Differential Cryptanalysis of the DES S1 Substitution Box

William A. Davis
Department of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
DavisW25@my.erau.edu

Laxima N. Kandel
Department of Computer Science
Embry-Riddle Aeronautical University
Daytona Beach, FL, USA
NiureKal@erau.edu

Abstract- Differential cryptanalysis is a classical chosen-plaintext attack technique that studies how specific input differences propagate through nonlinear components of a block cipher. In the Data Encryption Standard (DES), this behavior is governed primarily by the eight substitution boxes (S-boxes). This project implements and evaluates a differential analysis of the DES S1 S-box by measuring the output distribution resulting from fixed input differences. A Python script was developed to generate 20,000 random plaintext input pairs for two chosen differences, $\Delta X = 1$ and $\Delta X = 2$, and to compute the empirical output difference frequencies. The results show clear non-uniformity in S1's differential distribution. For $\Delta X = 1$, the most probable output differences occurred at $\Delta Y = 10$, 9, and 12, each with probabilities near 0.15-0.19. For $\Delta X = 2$, $\Delta Y = 12$ emerged as the dominant difference at approximately 0.18-0.19. These findings demonstrate how biased differential transitions can be exploited by an attacker to reduce key-search complexity in reduced-round variants of DES. The experiment successfully reproduces the statistical behavior expected from published DES differential characteristics and validates the Python simulation as an educational model of differential cryptanalysis.

Keywords- DES, differential cryptanalysis, S-box analysis, substitution boxes, block cipher security, differential characteristics, cryptanalysis techniques, Python implementation

I. INTRODUCTION

Differential cryptanalysis is one of the most influential attack methods used to study the internal behavior of block ciphers. The main idea is to examine how a specific input difference, written as ΔX, produces a corresponding output difference, written as ΔY, after passing through a nonlinear component. When certain output differences occur more often than others, the attacker can use these patterns to make informed guesses about the secret key or the structure of the cipher.

The Data Encryption Standard, or DES, is a well-known 16 round block cipher that relies heavily on eight substitution boxes, called S-boxes. Each S-box takes six input bits and outputs four bits using a fixed lookup table. Although these tables were designed to resist simple statistical attacks, they are not perfectly uniform. For many input differences, some output differences occur with noticeably higher probability. These biased transitions form differential characteristics that can be exploited in reduced round attacks and are central to understanding why DES is vulnerable to differential analysis.

This project focuses on the S1 S-box, which is the first nonlinear component processed in each DES round. The goal is to measure how often different output differences appear when applying specific input differences. A Python script was written to generate large sets of random input pairs, apply chosen input differences such as $\Delta X = 1$ and $\Delta X = 2$, and record the resulting output differences. By comparing the observed frequencies to known theoretical behavior, the experiment demonstrates the nonuniform structure of the S1 S-box and shows how these patterns are used in differential cryptanalysis.

II. BACKGROUND

Differential cryptanalysis is conceptually simple once the core idea is understood. The attack relies on observing how specific input differences propagate through nonlinear components of a cipher, such as S-boxes. In this project, the analysis is intentionally limited to the DES S1 S-box to provide a focused and reproducible demonstration of the technique. This section outlines the important practical considerations that make the experiment straightforward to implement using Python.

A. Selecting Input Differences

The first step in conducting a differential experiment is choosing which input differences to analyze. This project examines two fixed input differences, $\Delta X = 1$ and $\Delta X = 2$, because they are small binary differences that still produce meaningful output variations in the S1 box. These differences allow the experiment to capture a clear picture of how nonlinear components respond to controlled bit changes.

Once ΔX is selected, the experiment generates random plaintext values X and computes a paired value X' = X xor ΔX. These pairs are then passed through the S1 box, and the resulting output differences ΔY are recorded.

B. Automating the Experiment in Python

Python is well suited for this analysis because the S-box is a simple lookup table operation. The script used in this project generates 20,000 random input pairs for each chosen difference and applies the S1 transformation to measure the empirical frequency of each possible ΔY value.

The automation reduces the chance of human error and guarantees that the experiment is fast and repeatable. All results are produced in a matter of seconds, allowing multiple trials to be performed for consistency.

C. Interpreting Empirical Output Differences

Once the experiment is completed, the script counts how often each output difference occurs. This frequency distribution is the foundation of differential cryptanalysis. If the S1 box behaved like a perfectly random nonlinear function, all output differences would appear with similar probability. However, the empirical results show that certain ΔY values occur significantly more often.

