Study of Impossible Differential Cryptanalysis on Block Ciphers

Ng Wei En and Yeoh Jia Er
Victoria Junior College
20 Marine Vista, Singapore 449035

***Abstract*—Impossible Differential Cryptanalysis (IDC) is a powerful attack that has been successfully applied to several block ciphers in encryption schemes such as the Advanced Encryption Standard (AES). To explore and create new attack variants using IDC, a simple 48-bit Substitution Permutation Network (SPN) cipher called SmallCipher was created, with the novel method of using double distinguishers for the IDC attack. This new attack achieves a time complexity of $2^{29.587}$ chosen plaintexts, as compared to $2^{29.733}$ 6-round encryptions with a single-distinguisher. It is theorised that more significant improvements in optimal time complexity would be observed for 96-bit SmallCipher, between $2^{64.001}$ and $2^{90.830}$ for the single-distinguisher attack, and between $2^{54.438}$ and $2^{90.830}$ for the double-distinguisher attack. Furthermore, the double-distinguisher attack is theorised to converge more quickly to the optimal time complexity as the number of plaintext-ciphertext pairs used increases. Our methods of searching for 6-round distinguishers presented in this paper and newly discovered attack variants can be extended to other SPN ciphers such as AES.**

I. Introduction

The Advanced Encryption Standard (AES), otherwise known as Rijndael, was established by the U.S. National Institute of Standards and Technology (NIST) in 2001 [1]. AES is a symmetric-key algorithm, meaning the same key is used for encrypting and decrypting data. It supersedes the Data Encryption Standard (DES) and was adopted by the U.S. government for encryption of classified information. Nowadays, it is widely used in many encryption software across the world.

This paper studies Impossible Differential Cryptanalysis (IDC), an extension of the differential attack, which was first used by Lars Knudsen on DEAL [2]. Differential attacks exploit differences in the inputs of a cipher and its effects on the resultant difference. IDC extends on this idea by using "impossible differentials" (differentials that occur with zero probability) in order to eliminate wrong keys. An efficient method for finding impossible differentials, called a miss-in-the-middle attack, consists of finding "two events with probability one, whose conditions cannot be met together" [3]. It has since been successfully applied to several block ciphers in encryption schemes.

The best known chosen-plaintext attack against 7-round AES-128 at this time of writing uses IDC, and requires $2^{106.2}$ chosen plaintexts, $2^{94.2}$ bytes of memory, and a time complexity of approximately $2^{110.2}$ 7-round encryptions [7]. Other 7-round chosen-plaintext attacks are covered in Table I for comparison. Motivated by the effectiveness of IDC on 7-round AES-128 in comparison to other attacks, the paper aims to explore and extend IDC attacks further, and implement the attacks in practice to obtain fast and efficient results. As implementing state-of-art IDC attacks on Substitution Permutation Networks (SPN) ciphers like AES is impractical, a simpler but similar SPN cipher, named SmallCipher, was created to facilitate practical execution of new attack variants.

TABLE I
A summary of chosen-plaintext attacks against 7-round AES.

Attack Type	Data	Time	Memory
Brute Force		2^{128}	
Square [4]	$2^{128} - 2^{119}$	2^{128}	2^{68}
Collision [5]	2^{32}	2^{128}	2^{100}
Meet-in-the-middle [6]	2^{80}	2^{123}	2^{126}
IDC [7]	$2^{106.2}$	$2^{110.2}$	$2^{94.2}$

A. SmallCipher

SmallCipher is a 6-round symmetric block cipher that takes in plaintexts and keys of 48 bits, represented by a 4 by 3 matrix of nibbles where each nibble is made up of 4 bits.

Each round function is composed of the following operations in order [8].

