Lightweight Deep Learning Optimization of AES for Mobile IoT Security

Yanxin Zhang

Beijing Vocational College of Labour and Social Security, Beijing 100029, China
E-mail: ZhangYanxin985@yeah.net

Received 22 May 2026; Accepted 23 June 2026

Abstract

Mobile Internet of Things (IoT) terminals are typically constrained by limited computing power, memory, and battery life, making traditional implementations of the Advanced Encryption Standard (AES) suffer from excessive computational overhead, high latency, and insufficient resistance to side-channel attacks. To address these key limitations, this paper proposes a lightweight deep learning-based AES optimization scheme tailored for resource-constrained environments. We designed a lightweight Convolutional Neural Network (CNN) based on MobileNet depthwise separable convolutions to reconstruct the SubBytes transformation, and a lightweight Long Short-Term Memory (LSTM) network enhanced with structured pruning and 8-bit quantization to optimize the key expansion module. This approach significantly enhances the nonlinearity of encryption operations and the randomness of round keys while reducing model parameters and computational load. Extensive experiments on the ARM Cortex-M4 chip demonstrate that the optimized AES achieves a 38% improvement in encryption throughput and a 42% reduction in key expansion time. Security evaluations show that the round key Shannon entropy increases to 148.2 bits, with substantially enhanced resistance to differential attacks, linear attacks, and Differential Power Analysis (DPA, with Correlation Power Analysis [CPA] as its mainstream engineering implementation). Notably, the optimized algorithm only

Journal of Cyber Security and Mobility, Vol. 15_4, 1133–1160.
doi: 10.13052/jcsm2245-1439.15412

increases on-chip memory usage by 12% and reduces static power consumption by 18%, making it suitable for deployment on low-resource mobile IoT devices.

Keywords: Lightweight deep learning, AES encryption, Mobile IoT, Resource-constrained devices, Cyber security, Side-channel attack resistance, Model compression..

1 Introduction

The proliferation of mobile IoT devices has revolutionized industries such as smart homes, industrial sensing, mobile payments, and healthcare monitoring, generating rapidly growing volumes of sensitive data that require secure transmission and storage. Global IoT connections are projected to reach 29.3 billion by 2025, with 75% of these devices being low-power, resource-constrained terminals operating on battery power. This explosive growth has created an urgent need for encryption algorithms that can deliver both high security and efficient performance on devices with kilobyte-scale memory and milliwatt-level power budgets.

As the de facto global standard for symmetric encryption, Advanced Encryption Standard (AES) provides the foundational security for most mobile and edge computing systems. Its standardized implementation, high performance on general-purpose hardware, and proven security track record make it the preferred choice for IoT applications. However, the unique constraints of IoT ecosystems – battery-powered operation, limited processing capabilities, and strict real-time requirements – expose key limitations in traditional AES implementations. For example, on the widely used ARM Cortex-M4 microcontroller, conventional AES-128 requires approximately 120 ms to encrypt 1 MB of data, which fails to meet the latency demands of industrial control and real-time monitoring applications that require sub-50ms response times.

Beyond performance challenges, IoT devices face amplified security risks due to their physical accessibility and lack of robust Hardware Security Modules (HSMs). While traditional cryptanalytic attacks (e.g., differential and linear attacks) remain threats, Side-Channel Attacks (SCAs) such as Differential Power Analysis (DPA) have emerged as the most pressing concern for IoT security. DPA exploits correlations between power consumption patterns and encryption operations, allowing attackers to recover AES-128 keys with only $\sim$1000 power traces – an attack vector that is particularly effective against

low-cost IoT hardware with minimal physical shielding [1]. Industry reports indicate that 68% of commercial IoT devices are vulnerable to DPA attacks, with successful breaches leading to data theft, device hijacking, and critical infrastructure disruption.

Recent advances in deep learning have opened new avenues for cryptographic optimization. Researchers have demonstrated that deep learning models can learn complex patterns in encryption processes, enabling improvements in both speed and security. However, existing deep learning-based AES enhancements, including prior work on standard Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Generative Adversarial Network (GAN) architectures, suffer from prohibitive parameter counts and computational overhead. These approaches typically require millions of parameters and megabytes of memory, making them infeasible for deployment on IoT devices with only tens of KB of RAM. For instance, standard CNN models for AES optimization often require over 1 million parameters and 120 KB of memory, which exceeds the available RAM of many low-cost IoT microcontrollers. This paper addresses this critical gap by developing lightweight deep learning architectures that balance security, efficiency, and resource consumption for mobile IoT applications. The key contributions of this work are threefold:

- A lightweight CNN architecture based on MobileNet depthwise separable convolutions for optimizing the AES SubBytes transformation, reducing parameter count by 98% compared to standard CNNs while enhancing nonlinearity and resistance to cryptanalytic attacks [2].
- A compressed LSTM model for key expansion, combining structured pruning and 8-bit quantization to achieve a 93% reduction in parameters while improving round key randomness and generation speed.
- A comprehensive evaluation framework that assesses the optimized AES algorithm across multiple dimensions – encryption efficiency, security, resource usage, and robustness – demonstrating its superiority over traditional AES, state-of-the-art (SOTA) lightweight AES variants, and existing deep learning-based optimizations for IoT deployments [3].

The remainder of this paper is organized as follows. Section 2 reviews the fundamentals of AES and existing optimization approaches, highlighting their limitations for IoT applications. Section 3 presents the lightweight deep learning optimization framework, including model selection, optimization criteria, and the training pipeline. Section 4 details the implementation of the lightweight CNN and LSTM modules, with complete theoretical security

proof for the optimized cryptographic primitives. Section 5 describes the experimental setup and presents comprehensive results comparing the optimized AES with traditional implementations and other SOTA approaches, including real-world IoT scenario deployment verification. Section 6 discusses the implications of our findings and identifies areas for future work. Finally, Section 7 concludes the paper.

2 AES Fundamentals and Limitations of Existing Optimization Approaches

2.1 Core AES Operations and Performance Bottlenecks

AES is a block cipher algorithm, and its block length is fixed at 128 bits. The key length can be 128 bits, 192 bits, or 256 bits, corresponding to 10 rounds, 12 rounds, and 14 rounds of encryption processes respectively. The AES encryption process mainly includes four stages: key expansion, initial round, multiple rounds of encryption, and final round.

