Application of Machine Learning in DES Cryptanalysis

Stefan Andonov, Jovana Dobreva, Lina Lumburovska, Stefan Pavlov, Vesna Dimitrova, and Aleksandra Popovska-Mitrovikj

Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius, University of Skopje, R.N.Macedonia
stefan.andonov@finki.ukim.mk
jovana.dobreva@students.finki.ukim.mk
lina.lumburovska@students.finki.ukim.mk
stefan.pavlov@students.finki.ukim.mk
vesna.dimitrova@finki.ukim.mk
aleksandra.popovska.mitrovikj@finki.ukim.mk

Abstract. The usage of machine learning is expanding over all scientific fields and this branch is becoming more and more popular in the last years. In this paper we consider application of machine learning in the cryptanalysis, precisely in cryptanalysis of DES algorithm. This algorithm works in 16 rounds and we make two analyses: one for only one round and one for all rounds. We use different datasets and specific neural network for each analysis. We present results from several experiments for different datasets and different keys. Furthermore, we analyze and compare the obtained results, where we provide visual and textual presentation and we derive some conclusions.

Keywords: Machine learning · DES · Cryptanalysis · Neural networks · Datasets

1 Introduction

The process of encryption, i.e., transforming plaintext into ciphertext and its reverse process decryption can be done by two techniques: symmetric and asymmetric key cryptography. In this paper we chose symmetric cryptography which means using the same key for encryption and decryption. For each algorithm there are two key aspects used: algorithm type (define size of plaintext that should be encrypted per step) and algorithm mode (define cryptographic algorithm mode). The algorithm mode presents a combination of a series of the basic algorithm and some block cipher with feedback from previous steps. In symmetric cryptography there are block and stream ciphers and we focused on block ciphers choosing the DES algorithm (Data Encryption Standard).

DES is a block cipher. It was founded and developed from the company IBM with the support of NBS. The algorithm encrypts the data in a block of 64 bits (8 bytes) and uses a key with length of 56 bits. In simple words, each separate

block works on the following way: we have plaintext with 64 bits, we use a key of 56 bits and we get a ciphertext again with 64 bits. Initially, each eight position of the key is discarded and that is how we get 56 from 64 bits [1].

DES is based on two fundamental principles of cryptography: substitution (confusion) and transposition (diffusion). It has 16 steps and each step is called round. First of all, the initial plaintext with 64 bits is handed over to Initial Permutation (IP) function and this function is performed on the plaintext. After this, the permutation is separated in two halves of the block: Left Plaintext (LP) and Right Plaintext (RP), each consisted of 32 bits. This part of left and right plaintext is repeated in 16 rounds of the encryption process with its own key. From key with length of 56 bits, a different 48-bit subkey (round key) is generated using Key Transformation. Using the expansion permutation the Right Plaintext is expanded from 32 bits to 48 bits. Then both the key and the Right Plaintext have the same number of bits, 48 and they are XORed, so the result output is given in the next step. In order to go back to 32 bits, here the algorithm uses the S-box substitution. After substitution, these bits are permuted using P-box Permutation. The P-box output of 32 bits are XORed with the Left Plaintext and old Right Plaintext become the Left Plaintext and the process is called swapping. This process is repeated 16 times and after the completion of all 16 rounds the Final Permutation is performed.

This paper is a combination of both fields, cryptanalysis and machine learning or more specifically we use neural networks. Neural networks are a set of algorithms, that are designed to recognize a given pattern. Many cryptographic methods and algorithms have the disadvantage of requirement a large computational power, complexity and time consumption and they require some more powerful performance in order to provide better results. In such situations, neural networks are often applied [17]. Their usage in our paper are separately described in each section and depends on whether we use DES with one or with all rounds. We use Python programming language to implement our crypto-machine learning survey. In order to provide comparison between possible errors we tested neural networks on one round and on the whole algorithm (16 rounds) as well, where we used different datasets. The results we got will be separately described in the following sections.

This section is followed by a section where we give an overview of the related work to our research. It contains short analysis of previous papers cryptanalysis of DES and Triple-DES and how neural networks can be used for cryptanalysis. In the past, there were more attempts to combine both fields and we give brief overview of their usage. Additionaly, we give short overview how DES algorithm is a target for many attacks and what is their impact on the whole process. In Section 3, we present our experimental results, where we used a neural network for DES in one round and DES in all rounds. We analyze results obtained for different datasets and different neural networks. At the end, we give some conclusions.