1) SubNibbles (SN): A one-to-one substitution of each nibble in the input matrix.
2) ShiftRows (SR): An operation that rotates each row in the input matrix to the right by varying offsets (0 for the first and second row, 1 for the third row and 2 for the fourth row).
3) MixColumns (MC): Multiplication of each column in the input matrix with a fixed matrix in the Galois field $GF(2^4)$.
4) AddRoundKey (AK): XOR operation between the input matrix and the round key.

An initial key addition is performed before the first round, and MC is omitted in the last round. Round keys are generated based on a key schedule similar to the AES key schedule, which allows the derivation of previous round keys given a specific round key, thereby retrieving the master key.

B. Notations

The following notations are utilized: x_i^I denotes input of round i, while x_i^S , x_i^R , x_i^M , x_i^O denote the intermediate

values after the application of SubNibble (SN), ShiftRow (SR), MixColumn (MC) and AddRoundKey (AK) operations of round i respectively. The ith round key is denoted by k_i, with the master key denoted as k_0. When decrypting, the order of MixColumn MC and AK may be interchanged, by taking the XOR of x_i^R with an equivalent round key, $\omega_i = MC^{-1}(k_i)$. The intermediate value after AK with equivalent round key will be denoted as x_i^ω. $x_{i,col(j)}$ denotes the jth column of x_i.

Between two matrices of nibbles, corresponding nibbles with different values are referred to as "active nibbles", while those with equal value are referred to as "passive nibbles". A "differential" refers to the difference between two matrices. p_i is the probability that $MC^{-1}(x_i^M)$ leads to the differential of x_i^R (or in cases where MC and AK have been swapped, the probability that $MC^{-1}(x_i^M)$ leads to the differential of x_i^ω).

II. Methodology

In order to conduct IDC, distinguishers for SmallCipher need to be obtained before an attack can be carried out. A "distinguisher" is a series of propagation patterns of differentials which allow an adversary to exploit a cipher, so as to obtain additional information about secret keys. Programs for finding distinguishers were implemented in Python 3 [9] and the cipher and attacks in C/C++, compiled with GCC [10] using flags `-O3`. The source code is available at https://github.com/wei2912/idc.

A. Distinguisher Generation

For any pair of differentials (x, y), if the forward propagation of x by two rounds and backward propagation of y by one round leads to disjoint sets of differentials, there exist no plaintext pairs satisfying x whose corresponding output pairs after three rounds of encryption will satisfy y. Hence, (x, y) forms a 3-round impossible differential property (IDP). After finding an IDP (x, y), the program creates a forward extension of 3 rounds from y, creating a 6-round distinguisher with 2 key guessing rounds included.

B. Standard Attack

The attack is composed of three stages:

1) Plaintext-Ciphertext (PT-CT) Pair Generation: All 2^{12} plaintexts with the same values in the nibbles corresponding to the passive nibbles of x_1^I are gathered and encrypted. The ciphertexts are placed in a hash table, indexed by their nibble values corresponding to the active nibbles of x_6^O. In each row of the hash table, the ciphertexts are paired with each other to see if they satisfy the differentials at both x_1^I and x_6^O. A pairing of two ciphertexts and their corresponding plaintexts is known as a plaintext-ciphertext (PT-CT) pair. A fixed number of PT-CT pairs are gathered before moving onto Stage 2.

2) IDC on Partial (k_6, ω_5) Pairs: For each of the PT-CT pairs generated in Stage 1, all remaining partial k_6s that have not been eliminated by the previous PT-CT pairs are iterated through. For each partial k_6, a k_6 which satisfies the partial k_6 is formed.

