Security of Infocommunication Systems and Internet of Things

2026 Vol 4, No 1

https://doi.org/10.31861/sisiot2026.1.01019

Received 04 June 2026; revised 23 June 2026; accepted 25 June 2026; published 30 June 2026

Static Frequency-Based Canonical Coding as a Zero-Overhead Compression and Data Obfuscation Method for LoRaWAN IoT Sensors

Oleksiy Polikarovskykh[1,*] and Ihor Hula[2]

[1]Department of Technical Cybernetics and Information Technologies, Odesa National Maritime University, Odesa, Ukraine
[2]Department of Physics, Mathematics and Informatics, Khmelnytskyi National University, Khmelnytskyi, Ukraine

*Corresponding author (E-mail: polalexey@gmail.com)

ABSTRACT This paper proposes Static Frequency-Based Canonical Coding (SFCC) and its obfuscating extension SFCC-S – lightweight lossless compression methods for numeric IoT sensor data transmitted over LoRaWAN networks. Classical adaptive compression algorithms such as Huffman coding require transmitting a frequency table or dictionary alongside the compressed payload, consuming 28 – 128 bytes of the constrained LoRaWAN SF12 payload limit of 51 bytes per packet. The proposed approach eliminates this overhead by pre-sharing a static canonical code table between the sensor node and the receiving server, directly extending the static-context philosophy of the IETF SCHC standard (RFC 9011) from network headers to application-layer payloads. SFCC assigns variable-length binary codewords to a 14-symbol numeric sensor alphabet based on expected symbol frequencies, achieving a coding efficiency of 96.3% relative to the Shannon entropy bound. When codebook overhead is included, SFCC reduces total transmitted bits by a factor of 2.97 compared to dynamic Huffman coding. Over a 24-hour IoT simulation at 60 measurements per hour, SFCC reduces the number of LoRaWAN packets by 52.4%, projecting approximately 2x improvement in battery lifetime. SFCC-S extends the base method with a zero-overhead symbol permutation that reduces the observability of payload structure; this mechanism does not provide cryptographic security but mitigates statistical and format-based information leakage in constrained LPWAN environments. Both methods require only a 32-entry static lookup table, making them suitable for any MCU-class device.

KEYWORDS data compression, LoRaWAN, Internet of Things, canonical coding, payload obfuscation.

I. INTRODUCTION

The rapid proliferation of Internet of Things (IoT) devices has fundamentally transformed environmental monitoring, precision agriculture, smart cities, and industrial automation. According to industry forecasts, the number of connected IoT devices is expected to exceed 25 billion by 2030 [1, 2], with the vast majority operating on battery power in locations where infrastructure access is limited or impractical. In such deployments, energy autonomy is the primary design constraint, and radio transmission consistently emerges as the dominant source of energy consumption – accounting for up to 60% of total device energy expenditure [3]. It has been estimated that transmitting a single bit of information requires energy equivalent to approximately 1,000 processor instructions [3], which fundamentally shapes the design philosophy for constrained IoT nodes.

Among the available wireless technologies for IoT, LoRa and its network protocol LoRaWAN have emerged as leading solutions for low-power, long-range applications. LoRa uses Chirp Spread Spectrum (CSS) modulation to achieve communication ranges of several kilometers at sub-mW transmission power levels, making it particularly suitable for battery-powered sensors deployed in geographically dispersed environments [1, 4, 5]. However, LoRaWAN imposes strict payload constraints: depending on the spreading factor selected, the maximum payload per packet ranges from 51 bytes (SF12) to 222 bytes (SF7) [3, 6, 7]. At SF12, which provides the greatest range and noise immunity, the usable payload is limited to 51 bytes – a constraint that renders many classical compression algorithms counterproductive.

Data compression is widely recognized as a promising strategy for reducing the number of LoRa transmissions and thereby extending battery lifetime [1, 3, 4]. However, the application of classical lossless compression to LoRaWAN sensor nodes faces a fundamental tension: algorithms that adapt their coding to the statistical properties of the data (Huffman, Arithmetic, LZW) require transmitting a frequency table or dictionary alongside the compressed payload. For a sensor alphabet of 14 symbols – typical for numeric environmental data – the Huffman frequency table alone occupies 28 bytes, more than half the available SF12 payload. For larger alphabets, this overhead reaches 74 – 128 bytes, completely eliminating any compression benefit and in many cases expanding the transmitted data [3, 4, 8]. Machine learning-based compressors (CMIX, PAQ8PX, LSTM-compress) achieve superior compression ratios but exhibit negative energy balances on constrained microcontrollers due to their computational demands [4]. Temporal compression methods (LTC, RT-LRbTC) eliminate codebook overhead but introduce reconstruction error and depend on data correlation, making them unsuitable where exact sensor values are required [1].

