Bachelor's Thesis Computing Science

Radboud University Nijmegen

Avalanche Criteria Generalized To The Permutation Atrapos

Author:
Ayça Önal
s1114006

First supervisor/assessor:
Joan Daemen

Second assessor:
Silvia Mella

March 24, 2026

Abstract

This paper investigates avalanche effect on the permutation proposal Atrapos that operates on digits of a large finite field. To our knowledge, previously avalanche tests have not been evaluated in a non-binary domain. Evaluation is done specifically by the avalanche entropy (Hav) that measures the uncertainty of output digit changes from changing a single digit in the input. Atrapos permutation operates in the domain $\mathbb{F}_p$ with some prime number p. It is intended to be used in post-quantum cryptographic schemes such as CRYSTALS-Dilithium where $p = 8380417$ and CRYSTALS-Kyber where $p = 3329$. The study shows propagation at the bit and byte levels of the Advanced Encryption Standard (AES) by introducing input differences in a 128-bit input. Observing the established avalanche behavior of AES provides a baseline on analyzing the behaviour of Atrapos. This research suggests values for undecided parameters in Atrapos, assessed how closely the observed output behavior aligns with strict avalanche criteria.

Keywords: permutation-based cryptography · Atrapos·avalanche criteria·

Contents

Chapter 1

Introduction

Modern cryptographic primitives are constructed from permutations to ensure randomization of input by repeatedly applying a round function. A permutation is a bijective mapping on a state consisting of discrete elements. Elements are digits from the finite field $\mathbb{F}_q$ where q can be either a prime number or a power of a prime number. We look at the cases with $q = 2$, $q = 2^e$ for some integer $e > 1$ and $q = p$ where p is a prime number greater than 2.

Given a state over $\mathbb{F}_q^b$, the round function is an iterative sequence of transformations on a state with width b [10]. A single round consists of non-linear, shuffle, and linear mixing steps. Non-linearity is obtained from substitution of elements through usage of S-boxes. The shuffling state rearranges the positions of elements in the state. Finally, the linear mixing step combines elements of the state using linear operations over the given finite field. Combination of these three steps ensure that small input differences propagate rapidly across the entire state. The specifications of each step will be introduced in Chapter 3, in perspective of 128-bit permutation named Advanced Encryption Standard.

In permutation-based cryptography, the unknown key is absent from the internal application of rounds [5][4]. Therefore, the ideal behavior of unkeyed permutations can't be defined [4]. To introduce secrecy, a key is injected externally to the state before the round function. Well-known cryptographic permutations are *KECCAK-p* standardized by NIST as part of SHA-3, Xoodoo, and Ascon-p [**?**][11][6]. More information can be found on Related Work Section. A primary example of described permutation usage is the Sponge construction introduced in Preliminaries 2.3.

1.1 Security Objectives

The aim for iterated permutations is to achieve dependency of each output digit to every input digit in a complicated way. The relation is tested based on Strict Avalanche Criterion (SAC), introduced in Chapter 5.2. Assessing the degree of vulnerability for binary permutations begins with finding required number of rounds to satisfy the criteria SAC. Then, extended on a deeper level by using the avalanche entropy to measure the uncertainty of the change on output elements. However, avalanche tests are always conducted on primitives operating on bits as far as we know. The permutation Atrapos operates on states consists of digits in a larger finite field $\mathbb{F}_p$ with prime number p. The design intent is usability in post-quantum protocols such as CRYSTALS-Kyber in $\mathbb{F}_{3329}$ and CRYSTALS-Dilithium in $\mathbb{F}_{8380417}$. To evaluate such permutations, analyses must be conducted over non-binary fields. Due to it's design properties and intended use in post-quantum protocols, Atrapos becomes a suitable candidate for the research aim.

1.2 Research Intent

Main purpose in this research, is to have meaningful generalization of fine-grained avalanche tests for permutations that operate on elements from finite fields other than $\mathbb{F}_2$. The avalanche probability array $P_{\Delta F}$ contains, the probabilities of each output digit to take every possible value in the finite field $\mathbb{F}_p$ by the application of the permutation $\mathbb{F}$ on some input difference Δ. Since the target cryptographic primitive $\mathbb{F}$ operates on a non-binary field, the existing algorithm to compute $P_{\Delta F}$ requires modifications.

This research suggests the optimal value selection for undetermined parameters in Atrapos among all possible combinations. Suggestions are made based on the results provided by the H_{av} analysis.

Main contributions to already existing knowledge in this thesis are:

1. Generalization $P_{\Delta F}$ into a two-dimensional array containing a distribution for each output digit.

2. Analyzing effectiveness of our modifications on output digits using Avalanche Entropy (H_{av}).

3. Comparison of the outcome of bit-avalanche tests and byte-avalanche tests for AES.

4. Suggestions of Atrapos parameters based on conducted the Avalanche tests.

Chapter 2

Preliminaries

The foundation of permutations and cryptographic primitives are based on polynomial operations such as addition and multiplication. For different finite fields said operations perform differently. To better understand the analyses in this research, operations are defined elements from finite fields $\mathbb{F}_2$, $\mathbb{F}_{2^8}$, and $\mathbb{F}_p$ with a prime number p.

2.1 Polynomial Operations on Different Finite Fields

In the design of the Rijndael cipher, addition and multiplication on finite fields is defined through polynomial representations [9]. According to the specific finite field, operations differ. Elements of a finite field $\mathbb{F}_{p^e}$ where p is any prime number and e is indicating the power of p. The highest exponent that appears in a polynomial is called its degree. We can express $\mathbb{F}$ with the equation $b(x)$ where $b_i \in \mathbb{F}$ represents the coefficients of x [9].

$$b(x) = b_{n-1}.x^{n-1} + b_{n-2}.x^{n-2} + b_2x^2 + b_1x + b_0$$

Bits work on the finite field $\mathbb{F}_2$ where coefficients b_i can only be 0 or 1. A single byte can be represented by 8 bits which is a polynomial with 8 coefficients. We can store them as an 8-bit number [9]:

$$b(x) = b_7b_6b_5b_4b_3b_2b_1b_0$$

Each b_i represents whether the corresponding power of x appears in the polynomial. An example transformation on bytes is having the polynomial $x^6 + x^4 + x^2 + x + 1$ as the bit string 01010111.

2.1.1 Addition on Finite Field $\mathbb{F}_{2^e}$

Addition on the finite field $\mathbb{F}_2$ defines bit addition. Adding two bits is done by applying XOR to the bit sequence on modulo 2. In modulo 2 arithmetic,

addition follows the rules $1 + 1 \equiv 0 \pmod 2$, $1 + 0 \equiv 1 \pmod 2$, $0 + 1 \equiv 1 \pmod 2$, and $0 + 0 \equiv 0 \pmod 2$.

Addition on the finite field $\mathbb{F}_{2^8}$ defines byte level addition that can be done by applying bitwise XOR on polynomials with coefficients in $\mathbb{F}_2$. For example, adding the bytes 01010111 and 10000011 results in 0101110110.

2.1.2 Addition on Finite Field $\mathbb{F}_p$

On permutation using modular arithmetic with finite field $\mathbb{F}_p$ where p is a prime number larger than 2, addition takes $(a + b) \bmod p$. As an example we can assign $p = 5$.

$+_5$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Table 2.1: Addition table over $\mathbb{Z}_5$

2.1.3 Multiplication on Finite Field $\mathbb{F}_{2^e}$

Multiplication on the finite field $\mathbb{F}_2$ corresponds to bit level multiplication by Boolean AND operation. This means the result is 1 only if both input bits are 1; otherwise, the result is 0.

Multiplication on the finite field $\mathbb{F}_{2^8}$ corresponds to byte multiplication by algebraic product of corresponding polynomials. Then reducing the result modulo with a fixed irreducible polynomial [9]. The polynomial used for reduction in this research is

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Each polynomial with a degree smaller than that of $m(x)$ is the co-prime polynomial of $m(x)$. Therefore, multiplicative inverse modulo of $m(x)$ can be computed with the extended Euclidean algorithm. This property is used for the S-box construction in AES.

2.1.4 Multiplication on Finite Field $\mathbb{F}_p$

In finite fields with prime order of p, every non-zero element must have a unique multiplicative inverse under multiplication modulo p. Going back to

example $p = 5$, we will have the table:

x_5	0	1	2	3	4
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

2.2 Sponge

The Sponge construction builds an eXtendable- Output Function (XOF) from a permutation and a padding rule. An eXtendable-Output Function (XOF) is a cryptographic primitive that maps a variable-length input to a variable-length output [11]. SHAKE128 and SHAKE256 are XOFs built on the sponge construction which absorb input data into a fixed-size state and squeeze out a variable-length output.[11]. Relevant parameters to understand the sponge structure are:

- Amount of bits absorbed/squeezed per permutation, rate r
- Capacity c
- Permutation width b that can be computed as $b = c + r$

In the absorbing phase, variable-length input data is injected into the permutation's state and the squeezing phase extracts pseudorandom output bits. The security against collision approximately takes $2^{c/2}$ number of operations determined by birthday bound on binary fields [8]. Capacity $c \geq 256$ achieves minimum 128-bit security against collisions, which is desired in Advanced Encryption Standard introduced in Chapter 3.

A popular example of this design is Keccak which utilizes the sponge construction to create a secure hash function on Keccak-f permutations [6].