For example, when $\Delta X = 1$, output differences $\Delta Y =$ 10, 9, and 12 consistently appear with probabilities close to 0.15 to 0.19. When $\Delta X = 2$, the most common output difference is $\Delta Y = 12$, with probability around 0.18 to 0.19. These biased transitions are exactly what a differential attacker exploits in reduced round versions of DES.

D. Practical Significance

Although the full DES cipher is resistant to basic differential attacks when all 16 rounds are applied, biased S-box behaviors are essential for understanding why differential cryptanalysis such a powerful technique in modern cryptography is. This project highlights that even a single S-box shows measurable statistical structure. The Python experiment provides an accessible way to observe these patterns without needing to implement the full DES round function.

III. METHODOLOGY

The goal of this project is to measure how specific input differences propagate through the DES S1 substitution box. The methodology consists of three major components: implementing the S1 lookup table, generating controlled input differences, and recording empirical output differences from large sets of randomized trials.

A. S1 Substitution Box Implementation

The DES S1 substitution box is defined as a fixed 4 by 16 lookup table. Each input is a 6-bit value, where the first and last bits determine the row, and the middle four bits select the column. The Python script used in this project stores this table as a nested list. For each input value, the script extracts the row and column indices, performs the table lookup, and returns the corresponding 4-bit output.

This direct table method ensures the implementation behaves identically to the official DES S1 box specification. Because the S-box is the only nonlinear component being analyzed, an exact lookup is essential to avoid deviations in the statistical results.

B. Generating Pair Inputs

To carry out the differential analysis, the script repeatedly generates random 6 bit plaintext inputs X. For each X, a second value X' is created using the chosen input difference, computed as X xor ΔX. Using the xor operation ensures that the pair (X, X') always differs exactly by the specified ΔX value.

Two input differences are explored in this project, $\Delta X = 1$ and $\Delta X = 2$. These values were selected because they create small but meaningful bit changes, allowing clear observation of how the S1 box responds to structured differences.

C. Computing Output Differences

Each input pair is passed through the S1 box to obtain the outputs Y and Y'. The corresponding output difference is computed as ΔY = Y xor Y'. Because Y and Y' are 4 bit values, ΔY ranges from 0 to 15.

Over many trials, certain output differences appear far more frequently than others. These imbalances form the differential characteristics that an attacker may exploit in reduced round versions of DES.

D. Trial Count and Repetition

For each value of ΔX, the experiment performs 20,000 trials. This number is large enough to reveal stable statistical patterns while keeping the total runtime very short. The script was executed multiple times for each input difference to confirm consistency between independent runs.

The repeated trials produced nearly identical frequency distributions in every execution, demonstrating that the observed differential characteristics are not artifacts of random variation but reflect genuine structure in the S1 substitution box.

IV. RESULTS

The differential experiments were performed on the DES S1 substitution box using two fixed input differences, $\Delta X = 1$ and $\Delta X = 2$. For each input difference, the Python script generated 20,000 random input pairs and recorded the resulting output differences. The purpose of this section is to present the observed differential distributions and highlight the most significant characteristics that emerged from the data.

A. Results for ΔX = 1

TABLE I. COMBINED DIFFERENTIAL RESULTS (ΔX = 1)

Table I summarizes the combined differential output probabilities for input difference ΔX = 1.

ΔY	Average Count	Average Probability
10	3697	0.184875
9	3118	0.155925
12	3101	0.155050
13	1925	0.096275
3	1873	0.093675
6	1294	0.064725
7	1265	0.063275
11	1218	0.060925
15	1202	0.060125
5	671	0.033575
14	631	0.031575

Two independent trials were conducted using ΔX = 1. In both runs, the distribution of output differences exhibited strong consistency and clear statistical bias. The most frequent output differences were ΔY = 10, ΔY = 9, and ΔY = 12, with probabilities ranging between approximately 0.15 and 0.19. These values appeared far more often than would be expected from a uniform random mapping, where each ΔY from 0 to 15 would ideally occur with probability 0.0625.

Less common transitions, such as ΔY = 5, ΔY = 6, and ΔY = 14, occurred with probabilities ranging from roughly 0.03 to 0.06. The reproducibility of these results across separate runs confirms that the S1 substitution box produces stable and predictable differential patterns when subjected to ΔX = 1.

B. Results for ΔX = 2

TABLE II. COMBINED DIFFERENTIAL RESULTS (ΔX = 2)

Table II summarizes the combined differential results for input difference ΔX = 2.

