IMPROVEMENT ON BIT DIFFUSION ANALYSIS OF π-CIPHER

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF APPLIED MATHEMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

BEYZA BOZDEMİR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CRYPTOGRAPHY

SEPTEMBER 2016

Approval of the thesis:

IMPROVEMENT ON BIT DIFFUSION ANALYSIS OF π-CIPHER

submitted by **BEYZA BOZDEMİR** in partial fulfillment of the requirements for the degree of **Master of Science in Department of Cryptography, Middle East Technical University** by,

Prof. Dr. Bülent Karasözen
Director, Graduate School of **Applied Mathematics**

Prof. Dr. Ferruh Özbudak
Head of Department, **Cryptography**

Assoc. Prof. Dr. Ali Doğanaksoy
Supervisor, **Department of Mathematics, METU**

Asist. Prof. Dr. Fatih Sulak
Co-supervisor, **Department of Mathematics, Atılım University**

Examining Committee Members:

Prof. Dr. Marat Akhmet
Department of Mathematics, METU

Assoc. Prof. Dr. Ali Doğanaksoy
Department of Mathematics, METU

Assist. Prof. Dr. Nurdan Saran
Department of Computer Engineering, Çankaya University

Assist. Prof. Dr. Elif Saygı
Division of Elementary Mathematics Education, Hacettepe University

Assist. Prof. Dr. Fatih Sulak
Department of Mathematics, Atılım University

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name:	BEYZA BOZDEMİR
Signature :	

ABSTRACT

IMPROVEMENT ON BIT DIFFUSION ANALYSIS OF π-CIPHER

Bozdemir, Beyza
M.S., Department of Cryptography
Supervisor : Assoc. Prof. Dr. Ali Doğanaksoy
Co-Supervisor : Asist. Prof. Dr. Fatih Sulak

September 2016, 39 pages

π-Cipher, a sponge-based algorithm designed by Gligoroski et al. [20], is a second round algorithm of the CAESAR competition. The designers of π-Cipher analyzed the bit diffusion of the parts, $*$ operation and 1 round π-function for all variants of π-Cipher [20]. They showed the results with graphics; yet, they did not give any conclusion about these results.

We improve this analysis by applying Strict Avalanche Criterion (SAC) Test in the package of Cryptographic Randomness Testing designed by Doğanaksoy et al. [16] to $*$ operation for all w values and reduced round versions of π-function for $\pi16$-Cipher. We obtain that $*$ operation for all word sizes fails SAC Test whereas reduced round versions of $\pi16$-function pass the test.

Keywords: The CAESAR competition, π-Cipher, bit diffusion analysis, Strict Avalanche Criterion (SAC) test.

ÖZ

π-CIPHER ALGORİTMASININ BİT YAYILIM ANALİZİNİN GELİŞTİRİLMESİ

Bozdemir, Beyza
Yüksek Lisans, Kriptografi Bölümü
Tez Yöneticisi : Doç. Dr. Ali Doğanaksoy
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Fatih Sulak

Eylül 2016, 39 sayfa

Sünger tabanlı bir algoritma olan π-Cipher, Gligoroski ve ekibi tarafından tasarlanan CAESAR yarışmasının ikinci tur algoritmalarından biridir. π-Cipher algoritmasının tasarımcıları, $*$ operasyonunu ve 1 turluk π-fonksiyonunu bütün π-Cipher çeşitleri için incelemiş olup, iki yapıya ait bit yayılım analizini gerçekleştirmişlerdir [20]. Elde edilen analiz sonuçlarını grafiklerle göstermişler, fakat bu grafiklere dair açıklamalar ortaya koymamışlardır.

Doğanaksoy ve ekibi tarafından tasarlanan Kriptografik Rastgelelik Test Paketi içerisinde bulunan Katı Çığ Etkisi Testi yardımıyla biz bu analizi, $*$ operasyonunun bütün $w = 16, 32, 64$ değerlerine göre ve π-fonksiyonunun $w = 16$ değerine göre 1, 2 ve 3-turluk bütün zayıflatılmış versiyonlarına uygulayarak geliştirdik. $*$ operasyonu bütün w değerlerine göre Katı Çığ Etkisi Testinden kalırken, 1, 2 ve 3-turluk $w = 16$ için incelenen π-fonksiyonu testten geçmiş ve rastgele olduğu gösterilmiştir.

Anahtar Kelimeler : CAESAR yarışması, π-Cipher, bit yayılım analizi, Katı Çığ Etkisi Testi.

To my lovely family: my mother Ayteşe, my sisters Zehra, Sema, Elif, my brothers Emin İlyas, Abdulkadir and my father Halil

ACKNOWLEDGMENTS

I would like to express my very great appreciation to my thesis supervisor Assoc. Prof. Dr. Ali Doğanaksoy and co-supervisor Assist. Prof. Dr. Fatih Sulak for their patient guidance, enthusiastic encouragement and valuable advices during the development and preparation of this thesis. Their willingness to give their time and to share their experiences has brightened my path.

I thank all my friends 3G, Ayşe Candan, Betül A. Özdemir, Birgül Koç, Derya Karademir, Merve Öğünç and Tuğba Akbaş for friendship, sharing their valuable time with me, flavoring to my life, and their guiding on the process of the undergraduate and master studies.

I also thank to all my friends and everyone at Institute of Applied Mathematics.

I am heartily thankful to Onur Koçak for helping the implementation, also Neşe Koçak, Elif Saygı ve Zülfikar Saygı for being so cooperative during the Authenticated Encryption Group Studies.

Special thanks to Murat Cenk, Muhiddin Uğuz, Ömer Küçüksakallı, Baver Okutmuştur and Canan Bozkaya for their support and guidance.

Again, I would like to thank my huge family for their endless love and support during my whole life.

Finally, I would like to thank The Scientific and Technological Research Council of Turkey (TÜBİTAK) for supporting this thesis via TÜBİTAK 3501 project numbered 114F130 and titled "CAESAR Yarışmasına Katılan Kimlik Denetimini Sağlayan Algoritmaların Kriptanalizi".

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

SAC Test	Strict Avalanche Criterion Test
Enc	Encryption
Auth	Authentication
NIST	The US National Institute of Standards and Technology
AE	Authenticated Encryption
H	Header
AD	Associated Data
AEAD	Authenticated Encryption with Associated Data
PMN	Public Message Number
SMN	Secret Message Number
IV	Initialization Vector
N	Nonce
EtM	Encrypt-then-MAC
E&M	Encrypt-and-MAC
MtE	MAC-then-Encrypt
IND-CPA	Indistinguishability Chosen-Plaintext Attack
IND-CCA	Indistinguishability Chosen-Ciphertext Attack
INT-PTXT	Integrity of Plaintext
INT-CTXT	Integrity of Ciphertext
TRNG	True Random Number Generator
PRNG	Pseudorandom Number Generator
ARX	Modular Addition, Rotation, XOR
CIS	Common Internal State of π-Cipher
IS	Internal State of π-Cipher

CHAPTER 1

Introduction

In the simplest sense, cryptography enables two or more parties to provide the privacy and secrecy of the information when there is an adversary who can access their communication channel. In the more general sense, cryptography is a science that is about providing the trustworthy information, building secure and robust cryptosystems with guaranteeing the systems implemented and embedded into devices.

Cryptography which is a cornerstone of mathematics, computer science and communication security implies that a meaningful message, the plaintext, is converted to a meaningless message, the ciphertext, using the algorithms having mathematical background. The communicating parties, Alice and Bob, choose and share a secret key $k = (e, d)$ where e is encryption key and d is decryption key in the key space K before the communication to send messages each other in a secure way. After sharing the key, Alice encrypts the message m such that $E_e(m) = c$ where E is the encryption function and c is the ciphertext and delivers c to Bob. After receiving c from Alice, he decrypts c such that $D_d(c) = m$ where D is the decryption function, and so he recovers the message m. Therefore, they can safely send and take messages.

Cryptographic algorithms have to accomplish following goals [30]:

1. *Confidentiality*: The message is protected from the unauthorized parties. To keep the plaintext in secret, it is read by only the right person who has the decryption key.
2. *Data integrity*: The message is ensured to be unchanged from the unauthorized parties. Encryption scheme provides that the message is not proceeded any deletion, substitution or insertion.
3. *Entity authentication and data origin authentication*: Data origin authentication provides the data integrity while entity authentication means that communicators should identify each other during communication; that is, message sender must be authentic.
4. *Non-repudiation*: Sender should not deny the sent message by own, the communication, etc.

According to *The Codebreakers* [22], the traces of cryptography has been found about

4000 years ago in the time of Egyptians. Cryptography provides people to communicate securely over an insecure channel that an opponent, namely attacker can find the conversation between parties meaningless. Generally, cryptography is divided into two main categories.

1. Symmetric Key Cryptography,
2. Asymmetric Key (or Public Key) Cryptography.

In public key cryptography, there exist two different keys: private key and public key for encryption and decryption, respectively. The user generates public key and secret key for its own sake. Public key is available for the usage of anyone to encrypt the message while the private key is known only by the user to decrypt the ciphertext, and it is hard to derive from public key.

Public key cryptography is mainly used in two areas: public key encryption and digital signatures. In the digital signature schemes, the sender, Bob, signs the message with his private key, then anyone, Alice, can verify that the message is from Bob with using his public key. With digital signature schemes, Alice ensures that the message comes from Bob. That is, digital signature schemes provide the authentication of the message and the integrity of unmodified message in transfer at the same time. Moreover, Bob does not deny the message which is sent by himself. Therefore, digital signature provides the integrity, authentication and non-repudiation [34]. The public key encryption scheme is performed as follows [46]. User Bob generates his private and public keys, and then tells anyone including the attacker Eve his public key. Then, if Alice wants to send a message to Bob, she uses his public key to encrypt message, and sends the encrypted message to him. Even if Eve obtains the ciphertext, she would not be able to decrypt the ciphertext because she does not know Bob's private key. Therefore, the confidentiality of message is provided. After receiving the ciphertext from Bob, he can decrypt it via his private key, and obtain the message. Therefore, Alice and Bob can communicate in an insecure channel without meeting each other.

The important matter in the public key cryptography that the algorithms are based on the difficult mathematical problems such as one-way functions whose inverses are infeasible to compute. That is, while the encryption scheme of algorithms is easy to compute, the cryptanalysis of the decryption is computationally hard. For example, the factorization of integers, discrete logarithm problem and elliptic curves are the hard problems [34], and also they have no efficient solutions for big integers. Due to being hard problems, confidential message transmission, authentication, key exchange, bit commitment, secret sharing, e-voting, etc. are carried out with well-known public key cryptosystem examples such as RSA in 1977 [37], Zero-Knowledge Proof, Diffie-Hellman Key Exchange in 1976 [13], elliptic curve cryptography in the mid 1980s etc.