2 Related work

The DES has been a target for many attacks for a long time. Some of the attacks started by analyzing reduced-round DES and went up to the full-round DES. The most well known two attacks are differential cryptanalysis and linear cryptanalysis.

Differential cryptanalysis is a chosen-plaintext technique, developed by Biham and Shamir in [2], against reduced-round variants of the DES cipher, and later applied to the full 16-round DES.

Linear cryptanalysis is a statistical, known-plaintext attack on block ciphers. This technique has been more extensively developed by Matsui in attacks on the DES cipher [3].

In DES the S-boxes are the only non-linear transformation. All the remaining components, such as the data expansion and permutations are linear transformations. The full linear cryptanalysis of DES required 243 known-plaintexts.

Many other cryptanalytic attacks were developed afterwards, such as differential-linear cryptanalysis [4], related-key attacks [5, 6] and non-linear cryptanalysis [7].

In [4], most of the research is based on the usage of a recursive auto-associative artificial neural network. This type of system was unique at that time because it is self-organizing and therefore it does not require any design decisions from the user. Therefore, a unique property of the system allows for a variable number of symbols to be represented in each coded unit.

As we move on to related-key attacks, in [5, 6], two identical dynamical systems (neural networks), starting from initial conditions, can be synchronized by a common external signal which is coupled to the two systems. By doing that, the two neural networks, that are trained on their mutual output, synchronize to an identical time dependent weight vector. On this way, the synchronization by mutual learning can be easily applied to a secret key exchange protocol over a public channel.

The non-linear cryptanalysis, given in [7], analyze the security of a new key exchange protocol. This protocol is based as well on mutually learning neural networks. This paper represents a new potential source for public key cryptographic schemes which are not based on theoretical number functions and have small time and memory complexities.

Triple-DES was a target for a slightly different type of attacks due to its repetition of DES algorithm. Repeated encryption was bounded by some limitations as mentioned in [8] and [9]. In these papers, we have a conventional remote password authentication schemes. These schemes allow a serviceable server to authenticate the legitimacy of a remote server login user. Nevertheless, these schemes are not used for multi-server architecture environments.

Meet-in-the-middle attacks rendered the effective key to be of 112-bits length [10]. Few attacks on Triple-DES were successful in terms of numbers of known/chosen ciphertexts/plaintexts needed to break the algorithm such as the new attack introduced by Lucks which requires around 2^{32} known plaintexts, 2^{113} steps, 2^{90} single DES encryptions and 2^{88} memory [18]. Although it is the most

successful attack on Triple-DES in terms of number of needed known-plaintexts, it was, and still is considered impractical by the NIST. In this paper, a standard technique for attack of triple encryption is considered. The type of attack is meet-in-the-middle attack. One of the attacks used in [18], reduces the overall number of steps to roughly 2^{108}. Other attacks in [18] optimize the number of encryptions at the cost of increasing the number of other operations. Therefore, it is possible to break triple DES doing 2^{90} single encryptions and no more than 2^{113} faster operations.

There have been few attempts to use neural networks in cryptography. In 1998, Clark and Blank introduced a cryptographic system based on neural networks [11]. In [12], a connection between the theory of neural networks and cryptography is investigated. That is a new phenomenon and namely the synchronization of neural networks is leading to a new type of exchange of secret messages.

Another application of neural networks in cryptography was published by Kinzel and Kanter which introduced a way of using neural networks in secret key exchange over a public channel [13]. Klimov, Mityagin and Shamir introduced a method of cryptanalysis to the previous system in [14].

A scheme for remote password authentication was published in [15]. Besides these findings, another development was done on this field and it brought optimization in both, differential and linear cryptanalysis.

A known-plaintext attack was done in [16]. This attack is based on training a neural network to do the decryption process. The neural network is being fed with the ciphertext as its input and the plaintext is considered the reference output. After the training of neural network with a sufficient amount of plaintext-ciphertext pairs that are all encrypted with the same key, the neural network will be able to retrieve the plaintext from the ciphertext that has not been part of the training process, as this ciphertext is encrypted with the same key. Therefore, this type of attack is regarded as a global deduction attack.

