electronics

MDPI

Article

Secure and Lightweight Firmware Over-the-Air Update Mechanism for Internet of Things

Chae-Yeon Park [1], Sun-Jin Lee [2] and Il-Gu Lee [1,2,*]

[1] Department of Convergence Security Engineering, Sungshin Women's University, Seoul 02844, Republic of Korea; 20221097@sungshin.ac.kr
[2] Department of Future Convergence Technology Engineering, Sungshin Women's University, Seoul 02844, Republic of Korea; 220237017@sungshin.ac.kr
* Correspondence: iglee@sungshin.ac.kr

Abstract: The Internet of Things (IoT) necessitates secure and lightweight firmware over-the-air (FOTA) update mechanisms for remote device management and timely mitigation of security vulnerabilities. This study introduces an FOTA update method to mitigate man-in-the-middle attacks in resource-constrained environments. The proposed method minimizes firmware file size and encryption overhead through a dual-XOR operation and DEFLATE compression, while enhancing security via multiple transmission channels. It improves performance in terms of latency, memory usage, and power consumption but also maintains security against brute-force attacks during MITM attacks.

Keywords: internet of things; firmware over-the-air; cybersecurity; man-in-the-middle attacks; compression; encryption

Academic Editors: Yazan Otoum and Stefano Scanzio

Received: 11 March 2025
Revised: 10 April 2025
Accepted: 12 April 2025
Published: 14 April 2025

Citation: Park, C.-Y.; Lee, S.-J.; Lee, I.-G. Secure and Lightweight Firmware Over-the-Air Update Mechanism for Internet of Things. *Electronics* **2025**, *14*, 1583. https://doi.org/10.3390/electronics14081583

1. Introduction

The rapid proliferation of the Internet of Things (IoT), driven by advancements in information and communication technology, has resulted in the interconnection of billions of devices, forming expansive networks. The IoT is employed in various fields, including healthcare, industry, and domestic applications, and its convergence with big data, cloud services, and artificial intelligence fosters novel business opportunities and value creation. IoT devices tasked with real-time data acquisition and processing are often geographically dispersed and deployed on a large scale, rendering physical access challenging. Consequently, firmware over-the-air (FOTA) technology has become indispensable for remote device monitoring, diagnostics, and efficient firmware distribution and patching, including timely responses to security vulnerabilities [1].

The expanding influence of the IoT across daily life and various industries is paralleled by a surge in security threats. Notably, man-in-the-middle (MITM) attacks targeting gateways, application/firmware suppliers, and manufacturers—intermediaries between IoT devices and firmware providers—pose an increasing risk [2]. MITM attacks can disrupt device operation or lead to sensitive information leakage by enabling attackers to falsify or inject malicious code after intercepting transmitted firmware files. The FOTA mechanism is particularly susceptible to MITM attacks owing to its reliance on wireless communication in environments with limited physical control.

Various security enhancement technologies are being studied to address these attacks. End-to-end encryption (E2EE) is commonly employed as a security system in commercial networks. E2EE systems establish secure communication by encrypting data

between the client and server, preventing eavesdropping by potential attackers [3]. Over-the-air-rekeying (OTAR) is a wireless update technology that decentralizes equipment management. It enables remote key updates for communication security equipment, reducing the frequency of new key transmissions, mitigating key-related threats during transit, and accelerating key update procedures [4]. Furthermore, OTAR allows for isolating compromised nodes within the network, followed by remote key updates on the remaining nodes [4]. Encryption algorithms such as advanced encryption standard (AES) [5], SEED [6], and triple data encryption standard (3DES) [7], frequently utilized in OTAR, offer robust security. However, they often do not account for the resource constraints inherent in lightweight IoT devices. As a result, many lightweight cryptographic algorithms that are suitable for resource-constrained IoT environments have been developed. Taking two typical lightweight encryption algorithms as examples: CHACHA20 is faster than AES in CPUs without hardware acceleration [8]. However, because of the stream cipher nature of CHACHA20, it does not manage a value called nonce in IoT environments, which would destroy the cipher if leaked to an attacker. PRESENT-80 can be implemented with minimal energy per bit on hundreds of bytes of RAM/ROM, particularly if hardware accelerators are available [9]. However, 80-bit keys are highly vulnerable to brute-force attacks.

Therefore, this study proposes a methodology for reducing firmware file size using server-side lossless compression techniques and countering MITM attacks through a lightweight and secure FOTA mechanism that combines dual-XOR operations with two keys and multichannel transmission.

The main contributions of this study can be summarized as follows:

- Compression techniques within the lightweight FOTA update process are utilized to reduce latency and enhance security concurrently.
- A lightweight encryption technique is proposed to mitigate MITM attacks.
- A comparative evaluation of the proposed and conventional techniques for firmware files was conducted.

The remainder of this paper is organized as follows: Section 2 analyzes existing techniques designed to counter MITM attacks. Section 3 details the proposed technique. Section 4 describes the performance evaluation environment and analyzes the results. Finally, Section 5 presents the conclusions.

2. Related Work

Table 1 summarizes the contributions and limitations of significant prior studies on secure FOTA update mechanisms. In each study, we marked an "✕" if the element was not addressed, a "○" if it was fully addressed, and a "△" if it was partially addressed or not fully resolved.

Table 1. Related work on secure FOTA update mechanisms.

