석 사 학 위 논 문
Master's Thesis

라운드 단축 SIMON64/96, CHAM64/128, HIGHT에 대한 신경망 이용 차분공격

Differential Neural Cryptanalysis against Reduced-round SIMON64/96, CHAM64/128, and HIGHT

2020

백 승 근 (白 承 根 Baek, Seunggeun)

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

석 사 학 위 논 문

라운드 단축 SIMON64/96, CHAM64/128, HIGHT에 대한 신경망 이용 차분공격

2020

백 승 근

한 국 과 학 기 술 원

전산학부 (정보보호대학원)

라운드 단축 SIMON64/96, CHAM64/128, HIGHT에 대한 신경망 이용 차분공격

백 승 근

위 논문은 한국과학기술원 석사학위논문으로
학위논문 심사위원회의 심사를 통과하였음

2020년 6월 18일

심사위원장 김 광 조 (인)

심 사 위 원 손 수 엘 (인)

심 사 위 원 이 주 영 (인)

Differential Neural Cryptanalysis against Reduced-round SIMON64/96, CHAM64/128, and HIGHT

Seunggeun Baek

Advisor: Kwangjo Kim

A dissertation submitted to the faculty of
Korea Advanced Institute of Science and Technology in
partial fulfillment of the requirements for the degree of
Master of Science in Computer Science (Information Security)

Daejeon, Korea
June 18, 2020

Approved by

Kwangjo Kim
Professor of School of Computing

The study was conducted in accordance with Code of Research Ethics[1].

[1] Declaration of Ethical Conduct in Research: I, as a graduate student of Korea Advanced Institute of Science and Technology, hereby declare that I have not committed any act that may damage the credibility of my research. This includes, but is not limited to, falsification, thesis written by someone else, distortion of research findings, and plagiarism. I confirm that my thesis contains honest conclusions based on my own careful research under the guidance of my advisor.

MIS 백승근. 라운드 단축 SIMON64/96, CHAM64/128, HIGHT에 대한 신경망 이용 차분공격. 전산학부 (정보보호대학원) . 2020년. 36+iv 쪽. 지도교수: 김광조. (영문 논문)
Seunggeun Baek. Differential Neural Cryptanalysis against Reduced-round SIMON64/96, CHAM64/128, and HIGHT. School of Computing (Graduate School of Information Security) . 2020. 36+iv pages. Advisor: Kwangjo Kim. (Text in English)

초 록

비구별성의 측면에서 보았을 때, 암호체계에 대한 공격은 여러 암호문 사이를, 혹은 암호문과 무작위 값을 구분할 수 있는 효과적인 구별자를 학습하는 과정이라고 볼 수 있다. 예로부터 암호체계 공격과 기계학습 사이의 이론적 연관성이 연구되고 데이터에 기반한 공격 방법이 제시되었지만, 그러한 공격이 실제로 가능하게 된 것은 병렬 처리 하드웨어와 딥 러닝 알고리즘 등 여러 기술이 발전한 최근의 일이다. Gohr는 라운드 단축 경량 블록암호의 차분 특성을 신경망 분류기에 학습시켜 마지막 라운드 키를 얻는 방식을 제안했으나, Speck32/64라는 32비트 경량 블록암호 하나에 대해서만 실험이 진행되었다. 본 논문에서는 64비트의 키 길이를 갖는 (일반화된) Feistel 네트워크에 기반한, SIMON64/96, CHAM64/128, HIGHT 세 가지의 라운드 단축 경량 블록 암호를 대상으로 신경망을 이용한 구별자를 학습시켜 보고, Gohr가 제시한 공격 방식의 확장성과 네트워크의 구조에 따른 학습 가능성 및 정확도를 분석하였다. 또한 다양한 가정 하에서의 구별자의 유형을 제안하고 구분 성능을 실험적으로 평가하였다.

핵 심 낱 말 신경망 이용 차분공격, 데이터 기반 암호분석, 경량 블록암호, 구별자 공격, 딥 러닝

Abstract

From the perspective of indistinguishability, an attack on a cryptosystem can be modeled as a training process of efficient distinguishers between ciphertexts and random values, or among ciphertexts. Though the theoretical relationship between cryptanalysis and machine learning has been studied and data-driven cryptanalysis methods have been proposed, the attacks become practically available recently due to the progress of the technologies including the parallel processing hardware and the deep learning algorithms. Gohr proposed differential neural cryptanalysis by making neural classifiers learn differential properties of a reduced-round lightweight block cipher in order to obtain the final round key. However, only one 32-bit block cipher called Speck32/64 had been evaluated. In this paper, we train neural distinguishers against three 64-bit reduced-round lightweight ciphers with (generalized) Feistel network, including SIMON64/96, CHAM64/128, and HIGHT, to evaluate learnability and accuracy of the attacks. Various models of distinguishers under different assumptions have been proposed, and the performance of each distinguisher has been empirically assessed.

Keywords Differential neural cryptanalysis, data-driven cryptanalysis, lightweight block cipher, distinguishing attack, deep learning

Contents

List of Tables

List of Figures

Chapter 1. Introduction

1.1 Motivation

The data-driven analysis technique is one of the strategies to classify, distinguish, or approximate certain aspects of given data. Machine learning (ML) models provide computational tools and algorithms to learn these aspects to perform data-driven analysis. Often, machine learning models are considered to be capable of efficiently approximating any unknown relationships or functions.

Neural network (NN) is one of the ML models with emerging use, which simplifies and simulates the stimulation mechanism of neurons in brains. Neurons and the connections among the neurons are often placed in the form of layers. Deep learning (DL) is the learning procedure utilizing neural networks with multiple hidden layers; more hidden layers present in the model allows the model to approximate more complex functions.

Though DL models are being applied to every research field, there remains a very challenging class of targets, namely cryptographic primitives. Cryptographic primitives are some of the most complex functions, since finding data-driven relationships often leads to the break of the security requirements of the cryptosystems. Attacking cryptosystems using deep learning is called neural cryptanalysis.

It has been a long time since the utilization of data-driven analysis techniques for design and analysis of cryptosystems was first proposed. Rivest [21] mentioned that cryptography and ML are "sister fields," and many tools in ML can be also utilized to break the target cryptosystems. At that time, such methods were proposed on theoretical interests, due to the lack of computational power to perform actual experiments.

Since the performance of hardware has grown exponentially, now the amount of available computing resources becomes plentiful enough to perform data-driven analysis even for personal computers. Furthermore, advances in the data-driven techniques such as models, network structures in the case of DL models, and various efficient optimization methods allow the possible practicality of neural cryptanalysis.

Despite these advances, however, neural cryptanalysis is still in the early development stage due to the hardness of the tasks analyzing complex cryptosystems. For security, cryptosystems are designed to fulfill some requirements such as indistinguishability. One of the goals of neural cryptanalysis is to tear down the indistinguishability requirement via data-driven measures by teaching some distinguishers. Chapter 3 provides detailed descriptions of such distinguishers.

Some cryptosystems, including many public-key cryptosystems, are designed to have provable security. This indicates that the security of the cryptosystems relies on the difficulty of some NP-hard problems, or problems which are expected to be NP-hard. Breaking these cryptosystems leads to finding solutions for the underlying problems efficiently. In other words, neural cryptanalysis targeting these cryptosystems is equivalent to generating a heuristic measure to solve the underlying problems, which is considered to be intractable. Therefore, the targets of neural cryptanalysis are limited to relatively 'simple' cryptosystems, such as lightweight block ciphers.

A sparse number of previous publications have stated the models of neural cryptanalysis and the actual experimental results on the actual ciphers. Some publications such as Gohr's differential neural cryptanalysis [8] provided meaningful breakthroughs on the methodologies and performance of neural cryptanalysis. Other publications reported failures of the experiment, while some provided interpreta-

tions of results that can be misleading as the success of neural cryptanalysis. Detailed analysis on the related works is written in Chapter 4.

1.2 Problem Statement and Contribution

We provide the following problem statements, which are closely related to our contributions to this thesis.

First, plenty of different distinguishers can be established based on different combinations of datasets that we want to find relationships on. However, each previous work utilized the own model of the assumptions of the distinguishers. No common framework for selecting distinguishers are previously suggested, to the best of our knowledge. Therefore, we suggest the framework by systemizing various distinguisher models for neural cryptanalysis, focusing on indistinguishable-uniform function (IUF)[7] classifiers.

Second, the results of previous attacks are scattered and not been systemized effectively. Therefore, we summarize the previous approaches and applications of neural cryptanalysis on block ciphers. We also raise disputes on the results of some misleading publications, about whether they have proper models and analysis methods.

Finally but most importantly, there is a lack of comparative analysis on the neural cryptanalysis results for lightweight block ciphers. We performed neural cryptanalysis against several lightweight block ciphers including SIMON64/96 [4], CHAM64/128 [14], and HIGHT [11]. We limited the target cryptosystems into reduced-round 64-bit lightweight block ciphers having generalized Feistel network (GFN) structure. We observe and compare the performance of the attacks based on the configurations of GFN of the ciphers. We provide more analysis on how DL becomes effective on the block cipher and discuss whether DL is practically better than another ML model for the cryptanalysis tasks.

1.3 Organization

This paper is organized as follows. Chapter 2 provides some background information for the fields and theories related to neural cryptanalysis. Chapter 3 introduces attack scenarios or distinguishers that can be used for data-driven cryptanalysis of block ciphers. Chapter 4 describes previous results on neural cryptanalysis, as well as some criticism on misleading results.

Chapter 5 describes our neural cryptanalysis results on various lightweight ciphers. Chapter 6 provides some discussion on how the neural classifiers work, and provide several aspects of neural classifiers. Conclusions and future work are mentioned in Chapter 7.

Chapter 2. Background

2.1 Notations

The thesis is based on the following list of notations in Table 2.1.

Table 2.1: Notations

Domain	Notation	Description
Block Cipher	m	Key length, in bits
	K	The key space, equivalent to $\{0,1\}^m$
	n	The block size, in bits
	M	The message space, equivalent to $\{0,1\}^n$
	r	The round key size, in bits
	k	A key instance
	p	A plaintext instance
	c	A ciphertext instance, or a function output
	Enc, Enc^{-1}	The encryption and decryption algorithms
	Enc_k, Enc_k^{-1}	The encryption and decryption oracles with the key k
	$\&$	Bitwise AND operation
	$\oplus$	Bitwise XOR operation
	$\boxplus$	Arithmetic addition, within the corresponding processing unit
	$<< v$	Rotation, to v bits left
Distinguishing Games	A	Adversary
	A^{F_k}	Adversary, who is able to access an oracle F_k
	Adv(A)	Advantage of the adversary
	C	Challenger
	Pr[event]	Probability that the event occurs
	x	Challenger's binary choice from $\{0,1\}$; often 1 for the challenges generated by the adversary's choice and 0 for the randomly generated challenges.
	x'	Adversary's guess on the challenger's choice
	s	Adversary's state information
	Δ	An XOR difference constant for differential distinguishers
Machine Learning	τ	Portion of training data from the entire dataset
	α	Accuracy of a model
	$B(T,P)$	Binomial distribution with T trials and P probability
	$N(\mu,\sigma^2)$	Gaussian distribution with the average μ and the standard deviation σ
Related Work	?	Unspecified in the corresponding literature

2.2 Block Cipher

Block ciphers are symmetric cryptosystems that perform encryption and decryption by the unit of blocks. The structures and types of block ciphers are discussed in this section. Differential cryptanalysis, which is one of the popular and powerful cryptanalysis techniques, is also explained.

2.2.1 Structures of Block Cipher

In block ciphers, the blocks undergo several repeated steps of confusion and diffusion. The set of repetitive steps is called a round of the block cipher. The round keys, which are the values expanded from the key according to the key scheduling algorithm, are properly mixed on the rounds of the block cipher.

Substitution-permutation network (SPN) is a relatively straightforward way to repeat confusion and diffusion. A round of an SPN cipher can be described as a direct concatenation of the three steps: XORing a round key, applying substitution boxes (S-boxes), and applying a fixed bitwise permutation. In the step involving S-boxes, the block is divided into subblocks having d bits, and each subblock is converted using the S-box, which is a nonlinear bijective function from $\{0,1\}^d$ to $\{0,1\}^d$. The inverse S-box and the inverse permutation are used for the decryption.