3 Experiments

In this section we present and analyze obtained experimental results for one DES round and for full-round DES (16 rounds). The goal of our experiments is to perform a known ciphertext attack with usage of machine learning algorithms, or to be more precise with artificial neural networks. The whole process of the attack with ANNs has three phases:

- **Creation of dataset**
  As previously mentioned our goal is to determine the plaintext messages, based on the ciphertext for a given key. Therefore, we have to create a dataset where for a given key we will have multiple combination of ciphertexts and plaintext. We will do so by creation of a random $m \times n$ binary matrix. Each row in this matrix represents a random block with size n that can be used for encryption with one round of the DES algorithm or for encryption with

all 16 rounds of the DES algorithm. If we create the dataset for one round encryption, then $n = 32$ because the only thing that we need to predict in this case is the one half of the plaintext block that is an argument in the Feistel function. If we create the dataset for full DES encryption, then $n = 64$ because we need to determine the complete block.

$$Output_{m,n} = \begin{pmatrix} plain_{1,1} & plain_{1,2} & \cdots & plain_{1,n} \\ plain_{2,1} & plain_{2,2} & \cdots & plain_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ plain_{m,1} & plain_{m,2} & \cdots & plain_{m,n} \end{pmatrix} \tag{1}$$

Also, we need to create another $m \times n$ binary matrix where the i-th row will correspond to the encryption of the i-th row (block) of the input matrix with the given key.

$$Input_{m,n} = \begin{pmatrix} cipher_{1,1} & cipher_{1,2} & \cdots & cipher_{1,n} \\ cipher_{2,1} & cipher_{2,2} & \cdots & cipher_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ cipher_{m,1} & cipher_{m,2} & \cdots & cipher_{m,n} \end{pmatrix} \tag{2}$$

The matrix (1) will be used as the output of the artificial neural network, and the matrix (2) will be used as the input of the neural network.

– **Splitting the dataset into training and testing subsets**
Due to the complexity of this case, we decided that we need huge amounts of data in order to train good models for decryption of a given ciphertext. Because of the volume of the data, the best method to train and evaluate the model afterwards and reducing the time and computational resources, is the holdout method. We decided to split the complete dataset into two subset, a training dataset that represents randomly selected 80% of the original dataset and testing dataset that contains the rest 20% of the original datasets. The training dataset, can be split again with the ratio of 80-20 in order to validate the created models and tune the hyperparameters of the models.

– **Evaluation of the trained models for the testing dataset**
This is definitely not a regular machine learning problem, therefore we are introducing our own evaluation metrics that will be used to determine the accuracy of the trained models for known ciphertext attacks.
The outside error is calculated with the following formula:

$$Outside\ error = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} predicted(i,j) \oplus original(i,j)}{m * n}$$

where,

- $predicted(i,j)$ is j^{th} bit in the i^{th} block of the predicted text (output of the neural network)

- $original(i, j)$ is j^{th} bit in the i^{th} block of the plaintext (original output)
- m is the number of blocks in the dataset
- n , as previously explained, is the length of the blocks that are used in the input and output matrices.

On the other hand, the total accuracy is the percentage of accurately predicted (decrypted) blocks, i.e.,

$$Total\ accuracy = \frac{\#\text{accurately predicted blocks}}{m}$$

3.1 Experiments with one DES round

Datasets In these experiments we have created 1000 datasets, where each dataset corresponds to a different round 48 bit key. The round keys for each dataset was generated randomly (as the plaintext blocks) and a validation of the keys during their generation was enforced to guarantee that the keys are not weak keys.

Every dataset contained about 2^{19} pairs of (ciphertext, plaintext), where the plaintext and the ciphertext are 32 bit blocks, because the goal in this part is to predict the plaintext for one DES round.

Neural network The experiments were conducted with usage of the scikit-learn python package and the neural network has 4 hidden layers with 128, 256, 256 and 128 nodes in the corresponding layers. For optimization of the weights in the neural networks we use Stochastic gradient descent, for loss function the Mean Square Error is being used and the training was done in 300 epochs. The execution of all 1000 experiments was distributed on different nodes on a supercomputer in order to increase the necessary time. In Table 1 the results for 10 experiments with the smallest outside error, are shown.