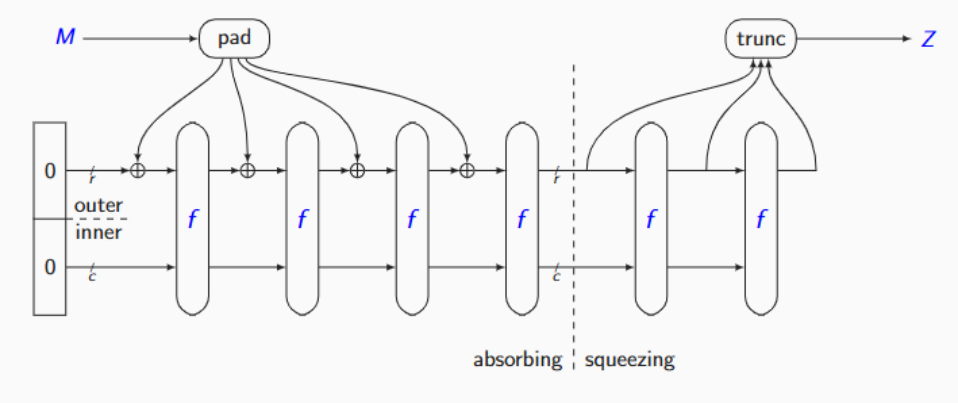

Figure 2.1: Sponge Structure

In permutations operating on digits, b is defined by row length ℓ as $b = 3\ell$. The rate on digits and capacity can be calculated by ℓ and 2ℓ respectively. The formula for collision resistance bound is computed based on the prime order p as $p^{c/2}$.

2.3 ML-KEM Scheme

Key Encapsulation Mechanisms (KEMs) are fundamental public-key cryptographic primitives used to establish a shared secret over an insecure public channel [2][11]. ML-KEM (Module-Lattice-Based Key Encapsulation Mechanism), formerly known as CRYSTALS-Kyber and standardized in FIPS 203, is a post-quantum key encapsulation mechanism based on module-lattice problems [2][11]. Keccak-based functions are essential components for the ML-KEM.

Chapter 3

Advanced Encryption Standard

Block ciphers are an encryption mechanism defined over blocks of length n. They take a n-bit input and use a secret key to provide a n-bit output [4]. The goal of block ciphers is to be hard to distinguish from a random n-bit permutation by an adversary that is unaware of the key. Block ciphers are frequently designed as iterated ciphers. The core principle is to achieve a high level of randomization through the repeated application of a round function. In each round, the round function is combined with a unique round key derived from the main secret key [3][10].

Best known example of block ciphers is the Advanced Encryption Standard (AES) that can be represented on either bit or byte level. On bit level the operation domain is $(\mathbb{F}_2)^{128}$, whereas on byte level it's $(\mathbb{F}_{2^8})^{16}$. For both representations the desired collision resistance is 128 bits determined by the collision resistance bound formula introduced in Chapter 2.2

Advanced Encryption Standard (AES) Structure

The Advanced Encryption Standard (AES) is a subset of the submission Rijndael developed by Joan Daemen and Vincent Rijmen that got established by NIST in 2001 [4][11]. AES has fixed block size of 128 bits and can have different sized keys such as 128, 192, 256 bits. With block size of 128 bit, operation of AES takes place on 16 bytes that are arranged into a 4 x 4 matrix, called the *state*.

AES applies a round function repeatedly to transform the internal state where each round consists of multiple steps. Before applying the round function, the original key is expanded using the Rijndael key schedule to produce a sequence of round keys [9]. Then, adding the round keys to all bytes of the state by bitwise XOR, before starting round operations. All

rounds except the last round has the sequence of operations

$$SubBytes \rightarrow ShiftRows \rightarrow MixColumns \rightarrow AddRoundKey \quad (3.1)$$

Operational overview:

- SubBytes replaces each byte with another non-linearly according to a lookup table, S-box.
- ShiftRows, shifts the last three rows of the state cyclically.
- MixColumns, mixes columns such that a change in one byte affects all bytes in the column by combining the four bytes in each column.
- AddRoundKey, adding the round key to all bytes of the state by bit-wise XOR.

On the final round, operation order is the same except MixColumns step is skipped [11]. Before doing any experiments on AES, a more detailed understanding of it's internal work is necessary.

3.1 SubBytes

SubBytes step is the only non-linear transformation of the round function. Every byte in the grid is swapped for a completely different byte using a fixed "lookup table" called an S-box as seen on Figure 3.1 [9][12]. The goal with non-linear transformation is ensure minimal correlation between input and output bits/bytes. The purpose of S-boxes is to create algebraical structure with no fixed points or opposite fixed points.

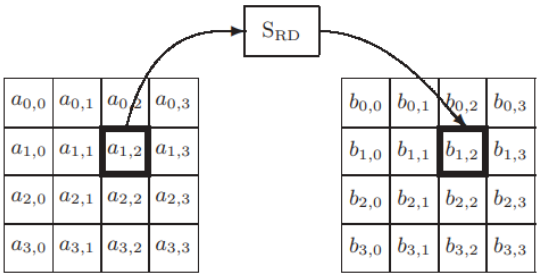

Figure 3.1: SubBytes

3.2 ShiftRows

ShiftRows is a linear step that cyclically shifts the rows of the state over different offsets. For AES on 128-bit block length, each row is shifted by

different amounts. The first row stays put, while the second, third, and fourth rows are shifted to the left by offsets of 1, 2, and 3 bytes, respectively [7][9]. This ensures that bytes the bytes in same column are now scattered into different columns as visualized by Figure 3.2. When combine with SubBytes step, the cipher achieves both non-linear confusion and byte-level diffusion simultaneously.

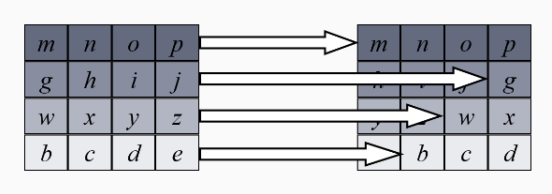

Figure 3.2: ShiftRows

3.3 MixColumns

MixColumns is a linear mixing operation within the Rijndael round transformation that provides diffusion by treating each 4-byte column of the state as a polynomial over $\mathbb{F}_{2^8}$.

Each column is multiplied modulo x^4+1 by a fixed, invertible polynomial c(x) that is co-prime with the modulo, defined by $c(x)$ [9]

$$c(x) = 03.x^3 + 01.x^2 + 01.x + 02 \tag{3.2}$$

This can be seen as matrix multiplication as explain in Section 2.2. Let $b(x) = c(x) \cdot a(x) \pmod{x^4+1}$. Then we can represent as

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

The specific coefficients (02,03,01,01) in $c(x)$ were selected to maximize the branch number to five, which is the mathematical upper limit for a transformation of these dimensions [9]. By mixing the bytes within each column a change in single input byte is guaranteed to affect all four bytes of the output column. The combination of SubBytes, ShiftRows and MixColumns ensures nonlinearity, permutation and diffusion.

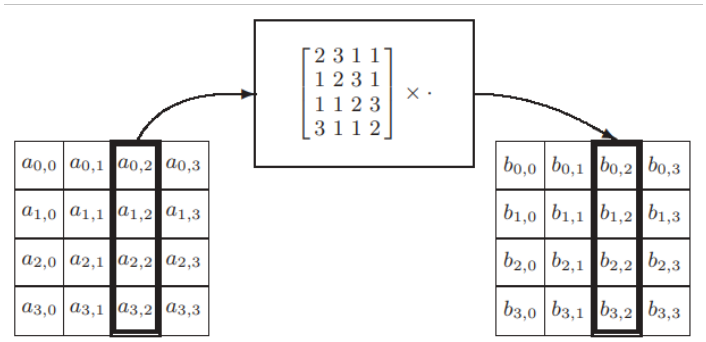

Figure 3.3: MixColumns

3.4 AddRoundKey

AddRoundKey is the step where the round keys derivated from the secret key, are combined with the state by bitwise XOR operation [9][7]. Each round key has the same length as the block length. This thesis takes the secret key as "0000000000000000". Hence, AddRoundKey loses it's significance because XOR operation with 0's gives the state without any change.

Chapter 4

Atrapos

In this section, we introduce the permutation proposal Atrapos, designed to operate on elements over the finite field for different post-quantum dimensions. We describe the structure of Atrapos and the underlying round function.

4.1 Atrapos Proposal

Atrapos is a permutation proposal operating on digits on a state a in the domain $\mathbb{F}_p$ with a prime number p. The state can be structured as a 2-dimensional array indexed by x and y to indicate positions as (x, y). An element on the state position can be denoted in 2-dimension as $a_{x,y}$ or in 1-dimension with the digit index i as a_i. Terms on the x-coordinate are evaluated modulo ℓ, and those on the y-coordinate with the number of rows modulo 3.

4.1.1 Atrapos in 2-Dimensional Space

The round function used by Atrapos in 2-dimension consists of 4 steps, where each step modifies at least one element $a_{x,y}$. Additional parameters are the number of rounds j, the round Constant c_j and shift offset denoted as an array of size 3, $r[3]$. The four steps are defined as

$$\theta : \forall x, y : a_{x,y} \leftarrow a_{x,y} + a_{x+1,y+1} + a_{x+4,y+4} + a_{x+5,y+5}$$
$$\rho : \forall x, y : a_{x,y} \leftarrow a_{x+r_y,y}$$
$$\iota : \forall x, y : a_{0,0} \leftarrow a_{0,0} + c_j$$
$$\gamma : \forall x : a_{x,0} \leftarrow a_{x,0} + a_{x,1}.a_{x,2}$$

The desired capacity c is between 320 bits and 384 bits. The lower bound is determined based on providing 128 bits of generic collision resistance

taking account of quantum computers. The upper bound is a conservative choice. Row length ℓ should be a prime number to prevent symmetries. The values for parameters r_1 and r_2 are yet to be determined, while r_0 is always fixed to 0.

4.1.2 Atrapos in 1-Dimensional Space

Throughout this research, analyzes are conducted on 1-dimensional representation of Atrapos for simplicity. Therefore, the steps of round function are redefined. The size of 1-dimensional array is equal to permutation width b. Current structure of Atrapos has 4 steps but our employment consists of three steps by skipping step ι.

The step θ is modified accordingly as

$$\theta : \forall i : a_i \leftarrow a_i + a_{i+1} + a_{i+4} + a_{i+5}$$

For the steps ρ and γ, first a new state $a_{x,y}$ is generated, and then each element position is realigned to a one-dimensional array. Two approaches can be used to convert the positions from two-dimensional space to one-dimensional representation.

1. if ℓ mod 3 = 1 : i = x + ((y - x) mod 3) . ℓ
2. if ℓ mod 3 = 2 : i = x + ((x - y) mod 3) . ℓ

A third method is unnecessary to define because of two main reasons. First, prime numbers greater than 3 can be assigned to ℓ. Second, only prime number that can satisfy $\ell \mod 3 = 0$ is 3.