In practical IoT deployments, data compression and

protection are typically implemented through a combination of universal algorithms and protocol mechanisms. The most common approach uses LZ-oriented algorithms (particularly LZ77 or LZ4) for compression, AES-128 symmetric encryption for confidentiality, DTLS-like protocols for channel protection, and Base64 encoding for transportation within text-based or binary APIs. However, such approaches were designed for networks with relatively large packets and stable connections, and are poorly adapted to the specific characteristics of LPWANs.

The IETF standard RFC 9011 [9] has already demonstrated an effective resolution to the analogous problem in the domain of network header compression: Static Context Header Compression (SCHC) eliminates per-packet header transmission by pre-sharing a static rule context between the sensor node and the network server. This static context approach achieves zero per-packet overhead for compression metadata. However, RFC 9011 explicitly limits SCHC to network and transport layer headers (IPv6, UDP, CoAP) and designates application payload compression as out of scope, leaving an unaddressed gap at the application data level.

This paper proposes Static Frequency-Canonical Coding (SFCC) – a lossless, prefix-free coding scheme for numeric IoT sensor data that applies the static context philosophy of SCHC to the application payload. SFCC assigns variable-length binary codewords to sensor alphabet symbols based on their expected frequency in the data stream, following the principle established by Morse code: more frequent symbols receive shorter codes. Code lengths are determined a priori using the Kraft inequality as a feasibility constraint and assigned using the canonical Huffman algorithm, which guarantees prefix-free decoding. The resulting code table is pre-shared between the sensor node and the receiving server – identical to the SCHC approach – and is never transmitted over the radio channel, achieving zero per-packet overhead.

We further propose SFCC-S, an extension of SFCC that applies a secret permutation of symbols within each code-length group. Since canonical codes assign codewords based on symbol ordering within each length level, permuting this order reassigns codewords without altering code lengths or compression efficiency. The result is a codebook that functions simultaneously as a compression table and a lightweight obfuscation key, complicating statistical analysis of intercepted packets without the computational overhead of full cryptographic encryption.

The proposed methods are evaluated against ASCII encoding, dynamic Huffman coding, and SFCC without permutation on a 14-symbol sensor alphabet representing numeric environmental measurements (temperature, humidity, atmospheric pressure). Experiments are conducted on simulated LoRaWAN traffic over a 24-hour period at 60 measurements per hour, measuring compressed payload size, encoding and decoding time, Kraft sum, prefix-freeness, and LoRaWAN packet count under SF12 constraints.

The main contributions of this paper are: a formal problem statement for zero-overhead lossless payload compression in LoRaWAN sensor nodes, grounded in information-theoretic bounds (Shannon entropy, Kraft inequality); the SFCC method – a static, prefix-free, canonical coding scheme for fixed sensor alphabets requiring zero per-packet metadata transmission; the SFCC-S extension – a secret-permutation variant of SFCC providing lightweight obfuscation at no additional transmission cost; experimental demonstration that SFCC achieves compression efficiency of 96.3% relative to the Shannon entropy bound while reducing total transmitted bits by a factor of 3× compared to dynamic Huffman coding when codebook overhead is included in the comparison; an empirical analysis showing that SFCC and SFCC-S reduce the number of LoRaWAN SF12 packets by approximately 53% compared to ASCII encoding, doubling the effective battery lifetime of sensor nodes.

II. LITERATURE REVIEW AND PROBLEM STATEMENT

The problem of reducing transmission overhead in energy-constrained LoRa sensor nodes has attracted significant research attention. Väänänen and Hämäläinen [1] conducted a comprehensive experimental evaluation of temporal compression algorithms – Lightweight Temporal Compression (LTC) and Real-Time Linear Regression-based Temporal Compression (RT-LRbTC) – implemented on an Arduino MKR WAN 1310 board powered by a 2000 mAh LiPo battery. Their measurements demonstrated that radio transmission is the dominant energy consumer in LoRa sensor nodes, and that compression algorithms can significantly reduce the number of transmission periods without measurably increasing the energy cost of computation. With a spreading factor SF12 and a compression ratio CR = 10 (LTC), battery lifetime extended from 343 to 596 days. The authors concluded that the algorithmic complexity of lightweight temporal methods is negligible compared to transmission energy, making compression a viable strategy for extending device lifetime. However, their work focused exclusively on temporal (lossy) compression of continuous environmental data and did not address the overhead imposed by transmitting the compression codebook alongside the payload.