In contrary to public key, symmetric key cryptosystem has a single key, secret (or private) key, which is agreed and shared when two parties meet before the conversation. To encrypt or decrypt a message, they use the same key. Due to using same key and the algorithms, symmetric key cryptosystem is faster than asymmetric key algorithms.

Therefore, symmetric key cryptography is preferred more in the daily life operations like using bluetooth, social networks or online shopping. For example, AES [12] is determined as a standard to encrypt the data in the electronic environments.

If Alice wants to use the symmetric key cryptosystem, she needs to choose the following ciphers:

- Block Ciphers,
- Stream Ciphers,
- Hash Functions,
- Message Authentication Codes (MAC),
- Authenticated Encryption Schemes.

Block ciphers are widely used encryption schemes that proceed on the blocks such that

$$E_K(P) := E : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n, and$$
$$E_K^{-1}(C) := D : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^n$$

where K is key, k is the key size, P is plaintext, C is ciphertext and n is the input and output size. The enciphering scheme divides the plaintext into the fixed length blocks and then encrypts or decrypts one block at a time. That is, a block cipher takes an n-bit block of plaintext with encrypting k-bit secret key and produces n-bit ciphertext. To be a unique decryption map, there exists a bijection map from $\{0,1\}^k \times \{0,1\}^n$ to $\{0,1\}^n$. Actually, the encryption/decryption is n-bit permutation since every plaintext is proceeded into ciphertext vice versa for every key in $\{0,1\}^k$. Due to the encryption/decryption on block by block, it provides an advantage. For example, if there is a mistake in one block, it does not affect the other blocks. However, it also provides disadvantages such that identical blocks are encrypted to identical blocks. To get rid of the disadvantages, one solution is to use the modes of operations. Block ciphers are generally based on an iterated product cipher [41] which combines two or more transformations such as substitution (S-box), permutation (P-box) or modular arithmetic with multiple similar rounds and different subkeys getting from the master key for each round to be more secure against the cryptanalysis.

The known block ciphers are Feistel ciphers, substitution-permutation networks, Lai-Massey ciphers. The well-known examples of block ciphers are DES [21], AES [12], IDEA [28], RC5 [36].

Stream ciphers [34] are encryption schemes on bits that plaintext bit is encrypted with the keystream bit at a time where plaintext bits $m_0m_1m_2\dots$ are the string of plaintext space P, and keystream bits $k_0k_1k_2\dots$ in the key space K are the randomly generated string of the defined alphabet. In other words, the encryption is that the plaintext bits are XORed with keystream bits, and so produces the ciphertext bits such that $E_{k_i}(m_i) = c_i$ where $c_i \in C$, the ciphertext space. Moreover, the decryption of ciphertext is that $D_{k_i}(c_i) = m_i$. Briefly, the decryption D and encryption E functions can be said that they are same because the operation is XOR. Because of the easiest encryption and decryption, the security of it depends on only keystream. However, the keystream bits are not directly the key bits. The construction of keystream is so important that key is used to generate it via keystream generator.

The stream ciphers are two types: synchronous and non-synchronous. While a non-synchronous stream cipher is that each keystream element depends on the previous plaintext or ciphertext bits as well as the key bits, synchronous stream cipher depends on only the key bits, but it is independent from the plaintext bits [46]. A synchronous stream cipher can be represented as a linear recurrence of degree l where the last term of keystream depends on the previous l terms. The best example of synchronous stream cipher is Linear Feedback Shift Register (LFSR). LFSRs are implemented efficiently in hardware due to XOR. Yet, the output of LFSRs has a linear recurrence relations. Due to the fact that the output of LFSRs or a keystream generator should look like a random sequence. Otherwise, the opponent Eve can guess the keystream bits so random keystream is crucial for the security of stream ciphers.

Hash functions $h : \{0,1\}^l \rightarrow \{0,1\}^n$ where $n < l$ such that $h(x) = y$ are functions not using any keys to squeeze arbitrary length of input message and produce a fixed length of output called as a *hash value* or *message digest*. These functions provide the integrity of information rather than confidentiality. That is, they guarantee that the data is not tempered. They are not invertible and reversible, i.e., they are one-way functions. Hash functions which construct a short *fingerprint* of input must have some properties:

1. *Preimage Resistance*: Given any hash value y, it should be computationally infeasible to find any input value x such that $h(x) = y$ where h is a hash function. This property is related to be one-way function.
2. *Second Preimage Resistance*: Given x, it should be infeasible to find $x \neq x'$ in the domain of h with $h(x) = h(x')$ where h is a hash function.
3. *Collision Resistance*: It should be infeasible to find two distinct values x and x' in the domain of h such that $h(x) = h(x')$ where h is a hash function.

There are various hash function algorithms in the literature. Some are not used anymore due to the shown weaknesses. Hash functions can be undermined by applying successful attacks, not exactly to be broken. However, the shown weaknesses are enough to need more strengthened one. Because of that, SHA (Secure Hash Algorithm)-3 Competition is arranged to supply the need of a new hash function as a standard hash function. The winner of the last SHA Competition, SHA-3, is Keccak on October 2, 2012 [9].

Message Authentication Code (MAC) is a keyed cryptographic hash function. MAC takes inputs namely a secret key and an arbitrary length of message, and so produces an output called as a MAC value or *tag*. The differences of MACs from the hash functions are that firstly MACs use key, and secondly they provide not only data integrity but also data authenticity. MACs work into three parts: the random key generation, the sign of the given key and message, and the verification of the authenticity of the given key and message.

The foremost for MAC function to be secure is that MAC is hard to compute a valid tag of the given message without key. Therefore, MAC function should have some security requirements. For example, MAC have to be secure against existential forgery. Moreover, MAC is different from digital signature because a tag generation and verification is done by the same secret key while digital signature uses two different keys. Also, MAC does not guarantee the aim of non-repudiation.

1.1 Authenticated Encryption Scheme

Confidentiality and authentication are two important security goals to be basically provided. With technological developments especially developments on the Internet, some applications need confidentiality and authenticity at the same. The application systems use symmetric key encryption schemes such as block ciphers and hash functions for confidentiality, and they use one-way functions, hash functions, keyed and unkeyed hash functions such as MAC, MDC to provide the authenticity.

Authenticated encryption is a block cipher based mode of operation which achieve the security in terms of confidentiality and authenticity, i.e., AE provides confidentiality, integrity and authentication on the data will be proceeded, and also it enables to the verification in the decryption part. To give details of AE, the encryption scheme of AE algorithm takes a plaintext, a key and a initialization vector (IV) or a nonce (N) and then it outputs a ciphertext and maybe a tag in the processes of encryption and tag generation. Given a ciphertext, a key, a tag and IV or N, the decryption process of algorithm takes the ciphertext, the key, the tag and IV or N, and it returns either the plaintext or *INVALID*, a invalidation symbol $\perp$ in the decryption and verification stages. If in the verification process, given tag is invalid, then decryption of ciphertext is meaningless and outputs $\perp$.

Generic approaches on AE that use AE as a black boxes having two different keys K_e and K_m were suggested as follows [4]:

1. MAC-then-Encrypt (MtE): The first of MtE is the generation tag T of message P with K_m, the second phase is that newly generated tag is appended the message, and then new message $P||T$ is encrypted with using K_e and produced C. The decryption of MtE is that C is decrypted with K_e to get concatenated message $P||T$. After separation of P and T, the sent tag and newly created tag are compared each other. This decryption method is Decrypt-then-Verify. MtE is used in SSL/TLS.

2. Encrypt-and-MAC (E&M): The Encryption and MAC proceed in parallel; yet, two are occurred separately, and they are concatenated at the end of AE. In the process, the message is encrypted with K_e producing ciphertext C, and the tag T of message is generated with using K_m. Then the result of E&M looks like

$C||T$. The reverse side of AE is proceeded like decrypting the ciphertext C, and verifying the sent tag T with a new created tag T', i.e., it is Decrypt-then-Verify. This approach is used in SSH protocol.

3. Encrypt-then-MAC (EtM): First of all, the message is encrypted with K_e to produce C, and then the encrypted message with K_m is used for the generating the tag T. After that, C and T are concatenated each other. The reverse side of this approach is different from the formers. It is Verify-then-Decrypt that firstly T and C are separated, and new tag is generated to compare the sent one. If the verification is valid, then message is decrypted with K_e; otherwise, the decryption outputs an invalid symbol, $\perp$. EtM is used in the protocol IPSec.

All three approaches were examined by Bellare and Namprempre [6], and they have showed that among three of them, Encrypt-then-MAC has more secure in all cases such as IND-CPA, IND-CCA, INT-PTXT and INT-CTXT. EtM scheme [4] is efficient in the respect of verification and decryption that save the unnecessary cost of decryption.

1.1.1 Authenticated Encryption with Associated Data

Authenticated encryption with associated data (AEAD) scheme is an extended version of AE scheme. It is represented firstly by Rogaway [38]. In this scheme, the difference from AE is that there is an additional data called as Associated Data (AD) or Header (H) which is not required the need of encryption; yet, AD is important to provide the authenticity [4]. AEAD is encryption scheme that provides authenticity and privacy using AD. Moreover, AEAD provides a robust definition. Encryption and decryption of AEAD scheme shown as follows:

$$E_K^{N,AD}(M) = E_K(N, AD, M) = C \textit{ and } T,$$
$$\text{and}$$
$$D_K^{N,AD,T}(C) = D_K(N, AD, T, C) = M \textit{ or } error, \perp$$

where M: message, N: Nonce, AD: Associated Data, K: key, C: ciphertext, T: tag.

Two ways are existed to transform AE to AEAD [38]:

- *Nonce stealing*: AD is a part of Nonce (N). If the length of Nonce is 128 bit in the AE scheme, in the nonce stealing case, $|N| = 32 \Rightarrow |AD| = 96$ where N is like a counter. Sometimes, the length of AD is 32-bit for IPv4 addresses for a specific need.
- *Ciphertext translation*: AD is authenticated with the help of a function family, $F : K' \times AD \to \{0,1\}^\tau$ such that $\Delta : F_{K'}(AD)$ and $E_K^N(M) = C$ where C is the XOR in Δ to the last τ-bit.