4.2 Atrapos Algorithm

Algorithm 1 to1D

Parameters: Positions of x and y in 2-dimensional array
Output: Position index for 1-dimensional array
if $\ell \mod 3 = 1$ **then**
    index = x + ((y - x) mod 3) . ℓ
else if $\ell \mod 3 = 2$ **then**
    index = x + ((x - y) mod 3) . ℓ
end if
return index

We rewrite the round function to operate on 1-dimensional state of b digits as shown in Algortihm 2 by using Algorithm 1. On step ρ, row shuffling phase requires shifting constants per row. As mentioned previously,

first value r_0 is fixed to 0. Based on the outcome from avalanche tests, values for row parameters r_1 and r_2 will be suggested.

Algorithm 2 Round Function F using a state

Parameters: A state of b digits as a 1D array $A[0 \dots b-1]$, shift offsets r as an array with size corresponding to the number of rows
Output: Permuted state on modulo p
Initialize a temporary b-bit vector B to all zeroes ▷ B is used for step θ
Initialize a temporary b-bit vector C to all zeroes ▷ C is used for step ρ and step γ
Initialize fixed sized array r to store shift constants
$r_0 \leftarrow 0$
for all state bit positions i **do** ▷ Step θ
    $B[i] \leftarrow A[i] + A[i+1] + A[i+4] + A[i+5]$
end for
for all state bit positions i **do** ▷ Step ρ
    $x \leftarrow i \bmod \ell$
    $y \leftarrow i \bmod 3$
    Compute new $index \leftarrow$ to1D(x + r[y], y)
    $C[index] \leftarrow B[i]$
end for
for all state bit positions i **do** ▷ Step γ
    if index is in first row when turned to 2D **then**
        $row_0 \leftarrow to1D(i, 0)$
        $row_1 \leftarrow to1D(i, 1)$
        $row_2 \leftarrow to1D(2, p)$
        $C[i] \leftarrow row_0 + row_1.row_2$
    end if
end for
return C

Each step in Algorithm 2 relies on values derived from the previous step, making temporary b-bit vectors B and C essential. Vector B plays a role in step θ for the calculation of new bit positions based on the values from previous positions. Without vector B some new bit positions of new $A[i+1]$ will overwrite previously stored values required for computation of other bit positions.

Column shifting process on step ρ requires a temporary vector, namely Vector C in Algorithm 2. It prevents new column values to overwrite previous values essential for the computation of other columns. As an example, let's say column 1 will shift to new column position 4. The values in old column position 4 needs to be in the position of new column 7, based on Algorithm 1. However, without vector C, column position 4 will be overwritten by the values of newly computed column 1 and previous stored values of column 4 will disappear.

4.3 Atrapos Post-Quantum Dimensions

As mentioned earlier, Atrapos can operate on different post-quantum dimensions, namely Dilithium and Kyber. This section describes the structure of Atrapos based on dimension parameters defined over distinct finite fields.

4.3.1 Atrapos-Dilithium

Atrapos-Dilithium operates on the finite field $\mathbb{F}_{8380417}$. The aim is to ensure collision security strength at least equivalent to that of AES. Based on chosen prime number, capacity (2ℓ) and rate (ℓ) can be computed from the inequality of $p^{c/2} \geq 2^{128}$.

$$\begin{aligned} p^{c/2} &= 8380417^{2\ell/2} = 8380417^{\ell} \\ 8380417^{\ell} &\geq 2^{128} \\ log_2(8380417^{\ell}) &\geq 128 \\ \ell.log_2(8380417) &\geq 128 \\ \ell.(22,9985) &\geq 128 \\ \ell &\gtrsim 5,5 \end{aligned}$$

The smallest prime number larger than 5 is 7. Therefore, research will be conducted on $\ell = 7$ for the dimension Atrapos-Dilithium.

4.3.2 Atrapos-Kyber

Atrapos-Kyber operates on a smaller dimension than Atrapos-Dilithium with the finite field $\mathbb{F}_{3329}$. The same inequality used for Atrapos-Dilithium is applied for computing ℓ value on Atrapos-Kyber.

$$\begin{aligned} p^{c/2} &= 3329^{2\ell/2} = 3329^{\ell} \\ 33297^{\ell} &\geq 2^{128} \\ log_2(3329^{\ell}) &\geq 128 \\ \ell.log_2(3329) &\geq 128 \\ \ell.(11.7008) &\geq 128 \\ \ell &\gtrsim 11 \end{aligned}$$

In this research, ℓ is chosen as 17 for the dimension Atrapos-Kyber.

Chapter 5

Avalanche

Avalanche criterion (AC) measures the sensitivity of an algorithm to input modifications by analyzing the bit-level behavior of the permutation or cipher. A permutation satisfies AC when a single bit input change causes change on **approximately half of the output bits** [1][10]. An algorithm not satisfying AC implies poor randomization and vulnerability to possible cryptography attacks.

5.1 Strict Avalanche Criterion (SAC)

AC is insufficient towards advanced attacks as it doesn't fully capture the uniformity of output changes. In response to this limitation, the Strict Avalanche Criterion (SAC) was introduced in 1985 by Webster and Tavares [12]. The advantage of SAC over AC comes from ensuring independency between output bit changes [10]. Specifically, SAC requires **each output bit** to change **independently with probability** $\frac{1}{2}$ whenever a single input bit is flipped [12][10].

However, the standard SAC definition doesn't fully apply to non-binary domains. Generalization is necessary, since output elements can take on more than two possible values.

Definition 5.1 (Strict Avalanche Criterion (SAC))**.** Each output element should change in such a way that every possible value occurs with equal probability of $\frac{1}{q}$ when a single input element is changed, where q is the number of elements in the finite field.

SAC ensures uniform distribution across each output element by given Definition 5.1. In a finite field with q elements, the probability of an element remaining unchanged in a uniform propagation is $\frac{1}{q}$. Hence, the probability that an element changes to any other value is $1 - \frac{1}{q} = \frac{q-1}{q}$.

AES satisfies SAC differently when assessed at the bit level versus the byte level. On bit level, $q = 2$ and the fraction $\frac{1}{2}$ is enough to satisfy the

criterion. On byte level, the desired probability becomes $\frac{255}{256}$ from applying the formula on $q = 2^8 = 256$.

For Atrapos to satisfy SAC, the desired probability of change is determined based on prime p. For Atrapos-Dilithium, p is set to 8380417 as specified in Section 4.3.1. Therefore, SAC is satisfied if each output digit has the probability change of $\frac{8380416}{8380417}$. For Atrapos-Kyber, p is established as 3329 in Section 4.3.2. The desired probability for each output digit is $\frac{3328}{3329}$ to satisfy SAC in dimension Kyber.

5.2 Avalanche Metrics

In order to understand the operation of avalanche tests, it is necessary to define several terms and properties. Avalanche metrics extend the SAC concept at a fine-grained level by using the effects of input changes on the output of an algorithm. Three avalanche metrics are utilized in fine-grained analysis, namely Avalanche Dependence, Avalanche Weight, and Avalanche Entropy derived from the avalanche probability vector $P_{\Delta F}$.

Each metric analyzes different aspects regarding output elements concluded from given input. First metric, Avalanche Dependence is defined by how many output elements may change. Second metric, Avalanche Weight is defined by expected weight of output elements, meaning the average number of elements that actually do change. Last metric, Avalanche Entropy, which generalizes SAC, defines the uncertainty of output elements changed by a given input difference. Since Avalanche Entropy is the core metric on our analysis, only the formula of Avalanche Entropy is specified. The mathematical expression for Avalanche Entropy is given below on binary domain, where p_i denotes the probability of output bit flips with index i:

$$H_{\text{av}}(F, \Delta) = \sum_i \left(-p_i \log_2(p_i) - (1 - p_i) \log_2(1 - p_i)\right)$$

To conduct tests on Atrapos, H_{av} needs to be defined over larger finite fields. Given the permutation and input difference Δ, the probability that the output difference j occurs on the output digit indexed by i from Δ can be computed as:

$$H_{\text{av}}(F, \Delta) = -\sum_i \sum_{j \in \mathbb{F}_q} \Pr(B_i = j) \log_2 \Pr(B_i = j)$$

5.3 The Choice for Avalanche Entropy

Avalanche Dependence and Avalanche Weight, present a simpler assessment on output elements than H_{av}. On their own, both metrics do not capture the

independence and uniformity of output element changes. Avalanche Dependence, does not account for the properties of distribution or independence, and focuses on the amount of changed elements. Avalanche Weight, focuses on the magnitude rather than the unpredictability of output change. In contrast, H_{av} incorporates both the amount and the probability distribution of changed output elements. For explained reasons, Avalanche Entropy is chosen as the primary metric in this research, as it provides a deeper evaluation of robustness.

5.4 Defining the Avalanche Probability Array

Avalanche probability array is defined to store probabilities of every output change on each output element. The algorithm is initialized as a vector to be applicable on every finite field that will be analyzed upon. However, after determining the permutation width, avalanche probability vector will be used as an array. The algorithm requires slight changes depending on the chosen finite field.

Algorithm 3 computes the avalanche probability vector $P_{\Delta F}$ of a cryptographic primitive F that operates on a state of b elements due to the application of input difference Δ on the input. All possible values that an output element can take in its domain is represented by j. The probability of observing a change of j to output element i from the application of Δ, is denoted as $P_{\Delta F}[i][j]$.

The procedure begins by initializing each output probability temporarily as zero in vector count. Then, the round function F is applied to a randomly selected input state A both with and without Δ. Subtracting computed output states indicates the impact of Δ. Whenever there is an occurrence of some possible value j for the output element i, specific $count[i][j]$ value gets incremented by 1.

The computation is iterated over a total of M randomly generated input states. Randomly sampling M states ensures the results to representative of the entire input space, rather than being biased by a particular state A. After processing M states, the occurrences are converted to probabilities by division on M.

