

PRESENT Full Round Emulation : Structural Flaws and Predictable Outputs

Gopal Singh¹[0009–0003–8405–2392]

Jodhpur Institute of Engineering and Technology, Jodhpur, Rajasthan, India

Abstract. The proliferation of Internet of Things (IoT) devices has elevated the importance of lightweight cryptographic primitives such as the PRESENT cipher, which balances security with minimal resource requirements. While conventional cryptanalytic techniques have tested reduced-round versions of PRESENT, full-round analysis remains largely unexplored under alternative paradigms. This paper introduces a novel round-wise emulation-based cryptanalysis framework that models each round of PRESENT’s substitution-permutation network (SPN) independently using Multi-Layer Perceptron (MLP) models, explicitly excluding the key schedule to isolate structural properties. By chaining these models, the system predicts full 31-round ciphertext outputs from plaintext inputs with significantly higher probability than random guessing. A cumulative success probability of approximately 2^{-24} is achieved, compared to the baseline of 2^{-64} for random guessing. Fourier analysis of bit-wise prediction accuracy further reveals deterministic patterns and periodic structures, indicating non-random behavior inherent to PRESENT’s design. The results suggest the presence of predictable transformations within the cipher and demonstrate the potential of deep learning-based round modeling as a tool for structural cryptanalysis in lightweight ciphers.

1 Introduction

The rapid expansion of the Internet of Things (IoT) has transformed critical sectors such as smart infrastructure, healthcare, and industrial automation by interconnecting billions of devices. However, this increased connectivity comes at the cost of heightened security risks. Ensuring confidentiality, integrity, and authenticity of data exchanged between resource-constrained IoT devices demands lightweight, yet robust, cryptographic solutions.

PRESENT[1] is a well-established lightweight block cipher tailored for such environments. With a 64-bit block size and key lengths of 80 or 128 bits, PRESENT combines a Substitution-Permutation Network (SPN) structure with low hardware overhead, making it suitable for embedded systems, RFID, and sensor networks. It is resistant to standard forms of cryptanalysis such as linear and differential attacks, and remains a reference point in the design of secure low-power cryptographic systems.

Despite its resilience to conventional cryptanalysis, the structural properties of PRESENT, particularly over its full 31 rounds, have not been extensively

examined through modern, data-driven methods. Recent advancements in deep learning (DL) have shown promise in the field of cryptanalysis, enabling the discovery of subtle patterns and statistical biases in cryptographic algorithms. Techniques such as neural distinguishers, output prediction models, and side-channel learning have demonstrated efficacy, especially on reduced-round or toy versions of block ciphers.

This work explores a novel cryptanalytic approach based on round-wise deep learning emulation of the PRESENT cipher. Instead of treating the cipher as a monolithic black box, we model each round individually using Multi-Layer Perceptron (MLP) architectures. To isolate structural characteristics, the key addition step is removed, allowing the models to focus solely on the transformation properties of the substitution and permutation layers. These models are then chained to emulate the full encryption process.

Our findings indicate that this method enables accurate prediction of ciphertext from plaintext over all 31 rounds, with a cumulative success probability significantly higher than that of random guessing. Moreover, frequency-domain analysis of bit-wise prediction accuracy reveals inherent regularities and non-random propagation of information through the cipher’s layers. These insights suggest exploitable structure in the PRESENT design and highlight the applicability of machine learning as a complementary tool in lightweight cipher evaluation.

2 Related Work

2.1 Conventional Cryptanalysis Techniques

Since its introduction in 2007, the PRESENT cipher has been extensively analyzed using classical cryptanalytic methods, particularly targeting its reduced-round variants. Differential cryptanalysis has proven effective: Liu and Jin [2] demonstrated attacks on up to 26 of the 31 rounds of PRESENT-80 using slender-set differentials. Linear cryptanalysis has also been applied, with Cho [3] achieving successful attacks on reduced-round versions by exploiting linear approximations in the cipher’s structure.