Table 1. Experimental results for outside error and total accuracy

Dataset no.	Net. Layout	Outside error	Total accuracy
1	128-256-256-128	0.16379	0.06595
2	128-256-256-128	0.16411	0.06018
3	128-256-256-128	0.16452	0.06159
4	128-256-256-128	0.16471	0.06494
5	128-256-256-128	0.16471	0.06517
6	128-256-256-128	0.16521	0.06001
7	128-256-256-128	0.16522	0.06027
8	128-256-256-128	0.16524	0.06279
9	128-256-256-128	0.16527	0.05869
10	128-256-256-128	0.16543	0.06167

Results with different dataset size In order to demonstrate the importance of the volume of the dataset used in training of the machine learning models we have conducted experiments with different sizes of the dataset. We did this for three randomly selected keys from the 1000 keys that we previously generated, and also we chose subsets from the original dataset with different lengths that were incremented by 50 000 samples in each iteration. We can notice in Fig.1. that as the dataset size increases, the outside error for prediction of the plaintext decreases.

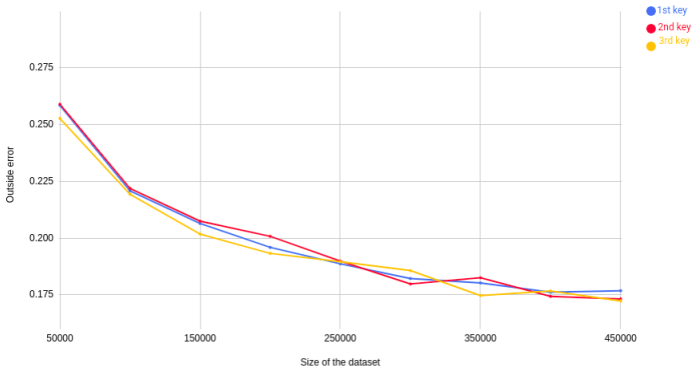

Fig. 1. Visualization of the dependency of the outside error on the dataset size

Also, on Fig.2. we can notice that the total accuracy increases as the size of the database increases.

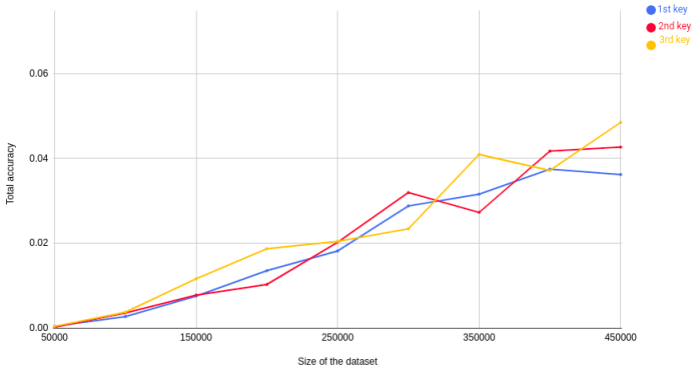

Fig. 2. Visualization of the dependency of the total accuracy on the dataset size

3.2 Experiments with all DES rounds

The main idea was to preform the decrypton of DES. Therefore, after we obtained the results from DES with one round and they shown good outcome, we decide to restructure the neural network and build it for all DES rounds decryption.

Dataset Each dataset used in this case is build of 64-bit inputs and 56-bit key, where one key represents one dataset and the inputs in each are unique. The visual representation of the input and the output of the neural network is given with the following matrices.

We trained the neural network with 2^{17} pairs of (ciphertext, plaintext), where 20% of it are used for validation.

For the testing of the network we use 2^{15} inputs (ciphertexts), that will give predicted plaintext, which is then compared with the original one. The output of the neural network are numbers in the interval [0,1], so we round them to 0 or 1.

Neural network The neural network was built of one input layer where there are 64 encrypted bits as input and appropriately an output layer with 64 bits as output, that is the prediction of the decrypted data. In the middle, there are four hidden layers with different inputs. The network was trained in 150 epochs and the required time is approximately 35-40 minutes.

