Linear Attack on Round-Reduced DES Using Deep Learning

Botao Hou[1,2], Yongqiang Li[1,2], Haoyue Zhao[1], and Bin Wu[1,2](✉)

[1] State Key Laboratory of Information Security,
Institute of Information Engineering, CAS, Beijing, China
{houbotao,liyongqiang,wubin}@iie.ac.cn, zhaohaoyue1@gmail.com
[2] School of Cyber Security, University of Chinese Academy of Sciences,
Beijing, China

Abstract. Linear attack is a powerful known-plaintext cryptanalysis method on block ciphers, which has been successfully applied in DES, KATAN, SPECK and other ciphers. In this paper, we use deep learning networks to achieve linear attack on DES with plain-cipher pairs. Comparing with traditional linear attack algorithm, our work requires less knowledge about complex cryptanalysis as neural network can work well by data-driven. Thus, this paper has three main contributions. First, a new linear attack architecture based on deep residual network was proposed to train discriminative neural networks with auto-generated plain-cipher pair data. The results indicate that trained neural networks can effectively learn algorithmic representations of the XOR distributions of given linear expression on DES. Second, several novel neural network-based algorithms were designed to efficiently enforce key recovery on round-reduced DES using trained networks with moderate full and partial bits of linear expression as inputs. Third, as far as we know, it is the first time that neural networks are used to achieve known-plaintext attack on complex block ciphers.

Keywords: Linear attack · Deep learning · DES

1 Introduction

Linear cryptanalysis is one of the most powerful analysis techniques used in modern block ciphers. It can achieve key recovery attacks utilizing non-zero correlation with bits of plain-cipher text and key, which is expressed in a linear approximate equation. The first linear cryptanalysis [2] was presented to break Data Encryption Standard (DES) successfully in 1994. Since DES [1] was published in 1977, its security has been focused by all over the world. In that paper, Matsui provided some linear equations on round-reduced DES and proposed a key recovery algorithm for known-plaintext attack in 8-round and even only-plaintext attack in 8 rounds. And Matsui [3] proposed an improved version for linear cryptanalysis and its application to the full 16-round DES. Later, Hermelin et al. [4] improved linear cryptanalysis into multiple approximations

L. Chen et al. (Eds.): ESORICS 2020, LNCS 12309, pp. 131–145, 2020.
https://doi.org/10.1007/978-3-030-59013-0_7

and achieved a faster attack. Obviously, all of those traditional linear cryptanalysis works need amounts of mathematical knowledge and manual theory deduction.

Recently, some works have been explored to combine deep learning and applicable statistical cryptanalytic techniques [10,12]. At first, Abadi and Andersen [5] trained two neural networks which allow them to communicate using given key without advanced cipher design, and another adversarial network was trained to prove that it cannot recover information without the key. However, their work did not explain what net construction is in cryptography. Soon, Coutinho et al. [11] improved simple adversarial network above with chosen-plaintext attack and obtained a unbreakable One-Time Pad algorithm in unsupervised condition which explored the effect of adversarial network in security. And then, some works tried to achieve cracking directly by simulating ciphers [13]. An unsupervised CycleGAN neural network [8], named CipherGAN, was used to crack Shift and Vigenere ciphers. Their work showed that neural network can learn detail relationship about encrypt and decrypt processes, but it was limited to fixed key. Comparing with traditional encrypt algorithms, modern block cryptographical algorithms are more complex so that previous methods can't work well, and some works began to apply some mature cryptanalysis methods to improve availability of attacking using machine learning [14]. Recently, some works [9] explored the possibility of applying machine learning on side channel attack of Advance Encryption Standard (AES), but generally side channel is considered not to be cryptanalysis in the sense we discussed. And Gohr [6] tried to apply deep learning on Speck, a lightweight block encryption algorithm. They constructed a network to more accurately learn the distribution of output difference with a fixed input difference. However, they didn't give attacks on more complex ciphers.

1.1 Our Contribution

First of all, we devise and train neural networks and expect that we can achieve efficient key recovery on DES using trained network models. Those network models should obtain the ability of distinguishing different distributions by observing given linear expression on round-reduced DES. Considering two different key recovery methods, one bit key recovery and multiple bits key recovery, we train corresponding network models in different ways.