ΔY	Average Count	Average Probability
12	3718	0.185900
10	2506	0.125325
3	2455	0.122775
11	1918	0.095900
13	1887	0.094350
9	1877	0.093850
5	1253	0.062675
14	1252	0.062600
7	1246	0.062300
6	1240	0.062000
15	646	0.032325

The experiments for ΔX = 2 also showed clear statistical structure that was consistent across both runs. The dominant output difference was ΔY = 12, which appeared with probability close to 0.18 to 0.19 in both trials. Other output differences, including ΔY = 10, ΔY = 3, ΔY = 11, and ΔY = 9, appeared with mid-range probabilities between 0.09 and 0.13.

The remaining differences, such as ΔY = 5, ΔY = 7, ΔY = 14, and ΔY = 6, occurred at noticeably lower rates. The strong emphasis on ΔY = 12 in the ΔX = 2 experiment suggests that certain bit patterns within the S1 structure align more favorably with this input difference, producing highly biased differential transitions.

C. Comparison of Both Differential Inputs

Both ΔX = 1 and ΔX = 2 produce nonuniform ΔY distributions, demonstrating that the S1 substitution box is not a perfect nonlinear mapping. However, ΔX = 2 results in a more sharply peaked distribution, with ΔY = 12 showing especially high frequency. This highlights the importance of input selection when constructing differential characteristics. Certain input differences reveal more exploitable structure in the S-box than others, which is exactly how differential cryptanalysis forms multi-round attack paths.

D. Key-Space Reduction Demonstration

To illustrate how differential bias can be used in an attack scenario, a simplified key-space reduction experiment was implemented. In a single round of DES, each S-box receives a 6-bit value formed by XORing the expanded right-half state with a corresponding 6-bit subkey. Using the strong differential characteristic identified earlier (ΔX = 2 consistently producing ΔY = 12 with probability ≈ 0.18-0.19), the experiment evaluates which 6-bit subkey candidates remain consistent with this biased transition.

The Python code generates input pairs (X, X') such that X' = X ⊕ 2 and passes them through S1 using an unknown randomly chosen 6-bit subkey. Only pairs whose output difference matches the dominant differential (ΔY = 12) are collected. Then, for all 64 possible subkey guesses k, the code checks how many of those collected pairs satisfy $S1(X \oplus k) \oplus S1(X' \oplus k) = 12$.

When the candidate keys are ranked by how many differential pairs they explain, the correct subkey consistently appears among the highest-scoring values while most incorrect keys score significantly lower. In a representative run, the true key scored 80 out of 80 differential pairs, while almost all other candidates scored around 46 or below. Only one additional key tied with the true key due to the simplified, single-round nature of this demonstration. In a full multi-round DES attack, additional rounds and characteristics would break such ties, leaving the true key uniquely identifiable.

This experiment demonstrates the practical value of differential bias: it reduces the effective key space from 64 subkey possibilities to only a few strong candidates, illustrating the exact mechanism by which differential cryptanalysis narrows key search in real block-cipher attacks.

V. DISCUSSION

The results of this experiment confirm the expected nonuniform behavior of the DES S1 substitution box when subjected to controlled input differences. For both ΔX = 1 and ΔX = 2, several output differences appeared much more frequently than others, and these consistent statistical biases form the basis for a classical differential cryptanalysis strategy. In particular, the fact that ΔY = 12 occurred with the highest probability across multiple runs demonstrates that certain bit transitions inside S1 are far

more predictable than a perfectly uniform nonlinear function.

These observations highlight the structural weaknesses of early symmetric designs such as DES. Although DES remains secure against practical differential attacks when all 16 rounds are applied, the behavior of individual round components shows clear patterns that can be exploited in reduced-round scenarios. The experiment confirms that even with relatively small sample sizes, the differential profile of an S-box can be measured and reproduced with high reliability.

This analysis also reinforces the importance of modern cipher design principles, where S-boxes are constructed specifically to resist differential and linear attacks. Contemporary ciphers such as AES use algebraically derived S-boxes with significantly flatter differential distributions. The contrast between DES S1 and modern S-boxes illustrates how far block cipher design has evolved since DES was standardized.

Finally, the experiment demonstrates the educational value of analyzing a single S-box in isolation. By removing the complexity of full DES round functions, it becomes easier to observe the underlying mathematical behaviors that differential cryptanalysis exploits. The patterns in S1 provide a clear example of how predictable nonlinear components can leak structural information about a cipher.