Reference	Proposed Method	Contribution	Limitation	Lightweight	Brute-Force Attack Vulnerability
[10]	An on-premise- based FOTA update framework.	Ensures stability by mitigating data transmission errors arising from unstable network conditions.	Managing hundreds of IoT devices in the field is cost prohibitive.	✕	○

Table 1. *Cont.*

Reference	Proposed Method	Contribution	Limitation	Lightweight	Brute-Force Attack Vulnerability
[11]	A FOTA update method employing a Merkle Tree structure root hash for data integrity authentication.	Reduces delay time compared to the conventional transport layer security (TLS) protocol; capable of mitigating spoofing and MITM attacks.	The TLS communication protocol remains slow and resource-intensive when applied to IoT environments.	△	×
[12]	A Timed Efficient Stream Loss-Tolerant Authentication protocol-based security scheme to enhance the authentication of the Satellite-Based Augmentation System.	Enhances security through transport channel encryption via over-the-air-rekeying (OTAR), which facilitates remote encryption key updates.	Vulnerability remains as the key itself is transmitted wirelessly; a successful MITM attack could compromise the key if the transmission channel is successfully eavesdropped.	△	○
[13]	A method to optimize the structure of the simple instruction multiple operation (SIMON) cryptographic algorithm by leveraging features of the SPECK algorithm.	The 32-, 48-, 64-, and 96-bit versions of optimized SIMON exhibit faster execution speeds than AES.	SIMON, with block sizes of 64, 96, and 128 bits, still demands more clock cycles than AES and remains unsuitable for resource-constrained IoT environments.	△	×
[14]	A hybrid encryption algorithm integrating AES and elliptic curve cryptography (ECC).	The encryption algorithm accounts for less than 1.23% of the FOTA system update time while strengthening the data transmission security of the vehicle FOTA system.	Integrating AES and ECC algorithms increases computational complexity and memory usage, potentially causing delays in real-time, resource-constrained IoT environments.	△	×
[15]	An algorithm incorporating the Dual XOR S-Box technique into the SubByte process of AES, reducing encryption and decryption rounds from the standard 10 rounds to 9.	The proposed algorithm exhibits approximately 30–34% faster performance than AES across various file sizes.	Experimental evaluation was limited to audio data, leaving its efficacy on actual firmware files unverified.	△	×
Our Model	FOTA update mechanism against MITM attack.	The Lightweight FOTA update mechanism using lossless compression techniques and dual-XOR operation, countering MITM attacks via multichannel transmission.		○	×

Conventionally, an on-premises approach is used for secure FOTA management to reduce errors due to the unstable network environment [10]. However, this method does not maximize the advantages of wireless updates. Therefore, we investigated a secure wireless FOTA update scheme that does not require 100% human presence at the terminal. Existing encryption algorithms are lightweight [11,13,14]. However, these lightweight encryption schemes are not suitable for lightweight IoT because the number of clock cycles is still more than two. It has been reported to speed up the AES by 30–34% [15]; however, it has not been tested using real firmware file data, and hence, we cannot verify its suitability for the FOTA environment. It increases security by updating the key on the wireless channel, but it is still vulnerable to eavesdropping attacks because the original key is transmitted on one channel [12].

Previous studies [11,13–15] have demonstrated resilience against brute-force attacks, but they still require lightweight implementation. In contrast, the system in [12] is vulnerable to such attacks and requires optimization for lightweight environments. The systems in [10] remain exposed to brute-force threats. To address these limitations, this study proposes a FOTA mechanism optimized for lightweight environments while effectively mitigating brute-force attack vulnerabilities.

3. Proposed Mechanism

Figure 1 illustrates the operational workflow of the proposed FOTA mechanism, which comprises two primary phases: key table sharing and the FOTA update process. During the key table sharing phase, both the server and the IoT device receive the key table via broadcast communication through a router. The FOTA update phase encompasses a seven-step operational process through which the server transmits the firmware file to the IoT device. Step 1 entails the preparation of a firmware binary file. In Step 2, the binary file undergoes lossless compression using the DEFLATE algorithm. Firmware is composed mainly of binary files [16]; if some bits are missing during the transfer and decompression process after compression, the original file cannot be restored normally. The DEFLATE algorithm is a popular lossless compression algorithm used in practice, and DEFLATE-based libraries (e.g., QAT zip) are open-source packages that can be easily integrated into various applications [17]. In addition, it can be dynamically applied with lower compression levels in delay-critical channel environments, which can be efficiently used in multichannel network environments in the future [17]. Therefore, we used DEFLATE, a lossless algorithm, to integrate sliding window compression (LZ77) and Huffman encoding to achieve lossless compression of redundant data. Step 3 involves the encryption of the compressed file via a dual-XOR operation. Equations (1)–(3) exemplify the dual-XOR operation method:

$$\mathrm{D} = B_p \oplus K_1, \tag{1}$$

$$B_e = \mathrm{D} \oplus K_2, \tag{2}$$

$$\forall i \in \{1, 2, \ldots, n\},\ {B_e}^i = {B_p}^i \oplus K_i \oplus K_{i+1} \tag{3}$$

After dividing the firmware file into multiple blocks, one such block is denoted as B_p. The size of each block is the same as the key length. If the length of the block (B_p) is not equal to the length of the key in the last block of the entire file, the blank space is filled by padding with the character 0. Each block B_p undergoes successive XOR operations with K_1 and K_2, where each key differs from those used in preceding blocks. The XOR operation between block B_p of the compressed firmware file and K_1 transforms B_p into D. Subsequently, B_e, representing the final encrypted file block, is derived from the XOR operation between D and K_2. Consequently, applying Equation (3) results in the entire firmware file being encrypted, as a total of n firmware file blocks are encrypted by applying