The Feistel network is the core structure of several block ciphers such as Data Encryption Standard (DES). In Feistel network with a round key k_i, the block on the round i is divided into two processing units (half-blocks) (X_i, Y_i). F is a nonlinear round function taking a half-block and a key as arguments and produces a half-block, which is not required to be invertible. The followings are the steps performed on each round of the Feistel cipher.

$$X'_i = X_i \oplus F(Y_i, k_i), Y'_i = Y_i \tag{2.1}$$

$$X_{i+1} = Y'_i, Y_{i+1} = X'_i \tag{2.2}$$

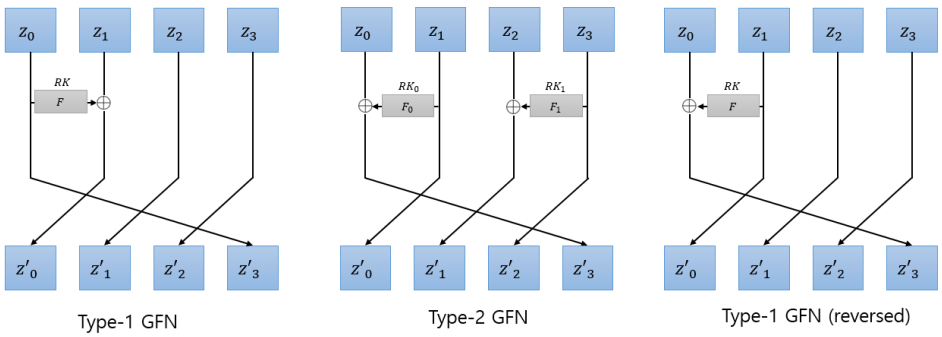


Figure 2.1: Structures of type-1 and type-2 GFN ciphers [10, Fig. 1]

The generalized Feistel network (GFN) provides extensions of the Feistel network. The type-1,2,3 ciphers suggested by Zheng et al. [28] are popular GFN structures in the design of block ciphers, which are depicted in Figure 2.1. More numerous variants of Feistel networks exist, such as unbalanced Feistel networks [23] or alternating Feistel networks using alternating round functions [18]. These variants are often considered as examples of GFN structures. [10]

2.2.2 Classification of Block Cipher by Complexity

General-purpose ciphers are designed for general applications for protecting the confidentiality of data. DES is one of the first general-purpose ciphers, though the scheme's security has been broken by various attacks including differential cryptanalysis and linear cryptanalysis. Currently, AES(Rijndael, the finalist in the AES competition) is in the most widespread use. Other than AES, plenty of general-purpose block ciphers have been proposed and/or currently in use, including MARS, RC6, Serpent, and Twofish.

General-purpose ciphers often utilize complex calculations or require storage to store precomputed S-boxes. However, data encryption is sometimes required in restricted environments. For instance, various use cases involving sensor networks and IoT devices such as a tiny implant sensor that capturing and exporting biosignals for medical purposes, require low-cost encryption in the terms of time, power, and memory. In these situations, using lightweight ciphers would be more appropriate.

Lightweight ciphers rely on relatively simple operations such as shifts, bitwise binary operations, and arithmetic operations. For example, lightweight block ciphers using only addition, rotation, and XOR operations to perform encryptions are called ARX ciphers.

Toy ciphers are block ciphers that having small key and block sizes and simple structures. Therefore, toy ciphers are not considered to be secure and rarely used in real life. Simplified Data Encryption Standard (SDES) [22] is one of the well-known toy ciphers. Toy ciphers are mostly used for educational purposes to teach how block ciphers work. Sometimes, toy ciphers are used for demonstration purpose of new cryptanalysis methods, as proofs of concepts. Plenty of earlier neural cryptanalysis publications demonstrated experiments trying to attack SDES as the proof of concept.

2.2.3 Target Ciphers

Aspects of the target lightweight block ciphers of this thesis are described here.

SIMON64/96 SIMON [4] is a family of lightweight Feistel ciphers. SIMON was designed by the National Security Agency, concurrently with Speck ciphers. As a family of lightweight ciphers, SIMON utilizes rotations, XORing, and bitwise AND operations to generate the following Feistel function.

$$F_{SIMON}(z,k) = ((z << 1)\&(z << 8)) \oplus (z << 2) \oplus k \tag{2.3}$$

We used SIMON64/96, a SIMON cipher having 64-bit block size and 96-bit key size, as our target. Full-round SIMON64/96 encryption takes 42 rounds.

CHAM64/128 CHAM64/128 [14] is a type-1 reversed alternating GFN having 64-bit block size and 4 processing units (16 bits each). Full-round CHAM64/128 encryption takes 80 rounds. CHAM64/128 is an ARX cipher; the operations are defined using arithmetic addition, rotation, and XORing. Different from the traditional Feistel ciphers, CHAM64/128 XORs the round number and mixes the blocks using arithmetic additions instead of XORing. Two types of operations are used in an alternating manner, as shown in Figure 2.2a.

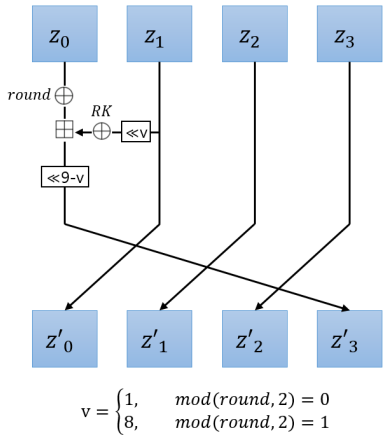
HIGHT HIGHT [11] is a type-2 GFN cipher having 64-bit block size and 8 processing units (8 bits each). Full-round HIGHT encryption takes 32 rounds. HIGHT applies two types of Feistel-like functions

composed of ARX operations as follows.

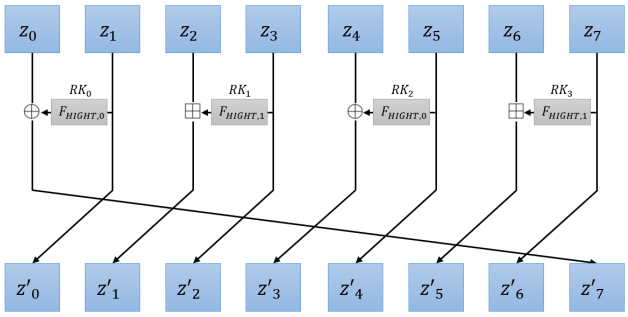
$$F_{HIGHT,0}(z,k) = (z <<< 1) \oplus (z <<< 2) \oplus (z <<< 7) \boxplus k \tag{2.4}$$

$$F_{HIGHT,1}(z,k) = (z <<< 3) \oplus (z <<< 4) \oplus (z <<< 6) \oplus k \tag{2.5}$$

Being a type-2 GFN, HIGHT invokes the functions on each pair of the processing units for every round, as in Figure 2.2b. Note that the results of $F_{HIGHT,1}$ are mixed to the corresponding processing unit using arithmetic addition instead of XORing.



(a) One round of CHAM64/128 [14, Fig. 1]



(b) One round of HIGHT [11, Fig. 3]

Figure 2.2: Round structures of CHAM64/128 and HIGHT, the target GFN ciphers

2.2.4 Security Notions of Block Cipher

Block ciphers are commonly modeled as super-pseudorandom permutations (SPRP), which were introduced by Luby and Rackoff [17]. A function $F : K \times M \to M$ having the key space K and the message space M should satisfy the followings to be an SPRP.

$$\text{(Permutation) } \forall k \in K, F_k : M \to M \text{ is a bijection.} \tag{2.6}$$

$$\text{(Efficiency) } \forall k \in K, F_k \text{ and } F_k^{-1} \text{ are efficiently computable.} \tag{2.7}$$

$$\text{(Super-pseudorandomness) } F \text{ does not have any super distinguishing circuit family.} \tag{2.8}$$

Roughly, a circuit is a super distinguishing circuit if the circuit is capable of correctly deciding whether the given (**normal, inverse**) oracles of a permutation came from (F_k, F_k^{-1}), or otherwise randomly selected (σ, σ^{-1}) from the set of possible permutations for M, with a significant advantage. A weaker notion called pseudorandomness can be similarly defined using only the target function F_k without involving the inverse F_k^{-1}.

Pseudorandomness and super-pseudorandomness are often compared to the IND-CPA and (IND-CPA + IND-CCA) notions of security in probabilistic cryptosystems, respectively. This is because both encryption and decryption oracles are given for the adversary's attempt to distinguish oracles in the definition of super-pseudorandomness, while only encryption oracle is given in the definition of pseudorandomness. However, most block ciphers are designed to have a similar structure for their encryption and decryption for the purpose of efficient computation. Therefore, it is often enough to analyze the cipher's security in the context of pseudorandomness, instead of super-pseudorandomness.

Desai and Miner [7] provided a useful security definition somewhat relaxed from pseudorandom permutations to capture pseudorandomness.

Definition 2.1. (Indistinguishable-uniform Permutation [7, Def. 2]) For a polynomial-time adversary A with two algorithms **find** and **guess**, $x \in \{0,1\}$, and a permutation generator $F : K \times M \to M$ having the key space $K = \text{Keys}(F)$ and the message space M, an indistinguishable-uniform permutation (IUP) experiment $\text{Exp}_F^{\text{IUP}}(A, x)$ is defined as follows.

$$k \leftarrow K \tag{2.9}$$

$$(p, s) \leftarrow A^{F_k}(\textbf{find}) \text{ while } p \text{ is not queried to } F_k \tag{2.10}$$

$$c_0 \leftarrow F_k(p); c_1 \xleftarrow{random} M \tag{2.11}$$

$$\text{return } x' := A(\textbf{guess}, c_x, s) \tag{2.12}$$

F is said to be an indistinguishable-uniform permutation if the advantage of the adversary A to guess x correctly, which is described as the following formula, is negligible.

$$\text{Adv}_F^{\text{IUP}}(A) = \Pr[\text{Exp}_F^{\text{IUP}}(A, 0) = 0] - \Pr[\text{Exp}_F^{\text{IUP}}(A, 1) = 0] \tag{2.13}$$

Definition 2.2. (Indistinguishable-point Permutation [7, Def. 2]) Based on the same notations and assumptions to Def. 2.1, an indistinguishable-point permutation (IPP) experiment $\text{Exp}_F^{\text{IPP}}(A, x)$ is defined as follows.

$$k \leftarrow K \tag{2.14}$$

$$(p_0, p_1, s) \leftarrow A^{F_k}(\textbf{find}) \text{ while } p_0, p_1 \text{ are not queried to } F_a \tag{2.15}$$

$$c \leftarrow F_k(p_x) \tag{2.16}$$

$$\text{return } x' := A(\textbf{guess}, c, s) \tag{2.17}$$

F is said to be an indistinguishable-point permutation if the advantage of the adversary A to guess b correctly, which is described as the following formula, is negligible.

$$\text{Adv}_F^{\text{IPP}}(A) = \Pr[\text{Exp}_F^{\text{IPP}}(A, 0) = 0] - \Pr[\text{Exp}_F^{\text{IPP}}(A, 1) = 0] \tag{2.18}$$

The authors showed that a PRP can be reduced into IUP, but there is no polynomial-time reduction from IUP to PRP. They also demonstrated IPP and IUP are equivalent definitions, by showing polynomial-time reductions from IPP to IUP and vice versa.

2.2.5 Differential Cryptanalysis of Block Cipher

Differential cryptanalysis (DC) is the analysis of XOR differences of ciphertext or intermediate values from the two plaintexts having a certain fixed difference. Biham and Shamir [5] first introduced differential cryptanalysis, and demonstrated the technique against DES which was successful.

Traditional but effective approaches of differential cryptanalysis focus on obtaining differential characteristics of nonlinear components of block ciphers which occur in high probability. Practically, the differential characteristics are found with computational methods such as mixed-integer linear programming (MILP). However, in current data-driven differential cryptanalysis methods, differences are mainly utilized as they can cancel out the round keys during the encryption procedures. The difference alone is not enough for the total break of ciphers. Still, more rounds of the cipher can be distinguished using differentials compared to the use of raw plaintext and ciphertext pairs only.

The known differential characteristics of reduced-round target ciphers are described in Table 2.2.

Table 2.2: Known differential characteristics of reduced-round target ciphers

Cipher	Rounds / Full Rounds	Prob. of a differential characteristic	References
SIMON64/96	10/42	2^{-18} (Related-key)	[25]
	11/42	2^{-22} (Related-key)	[25]
	12/42	2^{-30} (Related-key)	[25]
CHAM64/128	28/80	2^{-39}	[14]
	29/80	2^{-41}	[14]
	39/80	2^{-63}	[14]
HIGHT	11/32	2^{-45}	[27]
	12/32	2^{-53}	[27]
	13/32	2^{-61}	[27]

2.3 Machine Learning

2.3.1 Procedures of Supervised Learning

Supervised learning is a learning scenario when each instance of training data contains a corresponding label. The goal of supervised learning is to make the model predict the correct label of the training and testing data.

The feature extraction step is a preprocessing step that manipulates raw data into the model input. In neural cryptanalysis, each bit of the training data is encoded as zero or one and fed into the neurons in the input layer.

The training step is a procedure to optimize the parameters of the model so that the designated loss function becomes minimized. In binary classification tasks that most distinguishing attacks belong to, the final layer produces probabilities whether the predictions are zero or one. Mean squared error (MSE) of the classification results from the ground truth is frequently used as the loss function.

When all given training data has been used for training, an epoch of training is over. Validation takes place using designated validation data or randomly selected cross-validation data. Training continues until the validation loss converges or the designated number of epochs is reached. Test accuracy and loss of testing data, which is independent of the training data, is measured as the final evaluation. In a binary classification task on a balanced dataset, test accuracy of α =50% and MSE test loss of 0.25 is the baseline of random guessing.

2.3.2 Neural Network

Neural networks consist of layers of neurons and connections among them. Each connection has a weight value while each neuron has a bias value as the parameters. The activation level of each neuron is determined by the weighted sum of the incoming connections plus the bias of the current neuron, while an activation function is applied to the gross sum. Parameters are adjusted to reduce the loss by calculating gradients, which are often back-propagated from the output neuron.

Plenty of topologies of neuron connections has been utilized according to the target problem. Convolutional neural networks and recurrent neural networks are used for the problems having localities such as images, and sequential data such as sound, respectively. Fully connected networks and residual net-

works are often used for learning from general problems with independent features; inputs and outputs of cryptosystems would fall in this category.