Algorithm 3 Computation of the avalanche probability vector $P_{\Delta F}$ in non-binary

Parameters: a permutation operating on b-digit state in $\mathbb{F}_p$ applying a round function F
Inputs: An input difference Δ, number of samples M
Output: The avalanche probability vector $P_{\Delta F}$
Initialize the probability $b \times p$ bit vector of state probabilities $count_{[i][j]}$ to all zeroes
for M randomly generated states A **do**
    Compute $B = F(A + \Delta) - F(A)$
    for all state bit positions $\dot{I}$ **do**
        let B_i be the i-th digit of B
        $count[i][B_i] \leftarrow count[i][B_i] + 1$
    end for
end for
for all state bit positions i **do** ▷ Conversion to probabilities
    for all values of j **do**
        $count[i][B_i] \leftarrow count[i][B_i]/M$
    end for
end for
return $P_{\Delta F}$

5.4.1 Definition on Bit and Byte Level

For both bit and byte level computations the primitive $\mathbb{F}$ is defined as AES with block length of 128-bit. The round function takes the internal steps described in Chapter 3, specifically with the order of (3.1). In our avalanche analyses, experiment are conducted by the sample size 1000000 and initial key fixed to 0. Fixing the secret key makes the block cipher act as a permutation.

On bit level, the domain is $(\mathbb{F}_2)^{128}$ and the avalanche probability array of size 128 x 2. A bit can either have the value 0 or 1. Thus, only meaningful possible value for Δ is 1.

On byte level, the domain is $(\mathbb{F}_{2^8})^{16}$ which is a different representation for the block length 128. A byte can take 256 possible values by the definition of the finite field $\mathbb{F}_{2^8}$. The probability array has the size 16 x 256 which differs significantly from it's variation on bits level. Unlike bit level, on byte level Δ can have possible values from 0 to 255.

5.4.2 Definition on Non-binary Field

Computations of avalanche probability vector of a cryptographic primitive F is done on binary fields. To observe advanced results, modifications on the algorithm are required to be used for non-binary fields, specifically Atrapos permutation in the scope of this research.

In binary cryptography, addition (+) is equivalent to the bitwise XOR operation ($\oplus$)[10]. Unlike in binary fields, addition (+) and subtraction (-) over non-binary elements behave differently, demonstrating complications in finite fields beyond $\mathbb{F}_2$. Therefore, in Algorithm 3 subtraction operation is employed for linear cryptanalysis and modular differential cryptanalysis.

	Output Digit													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.195	0.220	0.194	0.190	0.221	0.221	0.194	1.000	0.217	0.203	0.217	0.217	0.198	1.000
1	0.199	0.201	0.198	0.191	0.192	0.192	0.205	-	0.181	0.198	0.181	0.181	0.198	-
2	0.205	0.202	0.191	0.198	0.190	0.190	0.201	-	0.208	0.193	0.208	0.208	0.199	-
3	0.198	0.171	0.194	0.246	0.183	0.183	0.193	-	0.209	0.187	0.209	0.209	0.190	-
4	0.203	0.206	0.223	0.175	0.214	0.214	0.207	-	0.185	0.219	0.185	0.185	0.215	-

Table 5.1: Toy Example of Avalanche Probability Array

For visualization, a toy example of avalanche probability array is created when the prime p is 5. The probabilities of each difference occurrence for each 13 output digit. The probabilities of output change across all output digits should sum to 1, as presented in Table 5.1. Entries corresponding to a probability of zero are denoted by "-" to aid legibility.

Chapter 6

Experiments on AES

The research is conducted by applying the Avalanche effect, specifically the Hav measure, to analyze the diffusion properties of AES and the Atrapos permutation proposal. The analysis on this chapter first examines AES at the bit level, where the effects of single-bit input changes are observed on output bits, and is later extended to byte level.

6.1 AES Experiments With $\mathbb{F}_2$

First, we apply avalanche tests on AES for bit level. AES on 128-bit block has the mapping of $(\mathbb{F}_2)^{128} \rightarrow (\mathbb{F}_2)^{128}$. As explained on section 3.1, every round except the last applies the MixColumns step. To include the case where MixColumns is absent on the last round, we introduce the term "half round". For a detailed analysis, AES has been applied on

- Half Round which is a single round without MixColumns
- 1 full round with MixColumns
- 1,5 rounds with 1 full round and 1 Half round
- 2 full rounds with MixColumns
- 2,5 rounds with 2 full rounds and 1 Half round

The analysis focuses on the probabilities of flipping bits from 0 to 7 individually over the specified rounds. Every probability table in below subsections for bit flipping experiments, presents only the probabilities where the output bit is 1, $P(bit = 1)$. Given the binary nature of bits, the probability for a bit to be 0, $P(bit = 0)$, can be computed directly from $1 - P(bit = 1)$.

6.1.1 Flipping Bits With Half Round

When flipping input bit x by half a round, the probabilities of the output bits are shown in Table 6.1. Independent of the flipped input bit, always exactly 8 output bits flip.

Output Bit	**Flipped Input Bit** x							
	0	1	2	3	4	5	6	7
0	0.515	0.469	0.515	0.531	0.453	0.453	0.531	0.516
1	0.516	0.485	0.516	0.531	0.500	0.516	0.532	0.562
2	0.453	0.563	0.500	0.470	0.454	0.515	0.469	0.516
3	0.563	0.501	0.468	0.453	0.515	0.468	0.516	0.531
4	0.453	0.483	0.562	0.500	0.499	0.469	0.469	0.484
5	0.484	0.453	0.500	0.531	0.500	0.547	0.531	0.531
6	0.454	0.500	0.533	0.499	0.547	0.531	0.531	0.484
7	0.501	0.532	0.499	0.547	0.532	0.532	0.485	0.515

Table 6.1: Output Bit Flip Probability Table Half Round

To satisfy SAC, each output bit should be equally likely to be 0 or 1 by definition in section 5.1. Table 6.1 indicates noticeable deviations from intended probability value. For instance, certain bits take the value 1 12.4% more frequently with a probability of 0.562, while others may occur 9.4% less frequently with a probability of 0.453. These results shows significant imbalance from applying only half a round, indicating poor SAC behaviour.

6.1.2 Flipping Bits With 1 Round

Output Bit	**Flipped Input Bit** x							
	0	1	2	3	4	5	6	7
0	0.501	0.516	0.516	0.484	0.562	0.454	0.485	0.453
1	0.531	0.516	0.484	0.563	0.485	0.485	0.454	0.500
2	0.499	0.499	0.515	0.500	0.500	0.562	0.500	0.531
3	0.548	0.531	0.531	0.500	0.484	0.499	0.531	0.500
4	0.531	0.469	0.500	0.468	0.500	0.500	0.500	0.546
5	0.530	0.516	0.516	0.469	0.547	0.469	0.547	0.531
6	0.484	0.532	0.532	0.484	0.469	0.469	0.532	0.530
7	0.516	0.548	0.563	0.453	0.500	0.483	0.532	0.485

Table 6.2: Output Bit Flip Probability Table 1 Round

The probabilities of the output bits for flipping input bit x on 1 full round are shown in Table 6.2. Independent of the flipped input bit, from the output bits, always the first 32 bits get flipped. The reason on the increase for the number of flipped output bits is due to the application of MixColumns. A single input bit flip affects multiple bytes simultaneously, causing the increase in output bit flips. Using only half round, the flipped input bit affects only the positions determined by the SubBytes and ShiftRows operations described in section 3.1 and 3.2. Only 8 bits are effected because both steps operates on a single byte.

As the probability tables grow increasingly large, only the probabilities of the first 8 bits will be reported rather than presenting each 128 bit. Even after one round, significant imbalances remain due to the frequently high deviations observed from Table 6.2.

6.1.3 Flipping Bits With 1.5 Rounds

Output Bit	**Flipped Input Bit** x							
	0	1	2	3	4	5	6	7
0	0.503	0.500	0.499	0.504	0.502	0.503	0.504	0.504
1	0.503	0.501	0.502	0.498	0.502	0.503	0.500	0.498
2	0.501	0.502	0.497	0.502	0.504	0.506	0.499	0.501
3	0.502	0.503	0.502	0.500	0.500	0.499	0.506	0.503
4	0.502	0.506	0.504	0.506	0.501	0.497	0.499	0.502
5	0.506	0.503	0.504	0.503	0.504	0.504	0.504	0.499
6	0.500	0.503	0.505	0.504	0.499	0.504	0.501	0.501
7	0.499	0.500	0.500	0.506	0.503	0.506	0.503	0.502

Table 6.3: Output Bit Flip Probability Table 1,5 Rounds

This time we apply 1 full round and 1 half round on AES. The first 8 bits are flipped regardless of the flipped input, similar to applying only half a round. On 1,5 rounds MixColumns step is included to AES. In the first full round, MixColumns mixes the bits of each column, causing the initial input bit flip to affect multiple positions within that column. However, the half round after that depends on steps SubBytes and ShiftRows which operates within individual rows. Therefore, one single input bit flip doesn't affect more than it's initial byte group.

Compared to Table 6.1 and 6.2, significant improvement can be observed to satisfy SAC. The highest observed bit probability exceeds 1.2%, while the lowest falls 0.6% below the desired SAC behaviour. On average, output bit probabilities drifts approximately 0.5% from ideal probability value. How-

ever, it is necessary to apply more rounds for complete SAC satisfaction across each output bit.

6.1.4 Flipping Bits With 2 Rounds

Output Bit	**Flipped Input Bit** x							
	0	1	2	3	4	5	6	7
0	0.505	0.500	0.501	0.503	0.502	0.502	0.502	0.505
1	0.499	0.506	0.502	0.503	0.502	0.502	0.502	0.500
2	0.501	0.504	0.503	0.501	0.504	0.503	0.505	0.499
3	0.504	0.502	0.502	0.503	0.504	0.501	0.500	0.505
4	0.503	0.501	0.506	0.505	0.502	0.502	0.498	0.499
5	0.499	0.501	0.504	0.502	0.500	0.504	0.505	0.504
6	0.501	0.503	0.504	0.505	0.501	0.498	0.505	0.502
7	0.502	0.503	0.499	0.500	0.496	0.503	0.506	0.504

Table 6.4: Output Bit Flip Probability Table 2 Rounds

Flipping input bit 0 to bit 7, across 2 full rounds caused all 128 output bits to be flipped. The probabilities are extremely close to 0.5 for every output bit similarly to applying 1,5 rounds. In previous experiments, flipping a single bit caused only a small number of output bit flips. The maximum observed number of flipped output bits is 32 bits that is not even half of the output bits.