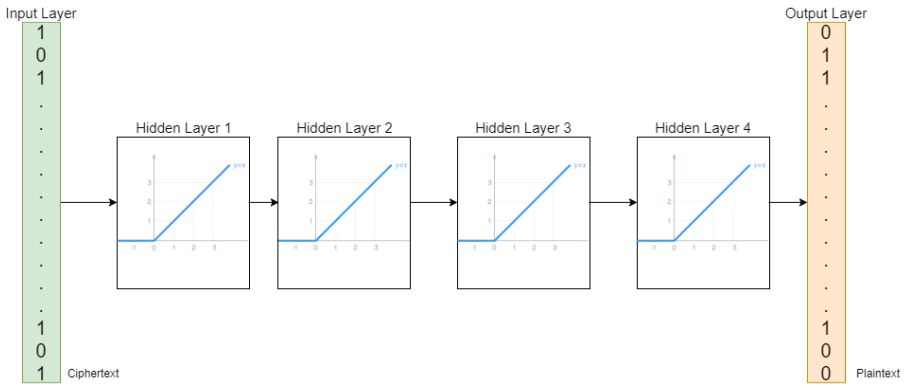

Fig. 3. Visual representation of the neural network

As shown on Fig. 3 we use ReLU as activation function. This function gives best results in practice, because its output is: $f(x) = max(0, x)$. Therefore, the output will be in the interval [0,1], and each number will represent the probability that its value is 1. To translate these probabilities into bits we are using round function that leads to the final result (prediction bits).

To optimize our results we used Stochastic gradient descent. Additionally, we applied the Mean Square Error as a function to minimize the loss.

Results Here, we present experimental results for full-round DES obtained for different neural network structures. For each experiment we calculate the outside error and the total accuracy.

In Table 2 results for the first considered network layout (128-256-256-128) and three different datasets are given.

Table 2. Experimental results for the first network

Dataset no.	Outside error	Total accuracy
1	0.08691	0.01397
2	0.10363	0.00607
3	0.12218	0.00210

From the results for the total accuracy in Table 2 we can conclude that with this network the percentage of accurately decrypted inputs is small. Therefore, we try with different structures of the neural network, that lead to better results presented on Table 3.

Table 3. Experimental results for different network layouts

Dataset no.	Net. Layout	Outside error	Total accuracy
3	256-512-512-256	0.00586	0.97020
1	256-512-512-256	0.00665	0.96673
2	256-512-512-256	0.00732	0.96565
2	128-256-512-256	0.00816	0.92515
1	128-256-512-256	0.00871	0.88050
3	128-256-512-256	0.00916	0.85683

The results in Table 3 showed that the most appropriate inputs for the hidden layers in the network are 256-512-512-256. For this network approximately 97% of the ciphertexts are accurately decrypted. On the other side, for the network with layout 128-256-512-256 this percentage is between 86% and 93%. Also, with the network 256-512-512-256 the values of outside error are smallest.

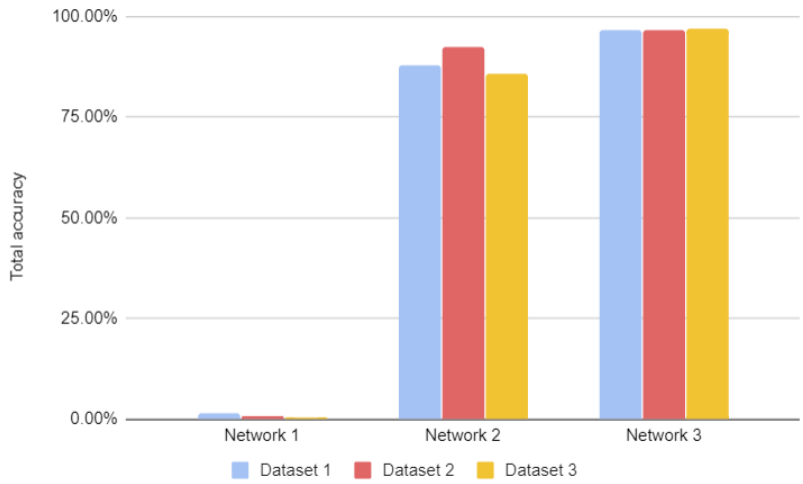

Fig. 4. Visualization of the dependency of the total accuracy on different networks

In Fig.4 we can visually compare results for total accuracy obtained with all three neural networks.