2.3.3 Other Machine Learning Models

Outside deep learning, there exist several machine learning models including decision tree, random forest, naive Bayes, and support vector machine.

A decision tree has a target feature and a threshold value for each node so that an instance of the data undergoes a decision process along the edges of the tree to get to the prediction value. Random forests are ensemble models made out of several decision trees. These tree-based classifiers often have excellent performances with the tasks having relatively few features, often numeric, as they make one decision using a single feature.

Compared to deep learning models, it is often perceived that other machine learning models are less capable of capturing the complex nonlinear relations between features and the label.

Chapter 3. Cryptanalysis Scenarios of Block Ciphers

We derived several cryptanalysis scenarios of block ciphers. Most of the scenarios are described as interactive games between an adversary A and a challenger C.

3.1 Basic Distinguishers

In the perspective of machine learning, deploying a distinguishing attack is equivalent to training a classifier to solve a supervised classification task of the appropriate output range. A classifier is called a successful distinguishing attacker if the winning probability of a game using the classification result is non-negligibly higher than the success probability of random guessing. Here, the experiment in definitions 2.1 and 2.2 are rephrased as the game format.

3.1.1 Ability of the Adversary

By default, the scenarios assume distinguishing attacks on the chosen-plaintext attack (CPA) model. The adversary selects the plaintext and obtains the encryption result using the encryption oracle Enc_k. The adversary tries to train a distinguishing model for the key k.

Some of the distinguisher notions assume known-plaintext attack (KPA), meaning that the concrete encryption oracle is not given (though the adversary can exploit the knowledge of cipher algorithm Enc). The adversary would try to attack the scheme using given (plaintext, ciphertext) pairs. The adversary may train a distinguishing model for general keys, that requires both plaintext and ciphertext to evaluate, by generating data on the adversary's own using Enc and randomly selected keys. For simplicity, these KPA distinguishers are suffixed by a prime symbol (').

Other distinguisher notions assume ciphertext only attack (COA), meaning that ciphertexts encrypted from uniformly random plaintexts are provided to the adversary, and the adversary cannot use the plaintexts as the training data. The adversary may train a distinguishing model to decide whether the ciphertext (or the function output) is valid, by generating data on the adversary's own using Enc and randomly selected keys. For simplicity, these COA distinguishers are suffixed by an asterisk symbol (*).

COA is not applicable for the encryption of block ciphers as they are permutations, meaning that any of the given ciphertexts can be decrypted into a valid plaintext. However, if it comes to other types of functions, a COA (output only attack is more accurate as the function is not the encryption and the output is not the ciphertext, but using some abuse of notations) could take place. Thus, we do not construct IPP* and IUP* games over permutations.

3.1.2 IPP Distinguishers

Having the IPP security is a required condition for an encryption oracle Enc_k to become a PRP. IPP describes an effort to distinguish two ciphertexts, or equivalently, distinguish a ciphertext from the encryption of a random value. In Definition 2.2, key selection and generation of a challenge are the roles of the challenger, while executing algorithms of **find** and **guess** is the role of the adversary. In the case

the adversary 'cheats', i.e. if the plaintext in the challenge is ever queried to the encryption oracle by the adversary, the adversary loses the game.

The attack is successful if the adversary's winning probability is non-negligibly greater than $1/2$. The detailed description of the IPP game is displayed in Table 3.1.

Table 3.1: IPP game

Challenger C		Adversary A
Pick $k \leftarrow \{0,1\}^m$	$\rightarrow$	Access Enc_k
Pick $x \leftarrow \{0,1\}$	$\leftarrow p$	Pick $p \leftarrow \{0,1\}^n$
$\begin{cases} p' \leftarrow \{0,1\}^n, c := Enc_k(p') & \text{if } x = 0 \\ c := Enc_k(p) & \text{if } x = 1 \end{cases}$	$c \rightarrow$	c
A wins, if $x = x'$ and p has not been queried to Enc_k by A.	$\leftarrow x'$	Guess x'

An IPP-KPA game, or simply IPP' can be constructed if the oracle Enc_k is not given for the adversary. The additional losing condition of the challenger which exists on IPP does not exist on IPP'. Similarly, the IPP' adversary is successful if the adversary's winning probability is significantly greater than $1/2$. The IPP' game is summarized in Table 3.2.

Table 3.2: IPP' game

Challenger C		Adversary A
Pick $k \leftarrow \{0,1\}^m$, $p \leftarrow \{0,1\}^n$, $x \leftarrow \{0,1\}$	$p \rightarrow$	p
$\begin{cases} p' \leftarrow \{0,1\}^n, c := Enc_k(p') & \text{if } x = 0 \\ c := Enc_k(p) & \text{if } x = 1 \end{cases}$	$c \rightarrow$	c
A wins, if $x = x'$.	$\leftarrow x'$	Guess x'

A successful IPP' adversary is a successful IPP adversary, and the reverse does not hold. One may trivially construct a successful IPP adversary from a successful IPP' adversary by query nothing to the given oracle Enc_k and delegate the challenge to the successful IPP' adversary to obtain a significant distinguishing advantage.

3.1.3 IUP Distinguishers

Different but equivalent to IPP, IUP (Definition 2.1) describes an effort to distinguish a ciphertext from a random value, to reflect the randomness of ciphertext. IUP and IUP' games can be similarly described to IPP and IPP' games, except the difference between 'a random value' and 'the encryption of a random value'. The description of the IUP and IUP' games are summarized in Tables 3.3 and 3.4, respectively. From the perspective of machine learning, this game also can be modeled into a supervised binary classification task, as same as other binary distinguishers.

The difference between the descriptions of IPP and IUP games, i.e. 'a random value' and 'the encryption of a random value', generates a quite major difference in how to interpret the distinguishing

Table 3.3: IUP game

Challenger C		Adversary A
Pick $k \leftarrow \{0,1\}^m$	$\rightarrow$	Access Enc_k
Pick $x \leftarrow \{0,1\}$	$\leftarrow p$	Pick $p \leftarrow \{0,1\}^n$
$\begin{cases} c \leftarrow \{0,1\}^n & \text{if } x = 0 \\ c := Enc_k(p) & \text{if } x = 1 \end{cases}$	$c \rightarrow$	c
A wins, if $x = x'$ and p has not been queried to Enc_k by A.	$\leftarrow x'$	Guess x'

task. In the IUP games, the 'random' challenges are generated by selecting a value from the codomain of the encryption. However, in the IPP games, the challenges are generated by non-uniformly picking a random value from the range of the encryption. Though the two games are the same for an ideal block cipher, the classifiers trained on a flawed block cipher (for example, a reduced-round cipher) may exhibit slightly different accuracy patterns.

Table 3.4: IUP' game

Challenger C		Adversary A
Pick $k \leftarrow \{0,1\}^m$, $p \leftarrow \{0,1\}^n$, $x \leftarrow \{0,1\}$	$p \rightarrow$	p
$\begin{cases} c \leftarrow \{0,1\}^n & \text{if } x = 0 \\ c := Enc_k(p) & \text{if } x = 1 \end{cases}$	$c \rightarrow$	c
A wins, if $x = x'$.	$\leftarrow x'$	Guess x'

3.2 Differential Distinguishers

Instead of training models to learn ciphertexts, one may train classifiers to learn on differentials of two related ciphertexts, in the perspective of differential cryptanalysis.

3.2.1 Difference Functions

An attacker may select a constant Δ indicating the XOR difference of two plaintexts for the different functions. The oracle Enc_k in the previous definitions of IPP(') and IUP(') games could be replaced into following difference functions which can be calculated using oracle invocations.

$$\text{(Difference, 'd')}\ Dif_{\Delta,k}(p) := Enc_k(p) \oplus Enc_k(p \oplus \Delta) \tag{3.1}$$

$$\text{(Concatenation, 'c')}\ Concat_{\Delta,k}(p) := Enc_k(p)||Enc_k(p \oplus \Delta) \tag{3.2}$$

$$\text{or equivalently, } Concat'_{\Delta,k}(p) := Dif_{\Delta,k}(p)||Enc_k(p) \tag{3.3}$$

Note that outputs of $Concat$ and $Concat'$ functions are mutually convertible via a simple XOR operation. However, actual performances of the distinguishers trained using the two different but equivalent forms

of data could be practically different as the data format may exert influence on how well the model captures the meaningful combination of the input.

Finally, we can construct a variant of the cIUF game, namely a mIUF game, by altering the challenge slightly. While *Concat* is applied when $x = 1$ as the same as the cIUF game, a random mask value is applied to the challenge instead of generating new random values when $x = 0$.

$$\text{(Masked, 'm')}\ Mask_{\Delta,k}(p, R) := Enc_k(p) \oplus R || Enc_k(p \oplus \Delta) \oplus R \tag{3.4}$$

$$\text{or equivalently, } Mask'_{\Delta,k}(p, R) := Dif_{\Delta,k}(p) || R \tag{3.5}$$

Note that in mIUF distinguishers, even though outputs of $Mask$ and $Mask'$ functions are not mutually convertible, the two functions have the same distinguishing capabilities. If we define $Mask''$ by the following equation.

$$Mask''_{\Delta,k}(p, R) := Dif_{\Delta,k}(p) || Enc_k(p) \oplus R \tag{3.6}$$

One can notice that $Mask$ and $Mask''$ could be mutually convertible via a simple XOR operation. Furthermore, since the uniformly random block R can completely randomize the $Enc_k(p) \oplus R$ term and there are no other dependencies of R on the data, $Mask'$ and $Mask''$ are equivalently distinguishable.

3.2.2 IUF and IPF Distinguishers

The difference functions are not pseudorandom permutations any more, and they should be modeled as pseudorandom functions. Therefore, definitions of IPF and IUF are used instead of IPP and IUP, respectively, to describe distinguishers for these difference functions. Note that while IUP and IPP are equivalent, IUF can be reduced to IPF but they cannot vice versa.

The structures of the IUP game and the IUF game are identical, except that the permutation under test (encryption oracle) is replaced to the function under test (such as the difference functions in 3.2.1).

Depending on the difference functions used in the games, cIUF, cIPF, dIUF, and dIPF games can be defined, as well as their KPA counterparts suffixed by a prime symbol.

Finally, IUF-COA games, or IUF* games, can be defined by following, when f is the any of the random function currently distinguishing.

Table 3.5: IUF* game on a function f

Challenger C		Adversary A
Pick $k \leftarrow \{0,1\}^m$, $p \leftarrow \{0,1\}^n$, $x \leftarrow \{0,1\}$		
$\begin{cases} c \leftarrow \{0,1\}^n & \text{if } x = 0 \\ c := f(p) & \text{if } x = 1 \end{cases}$	$c \rightarrow$	c
A wins, if $x = x'$.	$\leftarrow x'$	Guess x'

3.2.3 List of Differential Distinguishers

The list of distinguishability notions defined or mentioned so far are summarized in Table 3.6.

Table 3.7 provides a summary of how the challenges are generated for each differential distinguishing game. The adversary should distinguish the $x = 1$ cases from the $x = 0$ cases in order to win the corresponding distinguishing game.

Table 3.6: The family of security notions for distinguishing games

Distinguishing		Permutation	Function		
Type	Security	Enc_k [1]	$Concat_{\Delta,k}$ (c)	$Dif_{\Delta,k}$ (d)	$Mask_{\Delta,k}$ (m) [2]
Point	CPA	IPP	cIPF	dIPF	-
	KPA (')	IPP'	cIPF'	dIPF'	-
	COA (*) [3]	-	-	-	-
Uniform	CPA	IUP	cIUF	dIUF	mIUF
	KPA (')	IUP'	cIUF'	dIUF'	mIUF'
	COA (*)	-	cIUF*	dIUF*	mIUF*

[1] IPP* and IUP* games are impossible to break; any given ciphertext can be the encryption of a valid plaintext due to the property of permutations.

[2] Only mIUF(',*) games are defined since they are modified from the cIUF(',*) games.

[3] IPP* and IPF* games are impossible to break; while one may distinguish $(p, f(p))$ and $(p, f(R))$, $f(p)$ and $f(R)$ cannot be distinguished if the challenger also generates the p value from the random and does not release p to the adversary.

Table 3.7: The family of distinguishers on differentials, with the key k and the differential Δ

Games [1]	Distinguishes... $(x = 1)$	From... $(x = 0)$ [2]
cIPF(')	$p\|\|Enc_k(p)\|\|Enc_k(p \oplus \Delta)$	$p\|\|Enc_k(R)\|\|Enc_k(R \oplus \Delta)$
cIUF(')	$p\|\|Enc_k(p)\|\|Enc_k(p \oplus \Delta)$	$p\|\|R\|\|R_2$
$\leftrightarrow$	$p\|\|Dif_{\Delta,k}(p)\|\|Enc_k(p)$	$p\|\|R\|\|R_2$
cIUF*	$Enc_k(p)\|\|Enc_k(p \oplus \Delta)$	$R\|\|R_2$
$\leftrightarrow$	$Dif_{\Delta,k}(p)\|\|Enc_k(p)$	$R\|\|R_2$
dIPF(')	$p\|\|Dif_{\Delta,k}(p)$	$p\|\|Dif_{\Delta,k}(R)$
dIUF(')	$p\|\|Dif_{\Delta,k}(p)$	$p\|\|R$
dIUF*	$Dif_{\Delta,k}(p)$	R
mIUF(')	$p\|\|Enc_k(p)\|\|Enc_k(p \oplus \Delta)$	$p\|\|Enc_k(p) \oplus R\|\|Enc_k(p \oplus \Delta) \oplus R$
$\leftrightarrow$	$p\|\|Dif_{\Delta,k}(p)\|\|Enc_k(p)$	$p\|\|Dif_{\Delta,k}(p)\|\|R$
mIUF*	$Enc_k(p)\|\|Enc_k(p \oplus \Delta)$	$Enc_k(p) \oplus R\|\|Enc_k(p \oplus \Delta) \oplus R$
$\leftrightarrow$	$Dif_{\Delta,k}(p)\|\|Enc_k(p)$	$Dif_{\Delta,k}(p)\|\|R$

[1] $\leftrightarrow$ indicates an equivalent definition to the above row.