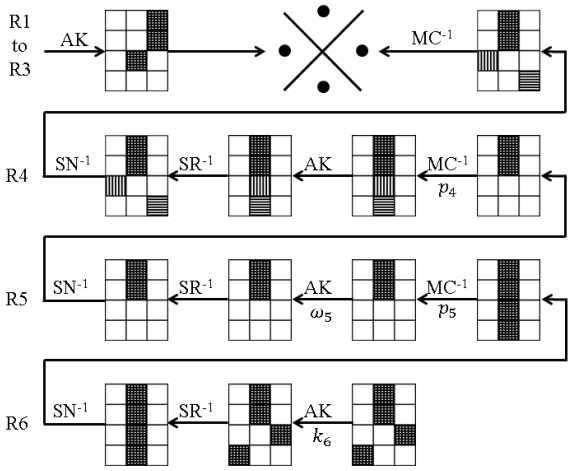

Fig. 1. Both Type I 6-round distinguishers combined into one.

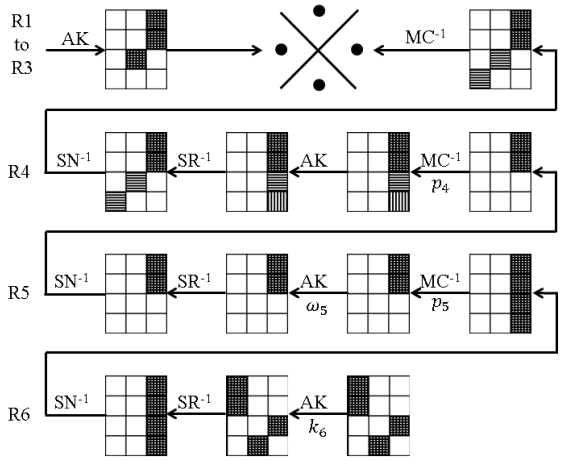

Fig. 2. Both Type II 6-round distinguishers combined into one.

The rest of the procedure follows:

1) Decrypt both ciphertexts with the k_6 by one round.
2) Observe the differential at x_5^ω. If the differential meets the IDP, iterate through all remaining partial (k_6, ω_5) pairs corresponding to that partial k_6. Each ω_5 contains the value of key nibbles corresponding to the active nibbles at x_5^ω. For each ω_5,
   a) Decrypt the two matrices at x_5^ω by one round.
   b) Observe the differential at x_4^ω. If the differential meets the IDP, eliminate the (k_6, ω_5) pair.
   c) If all of the remaining partial (k_6, ω_5) pairs corresponding to that partial k_6 were eliminated, the partial k_6 is eliminated.

After going through elimination of partial (k_6, ω_5) pairs, a reduced list of partial k_6s is obtained.

3) Brute Force: For each remaining partial k_6, all the possible values in the nibbles corresponding to the 8 passive nibbles at x_6^O are iterated through to form k_6s. For each k_6, the middle column of ω_5 is derived. If it satisfies the remaining

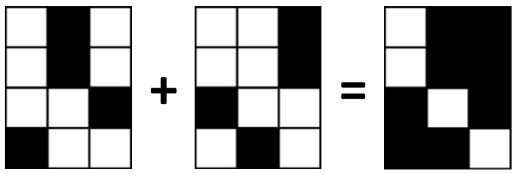

Fig. 3. Combining partial k_6 lists from both distinguishers. The shaded boxes indicate the active nibbles of the differential at x_6^O for each distinguisher.

partial ω_5s left, the rest of the keys until k_0 are derived and the keys are tested for correctness.

C. Single-Distinguisher Attack

The two 6-round distinguishers (Fig. 1) have the same propagation of differentials from x_6^O to x_5^I and can be combined to form a single-distinguisher with a doubled p_4. These two distinguishers shall be called Type I distinguishers, and the attack referred to as the "single-distinguisher attack". The horizontal and vertical lines represent the active nibbles in the corresponding differential of the first and second distinguisher respectively. There are another two 6-round distinguishers which have the same active nibbles in x_1^I and x_6^O, which shall be called Type II 6-round distinguishers (Fig. 2).

D. Double-Distinguisher Attack

The active nibbles in x_6^O of the Type I and II distinguishers do not overlap. Combining the partial k_6s generated from both 6-round distinguishers, the values of 8 nibbles can be retrieved and only the values of the remaining 4 nibbles need to be obtained through brute force. This effectively reduces the number of k_6s which need to be tested (Fig. 3). This attack is referred to as the "double-distinguisher attack".