Oliveira Júnior et al. [3] conducted a comprehensive experimental study of five classical lossless compression algorithms – Arithmetic, Huffman, LZ77, LZ78, and LZW – adapted for IoT devices and evaluated on an ESP32-based TTGO LoRa module. Two real datasets were used: concrete temperature monitoring (28-byte messages, 14-symbol alphabet) and GPS coordinates (22-byte messages, 13-symbol alphabet [0–9, ., ,, -]). The study is notable for evaluating a multi-dimensional set of metrics simultaneously: compression rate, execution time, heap memory usage, peak current, and global energy gain – a combination absent in most prior work. Key findings include: LZW achieved the highest compression rate (69% for temperature, 63% for GPS) and energy reduction of up to 22% under SF12 with deep sleep mode; statistical algorithms (Huffman, Arithmetic) showed greater stability in compression rate than dictionary-based methods; and execution time and heap memory are correlated but not causally related to compression rate. The authors explicitly identified the adaptive Huffman table as a necessary

overhead – the frequency table must be transmitted with each compressed message – and proposed as future work the use of static probability tables to eliminate this overhead. This observation directly motivates the zero-overhead approach of the proposed SFCC method. The study also developed an analytical energy model validated against measurements with a mean error of 0.7%, confirming that energy savings from compression are significant only from SF10 and above, where transmission energy dominates compression cost. A broader comparative study by Dias et al. [4] evaluated seven lossless compression algorithms – Huffman, LZW, BSC, CMIX, PAQ8PX, GMIX, and LSTM-compress – applied to three IoT data types: GPS coordinates (14-symbol alphabet), diversified sensor readings (64-symbol alphabet), and logistics identifiers (37-symbol alphabet). Experiments were conducted on a Raspberry Pi 5 with an RFM95W LoRa module and INA219 power sensors for real-time energy measurement. Their key finding was that metadata overhead dominates payload efficiency for small LoRa packets: the dynamic Huffman frequency table requires 28 – 128 bytes depending on alphabet size, which exceeds or nearly fills the SF12 payload limit of 51 bytes. As a result, Huffman coding failed to compress any SF12 packets for logistics data (37-symbol alphabet, 74-byte table). LZW achieved the best energy efficiency among classical methods, reducing LoRa transmission energy by up to 7.41% for logistics data at SF12. In contrast, ML-based algorithms (CMIX, PAQ8PX) achieved superior compression ratios (>80%) but exhibited negative energy gains due to computational overhead. The authors explicitly noted that metadata overheads including dynamic Huffman tables (28–128 bytes) affect payload efficiency for small packets – a finding that directly motivates the need for a zero-overhead coding scheme where the codebook is pre-shared between transmitter and receiver and requires no in-band transmission. Recent studies consistently indicate that for LoRaWAN systems the decisive factor is not only compression optimality, but the elimination of metadata overhead and minimization of computational complexity.

The IETF standard RFC 9011 [9] defines a profile of SCHC and Fragmentation for LoRaWAN networks. SCHC operates on the principle that both communicating parties share a static compression context – a set of rules agreed upon prior to communication – eliminating the need to transmit header fields that can be inferred from context. The standard defines an 8-bit RuleID encoded in the LoRaWAN FPort field, which serves as an index into the shared rule set. This allows header compression to be performed with zero per-packet overhead for the compression context itself. SCHC demonstrates the practical feasibility and standardization of the static-context approach in LoRaWAN [9, 10]: the shared rule table is provisioned once and never retransmitted, achieving the same zero-overhead property that motivates the proposed SFCC method. However, SCHC targets protocol header compression (IPv6/UDP headers) rather than application payload compression, and does not address the entropy coding of sensor data values. SCHC deliberately avoids adaptive compression structures such as dynamically transmitted trees, dictionaries, or probability tables, favoring deterministic decompression with minimal implementation complexity.