Here MixColumns is applied twice, causing the input bit flip to spread across all 128 bits of the state almost uniformly. In the second round, the shifted bytes from the first round now are in different columns by the second ShiftRows operation. Then, second MixColumns spreads the new column values to the entire state. As a result, all 128 bits got affected.

The avalanche effect is clear after 2 rounds by ensuring bit flips on every output bit. Even though it is not shown by Table 6.4, similar probabilities observed across all output bits. Considering minor deviation on entire 128-bit output, SAC is sufficiently satisfied.

6.1.5 Flipping Bits With 2.5 Rounds

Applying the analysis to 2.5 rounds allows us to investigate any improvement on output bits compared to application of 2 rounds. Both applications were able to propagate the initial bit flip across the entire 128-bit state. The advantage of 2,5 rounds compared to 2 rounds is further alteration of the output state by the steps SubBytes, ShiftRows, and AddRoundKey.

6.2 AES Experiments With $\mathbb{F}_{2^8}$

Now, we take the analyses a bit further by applying the mentioned rounds on bytes. AES on bytes has the mapping of $\mathbb{F}_{2^8} \rightarrow \mathbb{F}_{2^8}$, which indicates 256 value possibilities for each byte. The larger domain results in more complicated analysis. This section explores the effect of Δ values across rounds on:

- The number of fixed output bytes when the difference of 1 is applied on each byte separately.
- The entropy results of output bytes when each possible difference value from the domain $\mathbb{F}_{256}$ applied to input byte 0.
- Probabilities of output byte change.

6.2.1 Number of Fixed Output Bytes

The difference of 1 is applied to each byte individually in this subsection. For each round, a separate byte table is presented for deeper analysis. The left side shows the input byte where the difference is applied. Fixed bytes are shown by the symbol "-" and bytes with entropy values between 0 and 1 are shown by "x". The right side of the table indicates the number of fixed bytes for that round.

Change of Bytes With Half Round

	Output bytes																
1 at	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# fixed
0	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15
1	-	-	-	-	-	-	-	-	-	-	-	-	-	x	-	-	15
2	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	15
3	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	15
4	-	-	-	-	x	-	-	-	-	-	-	-	-	-	-	-	15
5	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	-	15
6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	-	15
7	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-	-	15
8	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	-	15
9	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-	-	15
10	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	-	15
11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	x	15
12	-	-	-	-	-	-	-	-	-	-	-	-	x	-	-	-	15
13	-	-	-	-	-	-	-	-	-	x	-	-	-	-	-	-	15
14	-	-	-	-	-	-	x	-	-	-	-	-	-	-	-	-	15
15	-	-	-	x	-	-	-	-	-	-	-	-	-	-	-	-	15

Table 6.5: Half Round on AES

Table 6.5 shows that each input byte change causes a change on only a single output byte from applying half a round of AES, without MixColumns. This means the transformation only permutes the input bytes.

Change of Bytes With 1 Round

	Output bytes																
1 at	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# fixed
0	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	12
1	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	12
2	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	12
3	-	-	-	-	x	x	x	x	-	-	-	-	-	-	-	-	12
4	-	-	-	-	x	x	x	x	-	-	-	-	-	-	-	-	12
5	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	12
6	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	12
7	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	12
8	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	12
9	-	-	-	-	x	x	x	x	-	-	-	-	-	-	-	-	12
10	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	12
11	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	12
12	-	-	-	-	-	-	-	-	-	-	-	-	x	x	x	x	12
13	-	-	-	-	-	-	-	-	x	x	x	x	-	-	-	-	12
14	-	-	-	-	x	x	x	x	-	-	-	-	-	-	-	-	12
15	x	x	x	x	-	-	-	-	-	-	-	-	-	-	-	-	12

Table 6.6: 1 Round on AES

Table 6.6 shows that when MixColumns applied once, 4 output bytes get effected. The effected bytes always appear in groups of four consecutive positions. This behavior is predicted, since MixColumns spreads the change to the column and impacting 4 bytes. Still more than half of the bytes are fixed which isn't enough for satisfying the avalanche behaviour.

Change of Bytes With 1.5 Rounds

As seen on Table 6.7, the amount of fixed numbered output bytes are the same as applying 1 round. 4 output bytes are affected from a single input byte change. The clear difference from Table 6.6 is the distribution of impacted bytes across the state by the extra permutation round, instead of appearing in groups of four.

	Output bytes																
1 at	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# fixed
0	x	-	-	-	-	-	-	x	-	-	x	-	-	x	-	-	12
1	-	-	-	x	-	-	x	-	-	x	-	-	x	-	-	-	12
2	-	-	x	-	-	x	-	-	x	-	-	-	-	-	-	x	12
3	-	x	-	-	x	-	-	-	-	-	-	x	-	-	x	-	12
4	-	x	-	-	x	-	-	-	-	-	-	x	-	-	x	-	12
5	x	-	-	-	-	-	-	x	-	-	x	-	-	x	-	-	12
6	-	-	-	x	-	-	x	-	-	x	-	-	x	-	-	-	12
7	-	-	x	-	-	x	-	-	x	-	-	-	-	-	-	x	12
8	-	-	x	-	-	x	-	-	x	-	-	-	-	-	-	x	12
9	-	x	-	-	x	-	-	-	-	-	-	x	-	-	x	-	12
10	x	-	-	-	-	-	-	x	-	-	x	-	-	x	-	-	12
11	-	-	-	x	-	-	x	-	-	x	-	-	x	-	-	-	12
12	-	-	-	x	-	-	x	-	-	x	-	-	x	-	-	-	12
13	-	-	x	-	-	x	-	-	x	-	-	-	-	-	-	x	12
14	-	x	-	-	x	-	-	-	-	-	-	x	-	-	x	-	12
15	x	-	-	-	-	-	-	x	-	-	x	-	-	x	-	-	12

Table 6.7: 1,5 Rounds on AES

Change of Bytes With 2 Rounds

	Output bytes																
1 at	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	# fixed
0	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
1	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
2	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
3	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
4	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
5	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
6	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
7	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
8	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
9	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
10	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
11	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
12	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
13	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
14	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0
15	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	0

Table 6.8: 2 Rounds on AES

The input difference on a single input byte spreads across all 16 output bytes after 2 rounds. Observations from Table 6.8 demonstrates the strongest possible avalanche behavior among the examined experiments in section

6.3.1.

Change of Bytes With 2.5 Rounds

Since 2 rounds already provides alteration of every output byte, increasing the round number won't improve avalanche effect further than permuting the state. This supports our assumption that AES achieves complete avalanche within just two rounds at byte level. Further rounds reinforce security through permutation and nonlinear mixing to increase complexity.

6.2.2 Entropy Calculation For Each Possible Difference Value

The highest avalanche entropy H_{av} for the finite field $\mathbb{F}_{256}$ is computed by $log_2(256) = 8$, considering all possible values are equally likely to occur. However, Δ value of 0 should be excluded from the input difference set. If the input difference is 0, the two inputs are identical which makes the outputs identical. The zero element includes it's own entropy of 0 that results in a slight reduction from ideal entropy 8 to $log_2(255) = 7.9943$. The aim is to get an entropy as closest to 7.9943 as possible. The following analysis is performed on comparing all possible Δ values on input byte 0. Maintaining the same input byte where the difference is applied is essential to eliminate the influence of other variables.

Entropy From Half Round

Table 6.5 already shows that the change of input byte 0 by $\Delta = 1$, impacts only the output byte 0. Every other Δ value influences the output bytes on a per-byte basis. From applying half a round without MixColumns step, observed maximum entropy on output byte 0 by $\Delta = 118$ is $6.98464 \approx log_2(127)$. The behaviour is far from reaching ideal entropy, since approximately half of the output bytes remain the same.

Entropy From 1 Round

Similar to the outcome of Table 6.6, 12 bytes remain fixed with 1 round of application. From all possible Δ values, the highest total entropy was observed from $\Delta = 78$. Table 6.9 shows entropy values per 4 effected bytes, which are identical to the entropy obtained after a half round.

Output Byte	0	1	2	3
Hav	6.98464	6.98464	6.98464	6.98464

Table 6.9: Hav values for 1 Round

Entropy From 1.5 Rounds

Our previous tests support that 1,5 rounds impact 4 output bytes. However, with the additional half round of nonlinearity and permutation, the values for H_{av} measure are increased to approximately 7.991. Even though all possible Δ choices have similar entropy values, $\Delta = 52$ produces the best entropy with a slight difference among all entropy results from distinct Δ values. Entropy values for each affected output byte can be found in Table 6.10 for mentioned Δ choice.

Output Byte	0	7	10	13
Hav	7.99134	7.99128	7.99151	7.99148

Table 6.10: Hav values for 1,5 Rounds

Table 6.10 indicates approximately ideal values of entropy compared to Table 6.9, even though the number of changed output bytes are equal. Taking SAC into consideration, more rounds will be applied to test if the H_{av} values can be increased.

Entropy From 2 Rounds

2 rounds is enough for to propagate Δ on all output bytes based on the observations from Table 6.11. Across all experiments involving every possible Δ, the resulting total entropy values remain nearly identical. With a minor increase, the maximum entropy is achieved by $\Delta = 133$ across every Δ value.

Output Byte	0	1	2	3	4	5	6	7
Hav	7.99122	7.99122	7.99122	7.99122	7.99150	7.99150	7.99150	7.99150
Output Byte	8	9	10	11	12	13	14	15
Hav	7.99151	7.99151	7.99151	7.99151	7.99134	7.99134	7.99134	7.99134

Table 6.11: Hav values for 2 Rounds

Entropy From 2.5 Rounds

Increasing the round number might seem unnecessary considering propagation of Δ across all output bytes. However, the change in H_{av} values indicates otherwise. The additional half round provides slight increase on maximum achieved entropy from 7.99139 to 7.99417. Although this technically represents an improvement in the measured propagation, H_{av} values are still below desired entropy of 7.9943.