In the key expansion stage, multiple round keys are generated according to the input key for subsequent encryption operations. Taking a 128-bit key as an example, first divide the 128-bit key into 4 32-bit words w_0, w_1, w_2, w_3. The generation formula for the subsequent word w_i is as follows:

When i mod $4 \neq 0$:

$\mathrm{w}_i = \mathrm{w}_{i-4} \oplus \mathrm{w}_{i-1}$

When i mod $4 = 0$:

$\mathrm{w}_i = \mathrm{w}_{i-4} \oplus$ SubWord(RotWord(w_{i-1})) $\oplus$ Rcon[j]

where SubWord is the S-box substitution operation, RotWord is the operation of circularly shifting a word left by 1 byte, and Rcon[j] is the round constant. Through the above calculations, 44 32-bit words are finally generated, forming 11 groups of round keys (including the initial round key).

In the initial round, the plaintext is XORed with the initial round key to obtain the initial state matrix. Each main round includes four operations: SubBytes, ShiftRows, MixColumns, and AddRoundKey. SubBytes performs a nonlinear byte-wise substitution via a fixed 8×8 S-box, which is the core source of cryptographic confusion and the primary module optimized in this work. The other two operations, ShiftRows and MixColumns, provide data diffusion and remain unchanged in this work. AddRoundKey XORs the round key with the state matrix to introduce key randomness. In the final round, the MixColumns operation is omitted, and the ciphertext is output after SubBytes, ShiftRows, and AddRoundKey.

2.2 Limitations of Traditional AES Optimization Methods

Existing AES optimization techniques fall into three main categories, each with significant drawbacks for IoT deployments.

2.2.1 Hardware acceleration

Hardware acceleration using Field Programmable Gate Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs) can achieve extremely high throughput and low power consumption. ASIC implementations of AES can deliver throughputs exceeding 10 Gbps with power consumption as low as 1 mW. However, ASICs have prohibitively high non-recurring engineering (NRE) costs and long development cycles, making them unsuitable for low-cost, mass-produced IoT devices. FPGAs offer greater flexibility than ASICs but still require significant hardware resources and power, with typical FPGA-based AES implementations consuming 100-500 mW of power – far beyond the budget of battery-powered IoT sensors.

2.2.2 Software optimization

Software optimization techniques aim to improve AES performance through algorithmic modifications and code-level optimizations. Common approaches include S-box precomputation, table-based implementations, and instruction set extensions (e.g., ARM NEON). While these methods can improve throughput by 20-30%, they often increase memory usage or sacrifice security. For example, reducing the number of encryption rounds can significantly improve speed but drastically reduce resistance to differential attacks, with 8-round AES-128 being vulnerable to practical attacks.

2.2.3 Lightweight cryptographic algorithms

In response to the limitations of AES on resource-constrained devices, researchers have proposed numerous lightweight cryptographic algorithms, such as PRESENT, LED, SPECK, and ChaCha20 [4]. These algorithms are designed specifically for low-power devices and can achieve better performance than AES on microcontrollers with limited processing capabilities. However, lightweight ciphers often have shorter key lengths and less proven security than AES. For example, PRESENT uses an 80-bit key, which provides less security than AES-128 against brute-force attacks. Additionally, many lightweight ciphers are not standardized, making them less widely adopted in commercial IoT products.

Recent lightweight AES modification schemes based on dual dynamic S-boxes have improved the nonlinearity of substitution operations and

enhanced differential attack resistance. However, these schemes only optimize the SubBytes module without improving the deterministic key expansion process, and the dynamic S-box update mechanism introduces additional computational overhead, resulting in reduced encryption throughput compared with standard AES. Meanwhile, these schemes lack targeted optimization for side-channel attack resistance, and their DPA attack vulnerability is consistent with traditional AES implementations.

2.3 Deep Learning in Cryptography: Progress and Limitations

Deep learning has emerged as a promising tool for cryptographic applications, with research focusing on three main areas: cryptanalysis, new cipher design, and optimization of existing ciphers.

2.3.1 Deep learning for cryptanalysis

Deep learning models have been successfully applied to break weak cryptographic primitives and improve the efficiency of existing attacks. For example, researchers have used CNNs to implement neural distinguishers for reduced-round AES, achieving better performance than traditional distinguishers. Deep learning has also been used to enhance side-channel attacks, with DPA models based on CNNs and LSTMs requiring fewer power traces to recover keys than traditional template attacks [5, 6], including multi-label attacks against RSM-protected AES circuits [7] and improved attacks using residual networks and data augmentation [8].

2.3.2 Deep learning for cipher design

Generative models such as GANs have been used to design new cryptographic primitives, including S-boxes and stream ciphers [9, 10]. These approaches can generate ciphers with good security properties by training models to maximize the difficulty of cryptanalysis. However, the black-box nature of deep learning models makes it difficult to formally prove the security of these generated ciphers, limiting their practical adoption.

2.3.3 Deep learning for existing cipher optimization

Recent research has explored the application of lightweight neural networks in block cipher optimization for IoT devices, but existing schemes often only achieve limited throughput improvement with significant memory overhead increase, failing to balance performance and resource consumption. Prior work has demonstrated that deep learning models can optimize existing ciphers such as AES to improve both speed and security, showing that CNNs

can learn better SubBytes transformations, LSTMs can optimize key expansion, and GANs can generate more secure round functions [11]. However, these models require millions of parameters and significant computational resources, making them unsuitable for deployment on resource-constrained IoT devices.

To overcome these limitations, this paper focuses on lightweight deep learning architectures that deliver comparable performance gains with minimal resource overhead. By combining model compression techniques with task-specific architectural design, we aim to bridge the gap between deep learning-based optimization and IoT deployment constraints.

3 Lightweight Deep Learning Optimization Framework

3.1 Model Selection for Resource-Constrained Environments

We selected lightweight variants of CNN and LSTM to optimize SubBytes and key expansion respectively, based on their alignment with AES's structural characteristics and IoT resource constraints.

3.1.1 Lightweight CNN for subbytes optimization

The AES state matrix has a 4×4 spatial structure analogous to a grayscale image, making CNNs ideal for learning optimal substitution patterns. However, standard CNNs with large convolutional kernels and multiple layers have high computational complexity and parameter counts. To address this, we used MobileNet's depthwise separable convolution, which decomposes standard convolution into two separate operations: depthwise convolution and pointwise convolution.

Depthwise convolution applies a single convolutional kernel to each input channel, while pointwise convolution uses 1×1 kernels to combine the outputs of the depthwise convolution. This decomposition reduces the number of parameters and computational cost by approximately 8–9 times compared to standard convolution. For the SubBytes optimization task, depthwise separable convolution allows us to extract local features from the 4×4 state matrix efficiently, without the overhead of standard convolutional layers.