The reviewed literature reveals a clear and consistent gap across all four sources. Oliveira Júnior et al. [3] explicitly proposed as future work the use of static probability tables for Huffman coding to eliminate the per-message codebook overhead – but did not implement or evaluate this approach. Väänänen and Hämäläinen [1] focused on lossy temporal methods that introduce reconstruction error. Dias et al. [4] confirmed that codebook overhead is a dominant limitation for small alphabets at SF12. RFC 9011 [9] demonstrated that static context sharing is standardized and effective for header compression but leaves application payload compression unaddressed. Existing payload compression methods for LoRa either: require dynamic codebook transmission (Huffman, LZW as typically implemented [3, 4]), consuming a significant fraction of the limited LoRa payload – up to 57% of the SF12 payload for a 14-symbol sensor alphabet; rely on lossy temporal approximation (LTC, RT-LRbTC [1]) that introduces reconstruction error and is unsuitable when exact values are required; target protocol headers rather than application data (SCHC/RFC 9011 [9]). No existing work proposes a lossless, prefix-free, zero-overhead payload coding scheme for numeric IoT sensor data that combines frequency-aware code length assignment with canonical code construction and optional symbol permutation for lightweight obfuscation. The proposed SFCC and SFCC-S methods address this gap directly, as summarized in Table 1.

TABLE 1. Comparison of related approaches.

Method	Lossless	Zero overhead	Prefix-free	Obfuscation	Target
LTC/RT-LRbTC [1]	No	Yes	–	No	Temporal sensor data
Huffman/LZW [3]	Yes	No	Yes	No	Payload (ESP32)
Huffman/LZW [4]	Yes	No	Yes	No	Payload (RPi5)
SCHC [9]	Yes	Yes	Yes	No	Protocol headers
SFCC	Yes	Yes	Yes	No	Sensor payload
SFCC-S	Yes	Yes	Yes	Yes	Sensor payload

In this sense, SFCC can be interpreted as a conceptual extension of the SCHC static-context philosophy from protocol headers to application-layer payload compression.

Thus, there is a need for a method that: introduces zero or nearly zero overhead; does not require the transmission of tables or dictionaries; employs an extremely simple decoder suitable for MCUs; provides compression for fixed sensor alphabets; and simultaneously complicates structural data analysis without relying on full cryptographic

robustness.

This niche is addressed by the proposed static canonical code with a fixed (potentially secret) table, which conceptually extends the philosophy of SCHC (RFC 9011) from the network-header level to the application-payload level. Unlike adaptive algorithms, the proposed approach does not require the transmission of dynamic trees, frequency tables, or dictionaries; instead, it relies on a pre-agreed static context that enables deterministic decoding with minimal computational overhead. As a result, the method is designed not only to achieve high compression efficiency, but also to minimize the overall transmission cost under the stringent LoRaWAN constraints on payload size, energy consumption, and MCU computational resources.

III. SCOPE OF WORK AND OBJECTIVES

This work proposes SFCC – a lossless, prefix-free payload coding scheme for numeric IoT sensor data that extends the static-context philosophy of SCHC to the application layer. The method assigns variable-length codewords to sensor alphabet symbols based on expected frequency, following the Morse code principle that more frequent symbols receive shorter codes. The resulting code table is pre-shared between the sensor node and the receiving server and is never transmitted over the radio channel, achieving zero per-packet overhead. An extension, SFCC-S, applies a secret permutation of symbols within each code-length group, parametrized by a pre-shared key. This provides lightweight obfuscation without altering compression efficiency or adding transmission overhead.

The specific objectives of this work are: to formally define the zero-overhead lossless compression problem for LoRaWAN sensor payloads; to construct the SFCC and SFCC-S code tables for a 14-symbol numeric sensor alphabet and verify their prefix-freeness and Kraft inequality compliance; to experimentally compare SFCC and SFCC-S against ASCII encoding and dynamic Huffman coding in terms of total transmitted bits, LoRaWAN packet count, and encoding speed; to demonstrate that the proposed methods reduce total transmission cost by a factor of 3× relative to dynamic Huffman coding when codebook overhead is included in the comparison.

IV. PROPOSED METHOD (SFCC / SFCC-S)

The proposed method is based on static canonical prefix coding for the transmission of IoT sensor telemetry data with a fixed symbol alphabet.

Unlike classical adaptive Huffman coding, the proposed approach does not require the transmission of frequency tables, coding trees, or dictionaries, since the codebook is pre-agreed between the transmitter and the receiver prior to deployment. The target application is environmental monitoring: sensor nodes transmit readings of the form $m_i = (T_i, H_i, P_i)$, where T_i is temperature (°C), H_i is relative humidity (%), and P_i is atmospheric pressure (hPa), formatted as decimal ASCII strings. Let the sensor alphabet be defined as the set

$$\mathcal{A} = \{a_1, a_2, ..., a_N\}, \quad (1)$$

where N denotes the number of valid symbols in the sensor messages. In this work, the alphabet is defined as

$$\mathcal{A} = \{0,1,2,...,9,.,,,-,\text{space}\}, \quad N = 14.$$

For each symbol a_i a fixed codeword length is assigned:

$$l_i \in \mathbb{N},$$

where: l_i is the length of the binary codeword for symbol a_i; $\mathbb{N}$ is the set of natural numbers.