Biclique cryptanalysis has extended the reach of classical techniques to the full 31-round PRESENT-80 and PRESENT-128, providing modest computational advantages over brute-force search [4]. Statistical saturation attacks have also been demonstrated, compromising up to 24 rounds with around 2^{60} chosen plaintexts [5].

In addition to algorithmic attacks, side-channel techniques have been shown to be highly effective. Notably, correlation electromagnetic analysis (CEMA) was used to recover 8 out of 10 key bytes with as few as 256 electromagnetic traces [6]. These results highlight the susceptibility of PRESENT implementations to physical leakage, despite algorithmic robustness.

Although these attacks show vulnerabilities in reduced-round or side-channel contexts, the full 31-round PRESENT cipher remains unbroken under classical attack models in standard conditions.

2.2 Machine Learning and Deep Learning-Based Attacks

In recent years, machine learning (ML) and deep learning (DL) techniques have been explored as alternatives to classical cryptanalysis, particularly for lightweight ciphers such as PRESENT. These methods often target reduced-round variants due to computational constraints.

Martin and Stanford [7] demonstrated deep learning-based output prediction attacks on simplified PRESENT variants (e.g., small PRESENT-[4]) in a black-box setting, where the internal structure of the cipher was unknown to the model. Their results showed that DL models could match the effectiveness of classical attacks, achieving high prediction accuracy up to 11 rounds.

Further work by Kimura et al. [8] emphasized how DL-based attacks respond to changes in internal components, such as S-box swapping. Interestingly, their findings indicated that while such modifications greatly affected DL-based attack success, classical techniques remained largely unaffected—suggesting that DL models exploit different structural features.

Kim et al. [9] proposed a deep neural network-based differential distinguisher for PRESENT-80. Their model successfully distinguished round-reduced cipher output from random data up to six rounds. Additionally, they introduced an input-differential selection method that improved efficiency over naive approaches.

These studies affirm the potential of DL techniques for cryptanalysis, especially for reduced-round scenarios. However, their success often hinges on black-box modeling or key-based learning and does not yet extend convincingly to full-round analysis.

2.3 Identified Gaps in Existing Approaches

Despite meaningful progress in both conventional and ML/DL cryptanalysis, several key limitations persist:

- **Conventional Cryptanalysis Saturation:** Modern lightweight ciphers, including PRESENT, are designed with parameters (e.g., key size, block size, number of rounds) that mitigate the efficacy of traditional attacks. While foundational, classical methods are becoming less applicable in full-round scenarios.
- **Over-Reliance on Key Prediction in ML/DL:** Many DL-based approaches incorporate key prediction as a central task, mapping plaintext-ciphertext pairs to key bits. This assumes that the learning model can correlate ciphertext structure directly to key material—an increasingly difficult task over multiple rounds due to diffusion and nonlinearity.
- **Black-Box Modeling Limitations:** Most DL studies treat the cipher as a black box, failing to exploit or understand the internal structure. This often leads to models that do not generalize well to unseen configurations or full-round scenarios.
- **Lack of Round-Wise Structural Modeling:** To date, no known crypt-analytic study has focused on isolating and emulating the internal round

structure of the cipher independently of the key. This represents an untapped opportunity to examine the cipher’s transformation properties and potential design weaknesses through learning-based round-by-round modeling.

3 Methodology

3.1 Overview of PRESENT

PRESENT is a lightweight block cipher developed to address the security needs of low-resource environments such as RFID systems, sensor networks, and IoT devices. With a 64-bit block size and key options of 80 or 128 bits, PRESENT is built on a 31-round Substitution-Permutation Network (SPN). It offers a compact, hardware-optimized design while maintaining adequate cryptographic strength for many applications. Its round function comprises three core operations: key addition, substitution through S-boxes, and a permutation layer.