For one bit key recovery on round-reduced DES, we propose a new neural network attack framework that can successfully distinguish two different binomial distributions. Those distributions perform two different situations of n-round linear approximation expression. Using the trained network models, we established corresponding one bit key recovery algorithm and achieved successful key recovering on 3, 4 and 5 rounds DES. In order to know the availability of our models, we calculate the expected efficiency for round-reduced DES that use Bayesian model. Experimental results indicate that the performance of our models is very closed to theoretical value.

In multiple bits key recovery, another neural network model is proposed to train as a discriminator for distributions produced by real and random effective key bits. And this model is used in proposed multiple bits key recovery algorithm. We tested the performance of this algorithm on 4 rounds DES and obtained effective key rank.

1.2 Paper Organization

The rest of the paper is organized as follows. In Sect. 2, we present a brief description of the cryptographic modules employed in our linear cryptanalysis. In Sect. 3, we introduce our detail scheme of neural networks. The result of neural discriminators and corresponding key recovery attacks are in Sect. 4. Section 5 is the conclusion about our scheme in short.

2 Preliminaries

Before introducing our architecture, we briefly review some cryptographic building modules deployed in linear cryptanalysis method on DES and two classical key recovery attack algorithms.

2.1 DES

DES is a iterative cryptographic algorithm with Feistel structure, which has a profound impact on the design of later ciphers. DES uses 56 bits key to protect message with block divided into 64 bits. Omitting the initial permutation IP and the final permutation IP^{-1} in full DES, we call input and output of round iterations as plain text block P and cipher text block C. Each block will be divided into two 32 bits blocks (L, R), which will be encrypted by total 16 rounds. More details can be seen in [1].

For rth round, the output L_r and R_r are computed as follows.

$$\begin{aligned} L_r &= R_{r-1} \\ R_r &= L_{r-1} \oplus F(R_{r-1}, K_r) \end{aligned} \tag{1}$$

Where F(.) is the non-linear function called F function, it contains four operations which include extension operation E of R_r, bitwise XOR operation between subkey and extended R_r, S-box operation S and final permutation operation. F function is briefly expressed as:

$$F(R_{r-1}, K_r) = S(E(R_{r-1}) \oplus K_r) \tag{2}$$

2.2 Linear Attack

Linear Approximate Equation. Linear attack has been widely used to break block cipher algorithms. Indeed, given plain text P, master key K and corresponding cipher text C, linear approximate equation L try to describe the linear relationship of bits in serval fixed locations like:

$$\alpha \cdot P \oplus \beta \cdot C = \gamma \cdot K \tag{3}$$

Algorithm 1. ONE BIT KEY RECOVERY ALGORITHM

```
Input:
$L_n$, $n-$round linear approximate equation
$Pr_{L_n}$, the probability of $L_n$
$Pair$, plain-cipher text pairs generated by key $K$
Output: output result
$N_{pc} \leftarrow$ the number of $Pair$
$N_L \leftarrow 0$
for $pair$ in $Pair$ do
    $L^l \leftarrow$ compare the left side of $L_n$
    if $L^l == 0$ then
        $N_L += 1$
if $N_L > N_{pc}/2$ then
    if $Pr_L > 1/2$ then
        return $L^r = 0$
    else
        return $L^r = 1$
else
    if $Pr_L > 1/2$ then
        return $L^r = 1$
    else
        return $L^r = 0$
```

Where α, β and γ are the bit location masks and $\alpha \cdot P$ is the bitwise addition for bits in locations marked by α in P. There we name the value of left side in L as L^l and the right side as L^r. Generally, equation L holds with the probability Pr_L of 1/2. But if there is an obvious deviation with 1/2 and Pr_L, we call this expression L as a well linear approximate equation. The bigger this deviation is, the quicker this expression could be distinguished from other expressions. Moreover, key recovery mentioned in follows is relative with Pr_L closely.

Key Recovery Attack. There are two different linear attack algorithms divided by number of key bits can be recovered. First one is one bit key recovery attack, relying on a well linear expression. Multiple bits key recovery is another, it generally depend on the linear equation which expended by (n-1)-round expression. Both of those attacks can work well in DES, and many effective linear expressions can be found [2].

One Bit Key Recovery. Given linear approximate equation L like Function 3, we can judge whether L^r is 0 or 1 with probability Pr_L. If we have N_{pc} plain-cipher text pairs generated by fixed key, we count the number N_L of those pairs that satisfy $L^l = 0$. If N_L has obvious difference with $N_{pc}/2$, we can judge this one bit key L^r depending on the symbol of difference with high success rate. The detailed recovery process is showed in Algorithm 1.