VI. CONCLUSION

This project implemented a practical differential cryptanalysis study on the DES S1 substitution box using a controlled Python-based experiment. Two input differences, $\Delta X = 1$ and $\Delta X = 2$, were applied over 20,000 randomized input pairs to measure the empirical distribution of output differences. The results consistently revealed nonuniform behavior, with certain output differences such as $\Delta Y = 10$, 12, and 9 appearing far more frequently than would be expected from a perfectly random nonlinear function.

The observed statistical biases match the well-known differential characteristics published for DES and confirm the structural weaknesses that make the cipher vulnerable in reduced-round attacks. Even though full 16-round DES remains difficult to break using basic differential techniques alone, S-box-level leakage is a foundational component of broader cryptanalytic strategies.

Overall, this experiment demonstrates that meaningful cryptanalytic insight can be gained from studying a single substitution box in isolation. The Python implementation was simple, reproducible, and provided clear evidence of biased differential transitions. These findings reinforce the importance of designing nonlinear components that exhibit high diffusion and minimal differential predictability in modern symmetric cryptography.

REFERENCES

[1] E. Biham and A. Shamir, “Differential cryptanalysis of DES-like cryptosystems,” Journal of Cryptology, vol. 4, no. 1, pp. 3-72, 1991.

[2] National Institute of Standards and Technology (NIST), “Data Encryption Standard (DES),” FIPS Publication 46-3, 1999.

[3] W. Stallings, Cryptography and Network Security: Principles and Practice, 7th ed. Boston, MA, USA: Pearson, 2017.

[4] J. Daemen and V. Rijmen, The Design of Rijndael: AES - The Advanced Encryption Standard. New York, NY, USA: Springer, 2002.

[5] C. Paar and J. Pelzl, Understanding Cryptography: A Textbook for Students and Practitioners. Berlin, Germany: Springer, 2009.

[6] M. Matsui, “Linear cryptanalysis method for DES cipher,” Advances in Cryptology, EUROCRYPT ’93, Lecture Notes in Computer Science, vol. 765, pp. 386-397, 1994.

[7] Python Software Foundation, “Python 3.13 Documentation.” [Online]. Available: https://www.python.org/doc/

APPENDIX A

PYTHON IMPLEMENTATION FOR DIFFERENTIAL ANALYSIS OF DES S1 S-BOX

sbox_diff_demo.py

C: > Users > willi > Desktop > DES_Final_Project > sbox_diff_demo.py > ...

```python
import random

S1 = [
    [14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7],
    [0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8],
    [4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0],
    [15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]
]

def s1_box(x):
    b0 = x & 1
    b5 = (x >> 5) & 1
    row = (b5 << 1) | b0
    col = (x >> 1) & 0b1111
    return S1[row][col]

def run_diff(delta_in, num_trials):
    c = {}
    for _ in range(num_trials):
        x = random.randint(0,63)
        x2 = x ^ delta_in
        y1 = s1_box(x)
        y2 = s1_box(x2)
        d = y1 ^ y2
        c[d] = c.get(d,0) + 1
    return c

def key_filter(delta_in, target_delta_out, num_pairs=40):
    true_key = random.randint(0,63)
    pairs = []
    while len(pairs) < num_pairs:
        x1 = random.randint(0,63)
        x2 = x1 ^ delta_in
        y1 = s1_box(x1 ^ true_key)
        y2 = s1_box(x2 ^ true_key)
        if (y1 ^ y2) == target_delta_out:
            pairs.append((x1,x2))
    scores = [0]*64
    for x1,x2 in pairs:
        for k in range(64):
            y1 = s1_box(x1 ^ k)
            y2 = s1_box(x2 ^ k)
            if (y1 ^ y2) == target_delta_out:
                scores[k] += 1
    r = sorted(range(64), key=lambda k: scores[k], reverse=True)
    print("true key:", true_key, f"({true_key:06b})")
    print("top key candidates:")
    for k in r[:8]:
        print(k, f"({k:06b})", "score:", scores[k])
    print()

def main():
    delta_in = 0b000010
    num_trials = 20000
    c = run_diff(delta_in, num_trials)
    for d,count in sorted(c.items(), key=lambda p:p[1], reverse=True):
        print(d, count, count/num_trials)
    key_filter(delta_in, 12, num_pairs=80)


if __name__ == "__main__":
    main()
```