In the SFCC method, codeword lengths are defined statically based on prior analysis of typical IoT data and symbol usage frequency. More frequent symbols are assigned shorter codewords. For the considered sensor alphabet, the following set of codeword lengths is used:

$$L = \{3, 4, 5\},$$

where:

- frequently used symbols (., ,, space, 1) are assigned 3-bit codewords (n_3=4);
- moderately frequent symbols (-, 0, 2, 3, 4) are assigned 4-bit codewords (n_4=5);
- rare symbols (5, 6, 7, 8, 9) are assigned 5-bit codewords (n_5=5).

Prefix validity is ensured by construction, as the proposed method follows canonical prefix coding rules. The assigned code lengths satisfy Kraft's inequality [11, 12]:

$$\sum_{i=1}^{N} 2^{-l_i} \le 1. \quad (2)$$

For the proposed length assignment, the Kraft sum evaluates to:

$$K = \frac{n_3}{2^3} + \frac{n_4}{2^4} + \frac{n_5}{2^5} = \frac{4}{8} + \frac{5}{16} + \frac{5}{32} = 0.969 \le 1,$$

where: n_l is the number of symbols assigned codeword length l, K = 0.969 confirms the existence of a valid prefix-free code, the reserve 1 - K = 0.031 corresponds to one unused 5-bit slot, available for future alphabet extension without modifying existing codewords.

Canonical code generation is performed by sorting symbols according to codeword length and lexicographical order:

$$(a_i, l_i) \rightarrow \text{sorted}(l_i, a_i).$$

The initial codeword is defined as the smallest binary value of length l_1:

$$c_1 = 0^{l_1},$$

where l_1 is the length of the first codeword (the shortest assigned length).

For each subsequent symbol, the codeword is computed recursively [12, 13] as:

$$c_i = (c_{i-1} + 1) << (l_i - l_{i-1}), \quad (3)$$

where: c_i is the codeword of the i-th symbol in sorted order, c_{i-1} is the codeword of the previous symbol, ≪ denotes the left bit-shift operation, l_i is the length of the current codeword, l_{i-1} is the length of the previous codeword.

Each codeword is represented as a binary string of fixed length l_i using leading zeros if necessary. The resulting SFCC and SFCC-S code tables are presented in Table 2.

TABLE 2. SFCC and SFCC-S Code Tables for the 14-Symbol Sensor Alphabet.

Symbol	p_i	SFCC	SFCC-S
'.'	0.164	000	011
','	0.148	001	000
'space'	0.148	010	010
'1'	0.066	011	001
'-'	0.049	1000	1011
'0'	0.049	1001	1001
'2'	0.082	1010	1000
'3'	0.066	1011	1100
'4'	0.033	1100	1010
'5'	0.049	11010	11100
'6'	0.033	11011	11110
'7'	0.033	11100	11011
'8'	0.066	11101	11101
'9'	0.016	11110	11010

Message encoding is performed as the concatenation of codewords:

$$B = c(x_1) \| c(x_2) \| \ldots \| c(x_M), \quad (4)$$

where: B is the resulting bitstream, x_k is the k-th symbol of the input message, k = 1, 2, …, M, M is the total number of symbols in the message, $c(x_k)$ is the codeword of symbol x_k, || denotes bit-string concatenation.

Decoding is performed by maintaining a bit buffer b, appending bits sequentially, and checking after each appended bit whether $b \in T^{-1}$:

$$T^{-1} : c_i \rightarrow a_i,$$

where: T^{-1} is the pre-shared inverse lookup table, c_i is the binary codeword, a_i is the corresponding decoded symbol.

Since the code is prefix-free, no lookahead or backtracking is required. The maximum buffer length is l_{max} = 5 bits, so the decoder requires a lookup table of only 2^5 = 32 entries, occupying 32 bytes of static memory – suitable for any MCU-class device.

The average codeword length [11, 13] is defined as:

$$\bar{L} = \sum_{i=1}^{N} p_i l_i, \quad (5)$$

where: p_i is the probability of occurrence of symbol a_i, l_i is the corresponding codeword length.