Equations (1) and (2) for each block. In this case, K_1 through K_{n+1} do not depend on each other consecutively; instead, each key is derived based on a seed value (private key) that is generated using a random number generator (RNG). Since XOR is a bitwise operation, even if several bits are corrupted during transmission, the original data can still be recovered, or errors can be detected effectively. In Step 4, the encrypted file and a private key are fragmented for transmission across multiple channels. The private key serves as a seed value for selecting K_1 and K_2 from the key table. This private key is generated randomly using a random number generator (RNG). The 6-GHz channel, exhibiting reduced interference and a superior data rate than the 5- and 2.5-GHz channels, is prioritized. Accordingly, B_e and the private key are partitioned and transmitted over optimized channels. The encrypted file is transmitted via the 6 GHz channel, whereas the private key is bifurcated into Private_Key1 and Private_Key2 transmitted over the 5- and 2.5-GHz channels, respectively. During this process, the private key is encrypted using the AES-128 algorithm. Therefore, even if an attacker eavesdrops on the packet over the channel, the original private key cannot be obtained owing to encryption of the packet. The use of multichannels increases the complexity of the attack because when an MITM attacker wants to obtain the secret key by eavesdropping on a channel, the attacker needs to successfully eavesdrop on both channels to obtain the complete recovery key. Especially, the proposed FOTA mechanism uses multilink operation simultaneous transmit and receive (MLO-STR). MLO-STR is a network situation where two or more channels can be used simultaneously. When channels are symmetrically occupied, MLO-STR can achieve up to 90% lower latency than single-link operation (SLO) [18]. In Step 5, the IoT device receives Private_Key1 and Private_Key2 from the server, retrieves K_1 and K_2, and decrypts the encrypted file. The FOTA update concludes with file decompression in Steps 6 and 7. The proposed method enhances security by distributing the RNG-based seed value of the encryption key (private key) across multiple channels. Moreover, the dual-XOR operation increases the number of key combinations necessary for a brute-force attack. Furthermore, file size and encryption complexity are minimized through the combined use of a dual-XOR operation and the DEFLATE compression algorithm.

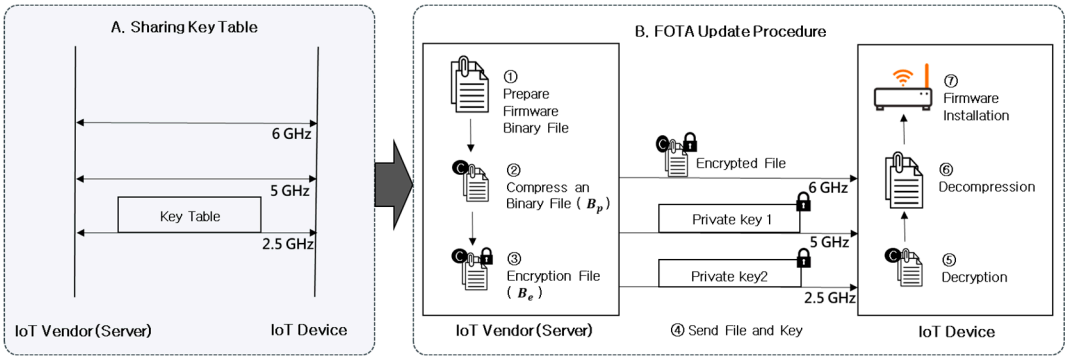

Figure 1. Flowchart of the proposed FOTA method.

4. Evaluation Results and Analysis

This study evaluated the latency, accuracy, and security of the proposed model and compared the results with those of the conventional model. Table 2 presents each model's encryption method and compression type. The conventional model (AES-Non-Compression) [19] encrypts the firmware file using AES-128 encryption without compression. The conventional model (AES-Compression) [20] employs both firmware file compression and AES-128 encryption. The proposed model (Dual-XOR-Compression) compresses the firmware file and subsequently encrypts it using a dual-XOR operation. This study employed five commercially available firmware images of varying sizes: Vueguera VN-

850NHD navigation firmware (1.2 MB) [21], Hyundai Phoneus A300 black box firmware (27.6 MB) [22], SKYREX SKY-3004F v0.8.0c CCTV firmware (2.4 MB) [23], ipTIME 14.28.8 router firmware (42.2 MB) [24], and Hyundai TNR UNIQ500 V5.0.6A black box firmware (27.6 MB) [25]. In a real-world scenario with channel noise, data or private keys may need to be retransmitted multiple times, resulting in increased delay. However, our performance evaluations for all models, namely, AES-Non-Compression, AES-Compression, and Dual-XOR-Compression, were carried out under the assumption of a noise-free network environment. The experimental code was written in Python 3.12 on a 13th Gen Intel (R) processor and 32.0 GB laptop with Windows 11 Home to evaluate the performance. The AES encryption algorithm uses an open-source code [26] implemented in Python and the crypto module provided by Python 3.12. Figures 2–5 were generated using the output data obtained from our implementation, and the graphs were plotted using matplotlib in Python.

The metrics defined in Equations (4)–(6) were used to assess the security of the proposed model:

$$N_{keys} = \frac{size\_of\_keytable}{|k|}, \tag{4}$$

$$N_{blocks} = \frac{firmware\_file\_size}{|k|}, \tag{5}$$

$$Number\ of\ seed\ pattern = \left(\frac{N_{keys} \times (N_{keys} - 1)}{2} \right)^{N_{blocks}} \tag{6}$$

where N_{keys} denotes the number of keys within the key table, and N_{blocks} represents the total number of blocks in the entire firmware file divided by the length of the keys. Both N_{keys} and N_{blocks} are defined with the length of the key as the denominator, which reflects the equivalence of the size of each encrypted block to the length of the key. The "number of seed pattern" refers to the number of potential seed combinations generated by the current key table. When an attacker is between the server and the IoT device transmission channel, the actual value that can be eavesdropped is the seed value, the private key, which is the basis for extracting the keys from the key table. In this case, the higher the number of seed patterns, the higher the difficulty of eavesdropping when an attacker tries to obtain the private key with a brute-force attack.