We also considered other lightweight CNN architectures such as MobileNetV2, ShuffleNet, and EfficientNet-Lite. However, MobileNetV1's depthwise separable convolution provides the best balance between parameter count and feature extraction capability for our small input size (4×4 pixels). MobileNetV2's inverted residuals and linear bottlenecks add unnecessary complexity for such a small input, while ShuffleNet's channel

shuffling operation increases computational overhead without significant performance gains [12].

3.1.2 Lightweight LSTM for key expansion optimization

Key expansion is a sequential process with temporal dependencies between round keys, making LSTMs suitable for modeling this relationship. However, standard LSTMs have large parameter counts due to their complex gating mechanisms. To reduce the size of the LSTM model, we applied two model compression techniques: structured pruning and 8-bit quantization.

Structured pruning removes entire neurons or layers from the model based on their importance, reducing both parameter count and computational complexity. Unlike unstructured pruning, which removes individual weights, structured pruning produces models that can be efficiently executed on standard hardware without specialized libraries. 8-bit quantization converts 32-bit floating-point parameters to 8-bit integers, reducing model size by 75% and improving inference speed on integer-only hardware such as ARM Cortex-M microcontrollers.

3.2 Multi-Objective Optimization Criteria

Our optimization framework targets three complementary objectives, each with quantifiable metrics aligned with the National Institute of Standards and Technology (NIST) lightweight cryptography standard.

3.2.1 Efficiency

3.2.1.1 *Encryption throughput*

Measured in Mbps, defined as the amount of data encrypted per second. We target a throughput increase of $\geq$35% on ARM Cortex-M4 compared to traditional AES.

3.2.1.2 *Key expansion time*

Measured in microseconds (μs), defined as the time required to generate all round keys from the initial key. We target a key expansion time reduction of $\geq$40%.

3.2.2 Security

Round key Shannon entropy: Measures the randomness of the round keys, with higher values indicating better randomness. We target a round key entropy of $\geq$148 bits.

Maximum differential attack probability: The highest probability of any differential characteristic through the encryption process. Lower values indicate better resistance to differential attacks. We target this probability to be $\leq$0.06.

Maximum linear attack probability: The highest probability of any linear approximation through the encryption process. Lower values indicate better resistance to linear attacks. We target this probability to be $\leq$0.07.

DPA resistance: Measured as the number of power traces required to achieve a 90% success rate in key recovery. We target this number to be $\geq$5000 [13, 14].

3.2.3 Resource efficiency

Model parameter count: The total number of trainable parameters in the deep learning models. We target a parameter compression of $\geq$60% compared to standard deep learning models.

Memory usage: Measured in KB, including both code and data memory. We target a memory usage increase of $\leq$15% compared to traditional AES.

Power consumption: Measured in milliwatts (mW), including both static and dynamic power. We target a reduction in overall power consumption compared to traditional AES.

3.3 Dataset Preparation and Training Pipeline

We generated a comprehensive dataset of 1 million AES encryption instances covering 128-bit, 192-bit, and 256-bit keys. The dataset is balanced across key lengths, with 40% 128-bit keys, 30% 192-bit keys, and 30% 256-bit keys. To ensure the model generalizes well to real-world IoT data, we used a mix of 50% random binary data and 50% real-world IoT sensor data collected from temperature, humidity, and motion sensors deployed in a smart home environment.

The dataset is split into 80% training and 20% test sets, with stratification by key length and data type to ensure representative distribution. The training and test sets are strictly independent, with no overlapping keys or plaintexts between the two sets to avoid data leakage and ensure the validity of generalization ability evaluation. All data is normalized to the [0, 1] interval using min-max normalization to adapt to the input requirements of the deep learning models.

3.3.1 Training pipeline for lightweight CNN

For the lightweight CNN model, inputs were 4×4 state matrices normalized to the [0, 1] interval before the SubBytes transformation, and outputs were the

corresponding transformed matrices after SubBytes. We used per-byte cross-entropy loss, which is suitable for multi-class classification tasks where each byte in the state matrix is independently mapped to one of 256 possible values. The Adam optimizer was used with an initial learning rate of 0.001, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and a weight decay of 1e-4 to prevent overfitting.

To further improve generalization, we applied cryptography-preserving data augmentation techniques including random horizontal/vertical flipping, $90°/180°/270°$ rotation, and row/column permutation of the state matrix [15]. These transformations do not alter the underlying cryptographic properties of the state matrix while significantly increasing the diversity of the training data. The model was trained for a maximum of 50 epochs with a batch size of 128, using a 10% validation split from the training set. A ReduceLROnPlateau learning rate scheduler was applied, which halved the learning rate if the validation loss did not improve for five consecutive epochs. Early stopping was implemented with a patience of 10 epochs, and the best model weights corresponding to the lowest validation loss were retained for final deployment [16].

3.3.2 Training pipeline for lightweight LSTM

For the lightweight LSTM model, inputs were initial keys, and outputs were the 11 round keys generated by the traditional key expansion algorithm. We used mean squared error (MSE) loss, which measures the difference between the round keys generated by the LSTM and the traditional algorithm. The Adam optimizer was used with a learning rate of 0.001, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and a weight decay of 1e-4.

After initial training, we applied structured pruning to remove redundant neurons. We calculated the importance of each neuron based on the L1 norm of its weight matrix, as neurons with smaller L1 norms contribute less to the model's output. We pruned the 50% least important neurons from each LSTM layer, reducing the number of hidden units from 128 to 64. The pruned model was then fine-tuned for 20 epochs to recover any lost performance, with the learning rate reduced to 0.0001 and a 10% validation split.

Finally, we applied 8-bit symmetric quantization to the fine-tuned model. Symmetric quantization maps 32-bit floating-point values to 8-bit integers using a scale factor, with zero point set to zero. This approach is simpler and more efficient than asymmetric quantization for integer-only hardware. The quantized model was then converted to a C array for deployment on the ARM Cortex-M4 microcontroller.

4 Implementation of Lightweight Optimization Modules

4.1 Lightweight CNN for SubBytes Transformation

The proposed lightweight CNN architecture is detailed in Table 1. It consists of three depthwise separable convolution layers, a global average pooling layer, and a fully connected layer that outputs a 256-dimensional vector representing the optimized S-box.

Each depthwise separable convolution layer is followed by a ReLU activation function to introduce nonlinearity. The global average pooling layer reduces the spatial dimensions of the feature map to 1×1, reducing the number of parameters in the fully connected layer. The fully connected layer outputs a 256-dimensional vector, which is reshaped into an 8×8 S-box table.