Algorithm 2. MULTIPLE BITS CANDIDATE KEY RANK ALGORITHM

Input:
L_n, $n-$round linear approximate equation
$Pair$, Plain-cipher text pairs generated by key K
Output: output $Rank_{key}$

```
$N_t \leftarrow$ the number of effective text bits in left $L_n$
$T_t \leftarrow \{0\}_{2^{N_t}}$
for $pair$ in $Pair$ do
    $e \leftarrow$ bits extracted from $pair$ following $L_n$
    $T_t[e] += 1$
$N_k \leftarrow$ the number of effective key bits in left $L_n$
$T_k \leftarrow \{0\}_{2^{N_k}}$
for $k$ in len($T_k$) do
    for $t$ in len($T_t$) do
        $L^l \leftarrow$ compare the left side of $L_n$
        if $L^l == 0$ then
            $T_k[k] += t$
$N_{pc} \leftarrow$ the number of $Pair$
for $k$ in $len(T_k)$ do
    $T_k[k] = T_k[k] - N_{pc}$
$Rank_{key} \leftarrow$ sort $T_k$ by descending value order
return $Rank_{key}$
```

Multiple Bits Key Recovery. Generally, if we attack n rounds DES, we have to obtain a (n-1)-round linear approximate equation L_{n-i} with Pr_L. Considering the effect of F function in first round and nth round, n-round expression L_n is described as:

$$\alpha \cdot P \oplus \beta \cdot C \oplus \mu \cdot F_1(P, K_1) \oplus \nu \cdot F_n(C, K_n) = \gamma \cdot K \tag{4}$$

Since L_n is expanded from L_{n-i}, Pr_{L_n} should be almost same with $Pr_{L_{n-i}}$, which makes us knowing the distribution of L_n^l. Obviously, this value is totally determined by some bits of plain-cipher text and key, and we call those bits as effective text bits and effective key bits respectively. Based on known Pr_L, we can recover those effective key bits as follows.

First, we list all possible effective key bits as key candidates. Considering that the probability Pr_{L_n} would almost equal to $Pr_{L_{n-i}}$ when K_1 and K_n are correctly guessed, this leads us to use maximum likelihood method in regard to those key candidates.

There we get N_{pc} plain-cipher text pairs generated with fixed key K. For each key candidate, compute L_n^l and add counter with 1 when it equals to 0. Sort all key candidates by the difference between counter and $N_{pc}/2$ as key rank. Generally, correct key bits will be in higher rank. The candidate key rank processing is showed in Algorithm 2.

3 Network Architectures

Our goal is to develop a learnable, end-to-end model for linear attack, and it should obtain statistical cryptanalytic characteristics. Thus, we proposed a new neural network architecture as a deep learning discriminator to distinguish different distributions. The diagram for our network is shown in Fig. 1.

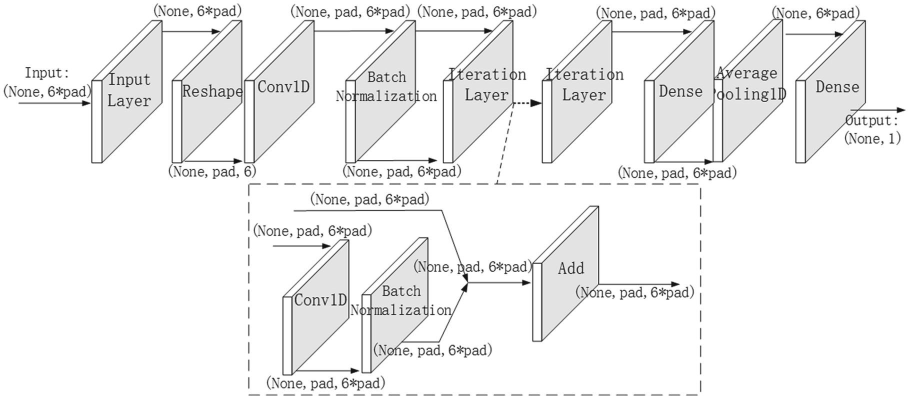


Fig. 1. Model overview. An universal neural network architecture used in our experiments

Those networks comprise three main components: input layer, iteration layer and predict layer. The iteration layer is built by classical residual neural network [7]. This network has been successfully applied in many domains. It consists of some residual blocks which add input layer to output layer and produce new output, the output will been sent to next block. The most important advantage of residual networks is that it can effectively avoid gradient dispersion when the number of layer increases.