The double-distinguisher attack is composed of four stages:

1) PT-CT Pair Generation: This stage is carried out similarly to the single-distinguisher attack. As both distinguishers have the same differential at x_1^I, the same group of plaintexts is encrypted and the additional number of chosen plaintexts needed is minimal.

2) IDC on Partial (k_6, ω_5) Pairs Corresponding to Type I Distinguishers: This stage is carried out similarly to the single-distinguisher attack with the use of the Type I distinguishers.

3) IDC on Partial (k_6, ω_5) Pairs Corresponding to Type II Distinguishers: This stage is carried out similarly to the single-distinguisher attack with the use of the Type II distinguishers.

4) Brute Force: The two lists of remaining partial k_6s in the list of partial (k_6, ω_5) pairs are combined, and all the possible values in the nibbles corresponding to the 4 remaining nibbles at x_6^O not covered by Type I or II distinguishers (Fig. 3 are iterated through. For each k_6 formed, the middle column and last column of ω_5 is derived. If it satisfies the remaining partial ω_5s left from conducting IDC on the first and second distinguisher, the rest of the keys till k_0 are derived and tested.

TABLE II
NUMBER OF CHOSEN PLAINTEXTS REQUIRED TO OBTAIN PT-CT PAIRS.

N	15679	31358	47037
Data	$\sim 2^{35.411}$	$\sim 2^{36.418}$	$\sim 2^{39.997}$

III. THEORETICAL MODEL

In order to pick the best distinguishers to use for attacks on 48-bit SmallCipher and estimate the performance of the attacks when extended to 96-bit SmallCipher, a theoretical model was constructed and calculations were made on the time complexity of the attacks. Details of the model are provided in the Appendix.

IV. RESULTS

The attacks were benchmarked on a Google Compute Engine instance [11] running Ubuntu 16.04, with one virtual CPU and 3.75 GB of memory.

A. Data Complexity

In order to decide the number of PT-CT pairs per distinguisher (N) to use, the number of partial (k_6, ω_5) pairs left after N rounds of filtering, $2^{4(4+2)}(1-2p_4p_5)^N = 2^{24}(1-2p_4p_5)^N$, was set to 2^{16}, 2^8 and 1. This gives $N =$ 15679, 31358 and 47037 respectively, which represents the number of PT-CT pairs per distinguisher to use. The number of encryptions required to generate a specific number of PT-CT pairs was measured thrice, and the mean was taken (Table II). In order to obtain the highest number of PT-CT pairs per distinguisher, 47037, approximately $2^{36.997}$ chosen plaintexts were required. This shows that the data requirements for the attacks are realistic.

B. Time Complexity

A single dataset was used for brute force and three datasets were used for the single and double-distinguisher attack. For each dataset, the programs were benchmarked thrice and the mean was taken.

The lower and upper bounds for the IDC stages and brute force were calculated, as well as the total time taken. For comparison, estimates of the time complexities were made by dividing the time taken for the IDC stages and the brute force by the time taken to conduct brute force ($\sim$ 207 days) and multiplying by 2^{48}, the theoretical time complexity of brute force.

Table III and IV present time complexities for the single and double-distinguisher attack. The upper row indicates the theoretical time complexity, while the lower row indicates the estimated time complexity from running the attack in practice.

The single-distinguisher attack has an estimated time complexity of $2^{29.733}$ whereas the double-distinguisher attack has an estimated time complexity of $2^{29.587}$, both requiring

TABLE III
CALCULATIONS OF TIME COMPLEXITIES OF SINGLE-DISTINGUISHER ATTACK.