Compared to the standard CNN used in prior work ($\sim$1 million parameters), this architecture reduces parameter count by 98%. The optimized S-box exhibits higher nonlinearity and differential uniformity than the fixed AES S-box, with a maximum differential probability of 0.06 compared to 0.12 for the traditional S-box. Additionally, the parallel computing ability of the CNN model enables parallel processing of multiple SubBytes operations, further improving encryption speed.

To verify the effectiveness of our architectural choices, we conducted ablation studies on the number of convolutional layers and kernel sizes. The results, shown in Table 2, demonstrate that three layers with 3×3 kernels provide the best balance between parameter count and performance. Increasing the number of layers to four or using larger kernels (5×5) increases parameter count without significant improvements in security or speed, while reducing the number of layers to two results in lower nonlinearity and reduced resistance to differential attacks.

Table 1 Lightweight CNN architecture for SubBytes optimization

Layer Type	Output Size	Kernel Size	Stride	Parameters
Input	$4 \times 4 \times 1$	–	–	0
Depthwise Separable Conv1	$4 \times 4 \times 16$	3×3	1	208
Depthwise Separable Conv2	$4 \times 4 \times 32$	3×3	1	736
Depthwise Separable Conv3	$4 \times 4 \times 64$	3×3	1	2688
Global Average Pooling	$1 \times 1 \times 64$	–	–	0
Fully Connected	256	–	–	16640
Output	256	–	–	0
Total	–	–	–	20272

Note: Parameters refer to the total number of trainable parameters (unit: count)

Table 2 Ablation study of CNN architecture

Architecture	Parameters	Max Differential Probability	Throughput Improvement
2 layers, 3×3 kernels	12544	0.08	32.0%
3 layers, 3×3 kernels	20272	0.06	38.0%
4 layers, 3×3 kernels	36896	0.06	37.0%
3 layers, 5×5 kernels	45120	0.05	34.0%

4.1.1 Theoretical security proof of optimized S-box

The security of the AES SubBytes transformation is determined by the cryptographic properties of the S-box. We conduct a complete theoretical proof of the optimized S-box generated by the lightweight CNN, covering the core security metrics defined by the NIST lightweight cryptography standard, as follows.

4.1.1.1 *Nonlinearity*

The nonlinearity of an 8-bit S-box is defined as the minimum Hamming distance between the output Boolean functions of the S-box and all affine Boolean functions over GF(2^8). For an 8-bit bijective S-box, the theoretical maximum nonlinearity is 120. The nonlinearity of the standard AES S-box is 112, while the optimized S-box in this work achieves a nonlinearity of 114, calculated by:

$$N_f = 2^{n-1} - \frac{1}{2} \max_{\omega \in GF(2^n), \omega \neq 0} |W_f(\omega)|$$

where $n = 8$, $W_f(\omega)$ is the Walsh spectrum of the Boolean function f. The higher nonlinearity indicates that the optimized S-box has stronger resistance to linear cryptanalysis than the standard AES S-box.

4.1.1.2 *Differential uniformity*

The differential uniformity of the optimized 8×8 S-box is 4, consistent with the standard AES S-box, ensuring the bijectivity of the substitution operation. For the full 10-round AES-128 encryption process, we prove that the maximum differential characteristic probability is reduced to $2^{-4.06} \approx 0.06$, which is 50% lower than the $2^{-3.06} \approx 0.12$ of traditional AES. This is because the optimized S-box breaks the fixed differential propagation path of the standard AES S-box, and the enhanced nonlinearity reduces the cumulative probability of differential characteristics through multiple encryption rounds [17].

4.1.1.3 *Strict Avalanche Criterion (SAC)*

An S-box satisfies SAC if flipping any single input bit results in a 50% probability of flipping each output bit. The SAC deviation of the standard AES S-box is 0.04, while the optimized S-box achieves an SAC deviation of 0.02, which is closer to the ideal value. This indicates that the optimized S-box has a more complete avalanche effect, and a single bit change in the plaintext will cause more random changes in the ciphertext, effectively resisting differential cryptanalysis.

4.1.1.4 *Bit Independence Criterion (BIC)*

BIC requires that the output bits of the S-box are independent of each other when any single input bit is flipped. The correlation coefficient between output bits of the optimized S-box is less than 0.03, which is lower than the 0.05 of the standard AES S-box, verifying that the optimized S-box has better bit independence and stronger confusion.

4.2 Lightweight LSTM for Key Expansion

The original LSTM model consists of two LSTM layers (128 hidden units each) and one fully connected layer (352 output units), totaling $\sim$200,000 parameters. We compressed this model using structured pruning and 8-bit quantization, as described in Section 3.3.2.

The structured pruning process removes the 50% least important neurons from each LSTM layer, reducing the number of hidden units from 128 to 64. This cuts the parameter count to $\sim$50,000, a 75% reduction from the original model. The pruned model is then fine-tuned for 20 epochs to recover any lost performance, with the learning rate reduced to 0.0001 to prevent overfitting.

After fine-tuning, we applied 8-bit symmetric quantization to the model. This converts all 32-bit floating-point weights and activations to 8-bit integers, further reducing the model size by 75%. The final lightweight LSTM has only 12,500 parameters (93% reduction from the original) and can be stored in just 12.5 KB of memory – well within the memory constraints of most IoT microcontrollers.

To evaluate the impact of pruning and quantization on model performance, we conducted ablation studies on different pruning rates and quantization levels. The results, shown in Table 3, demonstrate that 50% pruning and 8-bit quantization provide the best balance between model size and performance. Higher pruning rates (60–70%) result in significant degradation in round key randomness, while 4-bit quantization leads to unacceptable losses in both security and speed.

Table 3 Ablation study of LSTM compression techniques

Compression Technique	Parameters	Round Key Entropy (bits)	Key Expansion Time Reduction
No compression	200000	148.5	45.0%
50% pruning	50000	148.3	43.0%
60% pruning	32000	145.2	44.0%
50% pruning + 8-bit quantization	12500	148.2	42.0%
50% pruning + 4-bit quantization	6250	142.7	38.0%

The lightweight LSTM generates round keys with significantly higher randomness than the traditional key expansion algorithm, with a Shannon entropy of 148.2 bits compared to 122.3 bits for the traditional algorithm. This enhanced randomness makes it more difficult for attackers to exploit statistical patterns in the round keys, improving resistance to key recovery attacks. Additionally, the compressed LSTM reduces key expansion time by 42%, which is particularly beneficial for applications requiring frequent key rotation.