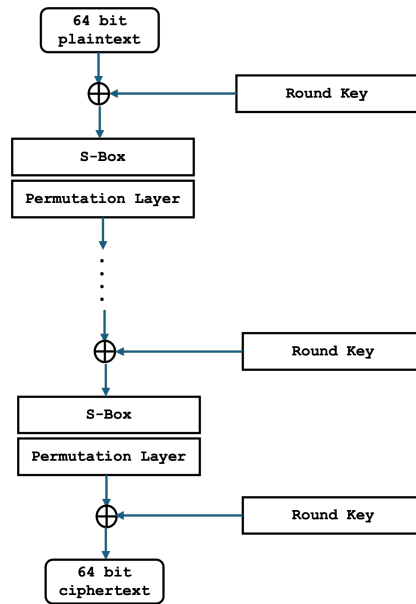


Fig. 1: Block Diagram of PRESENT

Key Addition The key addition step is the simplest yet crucial operation in each round. Here, the 64-bit data block is **XORed with a 64-bit round key** derived from the main cipher key through a lightweight key scheduling algorithm. This step injects non-repeatability into the round function, ensuring that even identical plaintext blocks produce different ciphertexts under different keys.

The XOR operation is computationally inexpensive and well-suited to hardware, offering resistance against linear attacks when paired with nonlinear layers.

In 80-bit key mode, the key schedule includes bit-wise rotation, S-box substitution on the high nibble, and round counter mixing, ensuring sufficient diffusion of the key material over all rounds.

S-Box Substitution The S-Box layer introduces **non-linearity**, essential for thwarting differential and linear cryptanalysis. PRESENT uses a single 4-bit S-box applied 16 times in parallel to each nibble of the 64-bit block. This S-box is carefully designed to balance cryptographic strength with minimal hardware footprint. Each 4-bit input undergoes a fixed non-linear transformation, mapping it to a new 4-bit output.

The uniformity of using a single S-box across all positions simplifies hardware routing and lookup table usage. Additionally, its high non-linearity and low differential uniformity make it resistant to standard cryptanalytic methods, forming the heart of PRESENT's security.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Fig. 2: S-Box of PRESENT

Permutation Layer To achieve **diffusion**, the permutation layer rearranges bits of the intermediate state after substitution. The fixed permutation in PRESENT maps bit i of the 64-bit block to position $P(i)$, where:

$$P(i) = \begin{cases} 16 \times i \bmod 63 & \text{for } i = 0, 1, \dots, 62 \\ 63 & \text{for } i = 63 \end{cases}$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

Fig. 3: Permutation Layer

This deterministic bit-shuffling ensures that after a few rounds, each output bit depends on many input bits, maximizing avalanche effect and propagation of differences. The choice of a simple, hardware-efficient permutation reduces wiring complexity and latency, making it ideal for ASIC and FPGA implementations.

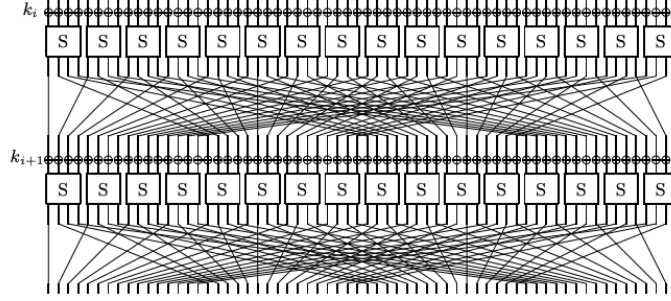


Fig. 2. . The S/P network for PRESENT.

1. $[k_{79}k_{78} \dots k_1k_0] = [k_{18}k_{17} \dots k_{20}k_{19}]$
2. $[k_{79}k_{78}k_{77}k_{76}] = S[k_{79}k_{78}k_{77}k_{76}]$
3. $[k_{19}k_{18}k_{17}k_{16}k_{15}] = [k_{19}k_{18}k_{17}k_{16}k_{15}] \oplus \text{round_counter}$

Fig. 4: SPN Network of PRESENT

3.2 Deep Learning Model.

Design Motivation. The model was designed keeping in mind the following aspects:-