6.2.3 Probability of Output Bytes

Application of Algorithm 3 on byte level, gives the avalanche probability array that contains probabilities of all possible output differences. $P_{\Delta F}$ aids in analyzing the uniform behaviour of the distribution. If the round function behaves ideally, each output difference should appear with roughly uniform frequency and sum up to 1. However, if the probability values are higher than others, it indicates a bias towards certain Δ value. This subsection observes $P_{\Delta F}$ based on the applications of round function. In the same way as the experiments on entropy calculation, all possible Δ values are applied to only input byte 0. Sample size M is taken as 1000000 throughout this subsection. It is worth noting that, such small probabilities tend to round up to slightly higher values on larger M. When M is set to 10000, the sum of all output difference probabilities equals exactly 1, as expected. If M is increased to 1000000, then the sum slightly exceeds 1.

This subsection is a continuation of subsection 6.3.2 on a deeper level. The desired difference probability for each output byte is $\frac{1}{256} \approx 0.004$ based on SAC.

Half Round

As found on subsection 6.3.2 by the application of Half Round, only impacted output byte is 0. Observing the contents of $P_{\Delta F}$, each 126 output differences contributes approximately with a probability of $0.008 = \frac{1}{125}$, while one difference contributes a probability of $0.016 \approx \frac{1}{64}$. This behaviour shows that byte 0 is strongly biased without the use of MixColumns.

The distribution of probabilities on the changed output byte consists of 126 occurrences of 0.008, 129 occurrences of 0, and a single occurrence of 0.016, which is far from ideal setting. The results indicates that avalanche effect is not uniform without MixColumns.

1 Round

Only the first four bytes are affected from Δ as seen from Table 6.6 by applying 1 round. The probability distribution of these 4 bytes remains the same as applying only half a round. For a single output byte, 126 occurrences of 0.008, 129 occurrences of 0, and a single occurrence of 0.016 were observed. Applying 1 round is still insufficient to satisfy SAC because of the noticeable imbalance between probabilities.

1.5 Rounds

Table 6.7 shows that output bytes 0, 7, 10 and 13 are the only bytes impacted by Δ on input byte 0. Analyzing $P_{\Delta F}$ supports Table 6.7 by revealing that all other bytes have probabilities equal to 1, except for the specified

ones. Compared to the probabilities achieved previously, the ideal probability value of 0.004 appeared on mentioned bytes. However, most output bytes still have complete certainty on their predicted values.

2 Rounds

As supported by Table 6.8, by 2 full rounds Δ propagates to every output byte. However, $P_{\Delta F}$ contains deviation from ideal behaviour with probabilities such as 0.003 and 0.005. It indicates that impacting every output byte doesn't necessarily guarantee satisfaction of SAC. Some differences are slightly more likely to occur due to the structure of the S-box and column mixing. Hence, observation of adding half a round is helpful to see whether the desirable avalanche effect can be completely satisfied.

2.5 Rounds

Applying 2 full rounds and one half round, results in the ideal probability 0.004 to occur exactly 4080 times. It is worth mentioning that, 4080 is not the full size of $P_{\Delta F}$ as 4080 = 16 x 255. This is due to absence of probabilities from $\Delta = 0$. As explained before, applying $\Delta = 0$ to the input bytes does not alter the input value. Overall, the aimed behaviour of full uniform distribution is accomplished by 2.5 rounds on Round function of AES.

Chapter 7

Experiments on Atrapos

The research proceeds to its primary focus of conducting avalanche tests on Atrapos permutation, which will be used on Dilithium and Kyber dimensions. This section evaluates the effectiveness of Atrapos in propagating input differences across its state by the same avalanche-based methodology applied to AES. The current Atrapos proposal contains undecided parameters that are yet to be set, specifically sample size M, r_1 and r_2. Conducting avalanche tests on different combinations of parameter values will assist in reducing ambiguity. To recommend optimal values for r_1 and r_2, tests were conducted using the Dilithium and Kyber dimensions. Whereas tests on identifying suitable values for M were carried out solely on Atrapos-Kyber. Further details about each experiment can be found in their devoted subsections.

7.1 Finding Optimal Shift Offsets

The shift offset values r determines which digits participate in the non-linear or mixing operations during that round. The phase for shuffling the rows in a round function requires as many random parameters as the row size, that is fixed to 3. The first value of r_0 is fixed to 0. Remaining parameters r_1 and r_2 are determined based on which combination of 3 values provides most uncertainty on output digits. The state is conceptually arranged into 3 rows, so each digit position belongs to a row determined by its index modulo 3. Digits that give the modulo result y, will be referred as modulo group y. An example would be digit 5 to be in group modulo 2 because $5 \bmod 3 \equiv 2$.

In Atrapos proposal, two dimensions are mentioned Atrapos-Dilithium with row length $\ell = 7$ and Atrapos-Kyber with $\ell = 17$. For both dimensions tests will be conducted on two different cases of computation of r_2 values. Then, the number of remaining fixed output digits will be analyzed depending on the round number. The optimal shift offsets are determined by identifying the minimum value within the set of row-wise maximal values

for the best propagation. The maximum value is taken from the number of remaining fixed digits across all modulo groups per r_1 and r_2 combination.

7.2 Shift Offsets For Atrapos-Dilithium

The analysis is conducted on Atrapos-Dilithium where $\ell = 7$. Therefore, the feasible set of values for the initial row index is $r_1 \in \{0, 1, 2, 3, 4, 5, 6\}$. The evaluation proceeds by systematically examining all valid assignments of r_2 computed from r_1 in two separate cases.

7.2.1 First Case of Atrapos-Dilithium

Determining r_2 is done by the given equation $r_2 = r_1 + 1$, ensuring that the two selected row indices are consecutive. For each combination of offset values the experiment is carried out across rounds. This fine-grained, round-by-round inspection enables a precise analysis on elimination of fixed digits. Thereby determining the minimum number of rounds required for the avalanche criterion to be satisfied.

0 rounds

This is the test case where round function is never applied. It indicates the initial state without any change on the output digits. For $\ell = 7$, 21 digits are fixed.

1 round

Round function is applied only once with different combination of r values. The output was affected either one digit or two digits depending on their modulo group, regardless of r choices. Digits from group modulo 1 interacted with fewer digit positions, leading to 19 digits to remain fixed. Whereas, digits in both groups of modulo 0 and modulo 2, contains 18 fixed digits.

2 rounds

Digit x	$r_1 = 0$ $r_2 = 1$	$r_1 = 1$ $r_2 = 2$	$r_1 = 2$ $r_2 = 3$	$r_1 = 3$ $r_2 = 4$	$r_1 = 4$ $r_2 = 5$	$r_1 = 5$ $r_2 = 6$	$r_1 = 6$ $r_2 = 7$
x mod 3 = 0	6	6	6	6	6	6	6
x mod 3 = 1	9	10	9	10	9	9	9
x mod 3 = 2	6	6	6	6	6	6	6

Table 7.1: Results modulo 3 for 3 rounds

Table 7.1 shows that application of 2 rounds, impacted digits in group modulo 1 differently depending on the r combinations. Output digits in group modulo 0 and modulo 2 indicates consistent behavior across all tested configurations, each having 6 fixed digits. The maximum values of every offset combination are $\{9, 10\}$. Taking $min(9, 10) = 9$, the suggestions can be made from combinations of
$r \in \{\{0, 0, 1\}, \{0, 2, 3\}, \{0, 4, 5\}, \{0, 5, 6\}, \{0, 6, 7\}\}$

3 rounds

Applying the round function 3 rounds, indicates every combination of r results in 0 fixed digits for all input digits. This means that any input difference introduced into the state propagates to all digits of the output state. Therefore, 3 rounds are enough to satisfy avalanche behaviour for the dimension Dilithium.

7.2.2 Second Case of Atrapos-Dilithium

The analysis is next conducted under an alternative formulation for the computation of r_2 from $r_2 = r_1 + 4$. The increased difference of 4 yields a distinct set of corresponding r_2 values compared to those examined in the prior analysis. This modification introduces a greater separation between the two selected row indices which observes the sensitivity of the avalanche characteristics depending on index spacing.

1 round

Round function is applied only once with different combination of r values with the new equation. Same observations from previous equation can be seen on second case. Digits from group modulo 1 interacted with fewer digit positions, which leads to 19 fixed digits. Whereas, digits in both groups of modulo 0 and modulo 2, 18 fixed digits remained regardless of modulo group.

2 rounds

Digit x	$r_1 = 0$ $r_2 = 4$	$r_1 = 1$ $r_2 = 5$	$r_1 = 2$ $r_2 = 6$	$r_1 = 3$ $r_2 = 7$	$r_1 = 4$ $r_2 = 8$	$r_1 = 5$ $r_2 = 9$	$r_1 = 6$ $r_2 = 10$
x mod 3 = 0	6	6	6	6	6	6	6
x mod 3 = 1	9	8	8	8	8	8	8
x mod 3 = 2	6	6	6	6	6	6	6

Table 7.2: Results modulo 3 for 2 rounds for second method

Table 7.2 shows that application of 2 rounds indicates slight improvement on the number of fixed digits compared to Table 7.1. Both tables have the similarity on showing consistent behaviour for the output digits in group modulo 0 and modulo 2 across all tested configurations, each having 6 fixed digits. The improvement of Table 7.2 is containing less fixed digits for group modulo 1. All configurations except $r = \{0, 0, 4\}$, leads us to the same number of remaining fixed digits, thus any combination can be used for better entropy among them.

3 rounds

Upon applying 3 rounds of the round function, all fixed digits are eliminated across every combination of r values. This means that any input difference introduced into the state propagates to all digits of the output state. Therefore, 3 rounds are enough to satisfy avalanche behaviour for the dimension Dilithium for both methods.