N	IDC with Type I	Brute Force	Total
15679	$2^{25.785}$ to $2^{27.563}$ 24.3 s ($\sim 2^{28.510}$)	2^{40} to $2^{43.020}$ exceeded (?)	$2^{40.000}$ to $2^{43.020}$ exceeded (?)
31358	$2^{25.790}$ to $2^{28.462}$ 25.5 s ($\sim 2^{28.584}$)	2^{32} to $2^{42.831}$ 1587.1 s ($\sim 2^{34.542}$)	$2^{32.019}$ to $2^{42.831}$ 1612.6 s ($\sim 2^{34.565}$)
47037	$\leq 2^{29.011}$ 25.7 s ($\sim 2^{28.594}$)	2^{24} to $2^{42.830}$ 30.9 s ($\sim 2^{28.860}$)	$\leq 2^{42.830}$ 56.6 s ($\sim 2^{29.733}$)

TABLE IV
CALCULATIONS OF TIME COMPLEXITIES OF DOUBLE-DISTINGUISHER ATTACK.

N	IDC with Type I and II	Brute Force	Total
15679	$2^{26.785}$ to $2^{28.563}$ 48.7 s ($\sim 2^{29.515}$)	2^{32} to $2^{42.836}$ exceeded (?)	$2^{32.038}$ to $2^{42.837}$ exceeded (?)
31358	$2^{26.790}$ to $2^{29.462}$ 51.2 s ($\sim 2^{29.589}$)	2^{16} to $2^{42.830}$ 1.3 s ($\sim 2^{24.232}$)	$2^{26.791}$ to $2^{42.830}$ 52.5 s ($\sim 2^{29.624}$)
47037	$\leq 2^{30.011}$ 51.2 s ($\sim 2^{29.589}$)	$\leq 2^{42.830}$ neg (~ 0)	$\leq 2^{42.830}$ 51.2 s ($\sim 2^{29.587}$)

$2^{36.997}$ chosen plaintexts. Both attacks have been shown to be significantly faster than brute force. The double-distinguisher attack is also marginally faster than the single-distinguisher attack, as conducting IDC with the Type II distinguishers and brute forcing on the remaining 4 nibbles was faster than conducting brute force on the remaining 8 nibbles.

V. CONCLUSION

When implemented in practice, the single-distinguisher attack on 48-bit SmallCipher had an estimated time complexity of $2^{29.733}$ whereas the double-distinguisher attack had a time complexity of $2^{29.587}$, both requiring $2^{36.997}$ chosen plaintexts. This is a significant improvement over brute force.

The attacks presented can be generalised to a 96-bit SmallCipher, with bytes instead of nibbles. This presents a theoretical time complexity between $2^{64.001}$ and $2^{90.830}$ for the single-distinguisher attack, and between $2^{54.438}$ and $2^{90.830}$ for the double-distinguisher. Based on Fig. 4, the lower bound on time complexity of the double-distinguisher attack converges faster to the optimal time complexity than the single-distinguisher attack as N increases. It is likely that the double-distinguisher attack will be significantly faster than the single-distinguisher attack in practice, for smaller values of N.

The methods of searching for 6-round distinguishers presented in this paper and our newly discovered attack variants can also be extended to other substitution permutation networks such as AES. The possibility of finding tighter bounds for the time complexity of single and double-distinguisher attacks can be explored, so as to obtain a more accurate estimate of the time taken.

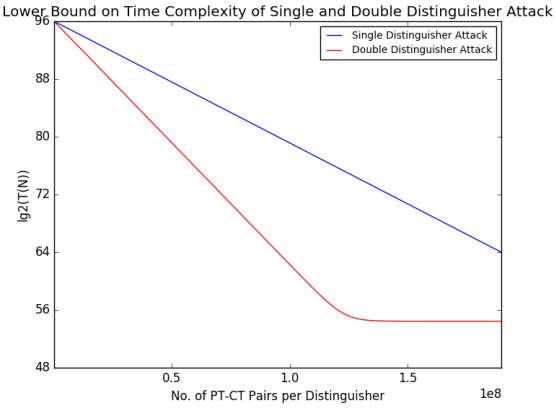

Fig. 4. Graph of lower bounds on time complexity of single and double-distinguisher attacks against 96-bit SmallCipher.