The input layer receives training data with fixed length and applies reshape layer into the data. We expect that our network should simulate XOR operation better and form some intermediate representation. For this reason, we transpose and apply convolution into input data so that we can expend the effect of each bit. After batch normalization layer, data will be sent into iteration layer. Each iteration layer has same structure with a convolution and normalization following. What' more, a skip connection is applied to add input layer and output layer and this operation may allow next layer can mix bits in block more like bitwise addition. Iteration layer will repeat 5 or 10 rounds in our experiments, and then the predict layer will be following. The predict layer provides a fully connect operation in order to combines all bits and a single linear layer to produce one bit predicted result.

In our key recovery experiments, this neural network will be fed with bit sequences and is expected to distinguish those input into two different distributions. For each sequence, it consists of 6 units and each unit will be padded with

Algorithm 3. ONE BIT DEEP LEARNING NETWORK KEY RECOVERY ALGORITHM

```
Input:
$L_n$, $n-$round linear approximate equation
$Net_{L_n}$, neural net discriminator trained by $L_n$
$Pair$, Plain-cipher text pairs generated by fixed key $K$
Output: output
$N_{pc} \leftarrow$ the number of $Pair$
$G \leftarrow Net_{L_n}(Pair)$
if $sum(G) > N_{pc}/2$ then
    return right of $L_n = 1$
else
    return right of $L_n = 0$
```

0 to fixed length, which is determined by max length of each mark in L_n. Generally, this length is not longer than 8 and we will pad the sequence to $6 \times 8 = 48$. Input layer changes the size of this sequence into 8×48, and it will be trained in this size till predict layer. Pooling operation condenses it into 48×1 and output it with 1 bit by dense layer.

In each epoch, we will check networks by validation data, and we save and update the best model according to its accuracy.

4 Attack Architecture

In this section, we will introduce two new linear attack architectures: one bit key recovery and multiple bits key recovery. We apply them in round-reduced DES, and both of them can distinguish different distributions well using deep learning net and realize expected key recovery.

4.1 One Bit Key Recovery

Given n-round linear approximation expression L_n as Function 3, and we know that it will hold with certain probability Pr_{L_n} in previous. There, we don't need to know exact value about Pr_{L_n} and more details, and we can also obtain one bit key information $\gamma \cdot K$. For this, we propose one bit key recovery algorithm showed in Algorithm 3 to recover mentioned bit using deep learning networks.

Train and Recover. Supposed that Pr_{L_n} is the probability linear expression L_n holds, if we ask that L_n^r is fixed to 0, the distribution of L_n^r will be almost binomial distribution which means 0 will appear with the probability equaling to Pr_{L_n}. While the binomial distribution will be inverse if L_n^r is fixed to 1. Thus, we mark those different distributions with corresponding labels and expect trained networks can effectively distinguish them by inputting some bit sequences.

In order to obtain those network models, we generate training and validation by several phases as follows:

1. Generate plain texts P and master keys K ordering uniformly distribution.
2. Encrypt P with K by $n-$round DES cipher and obtain cipher texts C.
3. Extract $P - C$ pairs into bits sequence EX_{pc} and K into EX_k depending on linear equation L_n.
4. Pad EX_{pc} with 0 into X following the order of $(\alpha \cdot P||\beta \cdot C)$.
5. Set label Y relying on XOR value distribution of each EX_k.

After generating enough data, neural network discriminator Net_{L_n} will be trained to predict right label Y. Obviously, if Net_{L_n} is train well, its correct output will help us directly to recover corresponding one bit key information. Thus, we apply trained network into Algorithm 3 to recover this key bit.

Recovery phase need N_{pc} plain-cipher pairs generated by fixed key K. Repetitively run Phase 3–4 above and we can obtain extracted text sequences of those pairs. Those text sequences are feed into Net_{L_n} and output their prediction. Considering with the accuracy of Net_{L_n}, the success rate of Algorithm 3 rely on N_{pc} and performance of algorithm will be shown in following experiments.

Goal Model. After training mentioned above, we indeed obtain a deep learning discriminator. This discriminator would first learn the simulation of XOR operation, and then obtain the ability that distinguish the difference with different binomial distribution performance.

Our deep learning model didn't know any information about those distributions and even didn't know XOR operation before training, all they obtaining is input seems like random bit sequences. Obviously, if we can obtain those distribution information about linear expression L, we can estimate the best result of those networks using Bayesian rule.