In the generic composition [38], there are two considered methods: Nonce-based with AD in the EtM and Nonce-based with AD in the MtE. The former one is that anyone encrypts the message with using N, and produces the ciphertext. Then its MAC is computed with using AD, N, ciphertext to obtain the tag T. Its output is $C||T$. The latter method is that with using AD, N, message the tag T is computed, and it is encrypted using N, message and T.

1.2 Recent Projects

Recent projects AES, NESSIE, CRYPTREC, eSTREAM, SHA-3 and PHC are open and transparent to decide the winner algorithms since these projects are known by all over the world, and all algorithms in the projects are analyzed. Therefore, some algorithms are eliminated from the stages of projects, and robust and secure algorithms are efficiently determined as the winners. The details of projects are given as follows.

The Advanced Encryption Standard (AES) competition [1] was a three-year competition to find a standard for encrypting the electronic data. The competition was organized by NIST. During years 1997-2000, 15 algorithms was presented as AES candidates, and designs was evaluated in terms of the security and the performance. Rijndael was selected as the winner of AES competitions among 5 finalists MARS, RC6, Serpent, Twofish, Rijndael on October 2, 2000. What distinguishes Rijndael from the others is that it has an efficient software and hardware performance, and it keeps less memory space. To use as a standard, some properties of Rijndael were modified, and it has been called as AES.

New European Schemes for Signatures, Integrity and Encryption (NESSIE) [32] was a project to manifest reliable cryptographic primitives in year between 2000 and 2003. In several categories such as block ciphers, public key encryption, etc. the cryptographic primitives were examined for the security claims. 42 algorithms were submitted the project. Among 42, 12 submissions were announced as "selectees". Some selectees are MISTY1, Camellia, Rijndael as block cipher, RSA-KEM as public key encryption, CBC-MAC as MAC algorithm and hash functions,etc.

Cryptography Research and Evaluation Committees (CRYPTREC) [11] is set by Japanese government to propose and investigate cryptographic primitives for the need of government and industries on 2000. The committee consists of Japanese academia, industry and government. The committee created "e-Government" in 2003. Similar to NESSIE, they selected some algorithms in some categories.

The ENCRYPT Stream Cipher Project (eSTREAM) [18] aimed to find a new stream cipher on the worldwide application in the years 2004-2008. eSTREAM was organized by the EU ECRYPT network. Algorithms on the competition were expected to be suitable for hardware or software applications. Moreover, they were permitted to be authenticated ciphers by producing output which is tag with ciphertext. In the competition eSTREAM, 7 algorithms with 4 software and 3 hardware based, were announced as portfolio. The portfolio algorithms were HC-128 (HC-256), Rabbit, Salsa20/12 (Salsa20/8, Salsa20), SOSEMANUK, Grain, MICKEY and Trivium.

The signal of *SHA-3 competition* [40] was given out by NIST with Cryptographic

Hash Workshop on 2005, and the second Cryptographic Hash Workshop on 2006. In 2007, SHA-3 was arranged to create a new hash function as a standard. The finalists, BLAKE, Grϕst, JH, Keccak and Skein were announced on 2010. The winner of competition was Keccak designed by Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche, and it ended a five-year competition. The standardization of SHA-3 was released on 2015.

Password Hashing Competition (PHC) [35] was announced in 2013 to construct password hash functions, and gave a standard password hash algorithm. The aim of competition answered the question of modern world applications which are protected towards the attackers. It was modeled similar to AES and SHA competitions. The PHC winner was revealed as Argon2.

1.3 CAESAR Competition

In recent years, successfully finished competitions like AES [1], eSTREAM [18], SHA-3 [40] have been organized to answer the demands of the industry and interests in the research community. Similarly, the CAESAR competition [7] has been initiated in 2014 in order to boost the design for the authenticated encryption tool which provides the privacy and authenticity together. The CAESAR competition is different from the previous competitions AES [1] and SHA-3 [40] done to determine the standard algorithm since the winner is going to be determined by the competition committee being created with the prominent academicians [7] not by NIST [33]. In addition, the competition allows the designers to tweak their algorithms. These features are similar to the competition of eSTREAM [18].

In the competition, most of algorithms are shown the traces of the construction of algorithms in the competitions AES and SHA-3. According to the construction of algorithms, the functions of AES are used directly or indirectly. Likewise, some algorithms are used the stream cipher based system or a sponge based construction like in the Keccak.

Nowadays, authenticated encryption is used almost everywhere such as online shopping, smart cards, social networks, etc. It is a hot topic that some cryptographic algorithms are in used to provide the authenticated encryption. The intensive cryptographic studies such as design and cryptanalysis of algorithms result a standard algorithm that does not provide the privacy and authenticity simultaneously. Because of that, some algorithms were designed to provide AE. Yet, the cryptanalysis of algorithms was not investigated in detail so the security of them are created the question marks. To solve the questions, the competition CAESAR was organized. CAESAR is the acronym of Competition for Authenticated Encryption: Security, Applicability, and Robustness.

The CAESAR competition has three-round elimination and the final round. Throughout three rounds of the competition, a number of candidates will be eliminated at the end of each round. Therefore, the security analysis of candidates is very important for the CAESAR competition. Having passed all three rounds, the algorithms are going to be determined by the votes of committee, and then they will be announced as the final portfolio.

There are 57 submissions for the first round of the CAESAR competition in terms of block based, sponge based, permutation based and compress function. However, some are withdrawn. Moreover, 19 of them are eliminated from the competition at the end of the first round. Therefore, there are 29 algorithms left for the second round of the CAESAR competition, and the third round started at the August, 2016 [8] with 17 algorithms. The final portfolio will be announced in December 2017 [7].

1.4 Random Numbers

1.4.1 Randomness

In statistics, an item from a set is random if each item of the set has an equal probability to be chosen. The key, IV or Nonce in an encryption session must be random to avoid predictability. Moreover, an encryption process is supposed to hide the redundancies in the plaintext space. That is, the resulting ciphertexts should be appearing random to avoid any obvious statistical cryptanalysis.

To test randomness, a random variable is defined over the space of binary sequences. Then depending on the distribution function of this random variable, a sequence or a set of sequences is accepted or rejected for being random.

In the 20^{th} century, science and its outcome, technology, has developed over and over, randomness is emerged to be so important. It has been understood that it has a relation with statistics, physics, probability theory, information theory, cryptology, and like the like. For example, in physics, thanks to the idea of random motions of molecules, statistical mechanics is developed, and it explains phenomena in thermodynamics and properties of gases [30]. The security of cryptographic systems depends heavily on the randomness of keys.

In cryptology, random numbers are used for the applications of private keys in symmetric systems, IV and nonce in the initialization step of algorithms, pin-password generation, key for MAC algorithms, RSA, banking or GSM process [30]. Random numbers are so important that they are the sign of the security criteria of these kinds of cryptographic systems. Therefore, the system having a high qualified random number generator is thought to be unbreakable. The systems need to ensure random number, namely key, due to attacks, privacy, authenticity or anonymity. The designers of an algorithm should give its security strength in bits which are guessed by attacker to break the system. If the ciphertext doesn't look like random bits, i.e., ciphertext has some clues about plaintext, then an opponent, Eve, will use them to get the key, or whole message. Thus, it can be said that algorithms must hide the statistical properties of ciphertext since sometimes it is not enough to be large key space without durable encryption function. The another need of random numbers in cryptology is to provide the privacy that requires uniform randomness. In addition to the importance of randomness on the key generation, it is important for many cryptographic algorithms, especially encryption algorithm. An encryption algorithm need to have a rule that is adequate to permit work to be random, and so ciphertext of that algorithm is indistinguishable from the random mapping. Moreover, the reasons for random numbers make cryptographic systems unpredictable and fresh. The freshness of systems are provided with help of using nonce, counter, IV or AD. The other reasons for randomness are noise and efficiency. While noise provides to hide the properties of input, efficiency can makes the system speed up.

Generating, storage or transmission of *good* random numbers [42] is a crucial problem. The problem of generation of random numbers may be overcome with rolling an ideal dice or tossing an ideal coin. However, a million times of experiment on dice or coin are not efficient. The other way to get random numbers is using the natural phenomena such as the sound from the environmental noise, elapsed time in the radioactive decay, mouse motions, etc. However, the generation of random numbers with these kinds of methods is also inefficient. The reason for this, using Pseudorandom Number Generators (PRNGs) is the most option to get random numbers that have long length with using small length of $seed$ which has true randomness. For example, full round AES is a PRNG with the given plaintext and key. PRNGs are based on the deterministic systems, and so they have a big period. Due to the determinism of generators with using seed from these natural phenomena, an attacker can guess some properties of generators, even s/he can get whole numbers which were produced or will be produced.

To ensure that PRNGs are reliable, they must be subjected to randomness tests which are designed to detect the property of randomness [30]. If a generator passes tests as many as possible, generator is thought that it proved to be confidential.

When deciding whether an algorithm is random, we need to investigate or test the randomness of produced sequences from the algorithm. If these sequences pass the randomness tests, i.e., sequences generated from the algorithm are random, then the algorithm can be accepted as random.

1.4.2 Random Sequences and Random Numbers

Given a binary sequence, if each term in that sequence has an equal probability, namely $1/2$ for being 1 and 0, then the sequence is a binary random sequence.

A process which produces an integer between $(0, n-1)$ is random if at each instance its outcome can take any value with an equal probability $\frac{1}{n}$. Any outcome of such a random process is called a random number.

Random numbers have the following properties [42]:

1. *Unpredictability*: Given any t bits of a binary sequence, it is computationally hard to predict the next bit of sequence.
2. *Uniformity*: In a random binary sequence, the numbers of $1's$ and $0's$ are al-

most same, i.e., zeros and ones spread uniformly throughout the random binary sequence.

3. *Independence*: Any two distinct terms of the sequence are equal with probability $\frac{1}{2}$.

1.4.3 Randomness Tests

Randomness tests are functions taking an input sequence and outputting a real value which are so important to seek the randomness of PRNGs [17]. There are two kinds of randomness tests: Statistical randomness tests and Cryptographic randomness tests. Each statistical test measures whether algorithms show the randomness property or not. Each statistical test has a distribution function which produces a value called as, p-value in $[0, 1]$. If p-value of any test applied on a sequence is resulted over a pre-determined threshold value, then it is concluded that the sequence is random. On the contrary, if the result is below this decided value, the sequence is labeled as non-random. In terms of results of statistical randomness test, cryptographic randomness tests are the same. However, cryptographic randomness tests search the weaknesses of algorithms in term of cryptographic primitives. During search on the randomness of cryptographic algorithms, they are seemed as PRNGs as mentioned before, and like PRNGs, they are investigated by using randomness tests. The reason for this, output of algorithms should be random looking and give no clue about the plaintext or key; therefore, when analyzing the output, guessing the algorithm could not be possible [49]. Algorithms could not be identified from a random mapping so analyzing the algorithms with randomness tests is important.