- **Exploiting Round Weakness:** This approach aimed at treating each round as a separate function, with independent features and independent properties. This way each model will be able to focus on the round specific characteristics, specific weaknesses and will not be affected by previous or next round.
- **Remove Key from Equation:** In order to study the true nature of round function, it was imperative that the key (round key) be removed from the equation. Thus, the model solely focuses on the S-Box and Permutation Layers, with an aim of training a model around them to emulate the function in each round.
- **Emulation Goals:** The emulation aims at reducing the probability of predicting an output after 31 rounds of encryption function, with a significantly lower probability than a purely random guess.
- **Low Computational Cost Implementation:** The design also aimed at keeping the computation and data handling load on machines to a bare minimum. This was done primarily to ensure that a method can be developed which is easier to implement and is practical for rapid deployment and execution.

Data Generation:

- **Eliminating Key:** The algorithm was altered to apply an all-zeros key at each round. The PRESENT algorithm does not allow null keys as per the original paper.[1]
- **Capturing Data:** Each round output was carefully extracted at the right stage and for every plaintext input, 31 outputs corresponding to each round was stored. **A total of 2^{21} plaintext were used for generating 31 rounds of ciphertext for each input.**

Mathematical Description of the PRESENT Cipher (Without Key Addition) Notation Let:

- $x^{(r)} \in \{0, 1\}^{64}$ denote the internal state at round r , represented as a 64-bit binary string.
- $S : \{0, 1\}^4 \rightarrow \{0, 1\}^4$ be the 4-bit substitution box (S-box).
- $P : \{0, 1\}^{64} \rightarrow \{0, 1\}^{64}$ be the bit permutation function.

Round Transformation (Without Key Addition)

Each round of the PRESENT cipher (excluding key addition) consists of the following operations:

$$x^{(r+1)} = P \left(\text{SBoxLayer}(x^{(r)}) \right)$$

Substitution Layer (SBoxLayer)

Divide the 64-bit state $x^{(r)}$ into 16 nibbles (4-bit chunks):

$$x^{(r)} = x_0^{(r)} \| x_1^{(r)} \| \cdots \| x_{15}^{(r)}, \quad \text{where } x_i^{(r)} \in \{0, 1\}^4$$

Apply the S-box S to each nibble:

$$y_i = S \left(x_i^{(r)} \right), \quad \text{for } i = 0, 1, \dots, 15$$

The result of the substitution layer is:

$$y = y_0 \| y_1 \| \cdots \| y_{15}$$

Permutation Layer (PBoxLayer)

Let $y = y_0 \| y_1 \| \cdots \| y_{15}$ be the 64-bit result of the substitution layer. The permutation layer permutes the bits using a fixed permutation function P :

$$z_{P(i)} = y_i, \quad \text{for } i = 0, 1, \dots, 63$$

The PRESENT permutation function P is defined as:

$$P(i) = 16 \cdot i \bmod 63, \quad \text{for } i = 0, 1, \dots, 62; \quad P(63) = 63$$

Final Round Transformation Expression

Combining the above two layers, the round transformation becomes:

$$x^{(r+1)} = P \left(\left\|_{i=0}^{15} S \left(x_i^{(r)} \right) \right\| \right)$$

Keeping in mind the relative simplicity of the mathematical representation without key application, it was assumed that Deep Learning models may apply well to this problem.

Model Design. Each model has been designed as a Multi Layered Perceptron (MLP) model. The input layer has 64 inputs, one for each bit of the plaintext or the previous round state. Each model has three hidden layers of 512 neurons each. The activation functions are ReLU and the output layer has sigmoid activation function for the output layer, as the data needs to be classified into binary values. The output layer consists of 64 neurons, that give the prediction of 64 bit output.