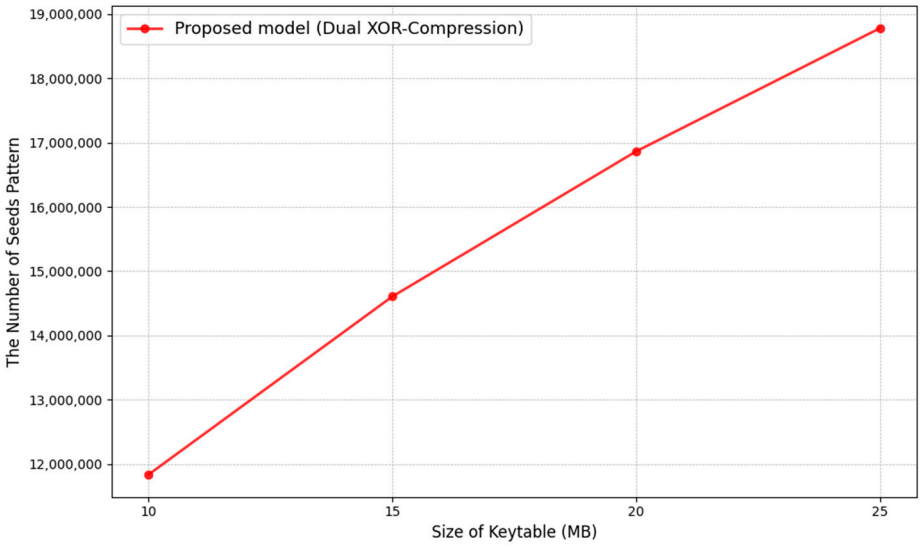

Figure 2. The number of seeds pattern of the proposed model (Dual-XOR-Compression).

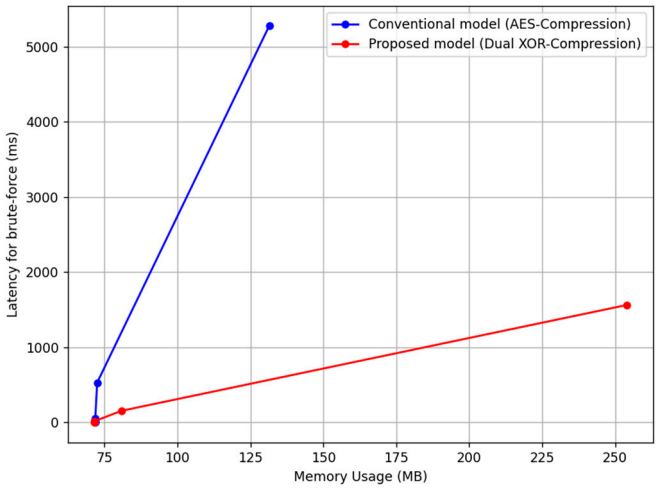

Figure 3. Latency comparison of the conventional model (AES-Compression) and the proposed model (Dual-XOR-Compression) brute-force attacks.

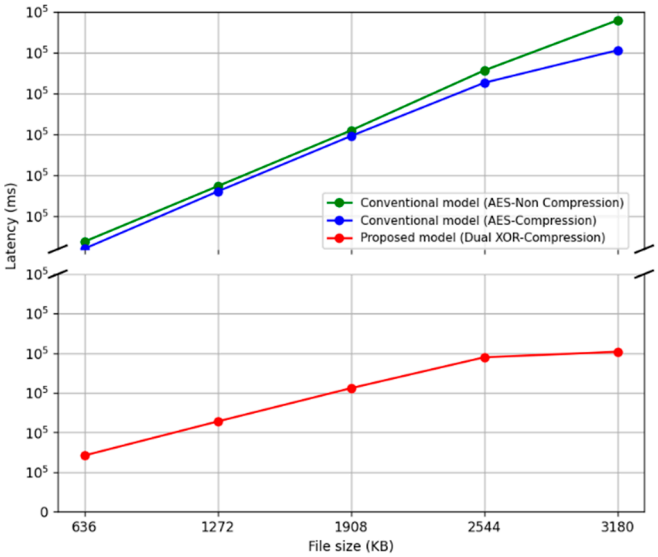

Figure 4. Latency comparison of the conventional and proposed models.

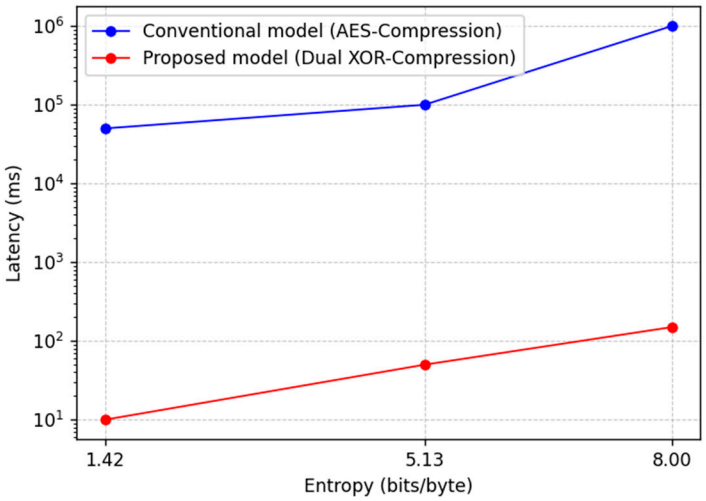

Figure 5. Latency comparison of the conventional and proposed methods.

Table 2. Encryption and compression methods of conventional and proposed models compared in this study.