If the algorithm fails in many of randomness tests, then algorithm is seemed $non-random$. However, if algorithm fails any one of them, while someone takes the algorithm non-random, the others accept the algorithm *random looking* after checking its randomness with further tests. Nonetheless, if an algorithm passes all randomness tests, the algorithm is approved as $random$. That means the algorithm produces sequences having significant characteristics like a random generator [30].

When searching on a short sequence to be applied to randomness tests, it wouldn't be right to mention that it is applicable to be random since the definition of randomness on the short is not easy. However, it is more precise to investigate the long sequences for randomness to apply randomness tests [42].

It is important that tests are applied on the reduced round versions of algorithms since analyzing the reduced round versions of algorithm is an important role on the design of the algorithms. If the algorithm shows a deviation from randomness in the first rounds, the designer(s) of algorithm need to increase the number of rounds, or if an algorithm is non-random in the last rounds, it can be thought that the algorithm must be non-random, and also it can be breakable in a simple cryptanalysis technique. Thus, finding which round the algorithm shows non-random properties is so crucial.

There are various test packages and statistical randomness tests in the literature [2, 5, 14, 15, 23, 25, 26, 29, 39, 47, 48].

1.5 Motivation

In this thesis, we study on the algorithms in the CAESAR Competition and the statistical randomness testing, especially cryptographic randomness testing designed by Doğanaksoy et al [16]. We analyze π-Cipher that is one of the second round algorithms in the CAESAR Competition. π-Cipher has a permutation function which is π-function consisting of $*$ operation. There is two consecutive transformations E_1 and E_2 in the π-function. These transformations use the $*$ operation [20]. We examine $*$ operation and π-function and analyze the bit diffusion analysis of the $*$ operation and π-function given by designers [20]. The authors gave the method of $*$ operation for all w word sizes and 10000 random inputs. The results of this method were shown only with the graphics. They did not give any conclusion about what the graphics indicate or what the given results mean. The second method they gave is for one round of π-function for all w variants with 1000 random inputs. In the same way, the results were shown only with the graphics and no conclusion about them. Because of the insufficient inputs and no conclusion about the obtain results, we analyze the parts of π-Cipher, $*$ operation for all $w = 16, 32, 64$ values and the reduced rounds of π-function for $w = 16$, and apply the SAC Test on these parts. We improve the bit diffusion analysis with using 2^{20} random inputs on $*$ operation for all w variants and reduced round versions (1, 2 and 3 rounds) of π-function for $w = 16$ to apply SAC Test and using the different interpretation technique for the obtained results.

CHAPTER 2

Cryptographic Randomness Testing

In this chapter we will introduce the cryptographic randomness tests package in which especially the Strict Avalanche Criterion Test. We use the SAC Test defined in the recent package is designed by Doğanaksoy et.al [16] to evaluate block ciphers and hash functions via cryptographic randomness tests. This package consists of 4 tests; SAC Test, Linear Span Test, Collision Test and Coverage Test.

2.1 Cryptographic Randomness Testing

Although random numbers play an important role in the cryptographic systems, the generation of random numbers is difficult. To generate a true random number, we can use the true number generators (TRNGs). Although these sources are nondeterministic to produce random numbers, the generation with these sources, storage and transfer of random numbers are problematic. The solution to this problem is to use the deterministic algorithms that are pseudorandom number generators (PRNGs). PRNGs take a random binary sequence of length k and produce a periodic random looking binary sequence of length $l >> k$ [30]. The outputs of these sources are pseudorandom. Because of pseudorandomness, the outputs must be checked whether they have certain non-randomness properties. They are subjected to the statistical tests which are designed to detect the characteristics expected from a random sequence. With that aim, NIST published a suite of randomness tests [5] which are used to evaluate the sequences to compare a truly random sequence via its probability; that is, the output of generator should not be distinguishable from the random sequence, that is, it should be *random looking*. Soto et al. apply NIST randomness tests suite to the candidate and finalist algorithms in AES competition [44, 45].

Cryptographic algorithms are constructed on the generation of quantities which are not easily predictable to provide the security of algorithms. Since an adversary does not have a clue to catch the leakage of the system or even to break the system, the generated quantities must have an adequate length and size or have randomness property, etc. Although the randomness property is needed to generate a key for asymmetric or symmetric systems, the keys generated from a deterministic source may cause the system to be broken if they show nonrandom properties [30].

Cryptographic primitives are indiscriminate from a random function. Block ciphers, hash functions or the other ciphers need to behave like a random mapping. Because of that, all cryptographic algorithms must satisfy cryptographic criteria. The nonlinearity of Boolean functions, collision resistance of hash functions are some criteria. For example, on the block cipher, one of the cryptographic primitives is Strict Avalanche Criterion. The criterion Strict Avalanche is that if one input of cipher is changed, then the ciphertext bits need to be affected by changing with the probability $one\ half$.

In short, the tests in the cryptographic randomness testing package designed by Doğanaksoy et.al [16] are as follows:

1. The Linear Span Test evaluates the linear dependence of ciphertext of the algorithm resulted from the encryption of highly linear dependent input set.
2. The Collision Test examines the collisions of some portions of ciphertext corresponding to a random subset of input set.
3. The Coverage Test looks for the size of ciphertext resulting from a random subset of input.
4. The SAC Test searches the SAC property of algorithms.

We use the Sac Test in this thesis. The details of the SAC test are the following section.

2.2 Strict Avalanche Criterion (SAC) Test

According to Claude Shannon [41], there exist two important principles: *confusion* and *diffusion* for all modern block ciphers to get strong ciphers. Diffusion is a criteria that one plaintext bit change affects on the almost all ciphertext bits without any shown statistics of a changed plaintext bit. In other words, each bit of plaintext and key should influence many ciphertext bits [24]. Confusion is the other important criteria that there is no evidence in the relation between key and ciphertext. The relation between the statistics of ciphertext and key must be as complex as possible. That is, the ciphertext depends on the plaintext statistics in a complicated manner to be broken by cryptanalyst [24]. In the block cipher, the confusion is provided by substitution layer, S-boxes; on the other hand, diffusion is provided by permutation layer, P-boxes. With using iterated substitution and permutation layers in a round, also using multiple rounds, the secure and efficient block cipher based algorithms are basically produced.

Ciphertext should not give any clue about plaintext, i.e., $\Pr(P = m|C) = 2^{-n}$ where n is the length of key which means that any chosen plaintext m corresponding to ciphertext C should spread randomly in P, and the other side of argument should be provided.

Similar to the importance of diffusion and confusion for block ciphers, collision resistance is one of desirable cryptographic properties for hash functions. If cryptographic algorithms do not have these kinds of properties with significant degree, then they are considered to have poor randomization. Infact, this situation is sufficient to break the algorithms. Hence, cryptographic randomness testing is crucial for the algorithms to determine security levels [49].

SAC Test is primarily recommended for S-boxes by Webster and Tavares in 1986 [51]. For an S-box, $S : \mathbb{F}_2^n \to \mathbb{F}_2^m$ satisfies Strict Avalanche Criterion (SAC) if for all $i \in 1, \cdots, n$ and $j \in 1, \cdots, m$ flipping the i^{th} input changes the output j with the probability $\frac{1}{2}$; therefore, the bias is 0.

Furthermore, SAC Test is defined in test package designed by Doğanaksoy et. al [16]. SAC test measures whether one input bit change affects any output bit changes with probability $\frac{1}{2}$ or not. To test SAC property, SAC Matrix is formed using 2^{20} different random inputs and corresponding outputs.

A sample of SAC Matrix for 2^{20} trials looks like as follows:

$$\begin{bmatrix}
524288 & 525371 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 526291 \\
523478 & 524270 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 522365 \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
\vdots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \vdots \\
523781 & 524376 & \dots & \dots & \dots & \dots & \dots & \dots & \dots & 524661
\end{bmatrix}_{n \times n}$$

When applying SAC test, random inputs plaintext, key or AD are generated, and key is fixed. The aim of SAC test is to control the property of SAC of algorithms. To control SAC property of plaintext and key, SAC Matrix is constructed, and if the values on the matrix are approximately equal to the half of 2^{20}, it will be assumed that the algorithm has the SAC property, and also it will be concluded that it looks like a random mapping.

If we take K as the number of hits which an entry gets, then the probability of it [50]

$$\Pr(K = k) = \frac{\binom{n}{k}}{2^n}$$

The value of the $n \times n$ SAC matrix is evaluated with using χ^2 Goodness of Fit Test with the probabilities derived from 2.1.

The one evaluation method of the cryptographic randomness test is χ^2 *Goodness of Fit Test* which we have used in the evaluation of SAC Test results. The test is used for

Table 2.1: SAC Test Ranges and probabilities for 2^{20} trials

Bin	Range	Probability
1	0-523447	0.100067
2	523448-523633	0.100039
3	523634-523759	0.100490
4	523760-523859	0.100400
5	523860-523944	0.099425
6	523945-524021	0.100249
7	524022-524092	0.099741
8	524093-524160	0.100666
9	524161-524225	0.099447
10	524226-1048576	0.099476

evaluating the categorical distribution of each entry of $n \times n$ SAC Matrix [49]. With the test, we can compare the observed data with the theoretical expected data in the distribution. Therefore, we can determine whether the observed values are consistent with the expected values, or if the χ^2 statistic results a large value, then we can interpret that the observed and the expected data are not close, and also this model is not suitable fit to the data. The details of the evaluation method on the application of SAC Test is as follows [49]:

- Divide the interval [0,1] into 10 equal subintervals,
- Apply a Goodness of Fit Distribution Test.

$$\chi^2 = \sum_{i=1}^{10} \frac{(F_i - m \cdot p_i)^2}{m \cdot p_i} \quad \text{and} \quad p\text{-value} = \texttt{igamc}\left(\frac{9}{2}, \frac{\chi^2}{2}\right)$$

where F_i: the number of p-values in subinterval i, m: the number of subsequences, p_i: the probability of subinterval i, $igamc$: the incomplete gamma function.

If the p-value is less than 0.01, then it is assumed that the algorithm is $non - random$.

SAC test is done as follows:

1. Set the $n \times n$ SAC Matrix entries to 0.
2. Get a random plaintext and compute the corresponding ciphertext (original output).
3. For each $1 \leq i \leq n$:
    - Flip the i^{th} bit of the input and get the corresponding output.
    - XOR the original output with the corresponding output.