7.3 Shift Offsets of Atrapos-Kyber

After concluding our analysis with smaller ℓ, the scope is extended to $\ell = 17$ on Atrapos -Kyber. This transition allows for an assessment of how the observed behaviour scales with increasing ℓ values. In this setting, the feasible set of values for the initial row index expands accordingly $r_1 \in \{0, 1, \ldots, 16\}$.

7.3.1 First Case of Atrapos-Kyber

This subsection applies the same equation used in section 7.2.1 on Atrapos-Kyber. Exhaustively examining all valid choices of r_1 enables a direct comparison with the findings obtained under subsection 7.2.1.

0 rounds

This test case indicates the initial state without any change on the output digits. For $\ell = 17$, 51 digits are fixed on any input digit.

1 round

Similar to Dilithium dimension, applying 1 round to the input is insufficient. The same number of digit change occurred on Atrapos-Kyber, even though ℓ is larger. Digits from group modulo 1 interacted with fewer digit positions, which leads to 49 digits to remain fixed. Whereas, digits in both groups of modulo 0 and modulo 2, 48 fixed digits remained.

2 rounds

Digit mod 3	r_1 **and** r_2 **values**																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	31	31	31	31	31	30	30	29	29	29	29	29	29	30	30	30	31
1	36	35	35	36	35	34	33	34	35	33	33	34	33	35	34	35	36
2	30	30	29	29	29	29	28	29	29	28	29	29	29	29	30	30	31

Table 7.3: Results modulo 3 for 2 rounds

Contrast to analysis of 2 rounds on Atrapos-Dilithium, no consistent behavior is observed across all tested configurations. The Table 7.3 shows that digits in group modulo 1 are more resilient to propagate then modulo groups 0 an 2. Considering all configurations, 33 is the minimum number that can be taken from the highest fixed digits sets. The suggestions can be made from
$r \in \{\{0, 6, 7\}, \{0, 9, 10\}, \{0, 10, 11\}, \{0, 12, 13\}\}$

3 rounds

Digit mod 3	r_1 **and** r_2 **values**																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	5	5	5	5	5	5	4	3	4	2	3	3	2	5	4	3	5
1	8	6	7	6	6	4	4	6	3	4	5	2	5	7	3	6	9
2	3	2	3	1	3	2	1	2	3	1	2	3	1	3	3	3	4

Table 7.4: Results modulo 3 for 3 rounds

The effects of having a larger ℓ is noticeable, since there are still remaining fixed digits after application of 3 rounds. Table 7.4 shows that having 3 rounds improved the uncertainty for each modulo group. However, more rounds are required to satisfy avalanche by it's description in Chapter 3. For now, the minimum of the highest remaining fixed digits is 3, occurring from the offsets $r = \{0, 11, 12\}$.

Even though $r = \{0, 9, 10\}$ has the least amount of remaining digits, $2 + 4 + 1 = 7$, compared to $3 + 2 + 3 = 8$ achieved by $r = \{0, 11, 12\}$, the distribution of $r = \{0, 11, 12\}$ is more uniform. Therefore, the choice $r = \{0, 11, 12\}$ is preferred against $r = \{0, 9, 10\}$

4 rounds

Upon exhaustive evaluation of all combinations of r values across every input digit, it is observed that no fixed digits remain after 4 rounds. This indicates that the input difference has fully propagated throughout the entire state, leaving no output position unaffected. Consequently, it can be concluded that the avalanche criterion is satisfied within 4 rounds for the configuration corresponding to $\ell = 17$. This result demonstrates that, regardless of the choice of row indices r_1 and r_2, the differential effect uniformly diffuses across all output positions within 4 rounds.

7.3.2 Second Case of Atrapos-Kyber

The analysis of is extended to Atrapos-Kyber, following the same equation introduced in 7.2.2. Conducting analysis of the same test cases on Atrapos-Kyber enables a direct comparison of the avalanche characteristics between the two schemes under identical index spacing conditions.

1 round

The remaining number of fixed digits is the same as the observations from 7.3.1. Digits from group modulo 1 have 49 fixed digits, while, digits in both groups of modulo 0 and modulo 2 still contains 48 fixed digits.

2 rounds

Digit mod 3	r_1 **and** r_2 **values**																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	32	32	30	30	31	32	31	30	31	31	31	30	30	32	32	30	31
1	36	35	35	33	34	35	34	33	35	34	34	34	33	36	36	34	35
2	32	30	29	29	31	31	28	28	31	31	29	29	30	32	30	28	30

Table 7.5: Results modulo 3 for 2 rounds for second method

The outcome is similar to the usage of first case on $\ell = 17$. However, on groups modulo 0 and modulo 2 Table 7.5 shows slight increase in the number of fixed digits compared to Table 7.3.

Observations on Table 7.5 indicates that 33 is the minimum number that can be taken from the highest fixed digits sets. The offset suggestions can be made from any combination that has 33 as their highest number of remaining fixed digits
$r \in \{\{0, 3, 7\}, \{0, 7, 11\}, \{0, 12, 16\}\}$

3 rounds

Digit mod 3	r_1 **and** r_2 **values**																
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	5	5	5	3	3	5	4	3	4	3	5	4	2	5	5	2	5
1	5	7	6	2	5	5	4	5	4	5	7	4	5	8	6	6	6
2	4	3	3	0	2	3	0	2	4	3	3	2	1	4	2	1	4

Table 7.6: Results modulo 3 for 3 rounds for second method

In contrast to Dilithium dimension, 3 rounds is insufficient for both methods based on Tables 7.4 and 7.6. Most significant difference seen from comparing these two tables is removal of fixed digits for some digit in modulo group 2, specifically with the combinations of $r = \{0, 3, 7\}$ and $r = \{0, 6, 10\}$. Even though the table is improved, not all output digits are affected by the introduced Δ. This indicates that the input differences have not yet fully propagated across the state, requiring additional rounds to satisfy SAC. Considering Table 7.6, the minimum of the highest remaining fixed digits is 3, occurring from the offsets $r = \{0, 3, 7\}$.

4 rounds

As it was observed from the first method, there are no fixed digits remaining after 4 rounds. This means the input difference has propagated to the entire state.

7.3.3 Conclusion From Both Methods

Various configurations of r values were tested through avalanche tests. The results indicate that increasing the row length ℓ requires an increase on the round numbers. Two different ℓ dimesnisons were tested by distinct cases of computing r_2 values. Some configurations showed similar performance across both dimensions, while others offered improvements for a particular dimension. After analyzing the combined results, a common set of r configurations was identified to provide impact on every output digit.

The criteria when selecting the configurations are to maximize propagation and minimize fixed digits for ensuring strong avalanche behavior. Based on these criteria, on Atrapos-Dilithium the optimal shift offsets can be chosen from $r \in \{\{0, 1, 5\}, \{0, 2, 6\}, \{0, 3, 7\}, \{0, 4, 8\}, \{0, 5, 9\}, \{0, 6, 10\}\}$.

For Atrapos-Kyber, the increase on ℓ requires more rounds to satisfy the avalanche criterion compared to Atrapos-Dilithium. Different combinations

can be suggested based on the number of rounds applied. Based on our findings on 7.3.2, the only suggested shift offset combination is $r = \{0, 3, 7\}$.

Across both dimensions, multiple r configurations are tested to satisfy SAC. Upon examining the intersection of the recommended r configurations for both dimensions, it is found that the sole common combination is $r = \{0, 3, 7\}$. Given that this configuration achieves the desired propagation properties across both dimensions, the continuation of this research shift offsets are assigned as such.

7.4 Changing Sample Size

Now that r values are established in the scope of this thesis, we can look at the relation between sample size M and Δ. Avalanche tests are conducted with the metric Hav described in Chapter 5.3. The aim is to test our suggested r combination with different settings to get the highest entropy. Since Atrapos-Kyber is defined by finite field $\mathbb{F}_{3329}$, the highest entropy can be $log(3329) = 11.7009$. The analysis should indicate how close we can get to the highest entropy without downgrading the performance. For efficiency purposes even though there remains 3 fixed digits, tests are conducted on 3 rounds. Therefore, total entropy will never be exactly 11.7009.

Previously discovered optimal shift offset values are used in the calculation of P_F.

The input difference Δ is a randomly selected integer, restricted by the range $[0, 3329]$. This bound is imposed to prevent collisions arising from reduction modulo 3329. Selecting values outside this range would risk producing collisions. The randomly selected values that will be used in the experiments are introduced as the set $\Delta = \{1, 3, 17, 29, 61, 302, 837, 1438, 2843\}$.

M	$\mathbf{diff}_1$	$\mathbf{diff}_3$	$\mathbf{diff}_{17}$	$\mathbf{diff}_{29}$	$\mathbf{diff}_{61}$	$\mathbf{diff}_{302}$	$\mathbf{diff}_{937}$	$\mathbf{diff}_{1438}$	$\mathbf{diff}_{2843}$
1000	9.68116	9.68523	9.68690	9.68033	9.68361	9.68166	9.67940	9.68394	9.68703
10000	11.43824	11.43877	11.43983	11.43781	11.43962	11.43930	11.43813	11.43718	11.43834
100000	11.67666	11.67656	11.67677	11.67677	11.67677	11.67677	11.67666	11.67677	11.67666
250000	11.69122	11.69122	11.69133	11.69133	11.69133	11.69133	11.69122	11.69122	11.69122
1000000	11.69844	11.69844	11.69844	11.69844	11.69844	11.69844	11.69844	11.69844	11.69844
2000000	11.69972	11.69972	11.69972	11.69961	11.69972	11.69972	11.69961	11.69972	11.69961
2500000	11.69993	11.69993	11.69993	11.69993	11.69993	11.69993	11.69993	11.69993	11.69993
3500000	11.70014	11.70014	11.70014	11.70014	11.70014	11.70014	11.70014	11.70014	11.70014
5000000	11.70036	11.70036	11.70036	11.70036	11.70036	11.70036	11.70036	11.70036	11.70036

Table 7.7: Effect on Total Hav by changes on Δ and M

Table 7.7 shows that all the Δ values behaves nearly identical on each

sample size. It can be concluded that total H_{av} results are independent from Δ choice. Until the sample size 1000000, improvements on the total entropy are noticeable. On the other hand, sample sizes larger than 1000000 have minor contributions to H_{av} values with excessive overhead. As seen in Table 7.7, the total entropy values remain largely stable, with minor variations on individual digits. For instance, slightly better entropy can be achieved by $M = 5000000$. However, computational cost increases drastically from significant time consumption. Given the negligible gain in entropy on overall assessment and excessive overhead, increasing the sample size beyond 1,000,000 is not necessary. Therefore, reaching entropy 11.69844 is considered to be sufficient enough for satisfying avalanche behaviour. As a conclusion, the suggestion for parameter sample size M is 1,000,000.