As Pr_L is the possibility of linear approximation expression L holding and discriminator B_L with Bayesian model obtains distribution features of L^l fully, if L^l of a bit sequence is 1, the accuracy of B_L correctly judging that this sequence belongs into $L^r = 1$ is shown following Function 5.

$$P(\gamma \cdot K = 1|1) = \frac{P(1|\gamma \cdot K = 1)P(\gamma \cdot K = 1)}{P(1|\gamma \cdot K = 1)P(\gamma \cdot K = 1) + P(1|\gamma \cdot K = 0)P(\gamma \cdot K = 0)} \tag{5}$$

Supposing that K is generated following uniform distribution, accuracy of B_L will be equal to Pr_L.

This Bayesian model will be our goal model of deep learning network. We replace network discriminator Net_L with this Bayesian discriminator B_L in Algorithm 3 and can get one bit key recovery. After reducing, we find that the relationship between success rate of one bit key recovery and number of plain-cipher pairs required is same with Lemma 2 in [2]. Thus, we can measure key recovery effect which uses deep learning networks with this lemma.

Experiment. All of our experiments are run in a uniform environment, models are trained on a workstation with NVIDIA GeForce GTX 1080Ti and Intel(R) E5-2609 1.7 GHz CPU.

$$P_H[7,18,24,29] \oplus P_L[15] \oplus C_H[7,18,24,29] \oplus C_L[15] = K_1[22] \oplus K_3[22] \tag{6}$$

$$\begin{aligned} P_H[7,18,24,29] \oplus P_L[15] \oplus C_H[15] \oplus C_L[7,18,24,27,28,29,30,31] \\ = K_1[22] \oplus K_3[22] \oplus K_4[42,43,45,46] \end{aligned} \tag{7}$$

$$\begin{aligned} P_H[15] \oplus P_L[7,18,24,27,28,29,30,31] \oplus C_H[15] \oplus C_L[7,18,24,27,28,29,30,31] \\ = K_1[42,43,45,46] \oplus K_2[22] \oplus K_4[22] \oplus K_5[42,43,45,46] \end{aligned} \tag{8}$$

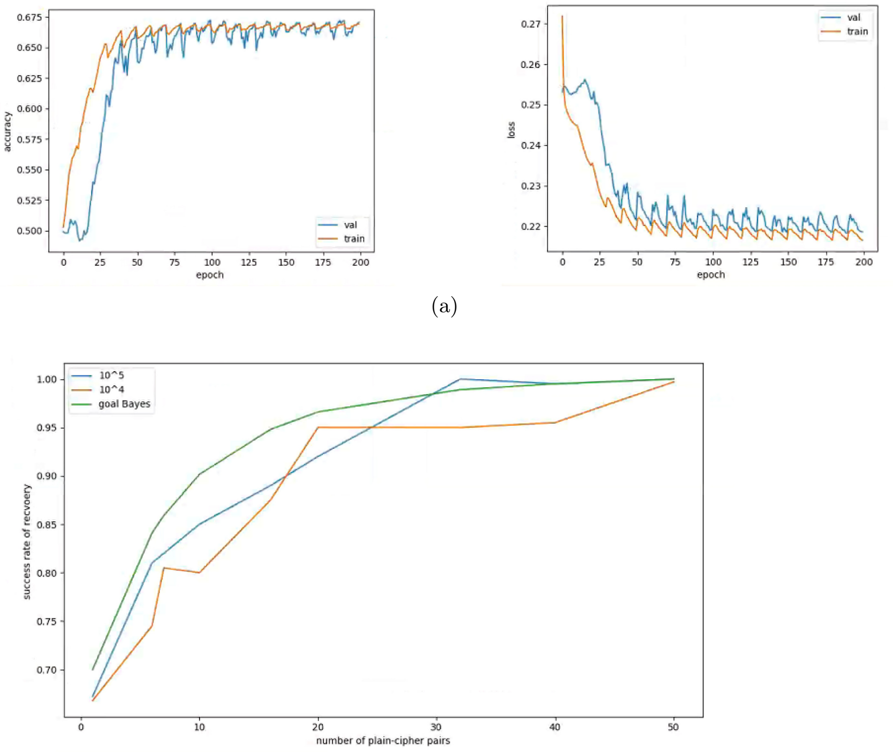
First, we tested the performance of one bit key recovery algorithm on L_3 which can be seen in Function 6. Model was trained for 200 epochs using the Adam optimizer [15] with a batch size of 1000 against MSE loss with $L2-$regularization. And there were 10^5 train data and 10^4 validation data used. Figure 2a shows the learn history of Net_{L_3}. The accuracy on validation data is 67.23% which is very closed to theoretical goal model which is 70%, same with Pr_{L_3}.