[2] R and R_2 indicate blocks uniformly drawn from the message space M.

3.3 Other Attack Scenarios

Other than the distinguishing attacks, there exist various attack scenarios against target ciphers.

3.3.1 Key Recovery (KR)

Key recovery attack is the ultimate goal of cryptanalysis. Attackers try to find the fixed key k when the encryption oracle Enc_k, and possibly the decryption oracle Enc_k^{-1} are given in a KR game. The KR game is defined in Table 3.8.

Table 3.8: KR game

Challenger C		Adversary A
Picks $k \leftarrow \{0,1\}^m$	$\rightarrow$	Accesses Enc_k and possibly Enc_k^{-1}.
		Evaluate the oracles with any polynomial number of queries, by choice.
A wins, if $k = k'$	$\leftarrow k'$	Guesses k'

3.3.2 Round Key Recovery (RKR)

While key recovery attacks often require the inversion of key scheduling, the round key recovery attack (RKR) is a weaker version of key recovery attacks. In the RKR/F game, the objective of the attacker is to find one of the fixed round keys of the given oracle. In the varying key situation, the attacker should find one of the round keys used in the encryption (RKR game) as described in Table 3.9.

Table 3.9: RKR game

Challenger C		Adversary A
Picks $k \leftarrow \{0,1\}^m$	$\rightarrow$	Accesses Enc_k and possibly Enc_k^{-1}.
		Evaluate the oracles with any polynomial number of queries, by choice.
A wins, if s is the jth round key scheduled from the key k.	$\leftarrow s, j$	Guesses s, j

Constructing a successful RKR attacker for the final round key is possible by utilizing a successful IUP attacker on one-round-reduced version of the cipher with 2^r additional queries. For all 2^r candidates of the final round key, the ciphertext in the challenge can be decrypted by one round and put into the one-reduced-round IUP attacker together with the plaintext. If the IUP attacker reports that the decrypted ciphertext seems to be a valid one, the round key candidate used to decrypt the ciphertext by one round is likely to be the correct one. If the IUP distinguisher model provides probability values, the adversary can select the round key candidate with the highest probability value to be a valid ciphertext.

3.3.3 Plaintext Recovery (PR)

Plaintext recovery attacks are attempts to mimic the decryption oracle of a given encryption oracle, which is the inversion task of a given function. The attacker can access to the encryption oracle Enc_k of the target cipher. The challenger makes a ciphertext as the challenge, and the attacker should guess the plaintext corresponding to the challenge, as described in Table 3.10.

Table 3.10: PR game

Challenger C		Adversary A
Picks $k \leftarrow \{0,1\}^m$	$\rightarrow$	Accesses Enc_k
Picks $p \leftarrow \{0,1\}^n$, $c := Enc_k(p)$	$c \rightarrow$	c
A wins if $p = p'$	$\leftarrow p'$	Guesses p'

3.3.4 Encryption Emulation (EE)

Encryption emulation attacks are attempts to mimic the given encryption oracle. The attacker can access to an encryption oracle Enc_k of the target cipher. The interaction between the challenger and the attacker is described in Table 3.11.

Table 3.11: EE game

Challenger C		Adversary A
Picks $k \leftarrow \{0,1\}^m$	$\rightarrow$	Accesses Enc_k
Picks $p \leftarrow \{0,1\}^n$ that not queried by A.	$p \rightarrow$	p
A wins, if $c' = Enc_k(p)$ and p has not been queried to Enc_k	$\leftarrow c'$	Guesses c'

3.3.5 Plaintext Identification (PI)

Given a ciphertext, plaintext identification attacks are attempts to find out which group the corresponding plaintext belongs to. The attacker can access to an encryption oracle Enc_k of the target cipher. Given a t-partition $\{P_i\}$ of $\{0,1\}^n$, the adversary should guess which partition the corresponding plaintext of the challenge ciphertext comes from, as shown in Table 3.12. This task is a supervised t-classification task.

Table 3.12: PI game for a t-partition $\{P_i\}$ of $\{0,1\}^n$

Challenger C		Adversary A
Picks $k \leftarrow \{0,1\}^m$, $x \leftarrow \{0,1,...,t-1\}$		Accesses Enc_k
$p \leftarrow P_x$, $c := Enc_k(p)$	$c \rightarrow$	c
A wins, if $x = x'$	$\leftarrow x'$	Guesses x'

Chapter 4. Related Work

4.1 Models of Neural Cryptanalysis

One of the major contributions of Albassal and Wahdan [3] is to first suggest the use of distinguishers for the construction of a successful RKR attacker, under the assumption of wrong-key randomization hypothesis.

Gohr [8] claimed that the hypothesis is too ideal for lightweight ciphers, and proposed the use of wrong-key response profiles which can be empirically enough to derive a successful RKR attacker from a one-round-reduced successful cIUF* attacker. While first introducing differential neural cryptanalysis, this is the first successful neural cryptanalysis attempt on a reduced-round lightweight cipher above 30% of the rounds.

4.2 Neural Cryptanalysis of Toy Cipher

Table 4.1 summarizes previous neural cryptanalysis attempts on toy ciphers. Albassal and Wahdan [3] only used their approach to perform a KR attack on a 4-round toy cipher called HypCipher. Another work by Wabbersen [24] exactly followed this methodology to break another 4-round toy cipher suggested by Heys [9].

Simplified DES (SDES) is a 2-round Feistel toy cipher; plenty of neural cryptanalysis results exist for SDES. Alallayah et al. [1] demonstrated a wide range of attacks (KR, PR, EE) while Danziger and Henriques [6] provided another KR attack on SDES. Meanwhile, Xiao, Hao and Yao [26] showed a successful EE attack on 2-round DES, but a failure on 3-round DES.

Table 4.1: Neural cryptanalysis on toy ciphers

Publications		AW04 [3]	AAAA12 [1]	DH14 [6]	XHY19 [26]	Wab19 [24]
Attack Scenario		RKR	KR;PR;EE	KR	EE	RKR
Target	Name	HypCipher	SDES	SDES	DES	Hey02 [9]
Block	Internal Structure	Feistel	Feistel	Feistel	Feistel	SPN
Cipher	Block Size (bit)	16	8	8	64	16
	# of Rounds	4	2	2	16	4
# of Rounds Successfully Attacked		4	2	2	2	4
Neural	Layer Type	Dense	Dense	Dense	Varying	Dense
Network	# of Hidden Layers	2	32	1	1	2
Properties	# of Neurons per Layer	16	32	?	1,000	256
	Activation Function	Sigmoid	?	?	Varying	ReLU
Training	Loss Function[1)]	SSE	MSE	?	MSE	Varying
Methods	Training Data (tuples)	$\leq$53	1,024	102,400	65,536	$\leq$8,000
	Avg. # of Epoch	?	7,869;1,640;2,861	?	350	200
Result		Success	Success	Success	Success	Success

[1)] MSE: Mean Squared Error, SSE: Sum Squared Error [(SSE) = (# of output neurons) $\times$ (MSE)]

Though DES is not considered as a toy cipher, the attack seems to be effective since 2-round Feistel ciphers with two processing units have the following trivial linear relationship:

$$\forall x \in \{0,1\}^{n/2}, \forall p \in \{0,1\}^{n}, p' := p \oplus IP^{-1}(x|0...0), \tag{4.1}$$

$$IP(Enc(p) \oplus Enc(p')) \;=\; ?...?|x \tag{4.2}$$

assuming Enc is a 2-round Feistel cipher (such as SDES and 2-round DES) with the initial permutation IP.

Feistel ciphers should have at least 3 rounds in order to discuss security. Luby-Rackoff construction [17] describes how to construct a pseudorandom permutation using a pseudorandom function via the Feistel network. $q << 2^{n/2}$ was given as the bound for the number of queries to make 3 and 4 round Feistel network become a PRP (i.e. secure against CPA) and an SPRP (i.e. secure against CPA+CCA), respectively. This bound was later shown as a tight bound for 3 or 4 rounds [20], though even more number of rounds would be required in the quantum settings; Kuwakado and Morii [15] showed that the security of 3-round Feistel network does not hold against quantum CPA.

4.3 Neural Cryptanalysis on Lightweight Cipher

Table 4.2 summarizes previous neural cryptanalysis attempts on lightweight block ciphers. The attacks suggested by Gohr [8] are cRoI* and RKR attacks on reduced-round Speck32/64 up to 8 and 11 rounds, respectively.

Meanwhile, Mishra, Murthy, and Pal [19] reported failure on PR attack for PRESENT, an SPN lightweight cipher. Jain and Mishra [13] described another failure on PR attack for FeW, a Feistel lightweight cipher. These two publications took direct approaches, which are turned out to be ineffective for lightweight ciphers.

Table 4.2: Neural cryptanalysis on lightweight block ciphers

Publications		Goh19 [8]	MMP19 [19]	JM19 [13]
Attack Scenario		cIUF*;mIUF*;RKR	PR	PR
Target	Name	Speck32/64	PRESENT	FeW
Block	Internal Structure	Feistel/ARX	SPN	Feistel
Cipher	Block Size (bit)	32	64	64
	# of Rounds	22	31	32
# of Attacked Rounds		8;8;11	31	32
Neural	Layer Type	CNN/Residual	Dense	Dense
Network	# of Hidden Layers	$\leq$10	1	2
Properties	# of Neurons per Layer	Varying	?	?
	Activation Function	ReLU	Sigmoid	Sigmoid
Training	Loss Function	MSE	MSE	MSE
Methods	Training Data (tuples)	10,000,000	10,000	10,000
	Avg. # of Epoch	200	?	?
Result		Success	Failure	Failure

4.4 Misleading Results

There exist some publications claiming success on PR attacks against general-purpose ciphers. Alani [2] claimed that only a simple fully connected network having 4 to 5 fully connected layers was enough to recover plaintext of DES and 3DES. Hu and Zhao [12] followed the same approach to attack AES. We claim that these results are consequences of overfitting, not the success of neural cryptanalysis. The summary of those claims is listed in Table 4.3. Other publications [26, 16] also reported that the approach was not reproducible.

Table 4.3: Disputable plaintext restoration attack results on full-round general-purpose ciphers

Publications		Ala12 [2]		HZ18 [12]	
Attack Scenario		PR	PR	PR	PR
Target	Name	DES	3DES	AES-128	AES-256
Block Cipher	Internal Structure	Feistel	Feistel	SPN	SPN
	Mode of Operation	ECB	ECB	ECB/CBC	ECB/CBC
	Block Size (bit)	64	64	128	128
Neural Network Properties	Layer Type	Dense	Dense	Cascaded	Cascaded
	# of Hidden Layers	4	4 to 5	4	4
	# of Neurons per Layer	$\leq$512	$\leq$1,024	$\leq$256	$\leq$256
	Activation Function	Sigmoid	Sigmoid	Sigmoid	Sigmoid
Training Methods	Loss Function	MSE	MSE	MSE	MSE
	Training Data (tuples)	2,048	4,096	$\approx$ 1741	$\approx$ 1741
	Avg. # of Epoch	352	239	45	41
Total Error[1)]		0.1110	0.1658	0.1734	0.2052

[1)] Measured on the entire data including both training and test data.

Accuracy values should be measured for validation or test data only, to properly show whether the trained models are indeed successful plaintext restoration attacks. However, the metrics of "total error"[12] are shown as the main results, which are measured by the full dataset containing the training dataset. Though the description of "outside error"[2] seems to be valid, we could not verify the accuracy metrics were correctly calculated due to the lack of source code.

In the attack attempt on AES [12], Hamming distance distribution between the bytewise model output and the ground-truth plaintext is given, which is the direct evidence that overfitting was occurred. In the overfitting scenario, the distribution can be expressed as a weighted sum of two binomial distributions:

$$\alpha_{bit} B(8, 1-\alpha_{bit}) + (1-\tau) B(8, 0.5) \tag{4.3}$$

where τ is the ratio of training data from the entire dataset, α_{bit} is the probability of producing correct bit for each bit of the training data, and $B(T, P)$ stands for the binomial distribution with T trials and P probability. If the training was successful, the distribution should exhibit only one peak, while the empirical result exhibits two peaks.

Assuming that the authors used the MATLAB default $\tau = 0.7$ (though the authors specified 15% test data ratio, another 15% of the total data are highly likely to be used as validation data in the default), their experiment results can be explained by α_{bit} values in Table 4.4. Those α_{bit} values minimize the

mean squared difference between Equation 2 and the observed frequencies. We can further calculate the estimation of the total error under overfitting by

$$E := (1 - \alpha_{bit})\tau + 0.5(1 - \tau) \tag{4.4}$$

which is close to the actual error the authors obtained as in Table 4.4.