For the proposed length assignment: $\bar{L} = 3.672$ bits/symbol. The source entropy is evaluated using Shannon's formula [11, 13]:

$$H = -\sum_i p_i \log_2 p_i = 3.535 \text{ bits/symbol}, \quad (6)$$

where: H is the source entropy in bits per symbol, p_i is the empirical probability of symbol a_i.

The proposed length assignment satisfies Shannon's source coding theorem [11, 14]:

$$H \le \bar{L} \le H + 1, \; 3.535 \le 3.672 \le 4.535,$$

confirming near-optimal coding efficiency of

$$\eta = \frac{H}{\bar{L}} \times 100\% = \frac{3.535}{3.672} \times 100\% = 96.3\%.$$

The SFCC-S (Static Frequency-Canonical Coding with Secret Permutation) method extends SFCC by introducing a secret permutation of symbols within groups of identical codeword lengths.

Let

$$\mathcal{G}_k = \{a_i : l_i = k\}, \quad (7)$$

denote the group of symbols assigned codeword length k, where $k \in \{3, 4, 5\}$. For each group, a secret permutation is defined:

$$\pi_k : \mathcal{G}_k \rightarrow \mathcal{G}_k, \quad (8)$$

where: π_k is a secret ordering of symbols in group $\mathcal{G}_k$, k is the codeword length index.

Unlike standard SFCC, where symbols within each group are ordered lexicographically, SFCC-S assigns codewords according to the secret permutation π_k. This reassigns which symbol receives which codeword without altering code lengths, Kraft sum, or prefix-freeness. The total key space of SFCC-S is:

$$|\mathcal{K}| = |\mathcal{G}_3|! \cdot |\mathcal{G}_4|! \cdot |\mathcal{G}_5|! = 4! \cdot 5! \cdot 5! = 345{,}600,$$

where: $|\mathcal{G}_k|!$ is the number of permutations of group $\mathcal{G}_k$, 4! = 24 is the number of permutations of the 4 symbols at length 3, 5! = 120 is the number of permutations of the 5 symbols at lengths 4 and 5 respectively. This corresponds to approximately $\log_2(345{,}600) \approx 18.4$ bits of key entropy [14], sufficient to disrupt pattern-based traffic analysis of captured payloads. The SFCC-S variant introduces a zero-overhead symbol permutation that reduces the observability of payload structure. It is important to note that this mechanism does not provide cryptographic security and is not intended to replace encryption, but rather to mitigate statistical and format-based information leakage in highly constrained LPWAN environments. SFCC-S is therefore intended as a lightweight payload obfuscation layer complementary to the existing LoRaWAN AES-128 session encryption, reducing directly observable structural information in transmitted payloads without adding transmission or computational overhead. SFCC-S preserves all properties of the base method: fixed codeword lengths per symbol group; prefix-free property and Kraft inequality; canonical construction within each length group; zero per-packet transmission overhead.

Since SFCC and SFCC-S do not transmit auxiliary tables or coding trees, the total transmission cost is determined solely by the encoded bitstream length:

$$C_{\text{SFCC}} = |B|,$$

where $|B|$ is the length of the encoded bitstream in bits. In contrast, dynamic Huffman coding requires transmitting the frequency table alongside the compressed data:

$$C_{\text{Huffman}} = |B_H| + |H_{\text{table}}|,$$

where $|H_{\text{table}}|$ is the codebook overhead. For the 14-symbol sensor alphabet, a standard implementation requires 14 × 4 = 56 bytes = 448 bits for the frequency table. For the evaluated test message of 61 symbols, the total transmission costs are:

$$C_{\text{Huffman}} = 218 + 448 = 666 \text{ bits}, \; C_{\text{SFCC}} = 224 \text{ bits},$$

yielding a factor-of-three reduction in total transmitted bits in favour of SFCC when codebook overhead is included in the comparison:

$$\frac{C_{\text{Huffman}}}{C_{\text{SFCC}}} = \frac{666}{224} = 2.97\,. \qquad (9)$$

Due to the absence of headers, minimal memory requirements, and deterministic decoding, SFCC and SFCC-S are well suited for LoRaWAN, embedded IoT devices, and MCU-based systems with strict constraints on payload size, energy consumption, and computational resources.