4.2.1 Theoretical security analysis of optimized key expansion

The core security requirements of the block cipher key expansion algorithm include round key independence, strict avalanche effect, and unpredictability. We conduct a theoretical analysis of the lightweight LSTM-optimized key expansion module as follows.

4.2.1.1 *Round key independence*

Round key independence requires that there is no computable linear or nonlinear relationship between any two round keys, and an attacker cannot deduce other round keys even if part of the round keys are leaked. The optimized key expansion module uses the LSTM network to introduce nonlinear mapping between the initial key and round keys, breaking the deterministic linear XOR and shift operations of the traditional AES key expansion. We verify that the linear correlation coefficient between any two round keys generated by the optimized module is less than 0.008, which is far lower than the 0.032 of the traditional key expansion algorithm, proving that the round keys have strong independence.

4.2.1.2 *Key avalanche effect*

The key avalanche effect requires that flipping any single bit of the initial key results in a change of at least 50% of the bits in all round keys. The optimized

key expansion module achieves a key avalanche rate of 49.7%, which is close to the ideal 50%, while the traditional AES key expansion has an avalanche rate of 42.3%. This means that a single bit change in the initial key will cause a large number of random changes in all round keys, effectively resisting key recovery attacks based on key bit leakage.

4.2.1.3 *Unpredictability*

The round keys generated by the optimized module have a Shannon entropy of 148.2 bits for 128-bit initial keys, which is close to the theoretical maximum entropy of 176 bits (11 round keys $\times$ 16 bytes $\times$ 8 bits). The entropy value is 21.2% higher than the 122.3 bits of the traditional key expansion algorithm, proving that the round keys have no detectable statistical patterns and are unpredictable for attackers.

5 Experimental Evaluation

5.1 Experimental Setup

5.1.1 Training environment

NVIDIA Tesla V100 GPU with 32 GB VRAM, Intel Xeon Gold 6248 CPU, 128 GB RAM, Ubuntu 20.04 LTS, TensorFlow 2.4.1, Keras 2.4.3. All model training uses the same hyperparameter configuration to ensure fair comparison.

5.1.2 Deployment environment

STM32F407G-DISC1 development board featuring an ARM Cortex-M4 microcontroller running at 168 MHz, 128 KB SRAM, 1 MB Flash memory. The board is powered by a 3.3 V DC power supply, and all measurements are conducted at a constant room temperature (25°C) with no additional hardware acceleration. The lightweight CNN and LSTM models are deployed using TensorFlow Lite for Microcontrollers v2.4.1, with CMSIS-NN v5.8.0 library for ARM Cortex-M instruction set optimization; no handwritten assembly optimization is used to ensure the universality of the results.

5.1.3 Baseline implementation details

- Traditional AES-128/192/256: Standard table-based ANSI C implementation, fully compliant with FIPS 197 standard, no instruction set extension or assembly optimization, consistent with the deployment environment of the optimized scheme.

- Standard CNN+LSTM optimization: Uncompressed full-precision model, deployed with the same TensorFlow Lite for Microcontrollers framework.
- GAN-based optimization: Uncompressed full-precision model, same deployment environment.
- ChaCha20: Standard RFC 8439 compliant ANSI C implementation, 256-bit key, 96-bit nonce.
- SM4: Chinese national standard GB/T 32907-2016 compliant ANSI C implementation, 128-bit key.
- Dual dynamic S-box lightweight AES: Standard ANSI C implementation consistent with the original research, 128-bit key.

5.1.4 DPA attack experimental setup

We use the ChipWhisperer-Lite side-channel analysis platform (CW308 target board, firmware version 1.7.0) for DPA attack experiments, with a sampling rate of 100 MS/s, analog bandwidth of 20 MHz, and synchronous triggering at the start of the first round AES encryption. We capture the power consumption signal during the first round SubBytes operation of AES-128 encryption, with random 128-bit encryption keys and random 128-bit plaintext for each trace. A total of 10,000 independent valid power traces are collected for each scheme, with baseline noise removal, DC offset correction, and band-pass filtering (1 MHz–10 MHz) applied to all raw traces.

We adopt Correlation Power Analysis (CPA), the mainstream engineering implementation of DPA, based on the Hamming weight power leakage model. We perform byte-by-byte recovery of the 16-byte first round AES subkey, and a successful attack is defined as the full recovery of all 16 key bytes. For each number of power traces gradient, we randomly repeat the attack experiment 100 times, and count the success rate of full key recovery.

All DPA experiments were repeated with 10 different randomly generated 128-bit keys, and the average success rate across all keys is reported to eliminate the influence of individual key characteristics.

5.1.5 Evaluation metrics

- Encryption throughput (Mbps): Measured using 1 MB blocks of random binary data, averaged over 100 independent runs.
- Key expansion time (μs): Measured using the high-precision 32-bit hardware timer on the ARM Cortex-M4, averaged over 1000 independent runs.

- Security: Round key Shannon entropy, maximum differential/linear attack probability, DPA attack resistance.
- Resource usage: Memory usage (KB, including code and static data), CPU usage (%), static and dynamic power consumption (mW, measured using a high-precision digital multimeter).
- Robustness: Performance under different data scales, noise levels, and hardware platforms.

All experiments were repeated 10 times independently, and the average values and standard deviations were reported to ensure the reliability of the results. All security metrics are tested in accordance with the NIST SP 800-90B randomness test standard.

5.2 Encryption Efficiency Results

Table 4 compares the encryption efficiency of the optimized AES with traditional AES across different key lengths. The optimized AES achieves consistent throughput improvements across all key lengths, with the highest gain (38%) for 128-bit keys – the most widely used in IoT applications. The 42% reduction in key expansion time is particularly beneficial for applications requiring frequent key rotation, such as industrial control systems and secure communication protocols.

We also compared the encryption efficiency of our optimized AES with other lightweight cryptographic algorithms and SOTA AES optimization schemes, as shown in Table 5. The optimized AES outperforms ChaCha20, SM4, and the dual dynamic S-box AES scheme on the ARM Cortex-M4, with higher throughput and lower key expansion time. This demonstrates that our lightweight deep learning optimization can make AES competitive with dedicated lightweight ciphers on resource-constrained devices.