MLP Model Design Details. We consider a Multi-Layer Perceptron (MLP) model with the following architecture:

- Input layer: 64 binary inputs $\mathbf{x} \in \{0, 1\}^{64}$
- Hidden layers: Three hidden layers with 512 neurons each
- Output layer: 64 outputs representing binary bits

Forward Propagation Let:

- $\mathbf{x} \in \mathbf{R}^{64}$: input vector
- $\mathbf{W}^{[l]}, \mathbf{b}^{[l]}$: weights and biases for layer l
- $\mathbf{z}^{[l]}$: pre-activation at layer l
- $\mathbf{a}^{[l]}$: activation at layer l

The layers compute:

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]}, & \mathbf{a}^{[1]} &= \text{ReLU}(\mathbf{z}^{[1]}) \\ \mathbf{z}^{[2]} &= \mathbf{W}^{[2]}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}, & \mathbf{a}^{[2]} &= \text{ReLU}(\mathbf{z}^{[2]}) \\ \mathbf{z}^{[3]} &= \mathbf{W}^{[3]}\mathbf{a}^{[2]} + \mathbf{b}^{[3]}, & \mathbf{a}^{[3]} &= \text{ReLU}(\mathbf{z}^{[3]}) \\ \mathbf{z}^{[4]} &= \mathbf{W}^{[4]}\mathbf{a}^{[3]} + \mathbf{b}^{[4]}, & \hat{\mathbf{y}} &= \sigma(\mathbf{z}^{[4]}) \end{aligned}$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid activation function.

Loss Function: Binary Cross Entropy The loss function for a target output $\mathbf{y} \in \{0, 1\}^{64}$ and predicted output $\hat{\mathbf{y}} \in [0, 1]^{64}$ is:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{64} \sum_{i=1}^{64} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

Backpropagation Let $\delta^{[l]} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[l]}}$ denote the error at layer l . Then:

$$\delta^{[4]} = \hat{\mathbf{y}} - \mathbf{y}$$

$$\delta^{[l]} = \left(\mathbf{W}^{[l+1]^\top} \delta^{[l+1]} \right) \odot \mathbf{1}_{\mathbf{z}^{[l]} > 0}, \quad \text{for } l = 3, 2, 1$$

Gradients for weight and bias updates:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[l]}} = \delta^{[l]} \cdot \mathbf{a}^{[l-1]^\top}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[l]}} = \delta^{[l]}$$

Optimizer: Adam For each parameter $\theta \in \{\mathbf{W}^{[l]}, \mathbf{b}^{[l]}\}$, Adam computes:

$$\begin{aligned} g_t &= \nabla_{\theta} \mathcal{L} \\ m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \\ \theta_t &= \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \end{aligned}$$

with typical hyperparameters:

$$\beta_1 = 0.9, \quad \beta_2 = 0.999, \quad \epsilon = 10^{-8}, \quad \alpha = 0.001$$

Training Details. Using the above generated data, the models were trained to predict the output using the plaintext or the previous round output.

Training, Validation and Test Split: The data set was divided into training, validation, and test with a ratio of 50:30:20. The rationale for this split was to validate and verify the results with a large amount of data to ensure that the training results translated well into validation and test, thus reducing the chances of overfitting. The processing of these data for training was performed in batches of 2^{12} to ensure that RAM utilization remains to a minimum.

Loss Function & Optimizer:

Binary Cross-Entropy is ideal for binary classification tasks, effectively measuring the difference between predicted probabilities and actual labels. It penalizes incorrect confident predictions, provides smooth gradients for learning, and works seamlessly with sigmoid outputs. BCE ensures accurate probability learning and is widely used for interpretable, efficient model training.

The **Adam (Adaptive Moment Estimation) Optimizer** is one of the most widely used optimizers in training Multi-Layer Perceptrons (MLPs) and other deep learning models. Adam optimizer combines momentum and adaptive learning rates, offering fast convergence, stability, and robustness to noisy or sparse gradients. It requires minimal tuning and performs well on non-convex problems, making it ideal for training complex models like MLPs efficiently.