- Write the XOR result of the original output with the corresponding output on the i^{th} row of SAC Matrix.

4. Repeat this process for 2^{20} different random inputs.

We use 2^{20} different random inputs and corresponding outputs sets. After this process, SAC Matrix is obtained. Using χ^2 *Goodness of Fit Test*, SAC Matrix is evaluated and p-value is obtained. Then the algorithm behaves like a random mapping is acquired, according to the corresponding p-value which helps us to estimate the security level of the algorithms [43].

The algorithm of SAC Test is as follows [50]:

Algorithm 2.2.1: SAC TEST(n)

```
$Construct\ n \times n\ integer\ arrays\ SAC[][];$
for $i \leftarrow 1$ to $n$
  do
    for $j \leftarrow 1$ to $n$
      do
        {$SAC[i][j] \leftarrow 0;$
for $i \leftarrow 1$ to $2^{20}$
  do
    $Take\ a\ random\ n - bit\ binary\ array\ a[];$
    $output1[] \leftarrow f(a[]);\ where\ f\ is\ the\ algorithm$
    for $j \leftarrow 1$ to $n$
      do
        $b[i] \leftarrow a[i] \oplus 1;$
        $output2[] \leftarrow f(b[]);$
        $SAC[i][j] \leftarrow SAC[i][j] + (output1[j] \oplus output2[j]);$
$Apply\ \chi^2\ of\ Goodness\ of\ Fit\ test\ to\ all\ entries\ of\ SAC[][];$

return $(p - value)$
```

If there is a difference on any entry (i, j) of $n \times n$ SAC matrix for an input, or there also exists a difference on the same entry of the obtained SAC matrix for other input, the difference will show that there is a signal of the relation between plaintext and ciphertext on that entry.

CHAPTER 3

π-Cipher

In this chapter, we will introduce our main algorithm π-Cipher which was a second round algorithm of the CAESAR Competition and the 3^{rd} round announced at August 2016. Moreover, we will express bit diffusion analysis given by designer, and attack done by others.

3.1 π-Cipher

πw-Cipher$n$ which is an AEAD cipher provides w-bit word sizes and n-bit security which is the length of key. π-Cipher has general design properties such as

- parallel,
- incremental,
- provably secure,
- Nonce based AEAD.

π-Cipher is an Encrypt-then-MAC algorithm; in fact, it is a stream $OAE2+$ algorithm. Its parallel and incremental design is similar to counter based XOR-MAC. It uses a counter based sponge based component, namely *triplex component*. The triplex component uses π-function which is a permutation on ARX operations.

Table 3.1: π-Cipher Variants

π-Cipher variants	w word size	$klen$ in bits	SMN in bits	PMN in bits	N Chunk	T in bits
π16-Cipher096	16	96	0 or 128	32	4	≤ 128
π32-Cipher128	32	128	0 or 256	128	4	≤ 256
π64-Cipher128	64	128	0 or 512	128	4	≤ 512
π64-Cipher256	64	256	0 or 512	128	4	≤ 512

π-Cipher has a flexible block size. Encryption/Authentication part of it uses a fixed-length key, message and AD in bytes; also, it accepts the fixed-length PMN and SMN. Moreover, its output is the ciphertext and a fixed-length tag in bytes.

π-Cipher has 4 different variants in the version v2.0:

- π16-Cipher96,
- π32-Cipher128,
- π64-Cipher128,
- π64-Cipher256

which is designed by Gligoroski et al., and the size of internal state varies according to the word w and $N = 4$ so it can be 256, 512 or 1024 bits respectively $w = 16, 32$ and 64. Moreover, it consists of three rounds [20].

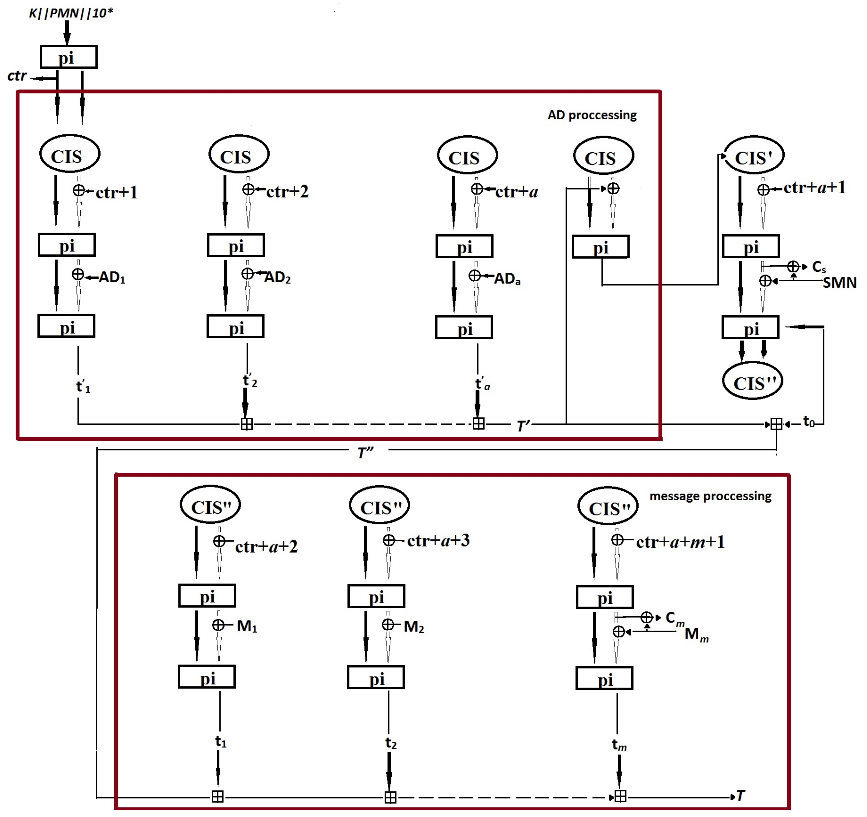

Figure 3.1: π-Cipher process of initialization, secret message number, associated data and plaintext; also, generation of tag.

As shown in Figure 3.1, the encryption scheme is divided into four parts: initialization, associated data processing, secret message number processing, processing plaintext with generation of tag.

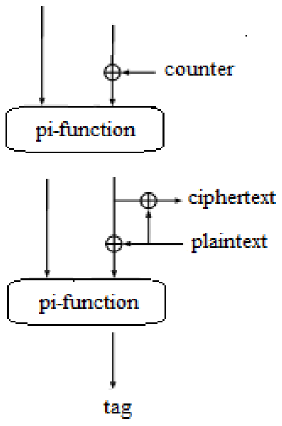

Figure 3.2: The triplex component for encryption (e-triplex).

Moreover, the encryption/authentication and decryption/verification of the algorithm have a new construction namely *triplex component* which is related to the dublex sponge as in the Figure 3.2.

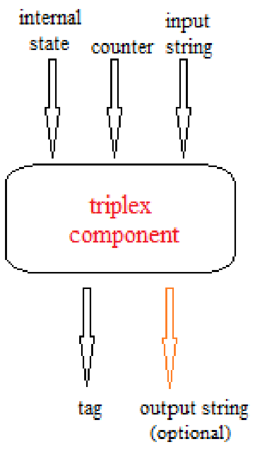

Figure 3.3: The triplex component.

The triplex component takes the internal state, counter and input string as inputs, and then it always outputs the authentication tag as in the Figure 3.3.

The 4 parts of π-Cipher use a permutation which proceeds the π-function. π-function is an ARX based permutation and is the core part of the algorithm. It consists of three rounds. Each round has two consecutive transformations called E_1 and E_2. These transformations are based on $*$ operation given in Figure 3.4. In other words,

$$Z = X * Y \equiv \sigma(\mu(X) \boxplus_4 \nu(Y))$$

where $\boxplus_4$ is the component-wise addition of two vectors of 4 dimensions in $\mathbb{Z}_{2^w}^4$, w= 16, 32, 64, and X, Y and Z in $\mathbb{Z}_{2^w}^4$ have different word sizes for types of π-Cipher [20].

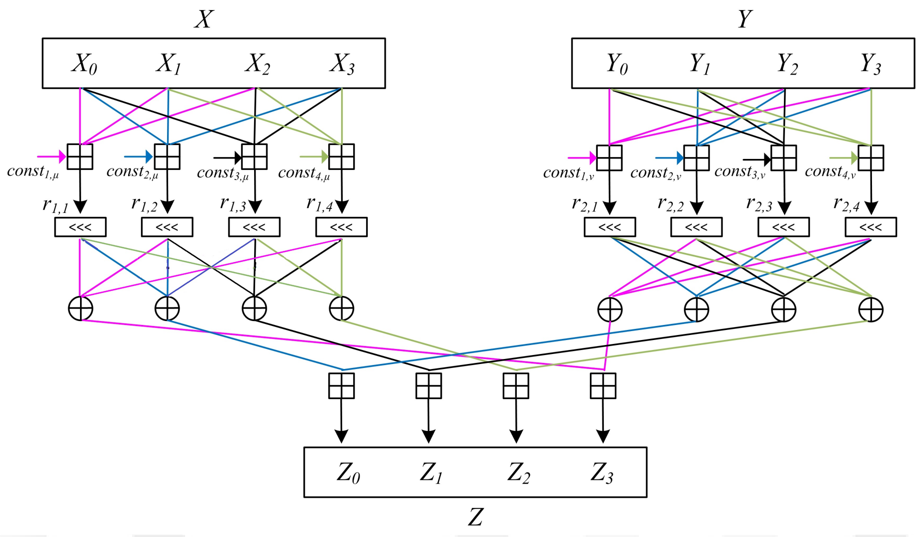

Figure 3.4: Graphical representation of $*$ operation.

Details of the transformations σ, μ, ν that provide the diffusion of nonlinear mixing of input variables can be found as follows:

- The transformation $\sigma : \mathbb{Z}_{2^w}^4 \to \mathbb{Z}_{2^w}^4$ is a permutation such that

  $$\sigma(X_0, X_1, X_2, X_3) = (X_3, X_0, X_1, X_2)$$

  where $(X_0, X_1, X_2, X_3) \in \mathbb{Z}_{2^w}^4$.

- The transformation $\mu : \mathbb{Z}_{2^w}^4 \to \mathbb{Z}_{2^w}^4$ is a permutation such that

  $$\mu(X) = A_2(ROTL(A_1 X \boxplus_4 C))$$

  where A_1 and A_2 are 4×4 matrices over $\mathbb{Z}_2$, $ROTL$ is a rotation and C is a constant defined in [20].