V. EXPERIMENTAL RESULTS

The proposed SFCC and SFCC-S methods were evaluated using a Python-based simulation implemented on a representative numeric sensor dataset. The test message consists of ten comma-separated floating-point sensor readings: 23.5, -12.3, 98.6, 45.2, -3.7, 100.0, 22.8, -15.4, 67.3, 88.1 comprising M = 61 symbols drawn from the 14-symbol alphabet $\mathcal{A}$. Four coding methods were compared: ASCII (8 bits per symbol, baseline), dynamic Huffman coding with adaptive frequency table, SFCC, and SFCC-S. All encoding and decoding operations were verified for correctness. Benchmark timing was performed over 10,000 repetitions using time.perf_counter() to obtain stable microsecond-level measurements. The IoT transmission simulation was conducted over a 24-hour period at a rate of 60 measurements per hour (1,440 total measurements), generating approximately 23,600 bytes of raw sensor data. The LoRaWAN SF12 payload constraint of 51 bytes per packet was applied throughout.

The empirical symbol frequencies, ranging from p = 0.016 for '9' to p = 0.164 for '.', confirm the suitability of the proposed length assignment presented in Table 2. The three most frequent symbols – '.', ',', and 'space' – account for 47.5% of all characters, justifying the assignment of 3-bit codewords to this group. The Shannon entropy of the observed distribution is H = 3.535 bits/symbol.

Table 3 summarizes the compression results for the test message. All three compressed methods achieve significant payload reduction compared to ASCII. Dynamic Huffman coding achieves the lowest raw data size (218 bits, 55.3% compression), slightly outperforming SFCC and SFCC-S (224 bits, 54.1%) due to its data-adaptive code lengths. However, this advantage is reversed when the codebook transmission overhead is included.

TABLE 3. Compression Efficiency Comparison (61-symbol test message).

Method	Bits	Header (bits)	Total (bits)	$\bar{L}$ (bit/sym)	η
ASCII	488	0	488	8.000	44.2%
Huffman (dynamic)	218	448	666	3.574	98.9%
Huffman (canonical)	218	~112	~330	3.574	98.9%
SFCC	224	0	**224**	3.672	96.3%
SFCC-S	224	0	**224**	3.672	96.3%

H = 3.535 bit/symbol, theoretical minimum = 215 bits

It should be noted that the reported Huffman overhead corresponds to a straightforward adaptive implementation transmitting explicit frequency counts. In practice, more compact variants such as canonical Huffman coding reduce this overhead by transmitting only codeword lengths rather than full frequency tables. For the considered alphabet (|Σ| = 14), canonical Huffman coding requires transmitting only the codeword lengths rather than full frequency counts, resulting in a side-information overhead of 14 symbols × 1 byte per length value = 14 bytes = 112 bits. Even under this optimized assumption, SFCC (224 bits) remains 32% more efficient, since it requires no signaling overhead whatsoever. This confirms that the advantage of SFCC is structural rather than implementation-specific: the elimination of all per-packet metadata is the decisive factor in short-payload LPWAN scenarios where minimizing end-to-end transmitted bits – rather than payload entropy alone – is the primary optimization objective.

This result is illustrated in Fig. 1, which shows the total transmission cost per method including header overhead. SFCC and SFCC-S achieve the lowest total cost, outperforming even the uncompressed ASCII baseline by a factor of 488 / 224 = 2.18×. The Huffman bar in Fig. 1 represents the adaptive implementation including full frequency table overhead; the canonical Huffman variant (~330 bits total) is discussed separately below.

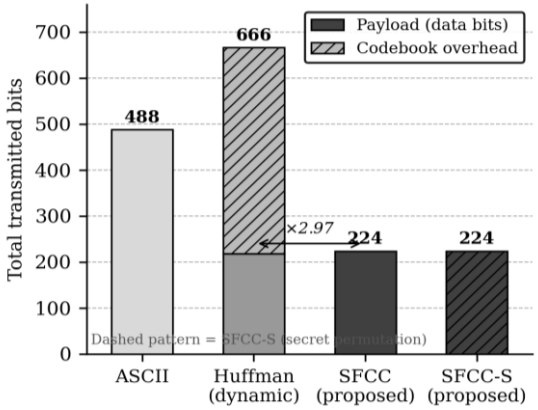

FIG. 1. Total transmission cost including codebook overhead. SFCC and SFCC-S transmit 224 bits compared to 666 bits for dynamic Huffman coding.

All three non-ASCII methods were verified to satisfy the Kraft inequality. Huffman coding achieves the theoretical maximum K = 1.000000, while SFCC and SFCC-S yield K = 0.969, leaving a reserve of one 5-bit slot for future alphabet extension. Prefix-freeness was verified for all methods, and lossless round-trip decoding $(\mathcal{D}(\mathcal{E}(x)) = x)$ was confirmed for all four methods on the test message.

