23] S. Halevi, H. Lin, After-the-fact leakage in public-key encryption, in: Theory of Cryptography Conference (TCC 2011), Springer, 2011, pp. 107–124.

[24] S. Faust, T. Rabin, L. Reyzin, E. Tromer, V. Vaikuntanathan, Protecting circuits from leakage: the computationally-bounded and noisy cases, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques (Eurocrypt 2010), Springer, 2010, pp. 135–156.

[25] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, T. Rabin, Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering, in: Theory of Cryptography Conference (TCC 2004), Springer, 2004, pp. 258–277.

[26] P. Rogaway, Authenticated-encryption with associated-data, in: Proceedings of the 9th ACM conference on Computer and communications security (ACM CCS 2002), ACM, 2002, pp. 98–107.