To be clear, our neural network knows nothing about XOR operation and detailed data distribution, but it can still perform well almost like goal model which knows all about knowledge. All of those show that the presented approach equips excellent learning capability of describing XOR distributions. What's more, we found that the increase of train data can significant improve the accuracy, and the network with 10^5 data is improved with 0.43% than 10^4 data.

Apply those models to recover key information and we found that the success rate of neural network models is only lower than theoretical Bayesian model slightly. The number of plain-cipher text pairs required in key recovery in different success rate based on those discriminators are shown in Fig. 2b. For each result, we run key recovery process for 2000 times to obtain moderate observations. We can see that our neural network can complete key recovery given small plain-cipher text set, and Net_{L_3} trained by 10^5 training data even performs better than theoretical success rate. Thus, those network model showed their capacity to distinguish different distributions.

Table 1. results of different models on corresponding linear expression L_n. Meanwhile, we show the average number of plain-cipher text pairs that can achieve key recovery success rate, each of them are test in 2000 times

Index	Network	Train data	Epoch	Depth	Accuracy	Number of $P-C$ pairs for success rate			
						85%	90%	95%	99%
1	B_3	–	–	–	0.7	6	10	17	32
2	Net_{L_3}	10^4	10^3	5	0.668	14	18	32	64
3	Net_{L_3}	10^5	5×10^3	5	0.6723	10	18	25	32
4	B_4	–	–	–	0.561	67	112	190	358
5	Net_{L_4}	10^6	5×10^4	5	0.5375	115	200	332	633
6	B_5	–	–	–	0.519	2770	4617	7849	14774
7	Net_{L_5}	10^6	5×10^4	10	0.5128	5130	8631	–	–



(a)

(b)

Fig. 2. (a) shows accuracy and loss of Net_3 in total train process data size of 10^5 respectively. Both valuation data and training data perform synchronously and indicate that network work well without over fitting. (b) shows key recovery performance of Bayesian model and our models. All of them will almost recover key information with success rate more than 99% when increase the number of plain-cipher pairs into 64. With same pair number, the success rate of neural network models only lower than theory Bayesian model slightly. And increasing the number of train data, neural networks will work better.

Excepted 3-round one bit key recovery, we also ran 4 and 5-round key recovery based on linear expression in Function 7 and Function 8. Comparing with L_3, the binomial distribution probability like Pr_{L_5} even decreases from 70% into 51.9% [2]. Obviously, the difficulty of distinguishing those two different distributions increases a lot. Table 1 shows the accuracy and key recovery of best models. There, B_n is the discriminator of goal model using Bayes mentioned above. And we can find that almost all of them can recover required one bit key with limited plain-cipher text pairs number. For neural discriminators Net_{L_5}, though it does not achieve success rate more than 95% with less than 20000 pairs plain-cipher text, it still performs its ability recovering key bit with success rate of even 90%.

Algorithm 4. MULTIPLE BITS DEEP LEARNING NETWORK CANDIDATE KEY RANK ALGORITHM

```
Input:
$L'_n$, $n-$round linear approximate equation
$Net_{L'_n}$, neural net discriminator trained by $L'_n$
$Pair$, Plain-cipher text pairs generated by key $K$
Output: output $Rank_{key}$
$N_k \leftarrow$ the number of effective subkey bits in left $L'_n$
$T_k \leftarrow \{0\}^{2^{N_k}}$
for key in len($T_k$) do
    $Ex \leftarrow$ bit sequences extracted from $(Pair, key)$ following $L'_n$
    $G_k \leftarrow Net_{L'_n}(Ex)$
    $T_k[key] = sum(G_{key})/len(G_{key})$
$Rank_k \leftarrow$ sort $T_k$ by descending value order
return $Rank_k$
```

4.2 Multiple Bits Key Recovery

Like Function 4, we can apply (n-1)-round linear approximation expression L_{n-1} to consecutive F-functions from the first round to the (n-1)th round or from the second round to the nth round of n rounds DES, and obtain n-round linear equation L'_n with some bits in F-functions. Because K_n is added in expression, we can try all possible effective bits in K_n and test whether the value of left L'_n satisfies the similar distribution like L_{n-i}, so that recover those effective bits. Thus, we propose new multiple bits key candidate recovery algorithm showed in Algorithm 4 to recover multiple bits using deep learning networks.