- The transformation $\nu : \mathbb{Z}_{2^w}^4 \to \mathbb{Z}_{2^w}^4$ is a permutation such that

  $$\nu(Y) = A_4(ROTL(A_3 Y \boxplus_4 C'))$$

  where A_3 and A_4 are 4×4 matrices over $\mathbb{Z}_2$, $ROTL$ is a rotation and C' is a constant defined in [20].

The π-function has two consecutive transformation E_1 and E_2 for each round. In the figure 3.5, the definition of E_1 is

$$E_1 : (\mathbb{Z}_{2^w}^4)^{N+1} \to (\mathbb{Z}_{2^w}^4)^N$$

such that

$$E_1(C_1, I_1, ..., I_N) = (J_1', ..., J_N')$$

where

$$J'_1 = C'' * I_1,$$
$$J'_i = J'_{i-1} * I_i$$

for $i = 2, ..., N$ and C'' is a 4-tuple of w-bit constant defined in [20] while the definition of E_2 is

$$E_2 : (\mathbb{Z}^4_{2^w})^{N+1} \rightarrow (\mathbb{Z}^4_{2^w})^N$$

such that

$$E_2(C_2, J'_1, ..., J'_N) = (J_1, ..., J_N)$$

where

$$J_N = J'_N * C''',$$
$$J_{N-i} = J'_{N-i} * J_{N-i+1}$$

for $i = 1, ..., N-1$ and C''' is a 4-tuple of w-bit constant defined in [20].

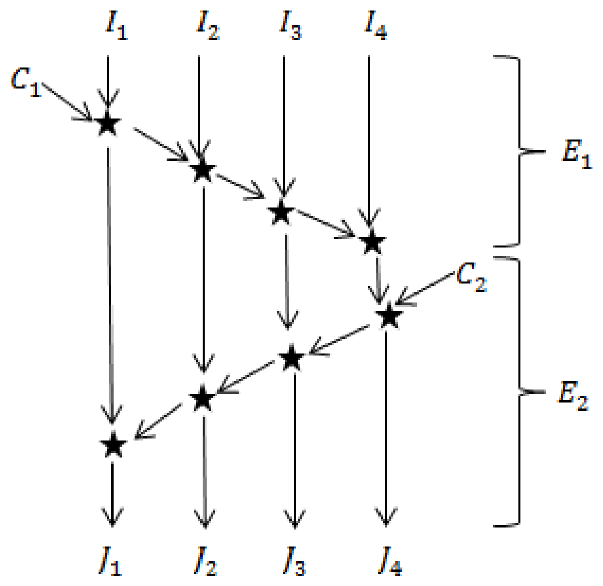

Figure 3.5: Graphical representation of E_1 and E_2 transformations in 1-round π-function

π-function is defined as follows for one round:

$$\pi(I_1, ..., I_N) = E_2(C_2, E_1(C_1, I_1, ..., I_N))$$

where N is taken the value 4 [20]. In version 2, the round number is reduced from 4 to 3. Thus, with this light, the π-function with 3 rounds is defined as

$$\pi(I_1, ..., I_N) = E_2(C_6, E_1(C_5, E_2(C_4, E_1(C_3, E_2(C_2, E_1(C_1, I_1, ..., I_N))))))$$

where C_i's are 4-tuple of w-bit constants for $i = 1, 2, 3, 4, 5, 6$ defined in [20].

3.2 Bit Diffusion Analysis of π-Cipher

The designers of π-Cipher give the bit diffusion analysis of $*$ operation of w=16, 32, 64 and one round π-function for π16-Cipher, π32-Cipher and π64-Cipher. They construct two experimental settings to evaluate the bit diffusion in the analysis. The first one of settings for $*$ operation is that they take 10000 randomly generated right and left inputs of $*$ operation, and then they analyze the propagation of one bit difference for 10000 inputs as follows:

1. Compute $Z = X * Y$ where X and Y are inputs and Z is output of $*$ operation.
2. Evaluate $Z' = X' * Y$ where X' is an input of $*$ operation such that
$$HammingDist(X, X') = 1.$$
3. Measure the Hamming distance between Z and Z'.

They repeat the same process for Y. After doing these, they represent the results in figures for X and Y in the values of $w = 16, 32, 64$ without any conclusion [20]. The second setting for one round π-function is that they take 1000 randomly generated inputs for IS of π-function, and then they examine the bit difference propagation for 1000 inputs as follows:

1. Compute the output of one round π-function of IS.
2. Evaluate the output of one round π-function of IS' where is an input of one bit change in IS.
3. Measure the Hamming distance between $\pi(IS)$ and $\pi(IS')$.

Do this on all of π16-Cipher, π32-Cipher and π64-Cipher. Then they give the results in figures in terms of minimum, average and maximum avalanche effect of one bit difference of π16-Cipher, π32-Cipher and π64-Cipher without any conclusion [20].

3.3 Cryptanalysis of π-Cipher

3.3.1 Cryptanalysis given by Leurent

In the cryptanalysis paper of π-Cipher [27], an extra property of algorithm is supposed to the tag second preimage resistance. The problem of the second preimage tag is a kind of Knapsack problem. The attack done by Leurent is applied with the efficient solution of the generalized birthday attack of Wagner since the word-wise addition, $\boxplus_8$, is used in the π-Cipher. This attack is against the m-sum problem that is found the values $l_1 \in L_1, l_2 \in L_2, \ldots, l_m \in L_m$ where m lists given $L_1, L_2, \ldots, L_m$ of n-bit such that $\oplus_{i=1}^{m} l_i = 0$. Similar to this, π-Cipher algorithm has the generation of tag of m-block message, i.e.,

$$T = T'' \boxplus_8 e(1, M_1) \boxplus_8 e(2, M_2) \boxplus_8 \ldots \boxplus_8 e(m, M_m)$$

where e means e-triplex, $\boxplus_8$ is a component-wise addition of 8-element vectors in $\mathbb{Z}_{2^w}$ and T'' is the associated data tag.

A tag second-preimage attack is done by building a message obtaining a fixed tag $\overline{T}$ with the assumed information of $T'' = 0$ and $\overline{T} = 0$. The obtained results are represented in the table [27].

3.3.2 Cryptanalysis given by Fuhr et al.

Fuhr and Leurent discover the forgeries because of π-Cipher padding process [19]. In the padding process, if the length of last block of plaintext or AD is inadequate, then $10 \ldots 0$ byte string is added to it; however, the padding function is supposed not to be injective. The fact that this causes the forgery attack is not applicable to the principle of nonce-based AEAD scheme. They claim that the forgery attack can be applied in the ciphertext by dropping some ciphertext bytes, and it can be successful with the probability 2^{-8} in the ciphertext only attack scenario.

3.3.3 Cryptanalysis given by Mihajloska et al.

In this work [31], the authors mentioned a new design of mode of operation with intermediate tags for π-Cipher which permits the tag verification of a long message on the device having less memory without giving unverified message. The mode of operation has such properties: parallelism, online encryption, and also nonce misuse resistance in the intermediate level for decryption. This level makes cipher robust with using SMN as a part of nonce. This mode of operation does not make the inverse primitive calls. That allows a single implementation of permutation for E/A and D/V. They give the security proof of privacy and authenticity with intermediate tags with the result of $min\{2^k, 2^c, 2^{b/2}\}$.

3.3.4 Cryptanalysis given by Alley et al.

Alley and Pieprzky show the state recovery attacks in [3] where is up to three rounds. In the bitrate of internal state, the known values used by the state recovery attack plus obtained values from exhaustive search are used for getting the rest in the internal state. With these attacks, one round of any variants of π-Cipher can be broken. Moreover, these attacks including divide-and-conquer attack can break two and three rounds of some variants faster than exhaustive search on the key for the version 1 of π-Cipher. In the version 2.0, the padding rule was changed; however, this attack works for one round of version 2.0 submission with the given distinguisher.

3.3.5 Cryptanalysis given by Boura et al.

The attack [10] is that authors propose a guess and determine attack on some variants of π-Cipher. They show the key recovery against 2.5-round of π-Cipher with time complexity approximately 2^{4w} where w is the word size and low data complexity. For example, the time complexity of attack for π16-Cipher96 with 2.5-round is 2^{72}. They mentioned that the security claims given by designers is very limited. Their attack complexity for 2 full rounds is 2^{72} while the designers claim that the security margin 96 and 128 bits for π16-Cipher96 and π16-Cipher128, respectively. Moreover, according to the first submission paper, it provides 128 and 256 bits of security for π16-Cipher128 and π32-Cipher256 in the 4 rounds; yet, the key recovery attack for 2.5 rounds is 2^{72} and 2^{137}, respectively. They claim that they exploit a weakness in the π-function.

CHAPTER 4

SAC Test Application on π-Cipher

Our contribution is that we improve the bit diffusion analysis of π-Cipher carried out by designers [20] with using more random inputs, the different technique, SAC Test, and the interpretation of the obtained results. In this chapter, we give the details of the application of SAC Test on the parts $*$ operation and π-function of π-Cipher. We use $*$ operation for $w = 16, 32, 64$ and π-function of one, two and three rounds for $\pi 16$-Cipher, reduced round versions of π-function for $w = 16$, to apply our method and get output sets. We explain how SAC Test is applied on $*$ operation and π-function and interpret the obtained results.

We use two ways to apply SAC Test on $*$ operation and π-function:

4.1 SAC Test Application on $*$ operation and its Results

In this section, we mention applying SAC Test to $*$ operation for $w = 16, 32, 64$ and interpreting the achieved results.

First, the diffusion property of the $*$ operation is analyzed since $*$ operation is the main operation of π-Cipher permutation. For this analysis, the below steps are followed.

1. Firstly choose a random Y and fix this value.
2. Choose a random X and compute the output of $*$ operation such that $Z = X * Y$. For each i where $i = 0, ..., n-1$.
3. Generate X_i by flipping the i^{th} bit of X and compute $Z_i = X_i * Y$.
4. Increment the $(i, j)^{th}$ entry of the $n \times n$ SAC matrix if j^{th} bit of $Z \oplus Z_i$ is 1.
5. Repeat the steps 2 to 4.

The procedure is carried out for 2^{20} different values of X. Also, the same steps are repeated for Y with fixed X as well. This way is pursued on $*$ operation for $w = 16, 32, 64$. We apply a χ^2 *Goodness of Fit Test* to the SAC matrix with the subinterval probabilities stated in the section of SAC Test [16].