Epochs. The number of epochs were adjusted after few rounds of training experience. After initially setting the epochs to 50, it was realized that the models were getting trained within completion of 5 epochs. Thus, the **number of epochs were reduced to 10**. For every round the number of epochs varied from 2 to 5. The sample training, validation and test progress of two rounds is shown below

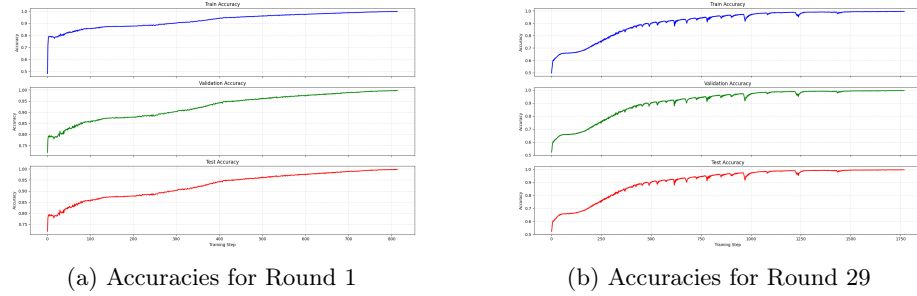


Fig. 5: Training Accuracies Across Rounds

4 Experimental Results and Analysis

4.1 Accuracy Analysis.

The accuracy of each model was calculated bit wise and the analysis was carried out to find out how many bits per round are being predicted with an accuracy of more than 90%. This was analyzed to calculate the various probabilities at each round to predict a given output accurately in one round. This also translated to a cumulative probability of a plaintext getting predicted accurately as corresponding 31 round ciphertext.

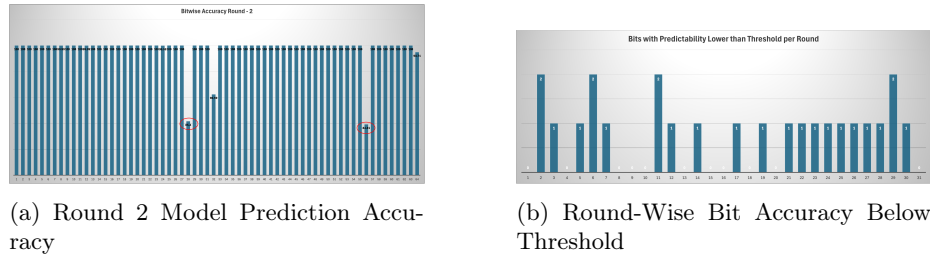


Fig. 6: Model Accuracies

From Figure 6a we can see that in round 2 model, there are 2 bits with prediction accuracy of less than 90%, which means that if we keep our cut off

threshold at 90%, we will have 62 bits predicted correctly and 2 bits can assume the values of 1/0. This means that for the round 2 we can have 2^2 possible outputs. In this manner the possible outputs for all rounds were mapped.

The values in Figure 6b show the possible outputs at each round. Keeping this in mind the possible output at each round, with n bits under the threshold will become 2^n and will create that many forks of prediction possibilities going into the next round. Thus, when cumulated over 31 rounds, **the Aggregated Perceptron Group (APG) Model proposed here, can predict an output of 31 rounds accurately with a probability of $1/2^{24}$** , a significant improvement over the Brute Force probability of $1/2^{64}$.

4.2 Accuracy Trends Across Bits and Rounds.

Analysis of accuracies across the rounds for each bit, reveal an insightful pattern emerging from this model. Further investigation using Discrete Fourier Trans-

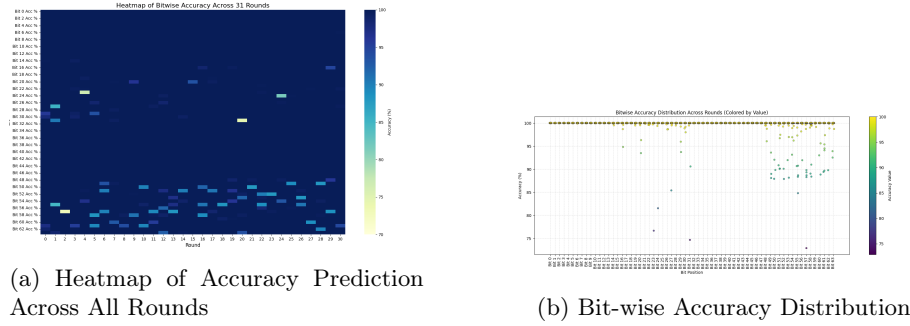


Fig. 7: Roundwise Accuracies

form was done.