Train and Recover. Similar with one bit key recovery algorithm, we also utilize the ability of deep learning networks that can distinguish different distributions. Of course, we have to consider the interference produced by right side of Function 4. In order to simplify our models, we suppose that the value of $\gamma \cdot K$ is 0 which may be happened with the probability of 1/2. Then for once right guess of key, the distribution of L'^{l}_n will still be almost binomial distribution which 0 will appear in the probability equal to $Pr_{L'_n}$, and we call it real distribution. While the distribution for one wrong guess of key will be uniform, and we name it random distribution. Those difference is what our networks should distinguish. If given bit sequences generated by correct fixed key, neural network discriminator $Net_{L'_n}$ should output label of real distribution as 1 with a big probability, otherwise, it should be random distribution as 0.

Also, we give the generation phases of training and validation data.

1. Generate plain texts P and label Y ordering uniformly distribution, and Num_P is the number of those P.
2. Generate master key K ordering uniformly distribution, and filtrate out $Num_P K$ that satisfy $\gamma \cdot K = 0$.

3. Encrypted P with K by $n-$round DES cipher and obtain cipher text C.
4. Extract $P-C$ pairs into bits sequence EX_{pc} and K into EX_k with linear equation L'_n.
5. For each label Y, do.
   - if $Y=1$, Pad EX_{pc} with 0 into X following the order of $(\alpha \cdot P||\beta \cdot C||\mu \cdot F_1||\nu \cdot F_n)$.
   - if $Y=0$, Pad EX_{pc} with 0 into X following the order of $(\alpha \cdot P||\beta \cdot C||Rand||Rand)$, which $Rand$ is generated ordering uniformly distribution.

As we know, $F_1(P_L, K_1)$ and $F_n(C_L, K_n)$ are determined by effective text bits and effective key bits. Because the number of effective key bits are few enough, we can research those bit keys exhaustively and call those keys as key candidate. For each possible key candidate, we test this key candidate with some plain-cipher text pairs and input corresponding bit sequences extracted following L'_n into network model $Net_{L'n}$. We count those output as the score which support that this key candidate is the right bits of master key required. Sort those key candidates with corresponding score in descending order and we call those as key rank. A well discriminator should have the ability ranking real right subkey higher.

Once we get a key rank, we can run an exhaustive key search for remaining several bits key. In each trying, we will choose a candidate bit key from key rank by order. Obviously, the higher the rank of right subkey is, the quicker whole key recovery will complete.

Goal Model. Also, our neural network discriminator $Net_{L'}$ need distinguish two binomial distributions. However, different with distributions in one bit key recovery, these binomial distributions should be with $p_{real} = Pr_{L'}$ and $p_{ran} = \frac{1}{2}$. Use Function 5 and we can obtain the theory accuracy of $B_{L'}$ with Bayesian model.

Experiment. We run our network models on the number of 10^5 training data and 10^4 validation data. And we tested the performance of multiple bits key recovery on L'_4 showed in Function 9 extended from L_3. Thus $Pr_{L'_4}$ will almost equal to Pr_{L_3} if effective key bits in K_4 is right.

$$P_H[15] \oplus P_L[7,18,24,29] \oplus C_H[7,18,24,29] \oplus C_L[15] \oplus F_1(P_L, K_1)[15] = K_2[22] \oplus K_4[22] \quad (9)$$

We trained this neural network about $4-$round with 200 epochs and each epoch is run in size of 5000. As no unit in L'_4 is more than 4 bits, we set *padding* as 5. And we contain 6×5 bits sequence, where the sixth unit is F_4 and it don't appear in Function 9, we will pad it into $\{0\}_5$. Real and random data determined by random label Y were sent to 5-depth residual network. And those two different distributions were separated with accuracy of 56.77%, while the accuracy of theoretical Bayesian model should be 58.3%.

Analysis the effective text and key bits in Function 9, we can easily ensure that the effective key bits effecting left side of L'_4 are $\{K_1[42], K_1[43], K_1[44], K_1[45], K_1[46], K_1[47]\}$, all of them are related to S-box S_1. Those 6 bits subkey are what we aim to recover. We list all possibility of 6 bits may take and get key candidate table with size of $2^6 = 64$.

Table 2. Multiple bits key recovery on 4-round DES. We list the average key rank on different number of plain-cipher pairs. They are measured through 200 rounds in replicated test.