Table 4.4: Estimation of total error under the overfitting assumption in [12] ($\tau = 0.7$)

Target	Bit Inaccuracy $1 - \alpha_{bit}$	Estimated Error E	Actual Error in the Literature
AES-128 (ECB)	0.0358	0.1751	0.1768
AES-128 (CBC)	0.0627	0.1939	0.2095
AES-256 (ECB)	0.0198	0.1639	0.1699
AES-256 (CBC)	0.0580	0.1906	0.1909

According to the analysis, the results only implies that their fully connected neural networks can overfit the training data produced by the practical ciphers. The results do not indicate that the plaintext restoration attack on any of the practical ciphers was successful.

Chapter 5. Evaluation

5.1 Methodology

5.1.1 Selection of Target Ciphers

We selected the target ciphers according to the following criteria.

- The ciphers should be lightweight block ciphers, which are implemented using rotations, bitwise operations, and arithmetic operations only.
- One of the ciphers should be a normal Feistel network, while the others should be GFNs with different generalization types.
- The ciphers should have the same 64-bit block size, while having different number of processing units.

SIMON64/96 [4], CHAM64/128 [14] and HIGHT [11] were selected as the targets. The specifications of the ciphers are summarized in Table 5.1.

For each block cipher, we manually picked a Δ value which is expected to cancel the diffusion for one round, by exploiting the relationship between the arithmetic addition and the XOR operation if possible. The choices are also shown in Table 5.1.

Table 5.1: Specifications of the target ciphers, with selected Δ values for differential neural cryptanalysis

Target	# of rounds	Block size (bits)	Processing unit size (bits)	Generalization type	Selected Δ in hex. representations
SIMON64/96	42	64	32	N/A	00000000 00000001
CHAM64/128	80	64	16	Type 1	8000 4000 0000 0000
HIGHT	34	64	8	Type 2, alternative	00 00 00 00 00 00 E9 80

5.1.2 Deep Learning Procedures

First, we prepared datasets which are the compositions of the challenges and corresponding answer bits of the various IUF and IPF binary classification games. 140,000 training pairs, 30,000 validation pairs, and 30,000 test pairs were prepared for each round and each cipher, having the equal number of $x = 0$ and $x = 1$ games.

Regarding the network structure, we benchmarked Gohr's network and configurations [8] for the fair comparison of our result to the previous work. We used the network with 2 hidden layers and configured the shape of the input layer using the number of processing units, reflecting the property of GFN ciphers.

We trained the models during 10 epochs and pick the stage with the highest validation accuracy. The models performed predictions for the test data, and the accuracy becomes the final accuracy of the target cipher reduced to the respective round.

For comparison to another machine learning model, we selected random forest models as they produce decent classification accuracy on a wide range of classification problems. We configured that a random forest model consists of 100 decision trees.

We skipped the experiment for 1-round and 2-round ciphers since it is apparent that the difference values are not propagated to the entire processing units for any Feistel ciphers, as shown in 4.2.

5.1.3 Meaningfulness of Distinguishers

To properly evaluate whether a trained distinguisher has a significant advantage in classifying $x = 0$ and $x = 1$ cases of each distinguishing game, one should establish a decision boundary of the accuracy. The boundary should be specified according to the cardinality of the testing data which actually produces the accuracy value.

Suppose that we perform the one-tailed Z-test to decide whether trained binary classifiers are better than random guessing on a balanced test dataset with n_{test} data. Random guessing follows $B(n_{test}, 0.5)$, which can be approximated into the normal distribution $N(\mu_{test}, \sigma^2_{test})$ where $\mu_{test} := n_{test}/2$ and $\sigma_{test} := \sqrt{n_{test}/4}$. Once we observe the mean accuracy a_{exp} from n_{exp} trials of i.i.d. experiment, we can calculate the z-score by the following:

$$z := \sqrt{n_{exp}}(n_{test}a_{exp} - \mu_{test})/\sigma_{test} = 2\sqrt{n_{exp}n_{test}}(a_{exp} - 0.5) \tag{5.1}$$

If $z > z_{0.95} \approx 1.645$, we can reject the null hypothesis on the 95% confidence interval, concluding that the trained classifiers would have non-negligible advantage over random guessing.

For one-shot experiment ($n_{exp} = 1$) with $n_{test} = 30,000$, we can calculate the accuracy level $a_{one-shot}$ required to reject the null hypothesis by the following:

$$z_{one-shot} = 2\sqrt{n_{exp}n_{test}}(a_{one-shot} - 0.5) > z_{0.95} \tag{5.2}$$

$$a_{one-shot} > 0.5 + 0.5z_{0.95}/\sqrt{30,000} \gtrsim 50.5\% \tag{5.3}$$

Therefore, we call the models having test accuracy more than or equal to 50.5% are meaningful distinguishers. However, the boundary is only for the reference and is not the absolute criterion as the accuracy might be high enough by chance. When the test accuracy keeps remaining below 50.5% while incrementing the rounds, we stopped the experiment for more rounds and concluded that meaningful distinguishers for the further rounds could be difficult to be trained using data-driven techniques.

5.1.4 Environment

Our experiment was conducted on Windows 10 64-bit, Intel Core i7-8700 @ 3.2GHz, 32GB RAM, and NVIDIA GeForce GTX 1070 Ti supporting CUDA 10.0. The data generation codes were written in C++, while the machine learning code was written in Python 3.6. The deep learning code is based on Tensorflow-GPU 1.14.0 and Keras 2.2.4., while we imported scikit-learn 0.21.2 for the implementation of random forest models.

5.2 Experimental Results

5.2.1 Summary

We provide a summary of the performance of trained distinguishers in Table 5.2. As there are no previous differential neural cryptanalysis results against SIMON64/96, CHAM64/128, and HIGHT to the best of our knowledge, we instead compare the results with the direct application of best known DC attacks, i.e. probabilities of differential characteristics in Table 2.2. This provides an indirect comparison of how much data is required and how much data is actually used.

Table 5.2: Summary of the performance of trained distinguishers, along with the known differential characteristics

Ciphers	Distinguishers or characteristics	Rounds, with a successful attack	Computation for training
SIMON64/96	cIUF(',*)	10 (23.8%)	1,400,000 ($\sim 2^{20.5}$)
	dIUF(')	10 (23.8%)	1,400,000 ($\sim 2^{20.5}$)
	dIUF*	11 (26.2%)	1,400,000 ($\sim 2^{20.5}$)
	mIUF*	7 (16.7%)	1,400,000 ($\sim 2^{20.5}$)
	(RDC)	(10)	($\sim 2^{18}$)
		(11)	($\sim 2^{22}$)
CHAM64/128	cIUF(',*)	28 (35%)	1,400,000 ($\sim 2^{20.5}$)
	dIUF(',*)	29 (36.3%)	1,400,000 ($\sim 2^{20.5}$)
	mIUF*	13 (16.3%)	29,400,000 ($\sim 2^{24.9}$)
	(DC)	(28)	($\sim 2^{39}$)
		(29)	($\sim 2^{41}$)
HIGHT	cIUF(',*)	9 (28.1%)	1,400,000 ($\sim 2^{20.5}$)
	dIUF(',*)	9 (28.1%)	1,400,000 ($\sim 2^{20.5}$)

5.2.2 SIMON64/96

The cIUF-type distinguishers for SIMON64/96 were successful up to 10 rounds of the cipher. The classification was almost perfect up to 7 rounds, and the accuracy started to drop from the 8 or 10 rounds with the S-shaped curves. This result is depicted in Figure 5.1a.

The result was similar for the dIUF-type distinguishers shown in Figure 5.1b. Some of the dIUF-type distinguishers were even slightly successful on the 11-round cipher; the dIUF* classifier we trained had 51.74% accuracy on the 11-round SIMON64/96. However, the dIPF' classifier only had meaningful classification up to 7 rounds.

As shown in Figure 5.1c, the mIUF* classifier could win on the distinguishing game for 7 rounds of SIMON64/96. Note that round 7 is the boundary of the reduced-round cipher where the accuracy of cIUF-type and dIUF-type classifiers start to decrease.

Original IPP(') and IUP(') games were even harder for the distinguishers. Given the plaintext, a neural network model could win the IPP' game up to only 3 rounds of SIMON64/96 cipher, whether the ciphertext in the challenge was the encryption of the particular plaintext or the encryption of another

random block. For the games IPP, IUP, and IUP', the classifiers learned almost nothing from the training data.

5.2.3 CHAM64/128

The cIUF-type distinguishers for CHAM64/128 were successful up to 29 rounds of the cipher. The classification was almost perfect up to 14 15 rounds, and the accuracy gradually dropped toward 50%. For the 27-round and 28-round CHAM64/128 cipher, the classification accuracy remains above the 50.5% accuracy level but becomes quite close to the level. The result was similar for the dIUF-type distinguishers of reduced-round CHAM64/128.

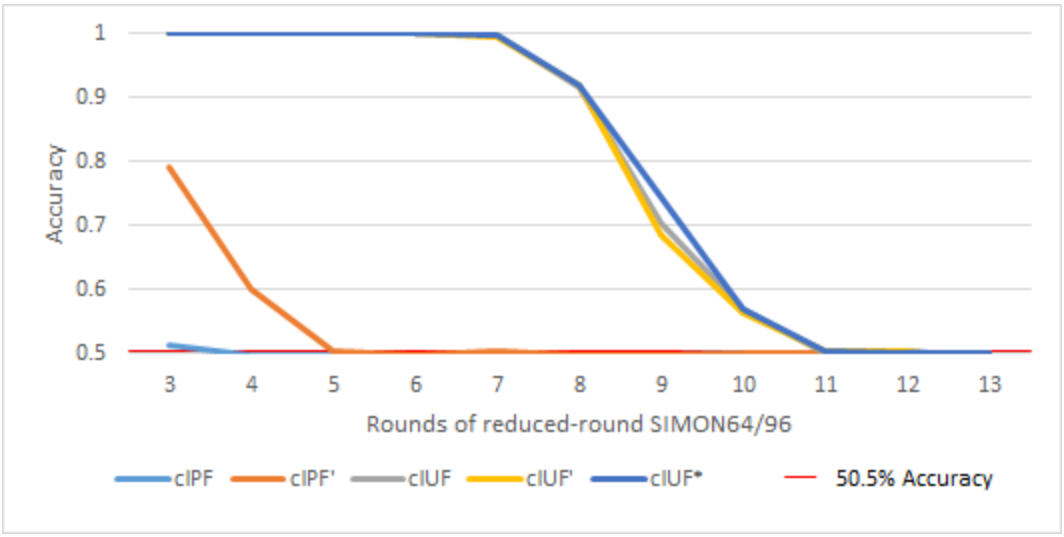
The accuracy drop forms a staircase pattern; the accuracy drops from 20 to 21, from 22 to 23, and from 24 to 25 rounds are relatively smaller than the drops from 21 to 22, from 23 to 24, and from 25 to 26 rounds. The staircase pattern seems to have emerged from the alternating nature of CHAM64/128. We could notice the instability on the performance of cIUF and dIUF classifiers, shown in gray lines of Figures 5.2a and 5.2b. The staircase patterns are exaggerated on the gray lines of cIUF classifiers, even showing the accuracy rises.

The cIPF' and mIUF(') distinguishers could distinguish the first three rounds of the cipher. The mIUF* distinguisher could distinguish 5-round and 6-round CHAM64/128, as depicted in Figure 5.2c.

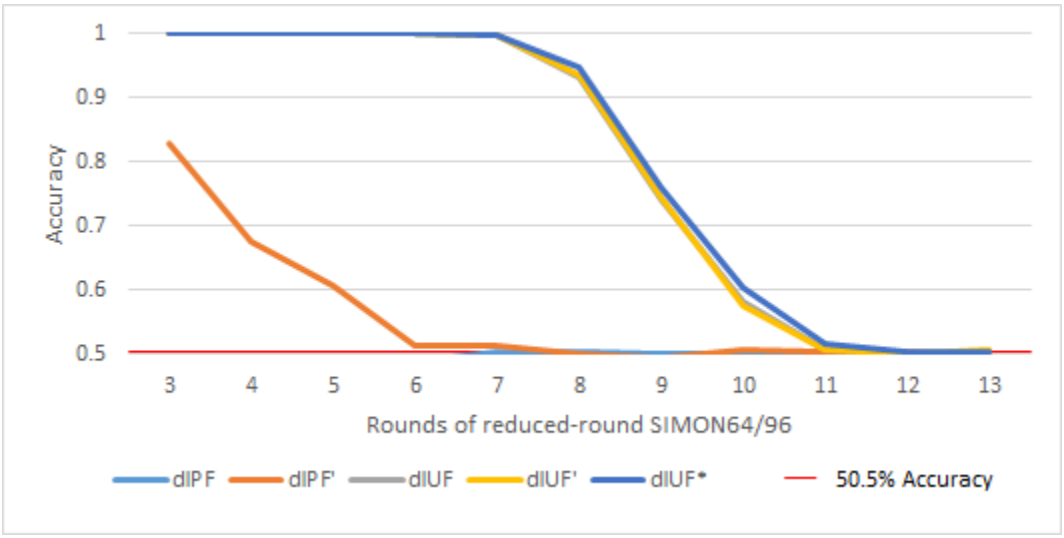
5.2.4 HIGHT

The cIUF-type distinguishers for HIGHT were successful up to 10 rounds of the cipher. The classification was almost perfect up to 8 rounds, and the accuracy rapidly dropped at the attack against the 9-round cipher with the almost straight line. The result was almost identical for the dIUF-type distinguishers of reduced-round HIGHT.