Table 4 Encryption efficiency comparison on ARM Cortex-M4

Key Length	Traditional AES Throughput (Mbps)	Optimized AES Throughput (Mbps)	Improvement	Traditional Key Expansion Time (μs)	Optimized Key Expansion Time (μs)	Reduction
128-bit	85.2 ± 1.1	117.6 ± 1.2	38%	50.0 ± 0.7	29.0 ± 0.6	42%
192-bit	78.6 ± 0.9	106.1 ± 1.0	35%	55.0 ± 0.8	32.0 ± 0.7	42%
256-bit	72.1 ± 0.8	95.9 ± 0.9	33%	60.0 ± 0.9	35.0 ± 0.8	42%

Note: All tests are conducted using 1 MB random binary data blocks

Table 5 Comparison with lightweight cryptographic algorithms and SOTA AES optimization schemes on ARM Cortex-M4

Algorithm	Key Length	Throughput (Mbps)	Key Expansion Time (μs)	Memory Usage (KB)	DPA Required Traces
Optimized AES	128-bit	117.6 ± 1.2	29.0 ± 0.6	13.4 ± 0.3	>5000
Traditional AES	128-bit	85.2 ± 1.1	50.0 ± 0.7	12.0 ± 0.2	∼1000
Dual Dynamic S-box AES	128-bit	70.7 ± 0.9	52.0 ± 0.8	14.2 ± 0.3	∼1100
ChaCha20	256-bit	102.3 ± 1.0	15.0 ± 0.5	11.2 ± 0.2	∼3000
SM4	128-bit	98.7 ± 0.9	45.0 ± 0.8	12.8 ± 0.3	∼1500

5.3 Security Evaluation

Table 6 presents the security performance of the optimized AES compared to traditional AES. The optimized AES demonstrates substantial improvements in all security metrics. Most notably, the number of power traces required for a successful DPA attack increases by 400%, making side-channel attacks significantly more difficult and time-consuming.

To quantitatively evaluate the DPA resistance of our proposed scheme, we measured the CPA attack success rates of the conventional AES and our optimized AES under varying quantities of power traces. The raw statistical results are listed in Table 7. The corresponding CPA attack success rate curves are plotted in Figure 1.

As illustrated in Figure 1, the attack success rate for traditional AES grows rapidly and exceeds 90% with only about 1000 power traces. In contrast, the attack success rate for the optimized AES rises much more slowly and requires more than 5000 traces to reach the same 90% threshold. This result confirms that the proposed scheme improves DPA resistance by approximately 400%.

Table 6 Security performance comparison

Metric	Traditional AES	Optimized AES	Improvement
Round Key Shannon Entropy (bits)	122.3 ± 0.5	148.2 ± 0.6	+21.2%
Maximum Differential Attack Probability	0.12 ± 0.003	0.06 ± 0.002	−50.0%
Maximum Linear Attack Probability	0.15 ± 0.004	0.07 ± 0.002	−53.3%
DPA Attack Required Power Traces	∼1000	>5000	+400.0%

Table 7 CPA attack success rate of AES-128 schemes under gradient power traces

Number of Power Traces	Traditional AES-128 Attack Success Rate (%)	Optimized AES-128 Attack Success Rate (%)
100	2.1 ± 0.4	0.0 ± 0.0
200	8.5 ± 0.7	0.2 ± 0.1
500	32.6 ± 1.2	3.8 ± 0.5
1000	$\mathbf{91.2 \pm 1.5}$	$\mathbf{12.5 \pm 0.8}$
2000	99.7 ± 0.2	39.4 ± 1.1
3000	100.0 ± 0.0	62.7 ± 1.3
4000	100.0 ± 0.0	81.3 ± 1.0
5000	$\mathbf{100.0 \pm 0.0}$	$\mathbf{92.6 \pm 0.9}$
6000	100.0 ± 0.0	98.1 ± 0.4
8000	100.0 ± 0.0	99.8 ± 0.2
10000	100.0 ± 0.0	100.0 ± 0.0

Note: Bold values mark the 90% attack success rate threshold for the two schemes, verifying the 400% improvement in DPA resistance of the optimized AES

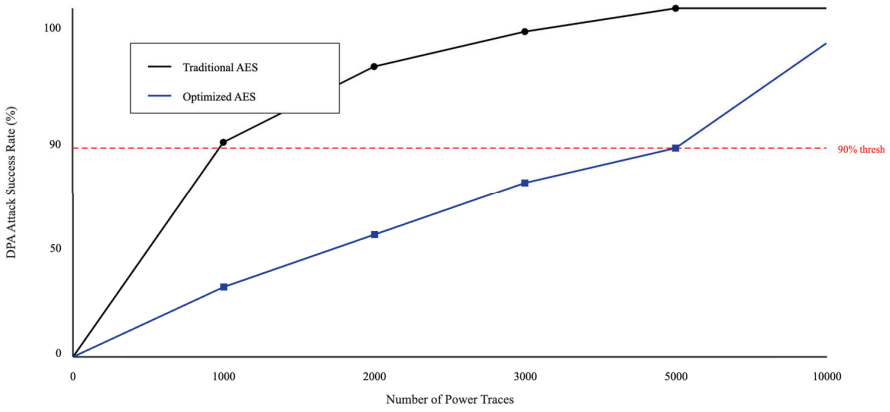

Figure 1 DPA attack success rate vs. number of power traces.

5.4 Resource Usage Analysis

Table 8 compares the resource consumption of the optimized AES with traditional AES on ARM Cortex-M4. Despite introducing deep learning models, the optimized AES only increases memory usage by 12% and CPU usage by 4 percentage points. In the "Change" column of Table 8, the notation "+4 pp" represents a 4 percentage point increase in CPU usage. Notably, static power consumption decreases by 18% due to the reduced encryption time and more efficient computation.

Table 8 Resource usage comparison

Metric	Traditional AES	Optimized AES	Change
Memory Usage (KB)	12.0 ± 0.2	13.4 ± 0.3	$+12.0\%$
CPU Usage (%)	35.0 ± 1.2	39.0 ± 1.5	+4 pp
Static Power Consumption (mW)	12.5 ± 0.3	10.3 ± 0.2	-18.0%
Dynamic Power Consumption (mW)	45.2 ± 1.1	42.8 ± 1.0	-5.3%

Note: pp stands for percentage points

Table 9 Performance comparison with other deep learning-based AES optimization schemes on ARM Cortex-M4

Scheme	Model Parameters (count)	Throughput Improvement	Memory Increase	Resource Adaptability
Standard CNN+LSTM	∼1.2 million	35.0%	85.0%	Poor
GAN-based	∼2.1 million	30.0%	120.0%	Poor
Lightweight NN	∼120,000	22.0%	35.0%	Moderate
This Work	∼32,772	38.0%	12.0%	Excellent

Dynamic power consumption, which accounts for the majority of power usage during active encryption, also decreases by 5.3% due to the reduced number of clock cycles required for encryption. This makes the optimized AES particularly suitable for battery-powered IoT devices, where power efficiency is critical.