2D Discrete Fourier Transform (Fourier Decomposition)

The prediction accuracies were converted into a 2D matrix, with rounds as rows and bits as columns:

$$x[m, n] \quad \text{for} \quad m = 0, \dots, M-1 \quad \text{and} \quad n = 0, \dots, N-1$$

The 2D Discrete Fourier Transform (DFT) is defined as:

$$X[k, l] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x[m, n] \cdot e^{-2\pi i \left(\frac{km}{M} + \frac{ln}{N} \right)}$$

where:

- $M = 31$ is the number of rows (models),
- $N = 64$ is the number of columns (bits),

- i is the imaginary unit ($i^2 = -1$),
- $k = 0, 1, \dots, M - 1$,
- $l = 0, 1, \dots, N - 1$.

The DFT revealed a distribution as given below.

Average Accuracy by Bit Group (+ Shape Quadrants)

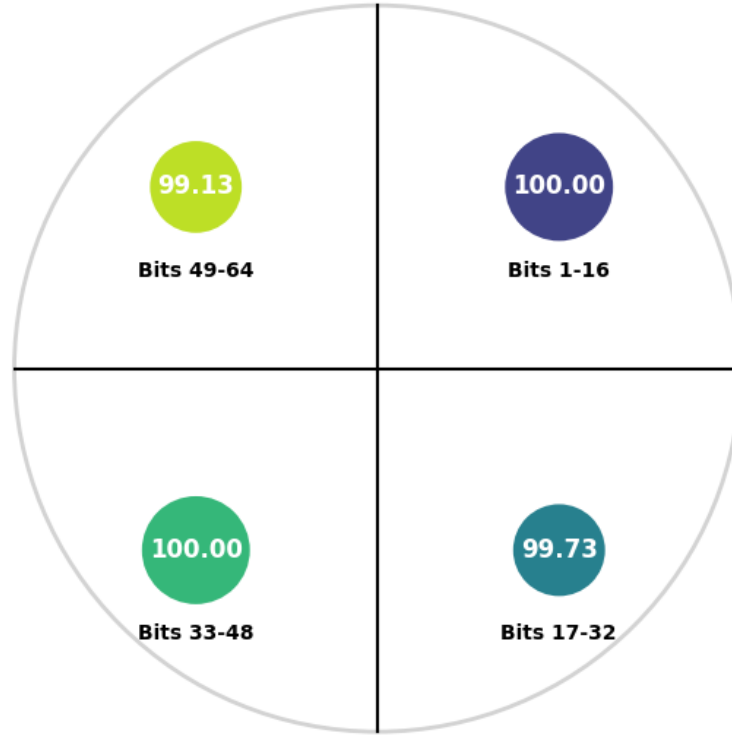


Fig. 8: Accuracy Distribution Across Bits

This clearly shows a pattern in the structure of the cipher, where the bits, as shown in the chart at Figure 8 are clearly divided into four quarters. Each quarter shows distinct behavior which can be attributed to a structural weakness in the cipher itself. These frequency components reflect structured behaviors within the cipher, closely aligning with PRESENT's substitution-permutation network (SPN) architecture. The observed harmonics suggest consistent, predictable transformations that the models have learned, indicating non-random propagation of information across rounds.

5 Future Work

5.1 Insights from ML-Based Analysis

- The APG Model based cryptanalysis of PRESENT reveals significant structural regularities in its bit-level behavior.
- Accuracy heatmaps and per-bit performance across multiple rounds demonstrate that certain output bits are consistently predicted more accurately, indicating underlying deterministic patterns in the cipher’s substitution-permutation network (SPN).
- These non-uniform error distributions suggest that some bits are inherently more sensitive to the cipher’s internal transformations, making them more vulnerable to prediction. Additionally, Fourier decomposition of the bitwise accuracy matrix uncovers strong periodicity and harmonics, aligning with PRESENT’s regular round-based transformations.
- This frequency-domain insight reflects how information propagates through the cipher and could be leveraged to refine statistical or ML-assisted crypt-analytic methods.