Model	Encryption Method	Compression Type
Conventional model (AES-Non-Compression)	AES-128 (bit)	Not compressed
Conventional model (AES-Compression)	AES-128 (bit)	DEFLATE
Proposed model (Dual-XOR-Compression)	Dual-XOR encryption	DEFLATE

Figure 2 presents the security evaluation results of the proposed model calculated using Equation (6). A Firmware_file_size of 15,885 kb, equivalent to that of a commercially available firmware file size, was used. This analysis defined the key length as 128 bits, which is in line with current NIST cryptographic recommendations, which proposes to increase the minimum required security strength from 112 bits to 128 bits [27]. As the value of N_{keys} increases due to an increase in the key table size, the number of seed patterns in the proposed model correspondingly increases. This increase in the number of seed patterns expands the range of possible key combinations, thereby increasing the complexity of attacks that attempt to deduce the seed value through random key extraction from the key table. Since the attacker does not know the private key, it must brute-force attack the keys used in each block to obtain the original data. This increases the complexity of the attack, and randomized brute force can ensure the confidentiality of the original data from the attack.

Table 3 presents the number of cases where an attacker can determine the encryption key using brute force without knowing the key table. The conventional (AES-128) model has a complexity of 2^{128} because it encrypts the entire file with one 128-bit key. On the other hand, the proposed (Dual-XOR) model has a complexity of $2^{128 \times N_{blocks}}$ because it encrypts each block with a different 128-bit key. In the proposed (Dual-XOR) model, an attacker must try a brute force attack $2^{N_{blocks}}$ times more than that in the conventional (AES-128) model to obtain the key, which improves attack complexity.

Table 3. Comparison of the number of key cases of the conventional (AES-128) and proposed (Dual-XOR) method.

Factor \ Model	Conventional (AES-128)	Proposed (Dulal-XOR)
The number of key cases	2^{128}	$2^{128 \times N_{blocks}}$

Figure 3 illustrates the latency under constrained memory usage during a brute-force attack on both the conventional model (AES-Compression) and the proposed model (Dual-XOR-Compression), conducted over 1000 iterations. A random 128-bit string was encrypted using a 128-bit key, and multiple random keys were tested to determine decryption feasibility. The proposed model (Dual-XOR-Compression) exhibited an approximately 3.4-fold lower latency than the conventional model (AES-Compression). However, neither successfully decrypted the encryption key within the imposed 250-MB memory constraint. This indicates that both the conventional model (AES-Compression) and the proposed model (Dual-XOR-Compression) are not effective against brute-force attacks with limited memory. The proposed model (Dual-XOR-Compression) is less secure than the conventional model (AES-Compression) when resources and time are unlimited, but it contributes to lightweighting while maintaining security when memory and time are limited.

Figure 4 presents the latency evaluation results for the conventional (AES-Non-Compression and AES-Compression) and proposed model based on firmware file size. Latency was measured as the duration from firmware file compression and encryption to decryption and decompression. Each model was tested 1000 times, and the average was computed. For all models, latency increased with increasing file size. However, the proposed model demonstrated reduced latency regardless of firmware file size, averaging 0.71% and 0.72% of that observed for the AES-Non-Compression and AES-Compression models, respectively.

Table 4 and Figure 5 illustrate the interrelationships among entropy, compression ratio, and latency categorized by dividing the entropy values of the firmware file into upper, middle, and lower ranges. Table 4 was obtained based on Equations (7) and (8) using three selected firmware files from the previously mentioned commercial firmware samples. Entropy, in the context of a firmware file, quantifies the randomness of the data and represents the average information content per symbol given a specific probability distribution, as expressed in Equation (7):

$$H(m) = -\sum_{i=1}^{M} P_i log_2 P_i, \quad (7)$$

where P_i represents the probability of symbol i being generated, and M denotes the number of possible symbols within the data. Equation (8) defines the lower and upper bounds of entropy [28]:

$$0 \leq H(m) \leq log_2 M. \quad (8)$$

Table 4. Entropy vs. compression ratio.

Entropy (Bits/Byte)	Compression Ratio (%)
1.42	93.16
5.13	64.28
8.0	0.62

The lower bound condition occurs when only one symbol has a probability of occurrence of 1, while all other symbols have a probability of 0. This implies that all data can be compressed into a single symbol as the file's entropy approaches 0. Conversely, the upper-bound condition is met when all symbols have equal probabilities of occurrence ($P_i = \frac{1}{M}$). In this scenario, uncertainty is maximized, all symbols are equally represented within the file, and further compression becomes challenging. This study set the upper entropy limit to 8, assuming data representation in bytes, where 1 byte can assume 256 distinct values. Additionally, the entropy range was partitioned into three intervals: lower (0–2.67), middle (2.68–5.33), and upper (5.34–8).

A compression algorithm reduces data size by identifying repetitive patterns or regularities. Consequently, higher entropy values, indicating greater randomness, hinder pattern identification and result in lower compression ratios. This, in turn, increases the amount of data to be processed, leading to increased latency. As depicted in Table 3, the compression ratio was 93.16% at an entropy value of 1.42, decreasing to 64.28% at 5.13 and further to 0.62% at 8.0. As shown in Figure 5, the conventional model exhibited latencies of 36,710 ms, 110,430 ms, and 616,800 ms at entropies of 1.42, 5.13, and 8.0, respectively. In contrast, the proposed model significantly reduced these latencies to 5.98 ms, 33.27 ms, and 117.6 ms, respectively. The proposed model achieved an average latency of only 0.72% compared to the conventional model, showing better performance regardless of firmware entropy.