We observe that the applications of SAC Test on $*$ operation for $w = 16, 32, 64$ for both inputs X and Y give the values of $p < 0.01$. Therefore, $*$ operation is *non-random* for the word sizes $w = 16, 32, 64$. The results for these bits are presented in Table 4.1 and Table 4.2.

Table 4.1: Results of SAC Test for X for $w = 16$

Position of the flipped bit	Avg. number of output bit changes
2	15.26
50	16.24
17	16.35
18	16.40

Table 4.2: Results of SAC Test for Y for $w = 16$

Position of the flipped bit	Avg. number of output bit changes
33	15.86
34	15.86
49	16.18
50	16.37

The fall in some bits such as 2, 17 and 33 shows that the difference in these bits does not spread out less than other points throughout the output 4.1. The results indicate that if there is a difference in the 2^{nd} bit of X, then 15.26 output bits change in average and this may be the starting point of a cryptanalysis.

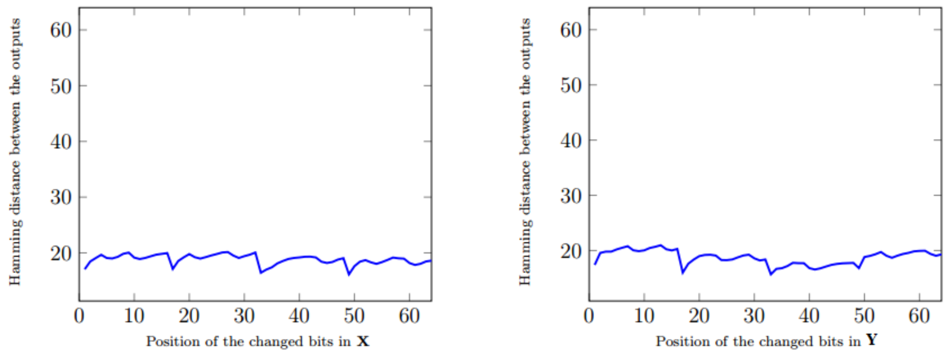

Figure 4.1: Bit diffusion of $*$ operation of input X and Y for $w = 16$ [20].

The only nonlinear part of the $*$ operation is the modular addition: the diffusion layer consists of two simple permutations and one rotation. Experimental result indicates that a single bit difference in the input affects 18 bits on average instead of the expected 32. Furthermore, we know that the difference in the leftmost bits of each modular addition results less number of bit changes in the output due to the differential characteristics of modular addition.

Similar to $*$ operation for $w = 16$, $*$ operation for $w = 32$ and $w = 64$ have the same drastic fall in the figures 4.1, 4.2 and 4.3 given by designers [20] in every 32-bit and 64-bit respectively.

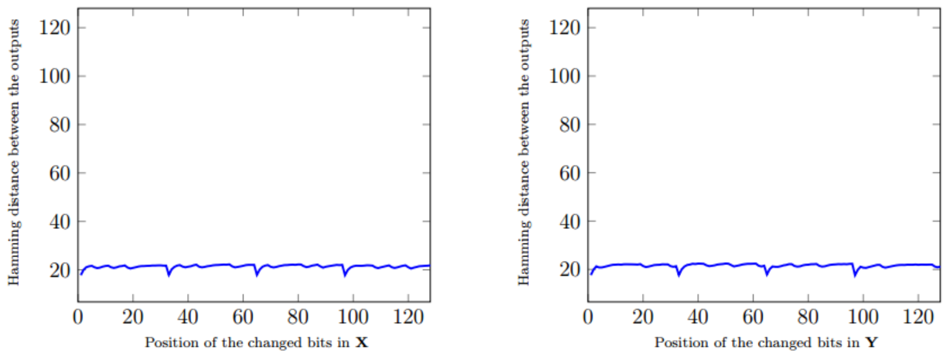

Figure 4.2: Bit diffusion of $*$ operation of input X and Y for $w = 32$ [20].

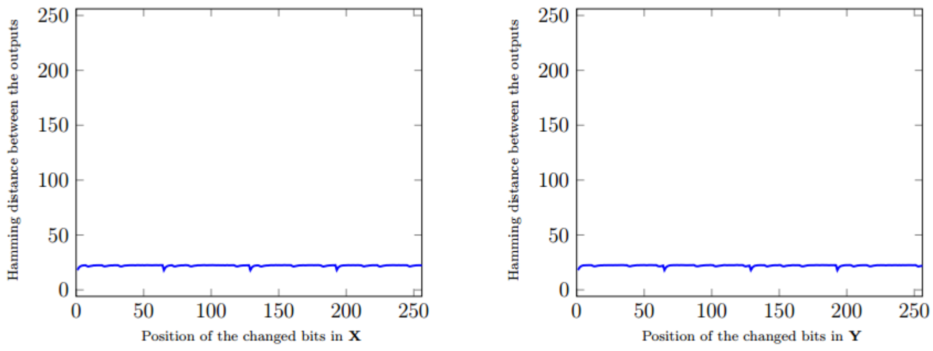

Figure 4.3: Bit diffusion of $*$ operation of input X and Y for $w = 64$ [20].

4.2 SAC Test Application on π-function and its Results

The second analysis method of the cipher using SAC Test is the statistical examination on the bit diffusion analysis of π-function, the internal state (IS). Similar to the first method, the procedure of first step is to take an input and compute its output, also the second step is to observe the examination the XOR of output corresponding the input which has the only one bit difference and the original output. To put in mathematical terms:

1. Take an input I in $(\mathbb{Z}_{2^w}^4)^N$ such that $\pi(I_0, \ldots, I_{b-1}) = J$ where $b = N \times 4 \times w$ and $N = 4$, so $b = 4 \times 4 \times w$.

2. Change the input bit I_i for $i = 0, \ldots, b-1$ and compute the corresponding output, i.e., $\pi(\neg I_0, \ldots, I_{b-1}) = J^1, \ldots, \pi(I_0, \ldots, \neg I_{b-1}) = J^b$.
3. XOR the corresponding output of one bit change input and the original output, i.e., $J^i \oplus J$ for $i = 1, \ldots, b$.
4. Write the result of XOR into i^{th} row of the $b \times b$ SAC matrix.

The procedure is performed for only one input, so we need to try it for 2^{20} different inputs due to the more random or non-random evaluation of diffusion of π-function from 1 to 3 round of π16-Cipher.

Table 4.3: SAC Test for π-function for $w = 16$

SAC Test results for π16-Cipher			
	1 round	2 rounds	3 rounds
p-value	0.969954	0.429349	0.774130

We apply SAC Test for the reduced round versions of π-function for π16-Cipher. We observe that all p-values obtained from the test are 0.969954, 0.429349 and 0.77413 for 1, 2 and 3 rounds of π-function, respectively. Since all p-values are greater than 0.01, we conclude that all versions of π-function for π16-Cipher are *random*.

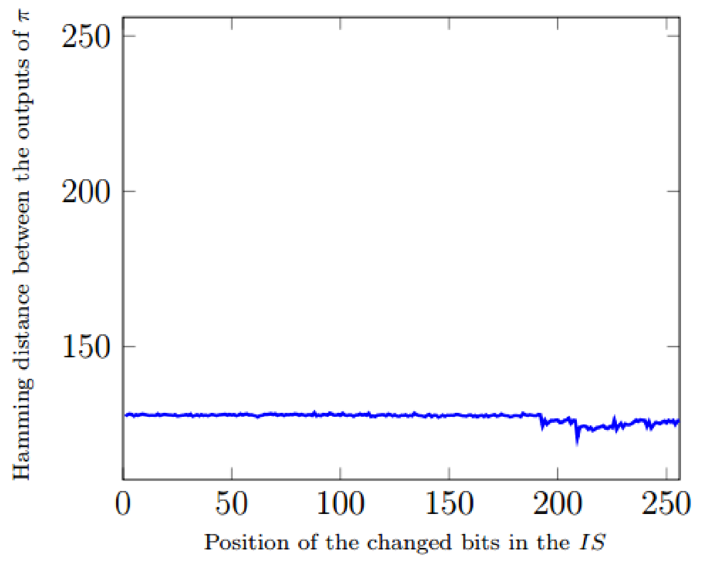

Figure 4.4: Bit diffusion of π-function for $w = 16$ in [20]

There are drastic falls in the last bits of one round π16-function in the figure 4.4 since π-function has two consecutive transformations E_1 and E_2 in the figure 3.5. Transformations E_1 and E_2 consist of $*$ operation which is ARX. The right part of the input is subjected to five $*$ operations whereas the left part of input is processed with eight $*$ operations as it can be seen from figure 3.5. That is, the differences in the last bits of

IS for π-function deal with less $*$ operations, and so there are certain decline at these output bits of π-function. Despite all, $\pi16$-function is random for all reduced rounds.

CHAPTER 5

Conclusion

In this thesis, we study on the topic of Authenticated Encryption algorithms in the CAESAR Competition and the statistical randomness testing, especially cryptographic randomness testing designed by Doğanaksoy et al [16]. We choose π-Cipher that is one of the second round algorithms in the CAESAR Competition, and then we study on it. We catch some points of algorithm which will have been improved. π-Cipher has a permutation π-function consisting of $*$ operation which is ARX operation. In the π-function, there is two consecutive transformations E_1 and E_2. These transformations use the $*$ operation with the defined constants in [20]. Firstly, we examine the working principles of $*$ operation and π-function. Then, we analyze the method of bit diffusion analysis of $*$ operation and π-function given by designers [20]. They gave the method for X and Y parts of $*$ operation with using all w word sizes and 10000 random inputs. The results of this method were shown only with the graphics for all w variants. They did not give any conclusion about what the graphics indicate or what the given results mean. In the same way, they gave the method for one round of π-function for all w variants with 1000 random inputs. The results were shown only with the graphics; yet, they gave them without any conclusion. Because of this, we decide to improve the methods given by designers and represent what the obtained results are. Therefore, we apply Strict Avalanche Criterion (SAC) Test to all versions for $*$ operation and $\pi 16$-Cipher for reduced round versions of π-function where SAC Test is one of the cryptographic randomness test proposed in the recent test package designed by Doğanaksoy et al. [16], and SAC test determines the number of rounds that algorithm behaves like a random mapping by analyzing the relation between inputs and outputs. With this method, it is aimed to get a single p-value related with the data set under consideration through a large set of p-values produced by SAC test. Finally, we analyze the parts of π-Cipher, $*$ operation and π-function, and apply the SAC Test on these parts. We improve the evaluation methods of bit diffusion analysis for $*$ operation and π-function given by the designers of the algorithm. According to the corresponding test results given in the Table 4.1, 4.2, 4.3 in Section 4, we determine that the algorithm behaves random up to how many rounds, and explain the diffusion of $*$ operation and π-function in further details. We explain that why $*$ operation has a non-random behavior, and show the results of random π-function for all reduced round versions according to the cryptographic randomness test, SAC Test.