ACKNOWLEDGMENT

The authors would like to thank DSO National Laboratories for providing facilities to aid in the completion of this research, their mentor Mr Low Yu Bin for his invaluable support and guidance, and everyone else who has helped them in one way or another.

APPENDIX
DERIVATION OF THEORETICAL MODEL

A. Probability Considerations

Each MC operation is associated with a probability cost. For each column with k active nibbles, there must be at least $5-k$ active nibbles after it passes through MC and similarly for MC^{-1}.

For both Type I and Type II distinguishers, there are three and two active nibbles in the middle column of x_4^ω and x_5^ω respectively. As each nibble can take on 2^4 values, $p_4 \approx \frac{15^3}{16^4}$ and $p_5 \approx \frac{15^2}{16^4}$.

The two Type I distinguishers lead to the same differential at x_4^O. If $MC^{-1}(x_4^O)$ leads to either of the two differentials at x_4^ω, each with a probability of p_4, then the IDP is satisfied. Therefore, the probability of ω_5 leading to the IDP is $2p_4$.

B. IDC Time Complexity

Let $T_{1,n}$ be the time complexity of the nth round of filtering, measured in 6-round encryptions, and T_1 be the time complexity of IDC.

In the first round of filtering, the program performs a guess on the four active nibbles of partial k_6s by testing all $2^{4\cdot4}$ partial k_6s to determine if they lead to the differential at x_5^ω. Each guess requires a single one round decryption.

$2^{4\cdot4} \cdot p_5$ partial k_6s will lead to the required differential at x_5^ω. At x_5^ω, there will be two active nibbles. Key guessing will

be performed on $2^{4\cdot2} \cdot 2^{4\cdot4} \cdot p_5$ partial ω_5s to determine if they lead to the differential at x_4^ω.

Summing up the time complexity from both steps and taking each guess to have a time complexity of 1/6th of a 6-round encryption,

$$T_{1,1} = \frac{1}{6} \cdot 2^{4\cdot4}(1 + 2^{4\cdot4} \cdot p_5) \tag{1}$$

For the second round of filtering, there are $2^{4\cdot2} \cdot 2^{4\cdot4} \cdot p_5 \cdot (1 - 2p_4)$ partial (k_6, ω_5) pairs that did not lead to the differential at x_4^ω during the second round of filtering, and must be guessed.

However, for a partial k_6 to be eliminated, all partial (k_6, ω_5) pairs for $0 \leq \omega_5 < 2^{4\cdot2}$ must be eliminated. As it is difficult to estimate the number of partial k_6s left after each round, a lower and upper bound of the number of partial k_6s is calculated instead.

1) Lower Bound on IDC: As it takes at least $2^{4\cdot4}$ partial (k_6, ω_5) pairs eliminated to eliminate one partial k_6, there can never be more than one partial k_6 eliminated per $2^{4\cdot4}$ partial (k_6, ω_5) pairs eliminated.

From the first round of filtering, there are $2^{4\cdot4}(1-p_5)$ partial k_6s that did not decrypt to meet the differential at x_5^ω, and $2^{4\cdot4} \cdot p_5 \cdot (1 - 2p_4)$ partial k_6s that met the differential at x_5^ω but not x_4^ω. Summing the two together, a lower bound on the number of partial k_6s guessed is $2^{4\cdot4} \cdot (1 - 2p_4p_5)$.