Figure 6 presents the accuracy results for both lossy compression (LC) and lossless compression (LZ) scenarios, comparing the conventional and proposed models. Assuming an ideal environment without channel noise, the AES-Non-Compression model involved only encryption and decryption processes. However, the AES-Compression and proposed models incorporated both encryption/decryption and lossless (de)compression processes. The JPEG algorithm was employed for LC, with the compression quality parameter set to 75, which is considered an optimal value [29,30]. The DEFLATE algorithm was used for LZ, with the compression level parameter set to 9, representing maximum compression.

$$Error\ Rate = \sum_{i=1}^{N} M(P_i, D_i) \tag{9}$$

$$M(P_i, D_i) = \begin{cases} 1, & (if\ P_i = D_i) \\ 0, & (if\ P_i \neq D_i) \end{cases} \tag{10}$$

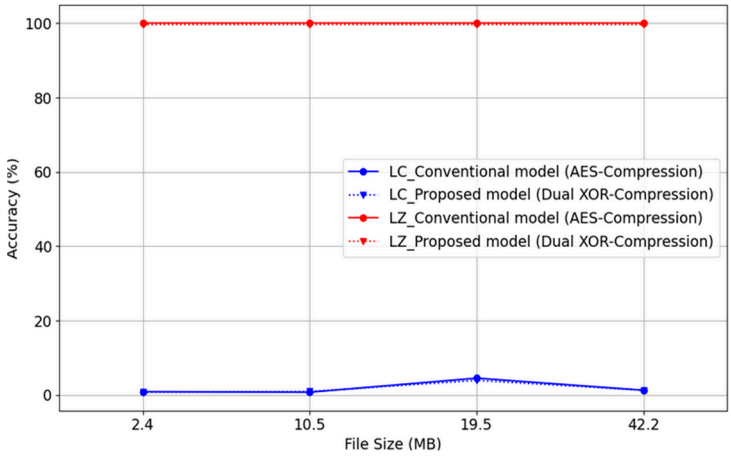

Figure 6. Accuracy comparison of conventional and proposed models based on the compression method employed.

Accuracy was measured using byte comparison and SHA-256 hash value comparison methods. The byte comparison method directly evaluates the file size and byte-level correspondence, as expressed in Equations (9) and (10). A single original block is denoted as P_i and a decompressed block as D_i. If $P_i = D_i$, the comparison returns 1; otherwise, it returns 0. The error rate is calculated by comparing the original and decompressed data in 16-byte blocks and counting the number of mismatched blocks. The SHA-256 hash function is valuable for verifying data integrity owing to the Avalanche effect, where a minor input change produces a significantly different output [31]. Hash value matches are represented as binary values: 0 for mismatch and 1 for match.

The experimental results indicate that, under LC, the conventional model exhibits byte agreement rates of 0.85, 0.72, 4.5, and 1.25%, whereas the proposed model shows rates of 0.72, 0.90, 3.9, and 1.25%. Hash values for both models converge to 0, indicating data mismatch. Conversely, under LZ, both models achieve 100% byte agreement, and hash values converge to 1. For a 2.4 MB file, LC achieved a compression ratio of 79.82%, whereas LZ achieved 0.63%. For a 10.5 MB file, the LC ratio was 79.89%, and the LZ ratio was −0.03%. The compression ratios for 19.5- 42.2-MB files were 80.20 and 51.65% for LC and 80.01 and 3.03% for LZ, respectively. Both the conventional and proposed models demonstrated degraded compression ratios under LZ but improved accuracy. Therefore, the proposed model maintains the accuracy of the conventional model, reduces latency, and enhances security.

Table 5 presents the power consumption results for the conventional (AES-Non-Compression), conventional (AES-Compression), and proposed (Dual-XOR-Compression) model. The base power of the Intel Core™ i7-1360P processor is defined as 28 W [32]. Accordingly, we define power consumption (j) as the product of the execution time of the conventional and proposed model and the base power of 28 W. The proposed model consumed approximately 261 times less power than the conventional model (AES-Non-Compression) and 179 less power than the conventional model (AES-Compression). These results demonstrate that the proposed Dual-XOR-Compression model achieves improved energy efficiency compared to the conventional AES-128 model. This enhancement highlights its suitability for deployment in lightweight and energy-constrained IoT environments.

Table 5. Comparison of the power consumption of the conventional model and proposed model.

Conventional (AES-Non-Compression)	Conventional (AES-Compression)	Proposed (Dual-XOR-Compression)
3916 (j)	2864 (j)	15 (j)

Table 6 presents the memory usage results for the conventional (AES-Non-Compression), conventional (AES-Compression), and proposed (Dual-XOR-Compression) model. To evaluate memory usage, the psutil library in Python was employed to measure the memory consumption of both the conventional and proposed model during their entire execution process. Measurements were taken at 0.1 ms intervals, and memory usage was accumulated throughout the execution. This procedure was repeated for 100 iterations, and the average memory usage across all iterations was calculated. The proposed model demonstrated a reduction in memory usage of approximately 23.8 MB compared to the conventional AES model without compression and 8.4 MB compared to the AES model with compression. In terms of reduced memory usage, the proposed model is more suitable for resource-constrained environments compared to the conventional model.

Table 6. Comparison of the memory usage of the conventional model and proposed model.