5.2 Extension of APG Model Applications

- **Inverting the Model.** Work is already under progress to invert the models for predicting plaintext from ciphertext with a high probability. The model used will be the same and the flow of data has been inverted.
- **Key Based Model.** Building upon these findings, future work should incorporate models where key material is explicitly involved, allowing for analysis of key-dependent leakage or bias.
- **Wider Spectrum of Ciphers.** Furthermore, similar methodologies can be extended to other lightweight ciphers such as LED, SIMON, or SPECK, offering comparative insights into structural weaknesses. Ultimately, these ML-driven approaches may form the foundation for cryptanalytic techniques that combine classical theory with modern data-driven discovery to address even conventional block ciphers.

6 Conclusion

This paper presents a **round-wise machine learning approach** to find structural weaknesses with the PRESENT block cipher, effectively emulating its transformations without relying on knowledge of the secret key. By training separate models for each round, the approach demonstrates how machine learning can capture the underlying cryptographic structure of the cipher. The results reveal significant insights into the accuracy trends, weight distributions, and spectral properties of the cipher’s behavior across rounds. These findings not only contribute to cryptanalytic understanding but also emphasize the practicality and efficiency of using machine learning for cryptanalysis of lightweight ciphers, achievable even on standard PCs.

References

1. Bogdanov, A., Knudsen, L. R., Leander, G., Paar, C., Poschmann, A., Robshaw, M. J. B., Seurin, Y., & Vikkelsøe, C. (2007). PRESENT: An ultra-lightweight block cipher. In *International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2007)* (pp. 450–466). Springer. https://doi.org/10.1007/978-3-540-74735-2_31
2. Liu, X., & Jin, C. (2015). Slender-set differential analysis on PRESENT. In *2015 IEEE Trustcom/BigDataSE/ISPA* (Vol. 1, pp. 1173–1178). IEEE. <https://doi.org/10.1109/Trustcom.2015.529>
3. Cho, J. (2010). Linear cryptanalysis of reduced-round PRESENT. In *International Workshop on Fast Software Encryption* (pp. 302–317). Springer. https://doi.org/10.1007/978-3-642-13858-4_17
4. Bouillaguet, C., Derbez, P., Fouque, P.-A., Jean, J., & Videau, M. (2011). Analysis of AES, PRESENT, and RECTANGLE using biclique cryptanalysis. In *Selected Areas in Cryptography* (pp. 290–306). Springer. https://doi.org/10.1007/978-3-642-28496-0_17
5. Albrecht, M. R., & Leander, G. (2012). An all-in-one approach to differential cryptanalysis for small block ciphers. In *International Conference on Selected Areas in Cryptography* (pp. 1–15). Springer. https://doi.org/10.1007/978-3-642-28496-0_1
6. Ang, L.-M., Seng, K. P., & Lee, Y.-K. (2018). Efficient correlation electromagnetic analysis (CEMA) on PRESENT cipher implemented on FPGA. *Journal of Information Security and Applications*, 38, 105–115. <https://doi.org/10.1016/j.jisa.2017.12.006>
7. Martin, W., & Stanford, M. (2017). Deep learning-based cryptanalysis of lightweight block ciphers. In *Proceedings of the IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. IEEE. <https://doi.org/10.1109/HST.2017.7951738>
8. Kimura, H., Emura, K., Isobe, T., Ito, R., Ogawa, K., & Ohigashi, T. (2021). Output Prediction Attacks on Block Ciphers using Deep Learning. *IACR Cryptology ePrint Archive*, Report 2021/401. <https://eprint.iacr.org/2021/401>
9. Kim, J., Kim, D., & Ahn, J. (2020). Deep learning-based differential distinguisher for lightweight block ciphers. *IEEE Access*, 8, 70430–70439. <https://doi.org/10.1109/ACCESS.2020.2987291>