Adding together the time complexity of guessing the remaining partial k_6s and the remaining partial (k_6, ω_5) pairs,

$$min(T_{1,2}) = \frac{1}{6} \cdot 2^{4\cdot4}(1 + 2^{4\cdot4}p_5)(1 - 2p_4p_5) \tag{2}$$

2) Upper Bound on IDC: Assume that no partial k_6 is eliminated until the end of key guessing, and there will always be $2^{4\cdot4}$ partial k_6s to guess while iterating the first round of filtering for each PT-CT pair. Therefore,

$$max(T_{1,2}) = \frac{1}{6} \cdot 2^{4\cdot4}(1 + 2^{4\cdot2} \cdot p_5 \cdot (1 - 2p_4p_5)) \tag{3}$$

However, this assumption only holds when the number of partial k_6s left after N rounds of filtering is at least 1, ie.

$$2^{4\cdot4}(1 - 2p_4p_5)^N \geq 1$$
$$\implies 0 \leq N \leq -(4 \cdot 4) \cdot \frac{\ln 2}{\ln(1 - 2p_4p_5)}$$

3) Overall Time Complexity on IDC: The overall time complexity $T_1 = \sum_{i=1}^{N} T_{1,i}$ for IDC is calculated by extending the above to N rounds of filtering, and described by these two equations:

$$min(T_1) = \frac{1}{6} \cdot 2^{4\cdot4} \cdot \left(\frac{1}{2p_4p_5} + \frac{2^{4\cdot4}}{2p_4}\right) \cdot \left(1 - (1 - 2p_4p_5)^N\right) \tag{4}$$

$$max(T_1) = \frac{1}{6} \cdot 2^{4\cdot4} \cdot \left(N + 1 + \frac{2^{4\cdot4}}{2p_4}\left(1 - (1 - 2p_4p_5)^N\right)\right) \tag{5}$$

C. Single-Distinguisher Attack

The single-distinguisher attack conducts IDC once, then conducts brute force on the remaining partial k_6s.

All possible values of the passive nibbles in x_6^O are iterated through and k_0 is derived from each k_6 by working backwards based on the key schedule. Deriving the keys and brute forcing incurs a cost of approximately one 6-round encryption.

Let T_2 be the time complexity of brute forcing the remaining keys.

1) Lower Bound on Brute Force: The filtering of k_6s based on the remaining (k_6, ω_5) pairs is assumed to pose no cost. The probability that a k_6 satisfies one of the remaining (k_6, ω_5) pairs is equivalent to the probability that a (k_6, ω_5) pair is not eliminated after going through N plaintext-ciphertext pairs, or $(12p_4p_5)^N$. Hence, the program only needs to perform encryptions using the remaining $2^{48}(12p_4p_5)^N$ k_6s.

$$min(T_2) = 2^{48}(1 - 2p_4p_5)^N \tag{6}$$

2) Upper Bound on Brute Force: The cost of deriving an ω_5 from a k_6 is approximately 1/12th of the time taken to perform a 6-round encryption. This is because deriving an ω_5 from a k_6 takes up 5 table lookups while performing a 6-round encryption takes up 72 table lookups.

Due to the properties of the distinguisher, only the middle column of ω_5 needs to be derived to determine if a k_6 has been eliminated. Hence, the cost of filtering a k_6 is $\frac{1}{12} \div 3 = \frac{1}{36}$ of the time taken to perform a 6-round encryption.

$$max(T_2) = 2^{48}\left(\frac{1}{36} + 2^{48}(1 - 2p_4p_5)^N\right) \tag{7}$$

3) Overall Time Complexity: The overall time complexity of the attack, $T = T_1 + T_2$.

$$min(T) = \frac{2^{14}}{3}\left(\frac{1}{2p_4p_5} + \frac{2^8}{p_4}\right)\left(1 - (1 - 2p_4p_5)^N\right) + 2^{48}(1 - 2p_4p_5)^N \tag{8}$$

$$max(T) = \frac{2^{15}}{3}\left(N + 1 + \frac{2^7}{p_4}(1 - (1 - 2p_4p_5)^N)\right) + 2^{48}\left(\frac{1}{36} + (1 - 2p_4p_5)^N\right) \tag{9}$$

$min(T)$ can be computed only for the bound $0 \leq N \leq -32 \cdot \frac{\ln 2}{\ln(1-2p_4p_5)}$.