Conventional (AES-Non-Compression)	Conventional (AES-Compression)	Proposed (Dual-XOR-Compression)
57.63 (mb)	42.18 (mb)	33.74 (mb)

5. Conclusions

The increasing utilization of IoT devices across various industries and diverse industrial sectors, coupled with their wireless interconnection, necessitates robust security measures. MITM attacks in wireless networks present a substantial threat, even within IoT environments. Therefore, an S-FOTA mechanism capable of mitigating MITM attacks is urgently required. Conventional encryption algorithms are often unsuitable for the resource-constrained, lightweight environments characteristic of many IoT devices. Consequently, this study proposed a FOTA update method designed to counter MITM attacks within such constrained environments. The proposed method minimizes file size and encryption overhead by employing LZ of firmware files and subsequent encryption via dual-XOR operations. Consequently, the proposed method reduced latency attributable to file size variations to an average of approximately 0.71% compared with conventional methods, while latency owing to entropy changes was reduced to approximately 0.72%. The proposed model consumed approximately 261 times less power than the conventional AES-128 model without compression and 179 times less than the model with compression. In addition, memory usage was reduced by approximately 23.8 MB and 8.4 MB compared

to the AES-Non-Compression and AES-Compression models, respectively. Furthermore, security was enhanced by distributing the RNG-based key table seed value (private key) across multiple transmission channels. When an attacker possesses knowledge of the key table and attempts a brute-force attack, each 5 MB increase in the key table size leads to an approximately unbounded exponential growth in attack complexity. When the attacker does not consider the key table and instead attempts a brute-force attack, the proposed model requires $2^{N_{blocks}}$ times more attempts than the conventional model to obtain the correct key due to the use of a different 128-bit key for each block. In a practical brute-force attack attempt, a random 128-bit string was encrypted using a 128-bit key, and multiple random keys were tested. Neither AES-128 nor dual-XOR successfully decrypted the encryption key within the imposed 250 MB memory constraint. This result indicates that both AES-128 and dual-XOR are resistant to brute-force attacks under limited memory conditions. The FOTA mechanism demonstrated 100% accuracy, equivalent to that of conventional models, while maintaining security against brute-force attacks. Additionally, it simplifies the encryption process, thereby reducing latency, energy consumption, and memory usage. Therefore, the proposed method optimized for secure and lightweight FOTA updates in resource-constrained environments. This work presents a multichannel transmission mechanism that performs a novel random seed-based XOR operation using key table for FOTA updates while maintaining compatibility with existing technologies through a combination of well-known methods such as XOR, DEFLATE, and random seed. Furthermore, it is not limited to the DEFLATE algorithm; therefore, it can support other lightweight, lossless algorithms better suited for lightweight IoT environments.

6. Limitation and Future Work

Both the proposed model and the conventional model were evaluated under identical conditions. This setup allows for a simple and intuitive interpretation of the performance differences, as the results are not affected by external variables. Therefore, the value of the simulation lies in the relative comparison between the two models. However, the limitation of this study is that all performance tests and simulations were performed in a "noise-free" environment. Therefore, in the future, the proposed mechanism will be applied to Raspberry Pi to build a real multichannel network environment to evaluate its performance in terms of memory usage, energy consumption of the device, delay time, and security. We plan to perform a comparative analysis between the proposed dual-XOR scheme and various existing lightweight encryption algorithms. In addition, we will conduct standardized security tests, such as the NIST test suite, as part of future work.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design, C.-Y.P. and I.-G.L.; analysis and interpretation of results, C.-Y.P., S.-J.L. and I.-G.L.; draft manuscript preparation, C.-Y.P., S.-J.L. and I.-G.L.; Project administration, I.-G.L.; Funding acquisition, I.-G.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Sungshin Women's University Research Grant of H20240090.

Data Availability Statement: The data used in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced encryption standard
ECC	Elliptic curve cryptography
FOTA	Firmware over-the-air
IoT	Internet of Things
LC	Lossy compression
LZ	Lossless compression
MITM	Man-in-the-middle
OTAR	Over-the-air-rekeying
RNG	Random number generator
TLS	Transport layer security