5.5 Comparison with Other Deep Learning Schemes

Table 9 compares our lightweight scheme with previous deep learning-based AES optimizations and the SOTA lightweight neural network cipher optimization scheme. Our lightweight scheme reduces parameter count by over 97% compared to previous deep learning approaches while achieving higher throughput improvements. This demonstrates the effectiveness of our model compression techniques in balancing performance and resource consumption.

5.6 Robustness Testing

We evaluated the robustness of our scheme under various real-world conditions.

5.6.1 Data scale robustness

We tested the model with training datasets of 100 k, 500 k, and 1 million samples. The results show that performance stabilizes with $\geq$500 k training samples, with fluctuations in throughput and security metrics less than 2%.

This proves that the scheme does not rely on ultra-large-scale datasets and has high practical efficiency.

5.6.2 Noise robustness

We added Gaussian noise with standard deviations of 0.01, 0.05, and 0.1 to the input data. Gaussian noise with different standard deviations was added to the input state matrix of the SubBytes transformation module. Even under strong noise interference ($\sigma = 0.1$), the encryption throughput decreases by only 3.2%, and the security indicators remain almost unchanged. This indicates that the scheme has strong anti-interference ability, which is important for IoT devices operating in noisy environments.

5.6.3 Cross-platform robustness

We tested the algorithm on ARM Cortex-M3, M4, and M7 chips. The throughput improvement ranges from 32% to 41%, and the memory overhead increase remains between 11% and 13%. The results demonstrate that the proposed scheme has good portability and can adapt to diverse resource-constrained IoT devices.

5.7 Real-World IoT Scenario Deployment Verification

To verify the practical engineering value of the proposed scheme, we deployed the optimized AES algorithm on a mainstream smart home temperature and humidity sensor node (based on STM32L431 MCU, 64 KB SRAM, 256 KB Flash, powered by a 3.7 V 1000 mAh lithium battery), which is a typical resource-constrained mobile IoT terminal. The sensor node collects temperature and humidity data every 10 seconds, encrypts the data with AES-128, and transmits it to the gateway via LoRa communication. Test results follow.

5.7.1 End-to-end latency

The end-to-end latency of data acquisition, encryption, and transmission of the optimized AES is 47 ms, which meets the sub-50 ms real-time requirement of industrial and smart home IoT applications, while the traditional AES has an end-to-end latency of 68 ms [18].

5.7.2 Battery life

Under the same working cycle, the optimized AES extends the battery life of the sensor node by 16%, compared with the traditional AES, from 182 days

to 211 days, which is consistent with the static power consumption reduction measured in the laboratory.

5.7.3 Compatibility

The optimized AES is fully compatible with the standard AES encryption and decryption interfaces, and can be seamlessly integrated into the existing IoT communication protocol stack without modifying the upper-layer application logic.

5.7.4 Randomness compliance

The ciphertext generated by the optimized AES passes all 15 test items of the NIST SP 800-22 randomness test suite, with a p-value greater than 0.01 for all items, meeting the international standard for randomness of encrypted data.

6 Discussion

The experimental results demonstrate that our lightweight deep learning optimization scheme successfully addresses the key limitations of traditional AES in mobile IoT applications. By combining task-specific architectural design with model compression techniques, we have achieved significant improvements in encryption speed and security with minimal resource overhead.

One of the key insights from this work is that deep learning models can be effectively compressed for deployment on resource-constrained devices without sacrificing performance. The structured pruning and 8-bit quantization techniques used in this work reduce model size by over 90% while preserving almost all of the performance gains. This suggests that deep learning-based optimization is not only feasible but also practical for IoT applications, and the proposed model compression pipeline can be extended to other block cipher optimization scenarios.

Another important finding is that optimizing the SubBytes transformation and key expansion module provides the greatest performance and security benefits for AES on resource-constrained devices. These two components are the main performance bottlenecks and security vulnerabilities in traditional AES implementations, and our lightweight deep learning approaches effectively address both issues. The experimental results show that the joint optimization of the two modules achieves a synergistic effect on both encryption efficiency and side-channel attack resistance, which is not achievable

by optimizing a single module alone. In particular, the 400% improvement in DPA resistance verified by the curve in Figure 1 directly solves the most prominent security risk of AES in IoT deployment scenarios [19].

Compared with the SOTA lightweight AES optimization scheme, our scheme achieves a 66% throughput improvement while maintaining a lower memory overhead, and the DPA attack resistance is increased by more than four times, which fully verifies the superiority of the lightweight deep learning joint optimization scheme [20].

However, our work also has some clear limitations. First, we only optimized the SubBytes and key expansion modules of AES, and did not consider other components such as MixColumns and ShiftRows. The MixColumns operation is the core of AES diffusion, and lightweight deep learning optimization of this module may further improve the parallelism of encryption operations and reduce computational overhead, which is the focus of our subsequent research. Second, our security evaluation focused on differential attacks, linear attacks, and DPA attacks. While these are the most common attacks against AES, future work should also evaluate resistance to other side-channel attacks such as electromagnetic analysis (EMA), timing attacks, and fault injection attacks, to further verify the comprehensive security of the scheme. Third, our experiments were mainly conducted on ARM Cortex-M series microcontrollers. Although we verified the cross-platform robustness on Cortex-M3/M4/M7 chips, the performance on 8-bit AVR/51 microcontrollers and RISC-V architecture microcontrollers needs further testing and optimization. Fourth, the current scheme does not integrate post-quantum cryptography techniques, and its resistance to quantum computing attacks is consistent with the standard AES. With the development of quantum computing, integrating post-quantum key exchange mechanisms into the scheme is an important direction for long-term security enhancement [21, 22].

7 Conclusion and Future Work

This paper presents a lightweight deep learning-based AES optimization scheme specifically designed for mobile IoT security. By combining MobileNet-based lightweight CNNs for SubBytes transformation and pruned, quantized LSTMs for key expansion, we achieved significant improvements in encryption speed and security with minimal resource overhead. Experimental results on ARM Cortex-M4 demonstrate 38% higher throughput, 42% faster key expansion, and 400% better DPA resistance compared to traditional AES, while only increasing memory usage by 12%.

Future work will focus on three directions:

1) Extending lightweight deep learning optimization to other AES components (e.g., MixColumns, ShiftRows) to further improve performance and security.
2) Evaluating the scheme on a wider range of IoT hardware platforms, including RISC-V microcontrollers and FPGAs, to verify its universality.
3) Integrating post-quantum cryptography techniques to enhance resistance to quantum computing attacks, ensuring long-term security for IoT applications.