4 Conclusion

In this paper we consider application of machine learning in DES cryptanalysis. We made a lot of experiments for different datasets, different keys and different neural networks. The goal of our experiments is to perform a known ciphertext attack with usage of machine learning algorithms, or to be more precise with the help of neural networks.

We analyse the outside error and the total accuracy, obtained for different neural networks, as measures for the success of the attack. The results show that for a specific neural network, approximately 97% of the ciphertexts are accurately decrypted. This large percentage is confirmation for the success of our cryptanalysis and achievement of the goal.

As a further research, we can perform and analyze this attack for other encryption algorithms.

References

1. Singh, Sombir, Sunil K. Maakar, and Sudesh Kumar. "A performance analysis of DES and RSA cryptography." International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) 2.3 (2013).
2. E. Biham, and A. Shamir, Differential cryptanalysis of DES-like Cryptosystems, Journal of Cryptology, Vol.4, No.1, pp.3-72,1991.
3. M. Matsui, Linear cryptanalysis Method for DES Cipher, in T. Helleseth (Ed.) Advances in Cryptology – EUROCRYPT'93, LNCS 765, pages 386-397, Springer Verlag, 1994.
4. S. K. Langford, Differential-Linear Cryptanalysis and Threshold Signatures, PhD thesis, Stanford University, USA, 1995.
5. E. Biham, New Types of Cryptanalytic Attacks using Related Keys, In T. Helleseth (Ed.) Advances in Cryptology EUROCRYPT'93, LNCS 765, pages 398-409, Springer Verlag, 1994.
6. L. R. Knudsen, Cryptanalysis of LOKI, in H. Imai, R. L. Rivest, and T. Matsumoto (Ed.s) Advances in Cryptology – Asiacrypt '91, LNCS 739, pages 22-35, Springer-Verlag, 1993.
7. T. Shimoyama, and T. Kaneko, Quadratic Relation of S-box and its Application to the Linear Attack of Full Round DSE, in H. Krawczyk (Ed). Advances in Cryptology – Crypto '98, LNCS 1462, pages 200-211. Springer-Verlag, 1998.
8. R. Merkle, M. Hellman, On the Security of Multiple Encryption, Communications of the ACM, Vol. 24, No 7, pp. 465-467, July 1981.
9. P. van Oorchot, and M.J. Wiener, A known-plaintext attack on two-key triple encryption in I. Damgrad (Ed.) Advances in Cryptology – EUROCRYPT'90, LNCS 473, pp.318-325, Springer-Verlag, 1991.
10. W. Diffei and M.E. Hellman, Exhaustive Cryptanalysis of the NBS Data Encryption Standard, Computer, Vol.10, No.6, pp.74-84, June 1997.
11. M. Clark, and D. Blank, A Neural-Network Based Cryptographic System, Proceedings of the 9th Midwest Artificial Intelligence and Cognitive Science Conference (MAICS'98), pp. 91-94 Dayton, USA, 1998.
12. T. Godhavari, N. R. Alamelu, R. Soundararajan, Cryptography Using Neural Network, Proceedings of 2005 Annual IEE INDICON, pp.258-261, 2005.
13. Kanter, W. Kinzel, E. Kanter, Secure exchange of information by synchronization of neural networks, Europhysics Letters, Vol.57, No.1, page 141, 2002.
14. A. Klimov, A. Mityagin, A. Shamir, Analysis of Neural Cryptography, in Y.Zheng (Ed.) Advance in Cryptology – ASIACRYPT 2002, LNCS 2501, pp.288-298, Springer, Berlin, Germany, 2002.
15. L. Li, L. Lin, M. Hwang, A remote password authentication scheme for multi-server architecture using neural networks, IEEE Transactions on Neural Networks, Vol. 12, No.6, pp.1498-1504, Nov.2001.
16. Mohammed M. Alani, Neuro-Cryptanalysis of DES and Triple-DES, Department of Computing, Middle East College, Muscat, Sultanate of Oman, p.2-4, 2011.
17. El-Zoghabi, A., Amr H. Yassin, and Hany H. Hussien. "Survey report on cryptography based on neural network." International Journal of Emerging Technology and Advanced Engineering 3.12 (2013): 456-462.
18. Lucks, S.: Attacking Triple Encryption, Theoretische Informatik, 68131 Mannheim A5, Germany, p.1-15 (1998)