D. Double-Distinguisher Attack

Each of the two distinguishers use $\frac{N}{2}$ PT-CT pairs, resulting in a total of N PT-CT pairs used.

IDC is performed on both distinguishers, causing the time complexity of key guessing to be doubled. Squaring the probability that a k_6 satisfies a remaining (k_6, ω_5) pair from each list, the probability that such a k_6 satisfies the condition for both lists obtained is $(1 - 2p_4p_5)^N$.

1) Lower Bound on Brute Force: The calculation is performed similarly to the single-distinguisher attack.

$$min(T_2) = 2^{48}(1 - 2p_4p_5)^N \tag{10}$$

2) Upper Bound on Brute Force: The middle column of ω_5 is derived for 2^{48} k_6s, incurring a cost of 1/36th of a 6-round encryption. After filtering based on the middle column, $2^{48}(12p_4p_5)^{N/2}$ k_6s remain. The last column of ω_5 is derived for each of these keys, leaving a final number of $2^{48}(12p_4p_5)^N$ k_6s to test.

$$max(T_2) = 2^{48}\left(\frac{1}{36} + \frac{1}{36}(1 - 2p_4p_5)^{N/2} + (1 - 2p_4p_5)^N\right) \tag{11}$$

3) Overall Time Complexity: The overall time complexity T is as follows.

$$min(T) = \frac{2^{15}}{3}\left(\frac{1}{p_4p_5} + \frac{2^8}{p_4}\right)\left(1 - (1 - 2p_4p_5)^{N/2}\right) + 2^{48}(1 - 2p_4p_5)^N \tag{12}$$

$$max(T) = \frac{2^{16}}{3}\left(\frac{N}{2} + 1 + \frac{2^7}{p_4}\left(1 - (1 - 2p_4p_5)^{N/2}\right)\right) + 2^{48}\left(\frac{1}{36} + \frac{1}{36}(1 - 2p_4p_5)^{N/2} + (1 - 2p_4p_5)^N\right) \tag{13}$$

REFERENCES

[1] N. F. Pub, "197: Advanced encryption standard (aes)," *Federal information processing standards publication*, vol. 197, no. 441, p. 0311, 2001.

[2] L. Knudsen, "Deal - a 128-bit block cipher," *complexity*, vol. 258, no. 2, p. 216, 1998.

[3] E. Biham, A. Biryukov, and A. Shamir, "Miss in the middle attacks on idea and khufu," in *International Workshop on Fast Software Encryption*. Springer, 1999, pp. 124–138.

[4] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner, and D. Whiting, "Improved cryptanalysis of rijndael," in *Fse*, vol. 1978. Springer, 2000, pp. 213–230.

[5] H. Gilbert and M. Minier, "A collision attack on 7 rounds of rijndael." in *AES Candidate Conference*, vol. 230, 2000, p. 241.

[6] H. Demirci, İ. Taşkın, M. Çoban, and A. Baysal, "Improved meet-in-the-middle attacks on aes," in *International Conference on Cryptology in India*. Springer, 2009, pp. 144–156.

[7] H. Mala, M. Dakhilalian, V. Rijmen, and M. Modarres-Hashemi, "Improved impossible differential cryptanalysis of 7-round aes-128," in *International Conference on Cryptology in India*. Springer, 2010, pp. 282–291.

[8] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer Science & Business Media, 2013.

[9] "Python 3.0 release." [Online]. Available: https://www.python.org/download/releases/3.0/

[10] "Gcc, the gnu compiler collection." [Online]. Available: https://gcc.gnu.org/

[11] "Virtual machine instances." [Online]. Available: https://cloud.google.com/compute/docs/instances/