REFERENCES

[1] AES, Advanced Encryption Standard, 1997-2000.

[2] P. Alcover, A. Guillamon, and M. Ruiz, New Randomness Test for Bit Sequences, in *Informatica*, pp. 339–356, 2013.

[3] J. Alley and J. Pieprzyk, *State Recovery Attacks against π-Cipher*, ACSW 16 Proceedings of the Australasian Computer Science Week Multiconference, February 1, 2016.

[4] E. Andeereva, *Analysis and Design of Authenticated Encryption Modes, MS. Thesis Supervisor Prof. Dr. Jörg Siekmann and Dipl.-Inform. Ammar Alkassar*, University of Saarland, Saarbrücken, Germany, August 29, 2005.

[5] L. E. Bassham, III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks, N. A. Heckert, J. F. Dray, and S. Vo, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Technical report, NIST, Gaithersburg, MD, United States, 2010.

[6] M. Bellare and C. Namprempre, *Authenticated Encryption: Relations among Notions and Analysis of The Generic Composition Paradigm*, IACR report, https://eprint.iacr.org/2000/025, July 14, 2007.

[7] D. J. Bernstein, CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, 2014.

[8] D. J. Bernstein, Google groups: cryptographic competitions, 2014.

[9] G. Bertoni, J. Daemen, J. M. Peeters, and G. V. Assche, *Keccak*, NIST, 2015.

[10] C. Boura, A. Chakraborti, G. Leurent, G. Paul, D. Saha, H. Soleimany, and V. Suder, *Key Recovery Attack Against 2.5-round π-Cipher*, FSE 2016 accepted paper, IACR report 2016/502, May 23, 2016.

[11] CRYPTREC, Cryptography Research and Evaluation Committees, 2000-2003.

[12] J. Daemen and V. Rijmen, The design of rijndael: Aes - the advanced encryption standard, 2001.

[13] W. Diffie and M. E. Hellman, *Diffie-Hellman Key Exchange*, volume IT-22, NO. 6, IEEE Transactions on Information Theory, MIT, Cambridge, Mass., November, 1976.

[14] A. Doğanaksoy, Ç. Çalık, F. Sulak, and M. Sönmez Turan, New Randomness Tests Using Random Walk, in *In 2nd National Conference Proceedings*, Ankara, Turkey, December, 15-17 2006.

[15] A. Doğanaksoy, F. Sulak, M. Uğuz, O. Şeker, and Z. Akcengiz, New Randomness Tests Based on Lenght of Runs, in *Mathematical Problems in Engineering*, March, 2015.

[16] A. Doğanaksoy, B. Ege, O. Koçak, and F. Sulak, Cryptographic Randomness Testing of Block Ciphers and Hash Functions, IACR Cryptology ePrint Archive, Report 2010/564, 2010, `http://eprint.iacr.org/2010/564`.

[17] A. Doğanaksoy, B. Ege, O. Koçak, and F. Sulak, Statistical Analysis of Reduced Round Compression Functions of SHA-3 Second Round Candidates, IACR Cryptology ePrint Archive, Report 2010/611, 2010, `https://eprint.iacr.org/2010/611.pdf`.

[18] eSTREAM, the ECRYPT Stream Cipher Project, 2004-2008.

[19] T. Fuhr and G. Leurent, *Observation on π-Cipher*, CAESAR competition mailing list, November, 2014.

[20] D. Gligoroski, H. Mihajloska, S. Samardjiska, H. Jacobsen, M. El-Hadedy, R. E. Jensen, and D. Otte, π-cipher v2.0, submission to the caesar competition, August 29, 2015.

[21] IBM Team, *Data Encryption Standard (DES)*, Federal Register, 1975.

[22] D. Kahn, *The Codebreakers: ,the Story of Secret Writing*, New York: Macmillan, 1967.

[23] V. Katos, A Randomness Test for Block Ciphers, in *Applied Mathematics Computation*, volume Volume 1, pp. 29–35, March, 2005.

[24] L. R. Knudsen and R. M. J.B., *The Block Cipher Companion*, Springer Heidelberg Dordrecht London New York, Boca Raton, FL, USA, 1st edition, 2011, ISBN 978-3-642-17341-7.

[25] D. E. Knuth, The Art of Computer Programming, in *Seminumerical Algorithms*, volume Volume 2 (3rd Ed.), Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997.

[26] P. L'Ecuyer and R. Simard, Testu01: A C Library for Empirical Testing of Random Number Generators, in *ACM Transactions on Mathematical Software*, volume 33, 2007.

[27] G. Leurent, Tag Second-preimage Attack against π-cipher, hal-00966794v2, 2014, `http://eprint.iacr.org/2010/564`.

[28] J. Massey and X. Lai, *International Data Encryption Algorithm (IDEA)*, Hasler Foundation, Ascom-Tech AG, 1991.

[29] U. Maurer, A Universal Statistical Test for Random Bit Generators, in *Journal of Cryptology*, volume 5, Issue 2, p. 89–105, Springer, January 1992.

[30] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 1996, ISBN 0849385237.

[31] H. Mihajloska, B. Mennink, and D. Gligoroski, *π-Cipher with Intermediate Tags*, pi-cipher.org, May 13, 2016.

[32] NESSIE, New European Schemes for Signatures, Integrity and Encryption, 2000-2003.

[33] NIST, US National Institute of Standards and Technology, 1901.

[34] C. Paar and J. Pelzl, *Understanding Cryptography*, Springer Heidelberg Dordrecht London New York, Inc., Boca Raton, FL, USA, 1st edition, 2010, ISBN 9783642041006.

[35] PHC, Password Hashing Competition, 2013-2015.

[36] R. Rivest, *Rivest Cipher (RC5)*, Proceedings of the Second International Workshop on Fast Software Encryption (FSE), 1994.

[37] R. L. Rivest, A. Shamir, and L. M. Adleman, *RSA*, PKCS# 1, ANSI X9.31, IEEE 1363, MIT, Cambridge, Mass., December 14, 1977.

[38] P. Rogaway, Authenticated Encryption with Associated Data, in ACM Conference on Computer and Communications Security, pp. 196–205, 2002.

[39] A. L. Ruhkin, Testing Randomness: A Suite of Statistical Procedures, in *Theory of Probability & Its Applications*, volume 1, pp. 111–132, 2001.

[40] SHA-3, The Third Secure Hash Algorithm Competition, 2007-2012.

[41] C. E. Shannon, *Communication Theory of Secrecy Systems*, volume 28(4), Bell System Technical Journal, New York, 1st edition, 1949, ISBN 0849385237.

[42] M. Sönmez Turan, *On Statistical Analysis of Synchronous Stream Ciphers, PhD Thesis Supervisor Assoc. Prof. Dr. Ali Doğanaksoy*, METU, Ankara, April, 2008.

[43] M. Sönmez Turan, Ç. Çalık, N. B. Saran, and A. Doğanaksoy, New Distinguishers Based on Random Mappings against Stream Ciphers, in S. W. Golomb, M. G. Parker, A. Pott, and A. Winterhof, editors, *Sequences and Their Applications SETA 2008*, volume 5203 of *Lecture Notes in Computer Science*, pp. 30–41, Springer Berlin Heidelberg, 2008, ISBN 9783540859116, `http://dx.doi.org/10.1007/978-3-540-85912-3_3`.

[44] J. Soto, Randomness Testing of the Advanced Encryption Standard Candidate Algorithms, Technical report, NIST, Gaithersburg, MD, United States, September 1999.

[45] J. Soto and L. Bassham, Randomness Testing of the Advanced Encryption Standard Finalist Candidates, in *NIST IR 6483, National Institute of Standards and Technology*, March 28, 2000.

[46] D. R. Stinson, *Cryptography Theory and Practice*, Chapman and Hall/CRC Taylor & Francis Group, Inc., Boca Raton, FL, USA, 3rd edition, 2006, ISBN 1584885084.

[47] F. Sulak, A New Statistical Randomness Test: Saturation Point Test, in *International Journal of Information Security Science*, volume 2, pp. 81–85, 2013.

[48] F. Sulak, New Statistical Randomness Tests: 4-bit Template Matching Tests, in *Turkish Journal of Mathematics*, 2016.

[49] F. Sulak, *Statistical Analysis of Block Ciphers and Hash Functions, PhD Thesis Supervisor Assoc. Prof. Dr. Ali Doğanaksoy*, METU, Ankara, February, 2011.

[50] F. Sulak, B. Ege, and O. Koçak, Statistical Analysis of Reduced Round Compression Functions of SHA-3 Finalists, in *International Journal of Research and Reviews in Applied Sciences*, volume 15, 2013.

[51] A. F. Webster and S. E. Tavares, On the Design of S-Boxes, in *Lecture notes in computer sciences; 218 on Advances in Cryptology-CRYPTO 85*, pp. 523–534, Springer-Verlag New York, Inc., 1986.

APPENDIX A

The bin values of $*$ operation for $w = 16, 32, 64$ and π-function for $w = 16$

Table A.1: SAC Test ranges and probabilities for $*$ operation of $w = 16$

Bin	Range	Probability
1	0-26	0.0843215
2	27-28	0.106545
3	29-30	0.163124
4	31-31	0.093362
5	32-32	0.0993468
6	33-33	0.0963362
7	34-34	0.087836
8	35-35	0.075288
9	36-37	0.106545
10	38-64	0.0843215

Table A.2: SAC Test ranges and probabilities for $*$ operation of $w = 32$

Bin	Range	Probability
1	0-55	0.0663124
2	56-59	0.146907
3	60-61	0.116131
4	62-62	0.0661531
5	63-63	0.0693032
6	64-65	0.139689
7	66-67	0.127369
8	68-69	0.102666
9	70-72	0.146907
10	73-128	0.0923413

Table A.3: SAC Test ranges and probabilities for $*$ operation of $w = 64$ and π-function of $w = 16$

Bin	Range	Probability
1	0-112	0.0262358
2	113-119	0.117759
3	120-122	0.101927
4	123-125	0.131444
5	126-128	0.147544
6	129-130	0.0977251
7	131-131	0.0464489
8	132-133	0.084995
9	134-138	0.151298
10	139-256	0.0946239