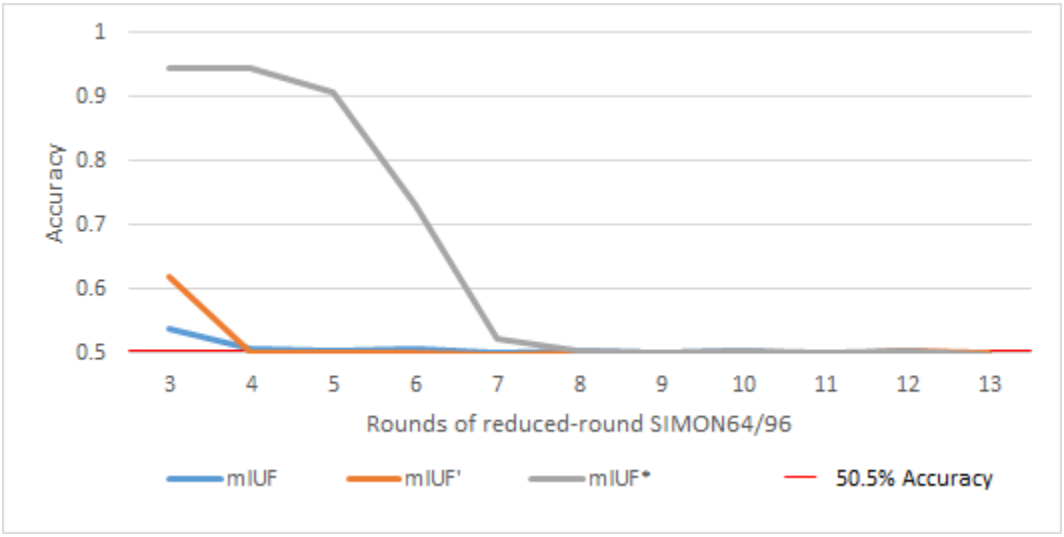
The mIUF-type classifiers could learn nothing from the corresponding data generated from HIGHT. All things we could observe from the results of mIUF(',*) distinguishers were the fluctuations of the accuracy lines around the 50% axis. This was similar for the original IPP(') and IUP(') games against reduced-round HIGHT.



(a) cIPF-type and cIUF-type distinguishers

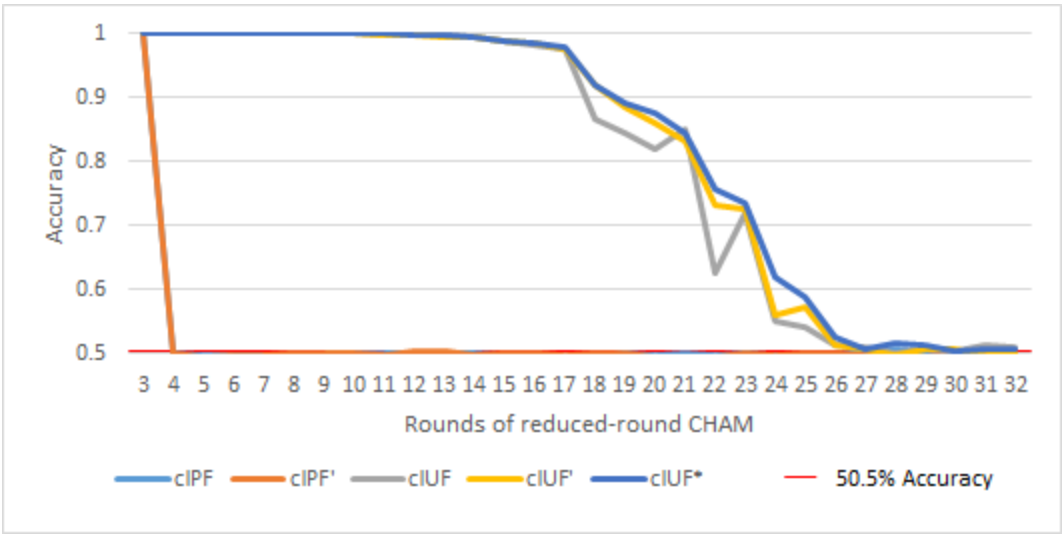


(b) dIPF-type and dIUF-type distinguishers

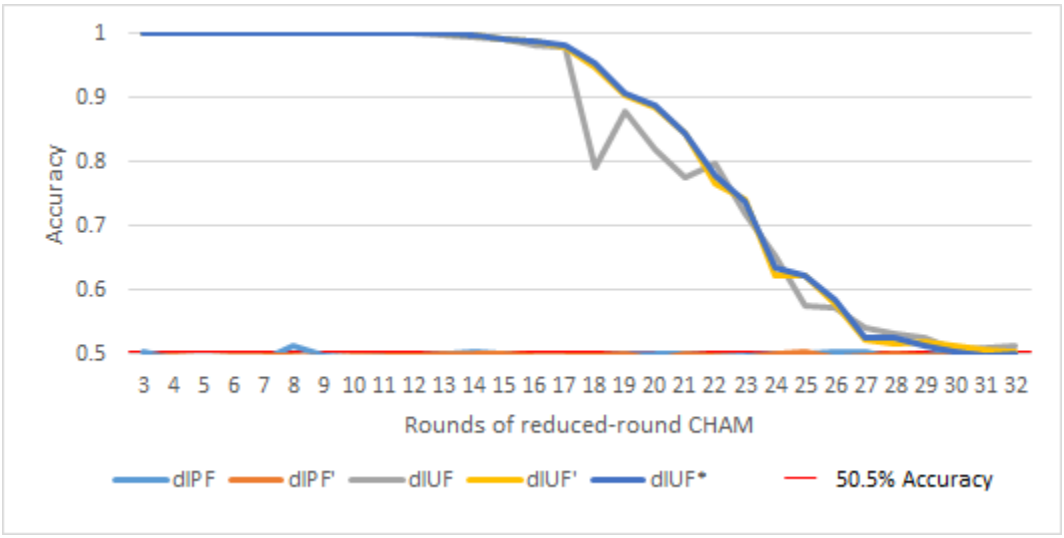


(c) mIUF-type distinguishers

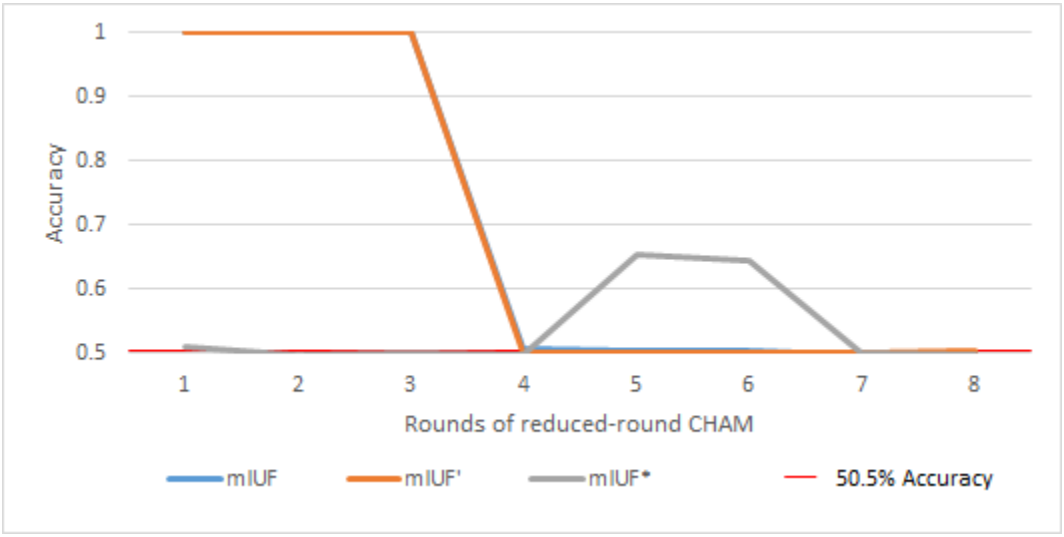
Figure 5.1: Accuracy of differential NN classifiers against reduced-round SIMON64/96



(a) cIPF-type and cIUF-type distinguishers

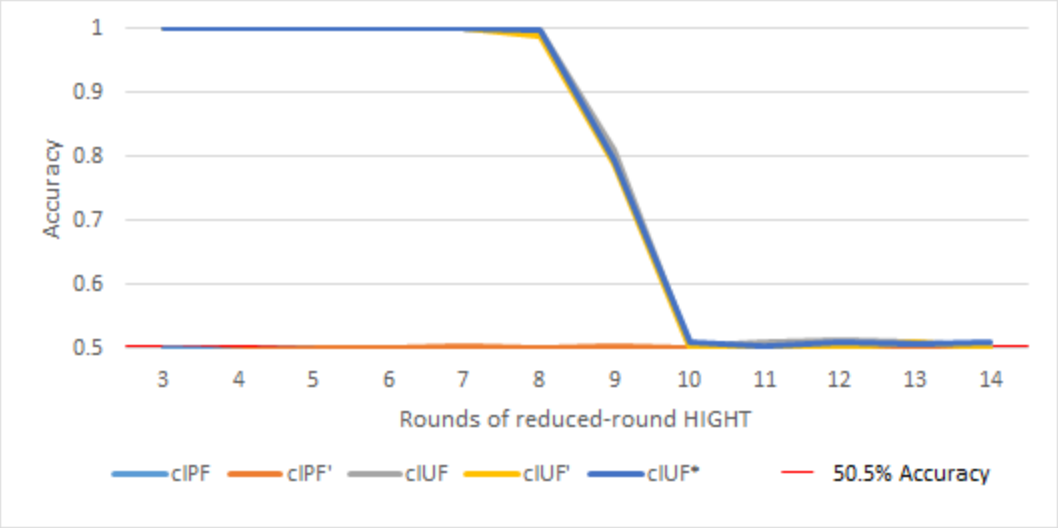


(b) dIPF-type and dIUF-type distinguishers

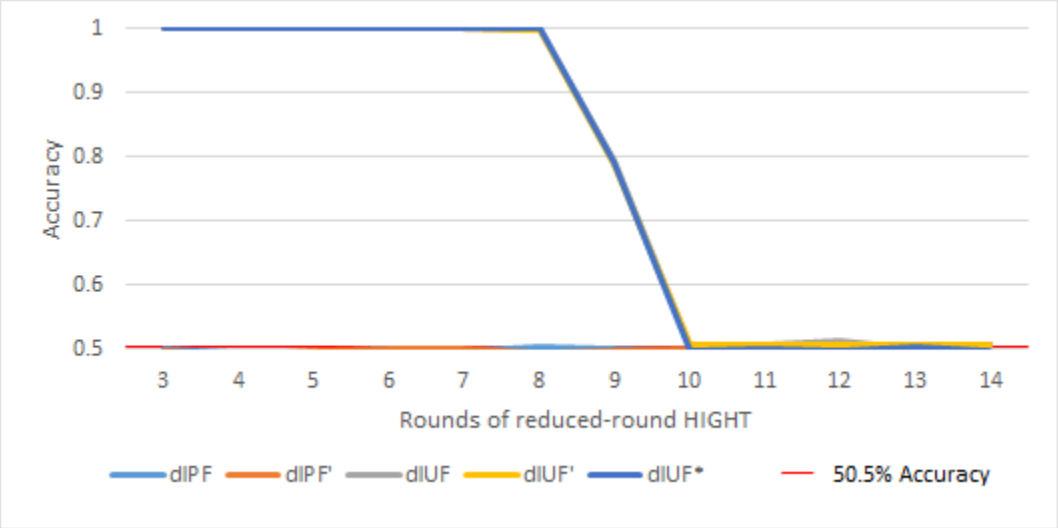


(c) mIUF-type distinguishers (see 6.2.3 for the mIUF* classifier trained with more epochs)

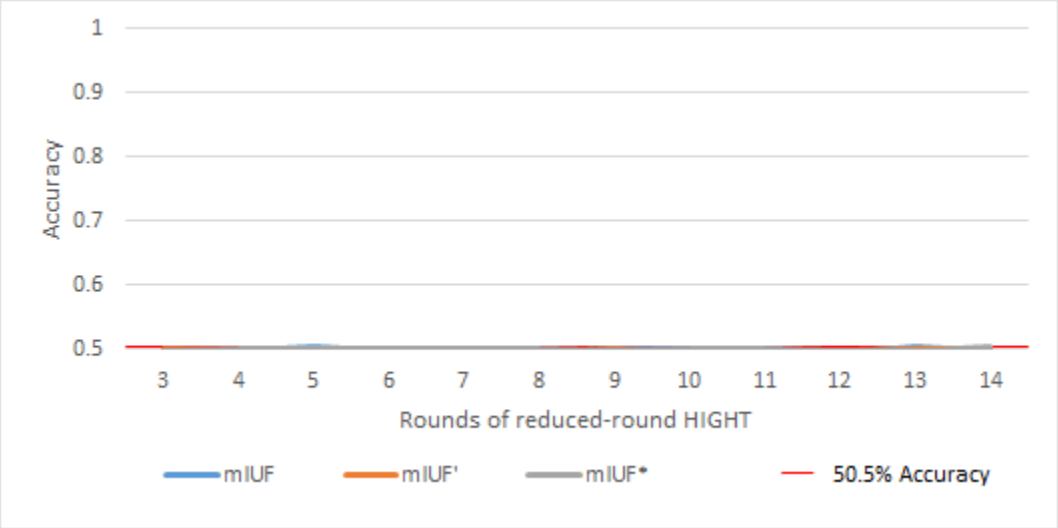
Figure 5.2: Accuracy of differential NN classifiers against reduced-round CHAM64/128



(a) cIPF-type and cIUF-type distinguishers



(b) dIPF-type and dIUF-type distinguishers



(c) mIUF-type distinguishers

Figure 5.3: Accuracy of differential NN classifiers against reduced-round HIGHT

Chapter 6. Analysis and Discussion

6.1 Analysis by GFN Types

We have conducted the experiment on three types of 64-bit Feistel ciphers: SIMON64/96 having the normal Feistel network, CHAM64/128 having the type-1 GFN, and HIGHT having the type-2 GFN. One can view a normal Feistel network as a degenerated type-1 GFN, as they have one invocation of the Feistel function per round and the rotation of the processing units follows. One can also view a normal Feistel network as a degenerated type-2 GFN, as they have invocations of the Feistel function for each adjacent pair of processing units, followed by the rotation of the processing units.