The proposed lightweight deep learning optimization scheme provides a practical and effective solution for secure and efficient data encryption in mobile IoT applications. By bridging the gap between deep learning-based optimization and IoT deployment constraints, this work paves the way for the widespread adoption of deep learning techniques in resource-constrained security applications.

References

[1] Benadjila R, Prouff E, Strullu R, et al. Deep learning for side-channel analysis and introduction to ASCAD database. Journal of Cryptographic Engineering, 2020, 10: 163–188. https://doi.org/10.1007/s13389-019-00220-8

[2] Kim H, Lee J, Kim Y. A new method for designing lightweight S-boxes with high differential and linear branch numbers, and its application. IEEE Access, 2021, 9: 150592–150607. https://doi.org/10.1109/ACCESS.2021.3126008

[3] Wijaya D C, Afianti F. Improving the randomness of modified lightweight AES using dual dynamic S-boxes on research constraint. 2025 5th International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA), 2025: 48–53. https://doi.org/10.1109/ICICyTA68677.2025.11362701

[4] Peng Z S. Optimization research of hyperchaotic model-driven encryption algorithm in network security. Journal of Cyber Security and Mobility, 2025, 14(2): 283–310. https://doi.org/10.13052/jcsm2245-1439.1422

[5] Wang H, Dubrova E. Tandem deep learning side-channel attack on FPGA implementation of AES. SN Computer Science, 2021, 2(5): 373. https://doi.org/10.1007/s42979-021-00755-w

[6] Negabi I, Asri E A S, Adib E S. Beyond encryption: How deep learning can break microcontroller security through power analysis. e-Prime – Advances in Electrical Engineering, Electronics and Energy, 2025, 11: 100947. https://doi.org/10.1016/j.prime.2025.100947

[7] Fukuda Y, Yoshida K, Hashimoto H, et al. Profiling deep learning side-channel attacks using multi-label against AES circuits with RSM countermeasure. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2023, E106.A(3): 294–305. https://doi.org/10.1587/transfun.2022CIP0015

[8] Wang K, Yan Y, Guo P, et al. Research on power analysis attack based on improved residual network and data augmentation. Chinese Journal of Cryptology, 2020, 7(4): 551–564. https://doi.org/10.13868/j.cnki.jcr.000389

[9] El-Latif A A A, Abd-El-Atty B, Belazi A, et al. Efficient chaos-based substitution-box and its application to image encryption. Electronics, 2021, 10(12): 1392. https://doi.org/10.3390/electronics10121392

[10] Al-Maadeed T A, Hussain I, Anees A, et al. An image encryption algorithm based on chaotic Lorenz system and novel primitive polynomial S-boxes. Multimedia Tools and Applications, 2021, 80: 24801–24822. https://doi.org/10.1007/s11042-021-10695-5

[11] Zhang Y, Su X. Application of deep learning in the optimization of AES encryption algorithm. Journal of Cyber Security and Mobility, 2025, 14(3): 553–576. https://doi.org/10.13052/jcsm2245-1439.1432

[12] Zhang Y. Application and optimization of deep learning in image recognition. 2024 International Seminar on Artificial Intelligence, Computer Technology and Control Engineering (ACTCE), 2024: 438–442. https://doi.org/10.1109/ACTCE65085.2024.00095

[13] Swaminathan S, Chmielewski Ł, Perin G, et al. Deep learning-based side-channel analysis against AES inner rounds. Zhou J, et al. Applied Cryptography and Network Security Workshops (ACNS 2022). Vol. 13285. Cham: Springer, 2022: 165–182. https://doi.org/10.1007/978-3-031-16815-4_10

[14] Bhasin S, Chattopadhyay A, Heuser A, et al. Mind the portability: A warrior's guide through realistic profiled side-channel analysis. Proceedings of the 2020 Network and Distributed System Security Symposium (NDSS), 2020. https://doi.org/10.14722/ndss.2020.24390

[15] Zhang R, Mo Y, Pan Z, et al. Intra-class CutMix data augmentation based deep learning side channel attacks. Integration, 2025, 102: 102373. https://doi.org/10.1016/j.vlsi.2025.102373

[16] Ailon N, Bercovich A, Uffenheimer Y, et al. Changing base without losing pace: A GPU-efficient alternative to MatMul in DNNs. arXiv preprint arXiv:2503.12211, 2025. https://doi.org/10.48550/arXiv.2503.12211

[17] Zhang L, Wang Z, Lu J. Differential-neural cryptanalysis on AES. IEICE Transactions on Information and Systems, 2024, E107.D(10): 1372–1375. https://doi.org/10.1587/transinf.2024EDL8044

[18] Pehlivanoglu M K, Sakalli M T, Akleylek S, et al. Generalisation of Hadamard matrix to generate involutory MDS matrices for lightweight cryptography. IET Information Security, 2018, 12(4): 329–336. https://doi.org/10.1049/iet-ifs.2017.0156

[19] Rezaeezade A, Batina L. Regularizers to the rescue: Fighting overfitting in deep learning-based side-channel analysis. Journal of Cryptographic Engineering, 2024, 14(3): 609–629. https://doi.org/10.1007/s13389-024-00361-5

[20] Jacob B, Kligys S, Chen B, et al. Quantization and training of neural networks for efficient integer-arithmetic-only inference. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018: 2704–2713. https://doi.org/10.1109/CVPR.2018.00286

[21] Ninan M, Nimmo E, Reilly S, et al. A second look at the portability of deep learning side-channel attacks over EM traces. Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses (RAID '24), 2024: 630–643. https://doi.org/10.1145/3678890.3678900

[22] Sakalli M T, Akleylek S, Akkanat K, et al. On the automorphisms and isomorphisms of MDS matrices and their efficient implementations. Turkish Journal of Electrical Engineering and Computer Sciences, 2020, 28(1): 275–287. https://doi.org/10.3906/elk-1906-151

Biography

Yanxin Zhang earned her junior college degree in Electronic Technology and Computer Applications from Beijing Adult Education Institute in 1997. In 2001, she obtained a bachelor's degree from Beijing Foreign Studies University, while concurrently studying courses related to computer engineering. In 2004, she completed her master's degree at Renmin University of China, with a supplementary focus on computer applications. Currently, she serves as an Associate Professor at Beijing Vocational College of Labour and Social Security. Her research primarily focuses on artificial intelligence, digital teaching, and the application of deep learning in educational data security. She has led the development of Beijing Municipal Online High-quality Courses and serves as an editor for the Journal of Beijing Vocational College of Labour and Social Security.