Chapter 8

Related Work

This chapter reviews the cryptographic primitives and supporting mechanisms that form the foundation for our work. In particular, iterated permutation KECCAK-p and the lightweight permutation Xoodoo. Understanding the design principles behind the related work provides context for deeper understanding of our analyses discussed in previous chapters.

8.1 Xoodoo

Xoodoo is an iterated permutation parameterized on number of rounds n_r. The design is inspired by the permutation KECCAK-p used in Keccak, SHAKE128 and SHAKE256 [6][11]. The state it is iterated on consists of 3 equally sized horizontal planes where each one made of 4 parallel 32-bit lanes. The state can be represented as a set of 128 columns of 3 bits in a 4 x 32 array [6]. The bits of the state are indexed by (x, y, z) where y represents planes; x represents lanes within a plane; z represents index bits in a lane [6]. These planes interact through mixing and nonlinear operations that occur per 3-bit columns. Outside of these specific interactions, the planes move independently from each other to incorporate the wide trail strategy.

The design idea of Xoodoo was adapted from KECCAK-p. The wide trail strategy is the core architectural approach used in Xoodoo alongside weak alignment property. This combination results in minimal trail clustering in differentials and correlations, making the permutation resistant to specific classes of attack, such as truncated differentials [6]. Xoodoo maintains this behavior while running efficiently on many types of devices, from low-end microcontrollers to high-end processors with vector instructions [6].

One of the principal improvements over earlier permutations is the notably rapid complete propagation of the input difference across all output elements. Statistical analysis of Xoodoo introduces several avalanche metrics to satisfy SAC with a rapid speed. From the introduced metrics, this research utilizes H_{av} metric as the core methodology.

8.2 Xoodoo Relation to Atrapos

The Atrapos permutation proposal is inspired by Xoodoo, particularly in its state organization and the iterative structure of its round function. Both primitives configures their internal state by structures on base 3, three-row in Atrapos and three-planes in Xoodoo. The difference is that Xoodoo uses three 128-bit planes, whereas Atrapos arranges digits in $\mathbb{F}_p$ into three rows across ℓ columns. The composition of round function in Atrapos by steps θ, ρ, ι, and γ that was described in section 4.1, mirrors the bit-oriented operations used in Xoodoo (θ, ρ, ι, χ). Nonlinearity step γ in Atrapos aligns with nonlinear transformations in Xoodoo. These similarities allows Atrapos to reach a high algebraic degree and rapid randomization. As a result, it is well suited for post-quantum standards like Kyber and Dilithium, much like how Xoodoo serves as the high-speed engine for parallelizable deck functions.

The Atrapos permutation differs from Xoodoo by some characteristics. First, the choice of main component that the operations are performed on. Atrapos uses digits within a finite field on modulo p which in this thesis taken as 3329, when Xoodoo operates on single bits.

Second, the flexibility of permutation's state size. While Xoodoo has a fixed 384-bit(48-byte) state [6], the width b of Atrapos can vary based on the choice of the prime p and the number of columns ℓ. As experimented in section 6.4.1, width b varies depending on the dimension where $\ell = 7$ or $\ell = 17$.

Third, the state geometry used for the structure. Xoodoo takes 384 bits into three planes of 128 bits. Atrapos arranges the digits into a two-dimensional array of 3 rows and ℓ columns.

Chapter 9

Conclusions

This research evaluated the propagation properties of the Advanced Encryption Standard and the Atrapos permutation proposal by applying the H_{av} avalanche measure. Conduction of avalanche tests aids in decision-making for selecting uncertain parameters. Moreover, the effectiveness of propagation in a chosen permutations can easily be assessed by this methodology.

9.1 Conclusion From Experimenting on AES

The results obtained from the bit-level analysis provide an assessment of propagation on individual bits. On the other hand, the byte-level experimental results yield a more informative characterization of the overall avalanche behaviour. At both levels of analysis, AES conducted on 0.5 and 1 rounds exhibits similar behaviour. This indicates that the AES S-box has a shortage, as no S-box achieves perfection. At 1.5, 2, and 2.5 rounds, the differential propagation improves considerably. However, the exclusion of output difference zero prevents reaching ideal behaviour. Nevertheless, AES demonstrates a high degree of sensitivity to input changes and is observed to satisfy the Strict Avalanche Criterion within as few as 2 rounds.

9.2 Conclusion From Experimenting on Atrapos

After validating the methodology using AES, the same avalanche-based analysis was applied to the Atrapos permutation proposal. Considering that the current proposal has not yet determined values for certain variables, this thesis recommends optimal values by conducting avalanche tests on different parameter combinations and observing how these choices influence propagation on output digits.

For the 3 shift offsets r_0, r_1, r_2 required for the round function, two different cases were considered on Atrapos-Dilithium and Atrapos-Kyber. The desired configuration should ensure that small changes in the input quickly

propagate through the internal state and eventually affect all output digits. Avalanche tests provided guidance on the effectiveness of each possible combination.

9.2.1 Suggestions on Atrapos-Dilithium

On Atrapos-Dilithium, omission of all fixed digits ensured in 3 rounds. Therefore, we analyzed r configurations based on the least total fixed-digit-occurrence in 2 rounds. Experiments conducted under the second case yielded results that are closer to a uniform distribution over the output digits. Considering every test experimented, the suggested shift offset configurations are given by $r \in \{\{0,1,5\},\{0,2,6\},\{0,3,7\},\{0,4,8\},\{0,5,9\},\{0,6,10\}\}$.

9.2.2 Suggestions on Atrapos-Kyber

On Atrapos-Kyber, a larger set of possible Δ values is present compared to Atrapos-Dilithium. Only after 4 rounds, propagation of Δ is spread across all output digits. The results obtained under the second case demonstrate reduction in the number of remaining fixed digits relative to those observed in the first case. For the first time across all experiments, one of the modulo groups achieved omission of fixed digits in some cases. Upon comparing every outcome across all evaluated r combinations, the optimal r values are selected as $r = \{0,3,7\}$.

9.2.3 Final Suggestions of r Values

From the possible suggestions of r combinations on both dimensions, it can be concluded that the second method provided greater change on the output. The suggested shift offset values of separate dimensions intersect only with the configuration $r = \{0,3,7\}$. Therefore, the final suggestions on each offset value r_0, r_1, r_2 are $\{0,3,7\}$ respectively.

9.2.4 Suggestions on Sample Size M

Finally, avalanche tests are conducted to observe the relation between sample size M and Δ by using our suggested r combination. The observation that entropy values remain nearly identical for various Δ choices suggests entropy values are depending on sample size rather than the choice of Δ. The progress on reaching the hightest entropy slows down after increasing the sample size to larger values than 1000000. Slowing down corresponds to both time aspect in seconds and entropy changes. As a conclusion, the optimal sample size is suggested as 1000000.

Bibliography

[1] Maki Mahdi Abdulhassan, Jamal Kamil K. Abbas, and Nada Qasim Muhammed. A robust statistical test for evaluating the avalanche effect in block ciphers. *SN Computer Science*, 6(1):875, 2025.

[2] Alessandro Amadori, Thomas Attema, Maxime Bombar, João Diogo Duarte, Vincent Dunning, Simona Etinski, Daniël van Gent, Matthieu Lequesne, Ward van der Schoot, Marc Stevens, and AIVD Cryptologists and Advisors. Guidelines for migrating to post-quantum cryptography: The pqc migration handbook. Technical report, AIVD, CWI, and TNO, Netherlands, December 2024. Revised and Extended Second Edition, first print December 2024, reprint August 2025.

[3] Parisa Amiri Eliasi, Koustabh Ghosh, and Joan Daemen. Mystrium: Wide block encryption effcient on entry-level processors. Technical Report 2024/1474, IACR Cryptology ePrint Archive, 2024.

[4] Joan Daemen. Introduction to design in symmetric cryptography. In *Designs, Codes and Cryptography*. Springer, 2011. Chapter.

[5] Joan Daemen. The future of symmetric cryptography: Deck functions. In *Future Directions in Symmetric Cryptography*. Springer, 2018. Chapter.

[6] Joan Daemen, Seth Hoffert, Gilles Van Assche, and Ronny Van Keer. The design of Xoodoo and Xoofff. *IACR Transactions on Symmetric Cryptology*, 2018(4):1–38, 2018.

[7] Joan Daemen and Bart Mennink. Lecture notes 4: Block ciphers. Course NWI-IBC023, Cryptography, Spring 2025, February 2025. Lecture notes, February 19, 2025.

[8] Joan Daemen and Bart Mennink. Lecture notes 7: Sponge functions. Course NWI-IBC023, Cryptography, Spring 2025, March 2025. Lecture notes, March 12, 2025.

[9] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES — The Advanced Encryption Standard*. Springer, 2002.

[10] K. Mohamed, M. N. Mohammed Pauzi, F. H. Hj Mohd Ali, and S. Ariffin. Analyse on avalanche effect in cryptography algorithm. In *Reimagining Resilient Sustainability: An Integrated Effort in Research, Practices & Education*, volume 3 of *European Proceedings of Multidisciplinary Sciences*, pages 610–618, 2022.

[11] National Institute of Standards and Technology. Federal information processing standards publication (fips) 203: Module-lattice-based key-encapsulation mechanism standard, August 2024.

[12] A. F. Webster and S. E. Tavares. On the design of s-boxes. In *Advances in Cryptology – CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 523–534. Springer, 1986.