As individual keys and Feistel functions are different among the ciphers, the comparison we provide in this section is not completely decoupled from the individual characteristics of the ciphers. However, we can still discuss the general tendency of how the effectiveness of differential neural classifiers varies by the network structures.

Here, we use the term 'cycle' to denote rounds of block cipher repeated by the number of processing units in the cipher.

6.1.1 Type-1 GFN Cipher

Ciphers with the type-1 GFN (or reversed type-1 GFN) usually require many rounds, as the difference values propagate slowly on the cipher. The full-round cipher of SIMON64/96 and CHAM64/128 consists of 21 and 20 cycles, respectively. If the starting difference presents only at the last processing unit, the difference should move to the first processing unit to spread the difference to the other processing units. Almost one cycle is required in this process.

It takes another cycle to let the difference present at every processing unit. Furthermore, the difference should be propagated to other bits in each processing unit. If the adversary started with just one bit of difference, the difference usually affects two or three bits on average on a single invocation of a Feistel function of lightweight ciphers. For instance, the Feistel function of SIMON64/96 specifies the three bits for the propagation, precisely, where the effective number of propagated bits is two, as two of the three different bits could be possibly masked by bitwise AND operations. The Feistel function of CHAM64/128 exploits arithmetic addition, which could affect many bits by once but at the exponentially diminishing probability. Therefore, 2 cycles or more, depending on the size of processing units and the characteristics of the functions, are additionally required to actually distribute the difference to every bit within the processing units.

6.1.2 Type-2 GFN Cipher

Ciphers with the type-2 GFN usually require less number of cycles compared to the type-1 GFNs. The full-round cipher of HIGHT consists of only 4 cycles. Only one round is required to let the difference ready for the propagation, different from one entire cycle in the type-1 GFN.

A cycle is required to let the difference present at every processing unit, similar to the type-1 GFN. After that, the first propagated difference should have gone through multiple Feistel functions for more in-block propagation, before the difference returns to the original block in which the propagation has

been started. In one cycle of HIGHT, the difference undergoes through 4 Feistel functions, which is enough to propagate the difference within an 8-bit processing unit.

6.1.3 Assessment

The performance of cIUF-type and dIUF-type distinguishers of SIMON64/96 started to drop at the end of the fourth cycle, and it required 5 cycles (23.8% of the total rounds) to reduce the accuracy down to random selections. The performance of cIUF-type and dIUF-type distinguishers of CHAM64/128 started to drop at the end of the fourth cycle, and it required more than 7 cycles (35% of the total rounds) to reduce the accuracy.

For HIGHT, one cycle and one more round (28.1% of the total rounds) were enough to be safe from cIUF-type and dIUF-type distinguishers.

These results coincide with the analysis of difference propagation for type-1 and type2 GFN ciphers. Therefore, we can conclude that the cIUF-type and dIUF-type distinguishers we have trained so far mainly gained their classification ability by capturing the bit locations and nonuniformity where the differences had less propagated.

6.2 Analysis by Distinguishing Tasks

6.2.1 Failure of Straightforward Neural Distinguishers

Training straightforward neural distinguishers is a difficult task. The ciphertext becomes random-looking even if only a few rounds are passed due to the addition of the unknown round key, while the round keys could be canceled out on differential settings. For all of the experiment, no IPP(') or IUP(') models could distinguish the plaintext-ciphertext pairs from any reduced-round ciphers having more than 4 rounds.

6.2.2 Evaluation on Differential Distinguishers

In theory, the cIUF-type distinguishers should produce higher accuracy than dIUF-type counterparts. Also, any IUF model should perform better than the corresponding IUF' model, and the IUF' model should be better than the corresponding IUF* model. This is because of the reductions among the designated models which can be derived using the available information.

However, the empirical results were not, and were even little opposite to the prediction. dIUF-type models had usually equal or slightly higher accuracy than cIUF-type models. In the accuracy patterns among cIUF-type models, and among dIUF-type models, IUF* models were similar or slightly better than IUF or IUF' models.

The models may probabilistically capture the distinguishing abilities, especially in the earlier stage of the training. Therefore, as more features which are not important for the classification are provided, the quality of the model could be diminished. IUF* models have no plaintext compared to IUF and IUF' models. dIUF-type models do not hold the two ciphertext values separately and only utilize their differences compared to cIUF models.

Therefore, we can assert that the presence of additional data such as plaintext values and separated ciphertext values exerted a negligible advantage on the differential distinguishing attacks. The accuracy comparisons were even reversed due to the confusion caused by these additional data.

6.2.3 Notes on mIUF-type Distinguishers

Gohr [8] utilized mIUF* classifiers for a cryptographic experiment called "real differences experiment", to examine whether the classifiers do not rely on the nonuniformity of the difference functions.

In a reduced-round cipher having large enough number of rounds, the output of mIUF* classifiers will be close to a random function, and the training would become unsuccessful. In a cipher having a small number of rounds, the difference has less entropy (having limited possible values) which can act as some advantage for remaining plain IUP* game, which is impossible without the additional information related to the plaintext. If the gap of the two effects, we would observe a bell-shaped curve. If they do not, we would not be able to train any meaningful mIUF* classifiers.

For CHAM64/128, it turned out that 10 epochs are not enough for the neural network model to capture the distinguishability of mIUF-type games. Here, we demonstrate the training results of mIUF* distinguishers in Figure 6.1. The same network was trained for 30 epochs, and the model with the best test accuracy was chosen among 7 models trained from random initial conditions.

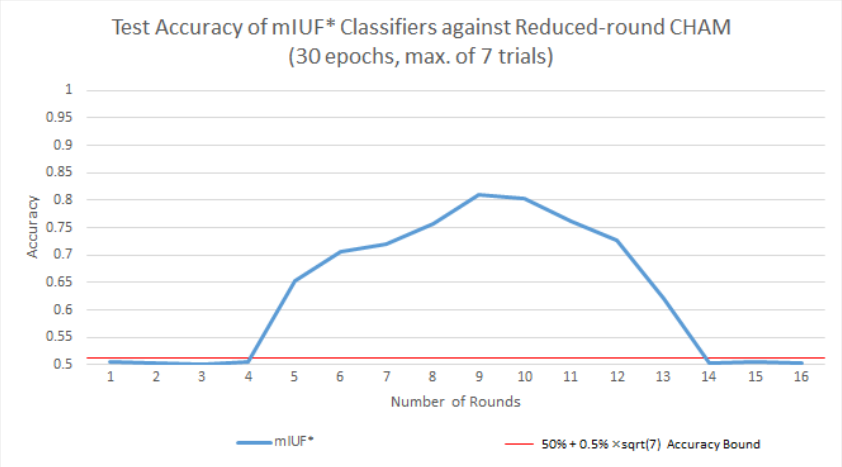


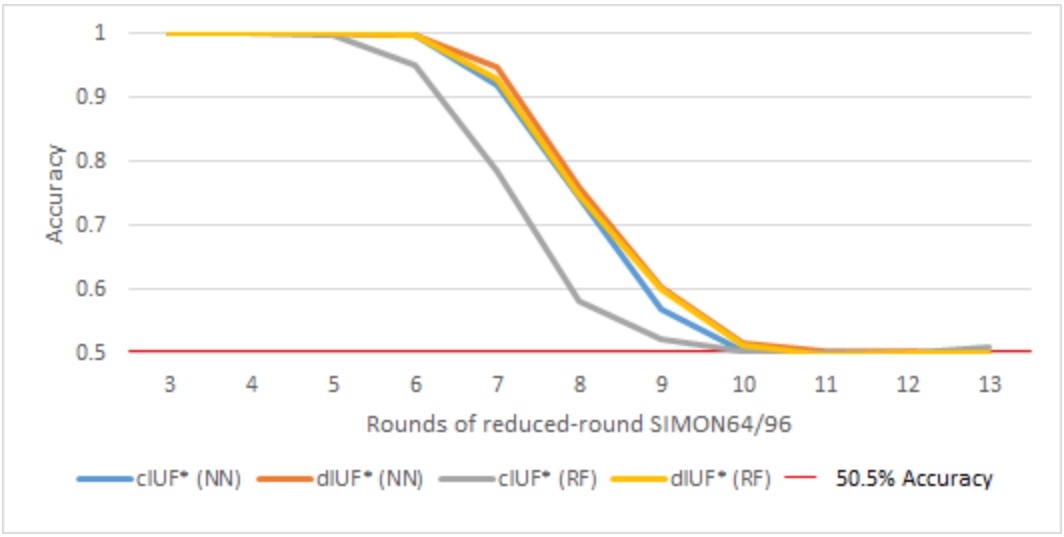
Figure 6.1: Test accuracy of mIUF* classifiers against Reduced-round CHAM64/128

For SIMON64/96 and CHAM64/128 reduced into 1 to 3 cycles, which are two type-1 ciphers, the mIUF* distinguishers could meaningfully classify the corresponding data produced by the ciphers. This demonstrates that the neural network models can learn more than the nonuniformity of the difference, but only before the accuracy of cIUF* and dIUF* classifiers remains almost perfect.

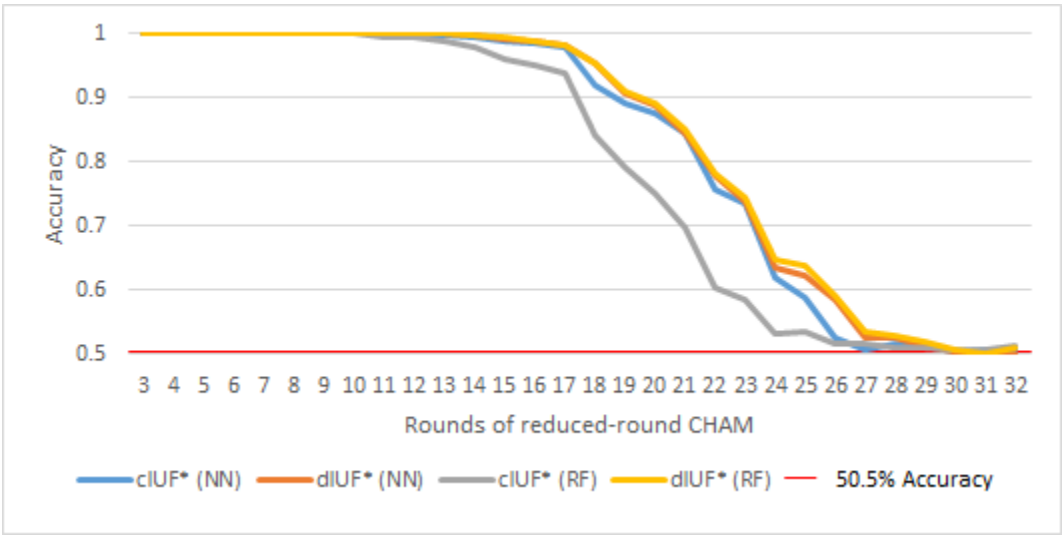
6.3 Comparison by Machine Learning Models

We tested both of DL models and random forest models for each distinguishing task. In the charts of Figure 6.2, the dIUF* models based on the random forest classifiers have almost identical accuracy compared to the neural network counterparts.