Network	Train data	depth	Accuracy	Average key rank in number of P-C pairs			
				32	64	128	256
$Net_{L'_4}$	10^5	5	0.5677	13	9	3	2

We set a random master key K which holds $\gamma \cdot K = 1$ asked by trained neural network $Net_{L'_4}$ above, and we obtained plain-cipher pairs $Pair$ with number of N_{pc} encrypted by K. Then we extract each pair following L'_4 and obtain bit sequence $(\alpha \cdot P || \beta \cdot C)$. Up to now, we have no information about F_1 in L'_4. For each key candidate K_{can}, we compute $\mu \cdot F_1(P_L, K_{can})$ and insert $\mu \cdot F_1$ into sequence. Record the prediction $Net_{L'_4}$ and get score of K_{can}.

Count all score of key candidate K_{can}, the rank of those key candidates with score is key rank. Research the rank of correct effective key bits, and we can test the performance of $Net_{L'_4}$ is showed in Table 2. As key ranks using $Net_{L'_4}$ are no lower than $2^5 = 32$ in those small number of plain-cipher pairs, all of those indicate that our neural network models can distinguish different distribution in multiple bits key recovery and are pretty effective for key ranking.

5 Conclusion

In this paper, we used deep learning network achieving linear attack in round-reduced DES. We proposed the network structure to distinguish different performance of linear expressions. Our experiments indicated that those deep learning networks have the capacity of learning complex static characteristics like XOR and distinguishing different distributions. In order to make networks perform better, we also designed two linear attack algorithms which apply network in one bit and multiple bits key recovery. These end-to-end architectures need almost few knowledge about distribution of linear expressions and performs well in our experiments. And the representations of our results are also useful for cryptanalysis on other more complex block ciphers.

For further work, we will continue to test the performance using deep learning networks to research linear approximations with limited advanced knowledge.

What's more, we found a problem effecting performance of net when we trained our network. Limited by number N_t of plain-cipher text bits, there are only 2^{N_t} text sequences in train text. However, training data is usually larger than this value and make some same input may have different label, and this may make network puzzled. The same situation also happened in [8], and we will explore those further more.

Acknowledgments. The authors appreciate the anonymous reviewers valuable comments, which improved the paper greatly. This work was supported by National Nature Science Foundation of China under Grants No. 61941116, No. 61772517 and No. U1936119, and National Key R&D Program of China under Grant No. 2019QY(Y)0602.

References

1. National Burean of Standards: Data Encryption Standard. U.S. Department of Commercc, Federal Information Processing Standards 46 (1977)
2. Matsui, M.: Linear cryptanalysis method for DES cipher. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 386–397. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_33
3. Matsui, M.: The first experimental cryptanalysis of the data encryption standard. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 1–11. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48658-5_1
4. Hermelin, M., Nyberg, K.: Linear cryptanalysis using multiple linear approximations. IACR Cryptology ePrint Archive (2011)
5. Abadi, M., Andersen,D.G.: Learning to protect communications with adversarial neural cryptography. arXiv Cryptography and Security (2017)
6. Gohr, A.: Improving attacks on round-reduced Speck32/64 using deep learning. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11693, pp. 150–179. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26951-7_6
7. He, K., et al.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. Gomez, A.N., et al.: Unsupervised cipher cracking using discrete GANs. arXiv: Learning (2018)
9. Gohr, A., Jacob, S., Schindler, W.: CHES 2018 side channel contest CTF - solution of the AES challenges. IACR Cryptology ePrint Archive (2019)
10. Lytvyn, V., Peleshchak, I., Peleshchak, R., Vysotska, V.: Information encryption based on the synthesis of a neural network and AES algorithm. In: 3rd International Conference on Advanced Information and Communications Technologies, pp. 447–450 (2019)
11. Coutinho, M., et al.: Learning perfectly secure cryptography to protect communications with adversarial neural cryptography. Sensors **18**(5), 1306 (2018)
12. Preishuber, M., et al.: Depreciating motivation and empirical security analysis of chaos-based image and video encryption. IEEE Trans. Inf. Forensics Secur. **13**(9), 2137–2150 (2018)
13. Greydanus, S.: Learning the enigma with recurrent neural networks. arXiv Neural and Evolutionary Computing (2017)

14. Paterson, K.G., Poettering, B., Schuldt, J.C.N.: Big bias hunting in amazonia: large-scale computation and exploitation of rc4 biases (invited paper). In: Sarkar, P., Iwata, T. (eds.) ASIACRYPT 2014. LNCS, vol. 8873, pp. 398–419. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45611-8_21
15. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference on Learning Representations (2015)