References

1. Understanding Firmware Over-The-Air (FOTA). Available online: https://kemsys.com/blog/firmware-over-the-air/ (accessed on 18 October 2024).
2. El Jaouhari, S.; Bouvet, E. Secure Firmware over-the-air updates for iot: Survey, challenges, and discussions. *Internet Things* **2022**, *18*, 100508. [CrossRef]
3. Kim, S.-Y.; Yun, S.-W.; Lee, E.-Y.; Bae, S.-H.; Lee, I.-G. Fast packet inspection for end-to-end encryption. *Electronics* **2020**, *9*, 1937. [CrossRef]
4. Telecommunications Industry Association. Mobile and Point-to-Point Communications Standards. 2002. Available online: https://tiaonline.org/committee/tr-45-i-mobile-and-point-to-point-communications-standards/ (accessed on 12 January 2025).
5. Giraud, C. DFA on AES. In *Advanced Encryption Standard, Proceedings of the AES 2004, Bonn, Germany, 10–12 May 2004*; Dobbertin, H., Rijmen, V., Sowa, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3373, pp. 27–41. [CrossRef]
6. Yoon, J.; Lee, S.; Cheon, D.H.; Lee, J.; Lee, H. *The SEED Encryption Algorithm*; Internet Engineering Task Force: Fremony, CA, USA, 2005. [CrossRef]
7. Coppersmith, D.; Johnson, D.B.; Matyas, S.M. A proposed mode for triple-DES encryption. *IBM J. Res. Dev.* **1996**, *40*, 253–262. [CrossRef]
8. Saraiva, D.A.F.; Leithardt, V.R.Q.; de Paula, D.; Sales Mendes, A.; González, G.V.; Crocker, P. PRISEC: Comparison of Symmetric Key Algorithms for IoT Devices. *Sensors* **2019**, *19*, 4312. [CrossRef]
9. Silva, C.; Cunha, V.A.; Barraca, J.P.; Aguiar, R.L. Analysis of the Cryptographic Algorithms in IoT Communications. *Inf. Syst. Front.* **2024**, *26*, 1243–1260. [CrossRef]
10. Bhargava, V.; Raghava, N.S. Version control system gateway to optimize firmware over the air (FOTA) update for IoT wireless devices. *Wirel. Pers. Commun.* **2024**, *134*, 249–265. [CrossRef]
11. Shin, Y.; Jeon, S. MQTree: Secure OTA protocol using MQTT and MerkleTree. *Sensors* **2024**, *24*, 1447. [CrossRef] [PubMed]
12. Anderson, J.; Lo, S.; Neish, A.; Walter, T. Authentication of satellite-based augmentation systems with over-the-air rekeying schemes. *Navig. J. Inst. Navig.* **2023**, *70*, navi.595. [CrossRef]
13. Alassaf, N.; Gutub, A.; Parah, S.A.; Al Ghamdi, M. Enhancing speed of SIMON: A light-weight-cryptographic algorithm for IoT applications. *Multimed. Tools Appl.* **2019**, *78*, 32633–32657. [CrossRef]
14. Cheng, A.; Yin, J.; Ma, D.; Dang, X. Application and research of hybrid encryption algorithm in vehicle FOTA system. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 4988–4993. [CrossRef]
15. Hazzaa, F.; Hasan, M.M.; Qashou, A.; Yousef, S. A new lightweight cryptosystem for IoT in smart city environments. *Mesopotamian J. CyberSecur.* **2024**, *4*, 46–58. [CrossRef]
16. Altinay, A.; Nash, J.; Kroes, T.; Rajasekaran, P.; Zhou, D.; Dabrowski, A.; Gens, D.; Na, Y.; Volckaert, S.; Giuffrida, C.; et al. BinRec: Dynamic binary lifting and recompilation. In *EuroSys '20, Proceedings of the Fifteenth European Conference on Computer Systems, Heraklion, Greece, 27–30 April 2020*; Association for Computing Machinery: New York, NY, USA, 2020; pp. 1–16. [CrossRef]
17. Xia, Q.; Ji, H.; Zhou, Y.; Kim, N.S. Hardware-Accelerated Kernel-Space Memory Compression Using Intel QAT. *IEEE Comput. Archit. Lett.* **2025**, *24*, 57–60. [CrossRef]
18. Qasaimeh, M.; Al-Qassas, R.S.; Ababneh, M. Software Design and Experimental Evaluation of a Reduced AES for IoT Applications. *Future Internet* **2021**, *13*, 273. [CrossRef]
19. Carrascosa-Zamacois, M.; Geraci, G.; Knightly, E.; Bellalta, B. Wi-Fi Multi-Link Operation: An Experimental Study of Latency and Throughput. *IEEE/ACM Trans. Netw.* **2023**, *32*, 308–322. [CrossRef]

20. Gadhiya, N.; Tailor, S.; Degadwala, S. A Review on Different Level Data Encryption through a Compression Techniques. In Proceedings of the 2024 International Conference on Inventive Computation Technologies (ICICT), Lalitpur, Nepal, 24–26 April 2024; pp. 1378–1381. [CrossRef]
21. Vugera. Download Center–Blackbox Firmware Update Guide. Vugera. Available online: https://vugera.cafe24.com/article/%EC%9E%90%EB%A3%8C%EC%8B%A4/7/7249/ (accessed on 26 March 2025).
22. Pontus. ETC Firmware Resources. Pontus. Available online: http://pontus.co.kr/board/etc_fw/view/6272 (accessed on 26 March 2025).
23. Skyrex. Download Center–Blackbox Firmware. Skyrex. Available online: https://www.skyrexcctv.com/gb1/bbs/board.php?bo_table=tl_pds&wr_id=1204&sst=wr_hit&sod=asc&sop=and&page=1 (accessed on 26 March 2025).
24. ipTIME. Customer Support–Firmware Downloads. ipTIME. Available online: https://iptime.com/iptime/?uid=25839&mod=document&page_id=16 (accessed on 26 March 2025).
25. HDTNR. Download Center–Product Firmware. Available online: http://hdtnr.co.kr/download/?mod=document&uid=89 (accessed on 26 March 2025).
26. Zhu, B. AES-Python: Simple and Pure-Python Implementation of the AES Block Cipher. Available online: https://github.com/bozhu/AES-Python (accessed on 25 March 2025).
27. National Institute of Standards and Technology (NIST). *SP 800-131A Rev. 3 (Initial Public Draft): Transitioning the Use of Cryptographic Algorithms and Key Lengths*; NIST: Gaithersburg, MD, USA, 2024. Available online: https://csrc.nist.gov/pubs/sp/800/131/a/r3/ipd (accessed on 6 April 2025).
28. Hansel, G.; Perrin, D.; Simon, I. Compression and entropy. In *STACS 92*; Finkel, A., Jantzen, M., Eds.; Springer: Berlin/Heidelberg, Germany, 1992; pp. 513–528. [CrossRef]
29. The Big Image Compression Tool Comparison. Available online: https://compress-or-die.com/The-big-image-compression-tool-comparison (accessed on 9 November 2024).
30. Tutorial: Error Level. Analysis. Available online: https://www.fotoforensics.com/tutorial-estq.php (accessed on 5 November 2024).
31. Upadhyay, D.; Gaikwad, N.; Zaman, M.; Sampalli, S. Investigating the avalanche effect of various cryptographically secure hash functions and hash-based applications. *IEEE Access* **2022**, *10*, 112472–112486. [CrossRef]
32. Intel. Intel® Core™ i7-1360P Processor (18M Cache, up to 5.00 GHz)—Product Specifications. Available online: https://www.intel.com/content/www/us/en/products/sku/232155/intel-core-i71360p-processor-18m-cache-up-to-5-00-ghz/specifications.html (accessed on 30 March 2025).