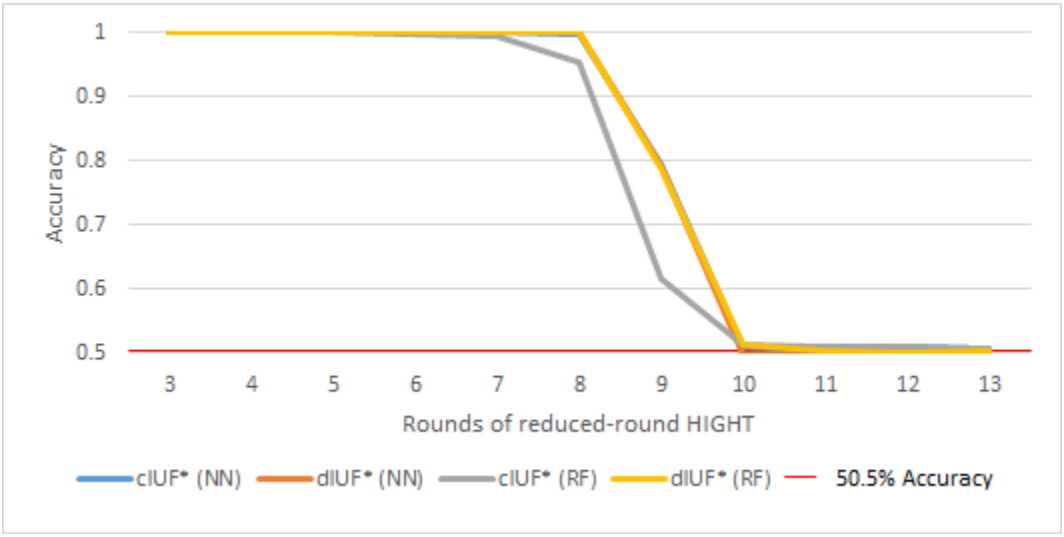
However, for cIUF* models, random forest models were clearly inferior to the neural network models. As the classification boundary is drawn along the axes, random forest models are inefficient to draw the boundary of XOR functions. If a feature is shown as XOR of two values, the model may face inefficiency in combining the two features. One could infer that the features from the Dif function contain aspects that more directly utilized in the classifier, compared to two separated values of $Concat$.



(a) Distinguishers against reduced-round SIMON64/96



(b) Distinguishers against reduced-round CHAM64/128



(c) Distinguishers against reduced-round HIGHT

Figure 6.2: Accuracy of cIUF* and dIUF* distinguishers, according to the ML models

Chapter 7. Concluding Remark

Along with the development of deep learning, plenty of attempts exist to analyze the security of block cipher using the data-driven approach. These approaches are diverse, or often disputable; the reason is due to the absence of systemization on attack scenarios and methodologies regarding this emerging field.

In this context, we identified and described 17 types of distinguishing games that could be utilized for data-driven cryptanalysis of block ciphers. To evaluate the performance of neural network classifiers over these distinguishing games, we trained every type of the differential classifiers against three 64-bit reduced-round lightweight block ciphers having GFN structures, namely SIMON64/96, CHAM64/128, and HIGHT. Based on the accuracy patterns, we evaluated the performance of each distinguishing scenario and provided some insights on how the neural network behaves to make a classification.

Most of the deep learning models can obtain their major distinguishing capability from the bit values where the initial difference has not yet propagated, relying on a relatively small amount of computation time and training data. Using these deep learning models, a cryptographer can pinpoint which round the diffusion step of a block cipher is mostly completed, and observe the effective bit difference propagation rate of the Feistel function.

Furthermore, there is some evidence that deep learning models can classify beyond this nonuniformity of functions, as seen in the examples of mIUF* games. Deep learning models would have the capability to learn complex functions if a large enough dataset is given with enough training time, but 10 epochs of 140,000 instances of data each could be small. Therefore, a scaled-up experiment on lightweight block ciphers with more data and time could be conducted as future work. Using more resources, distinguishing reduced-round ciphers with few more rounds compared to the results in this thesis could be possible.

Another next step of this study is to deploy RKR attacks using each successfully trained classifier. As our analysis aimed at 64-bit GFN ciphers, the scope of the experiment could be extended to the other block ciphers having different message length, more complex Feistel functions, or other structures such as SPN.

Even though deep learning technology rapidly evolves, breaking full round lightweight or general-purpose ciphers will be likely to remain as an almost impossible task, due to the exponentially increasing amount of time and dataset. To establish a breakthrough on the success of breaking more rounds of ciphers, new approaches and strategies should be suggested, possibly combined with the other known cryptanalysis approaches such as linear cryptanalysis.

Bibliography

[1] Khaled M Alallayah, Alaa H Alhamami, Waiel AbdElwahed, and Mohamed Amin. Applying neural networks for simplified data encryption standard (SDES) cipher system cryptanalysis. *Int. Arab J. Inf. Technol.*, 9(2):163–169, 2012.

[2] Mohammed M Alani. Neuro-cryptanalysis of DES and triple-DES. In *International Conference on Neural Information Processing*, pages 637–646. Springer, 2012.

[3] Ayman MB Albassal and A-MA Wahdan. Neural network based cryptanalysis of a feistel type block cipher. In *International Conference on Electrical, Electronic and Computer Engineering, 2004. ICEEC'04.*, pages 231–237. IEEE, 2004.

[4] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK lightweight block ciphers. In *Proceedings of the 52nd Annual Design Automation Conference*, pages 1–6, 2015.

[5] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of CRYPTOLOGY*, 4(1):3–72, 1991.

[6] Moisés Danziger and Marco Aurélio Amaral Henriques. Improved cryptanalysis combining differential and artificial neural network schemes. In *2014 International Telecommunications Symposium (ITS)*, pages 1–5. IEEE, 2014.

[7] Anand Desai and Sara Miner. Concrete security characterizations of PRFs and PRPs: Reductions and applications. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 503–516. Springer, 2000.

[8] Aron Gohr. Improving attacks on Speck32/64 using deep learning. *IACR Cryptology ePrint Archive*, 2019:37, 2019.

[9] Howard M Heys. A tutorial on linear and differential cryptanalysis. *Cryptologia*, 26(3):189–221, 2002.

[10] Viet Tung Hoang and Phillip Rogaway. On generalized feistel networks. In *Annual Cryptology Conference*, pages 613–630. Springer, 2010.

[11] Deukjo Hong, Jaechul Sung, Seokhie Hong, Jongin Lim, Sangjin Lee, Bon-Seok Koo, Changhoon Lee, Donghoon Chang, Jesang Lee, Kitae Jeong, et al. HIGHT: A new block cipher suitable for low-resource device. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 46–59. Springer, 2006.

[12] Xinyi Hu and Yaqun Zhao. Research on plaintext restoration of AES based on neural network. *Security and Communication Networks*, 2018, 2018.

[13] Aayush Jain and Girish Mishra. Analysis of lightweight block cipher FeW on the basis of neural network. In *Harmony Search and Nature Inspired Optimization Algorithms*, pages 1041–1047. Springer, 2019.

[14] Bonwook Koo, Dongyoung Roh, Hyeonjin Kim, Younghoon Jung, Dong-Geon Lee, and Daesung Kwon. CHAM: a family of lightweight block ciphers for resource-constrained devices. In *International Conference on Information Security and Cryptology*, pages 3–25. Springer, 2017.

[15] Hidenori Kuwakado and Masakatu Morii. Quantum distinguisher between the 3-round Feistel cipher and the random permutation. In *2010 IEEE International Symposium on Information Theory*, pages 2682–2685. IEEE, 2010.

[16] Linus Lagerhjelm. Extracting information from encrypted data using deep neural networks. Master's thesis, Umeå University, 2018.

[17] Michael Luby and Charles Rackoff. How to construct pseudorandom permutations from pseudorandom functions. *SIAM Journal on Computing*, 17(2):373–386, 1988.

[18] Stefan Lucks. Faster luby-rackoff ciphers. In *International Workshop on Fast Software Encryption*, pages 189–203. Springer, 1996.

[19] Girish Mishra, SVSSNVG Krishna Murthy, and SK Pal. Neural network based analysis of lightweight block cipher PRESENT. In *Harmony Search and Nature Inspired Optimization Algorithms*, pages 969–978. Springer, 2019.

[20] Jacques Patarin. New results on pseudorandom permutation generators based on the DES scheme. In *Annual International Cryptology Conference*, pages 301–312. Springer, 1991.

[21] Ronald L Rivest. Cryptography and machine learning. In *International Conference on the Theory and Application of Cryptology*, pages 427–439. Springer, 1991.

[22] Edward F Schaefer. A simplified data encryption standard algorithm. *Cryptologia*, 20(1):77–84, 1996.

[23] Bruce Schneier and John Kelsey. Unbalanced feistel networks and block cipher design. In *International Workshop on Fast Software Encryption*, pages 121–144. Springer, 1996.

[24] Jan Wabbersen. Cryptanalysis of block ciphers using feedforward neural networks. Master's thesis, Georg-August-Universität Göttingen, 2019.

[25] Xuzi Wang, Baofeng Wu, Lin Hou, and Dongdai Lin. Automatic search for related-key differential trails in SIMON-like block ciphers based on MILP. In *International Conference on Information Security*, pages 116–131. Springer, 2018.

[26] Ya Xiao, Qingying Hao, and Danfeng Yao. Neural cryptanalysis: Metrics, methodology, and applications in CPS ciphers. *arXiv preprint arXiv:1911.04020*, 2019.

[27] Jun Yin, Chuyan Ma, Lijun Lyu, Jian Song, Guang Zeng, Chuangui Ma, and Fushan Wei. Improved cryptanalysis of an ISO standard lightweight block cipher with refined MILP modelling. In *International Conference on Information Security and Cryptology*, pages 404–426. Springer, 2017.

[28] Yuliang Zheng, Tsutomu Matsumoto, and Hideki Imai. On the construction of block ciphers provably secure and not relying on any unproved hypotheses. In *Conference on the Theory and Application of Cryptology*, pages 461–480. Springer, 1989.

Acknowledgments in Korean

먼저 부족한 점이 많은 저에게 연구자가 가져야 할 능력과 자세에 대해 알려주시고, 여러 논문을 지도하여 주신 김광조 교수님께 감사의 말씀을 드립니다. 대한민국 1세대 암호 연구의 주역이신 교수님의 마지막 석사과정 지도학생이 되어 많은 것을 배울 수 있어서, 석사과정 한 순간 한 순간이 더욱 소중했지 않나 싶습니다. 또한, 코로나19로 어수선하고 바쁘셨을 와중에도 본 석사학위논문의 심사위원으로 흔쾌히 참석하여 주신 손수엘 교수님과 이주영 교수님께도 깊은 감사를 드립니다.

2년간 연구실에서 수많은 도움을 받았습니다. 항상 연구실의 구심점이 되었던 락용이 형, 비슷한 연구 주제를 바탕으로 많은 이야기를 나누었던 Harry, 연구실에 새로운 분위기를 가져다 준 Edwin, 여러 과제를 꿋꿋하게 수행하던 지은이 누나, 실험을 위해 많은 시간을 함께 하면서 고생을 나누었던 낙준이 형, 수업의 여러 프로젝트와 학회 발표까지 많은 것을 함께 했었던 나비 누나, 조교 활동이나 수업 수강을 같이 하면서 수학 분야 배경지식과 관련하여 여러 도움을 준 성호 형과 동연이, 짧은 시간 함께했지만 기억에 남는 형철이 형에게도 감사의 인사를 전합니다.

또한, 이 모든 것을 지지해 주신 부모님께 감사드립니다. 졸업 후에도 석사과정에서 얻은 여러 경험과 능력을 바탕으로 더 편리하고 좋은 세상을 만들어 나가기 위해 노력하겠습니다.

Curriculum Vitae in Korean

이 름: 백 승 근

학 력

2008. 3. – 2011. 2. 한국과학영재학교

2011. 2. – 2018. 8. 한국과학기술원 전산학부 (학사)

2018. 8. – 2020. 8. 한국과학기술원 정보보호대학원 (석사)

경 력

2019. 2. – 2019. 6. 한국과학기술원 정보보호론 일반조교

2019. 8. – 2019. 12. 한국과학기술원 고급 정보보호 일반조교

2020. 3. – 2020. 7. 한국과학기술원 정보보호개론 일반조교

연 구 과 제

2018. 8. – 2018. 10. 암호화폐와 스마트 컨트랙트 응용 시스템 설계 및 보안 취약성 분석 연구

2018. 8. – 2019. 12. 양자 난수 생성기의 보안성 및 성능 연구

2018. 8. – 2020. 7. 양자 컴퓨터 환경에서 래티스 문제를 이용한 다자간 인증키 교환 프로토콜 연구

연 구 업 적

1. **Seunggeun Baek**, Nabi Lee, and Kwangjo Kim, "Generic Analysis of E-voting Protocols by Simplified Blockchain," *2019 Symposium on Cryptography and Information Security (SCIS 2019)*, Jan., 22-25, 2019, Otsu, Japan.
2. **Seunggeun Baek**, and Kwangjo Kim, "Recent Advances of Neural Attacks against Block Ciphers," *2020 Symposium on Cryptography and Information Security (SCIS 2020)*, Jan., 28-31, 2020, Kochi, Japan.
3. Rakyong Choi, Dongyeon Hong, Seongho Han, **Seunggeun Baek**, Wooseok Kang, and Kwangjo Kim, "Design and Implementation of Constant-round Dynamic Group Key Exchange from RLWE," *IEEE Access*, Vol. 8, pp. 94610-94630, 2020.
4. Harry Tanuwidjaja, Rakyong Choi, **Seunggeun Baek**, and Kwangjo Kim, "Privacy-Preserving Deep Learning on Machine Learning as a Service - A Comprehensive Survey," (submitted to *IEEE Access*, 2020.05.25), 2020.
5. **Seunggeun Baek**, and Kwangjo Kim, "Neural Distinguishing Attacks against CHAM64/128 and HIGHT," (unpublished manuscript), 2020.