Table 4 presents the encoding and decoding throughput measured over 10,000 repetitions. SFCC and SFCC-S achieve encoding times of 7.06 and 7.14 µs respectively, comparable to dynamic Huffman (6.83 µs) and significantly faster than ASCII (21.60 µs), which performs 8 format operations per symbol.

TABLE 4. Encoding and Decoding Speed (10,000 repetitions, 61-symbol message).

Method	Encoding (µs)	Decoding (µs)
ASCII	21.60	17.60
Huffman	6.83	31.11
SFCC	7.06	31.42
SFCC-S	7.14	31.38

Decoding times are comparable across all compressed methods (31.1 – 31.4 μs), reflecting the shared bit-by-bit prefix lookup strategy. The constant decoding time of SFCC and SFCC-S, independent of message content, is a desirable property for real-time embedded applications. Notably, SFCC-S introduces no measurable decoding overhead compared to SFCC, since the secret permutation affects only the pre-shared table and has no runtime cost.

The 24-hour IoT simulation confirms the practical impact of the proposed methods at scale. Table 5 presents the results for 1,440 measurements totalling 23,602 bytes of raw sensor data.

TABLE 5. IoT Simulation Results (24 hours, 60 measurements/hour, LoRaWAN SF12, 51-byte payload limit).

Method	Bytes	Packets	Saving	w/ header
ASCII	23,602	462	0.0%	23,602 bytes
Huffman	11,204	219	52.5%	11,260 bytes
SFCC	11,226	220	52.4%	11,226 bytes
SFCC-S	11,226	220	52.4%	11,226 bytes

SFCC and SFCC-S reduce the number of LoRaWAN packets from 462 (ASCII) to 220 – a reduction of 52.4% – while transmitting fewer total bytes than Huffman (11,226 vs. 11,260) once codebook overhead is accounted for. The compression ratio of approximately 2× directly corresponds to a proportional reduction in radio-on time and, by extension, a projected battery lifetime improvement of approximately 2× compared to uncompressed ASCII transmission.

V. CONCLUSION

This paper proposed SFCC and its obfuscating extension SFCC-S as lightweight lossless compression methods for numeric IoT sensor data transmitted over LoRaWAN networks. The key design principle – pre-sharing a static code table between the sensor node and the receiving server, eliminating all per-packet codebook overhead – directly extends the static-context philosophy of the IETF SCHC standard (RFC 9011) from network headers to application-layer payloads.

The experimental evaluation on a 14-symbol numeric sensor alphabet demonstrated the following results: SFCC achieves a coding efficiency of 96.3% relative to the Shannon entropy bound ($\bar{L}$ = 3.672 vs. H = 3.535 bits/symbol), with lossless decoding verified for all test cases; when codebook overhead is included in the comparison, SFCC reduces total transmitted bits by a factor of 2.97× relative to dynamic Huffman coding (224 vs. 666 bits), despite a marginally higher raw compressed size; even when compared against the more compact canonical Huffman variant (~330 bits), SFCC (224 bits) remains 32% more efficient due to the complete absence of per-packet signaling overhead; over a 24-hour IoT simulation at 60 measurements per hour, SFCC reduces the number of LoRaWAN SF12 packets by 52.4% (from 462 to 220), projecting an approximately 2× improvement in sensor node battery lifetime; encoding speed of SFCC (7.06 μs) is comparable to dynamic Huffman (6.83 μs) and requires only a 32-entry static lookup table, making it suitable for any MCU-class device; SFCC-S introduces a secret symbol permutation as a lightweight payload obfuscation layer that disrupts pattern-based traffic analysis; it does not provide cryptographic security but adds no transmission overhead and no measurable runtime cost compared to SFCC.

The proposed methods address a gap identified consistently across related work [1, 3, 4, 9]: no existing lossless method simultaneously achieves zero per-packet overhead, prefix-free decoding, and MCU-compatible implementation for fixed sensor alphabets in LoRaWAN. SFCC fills this gap as a direct application-layer counterpart to SCHC header compression.

The proposed SFCC scheme assumes a stationary symbol distribution and a fixed sensor data format, and is therefore most effective for structured numeric telemetry with predictable character frequencies. Its compression efficiency may degrade if the source statistics deviate significantly from the assumed model – for example, when transmitting alphanumeric identifiers or variable-format messages. In such cases, the static code table should be recalibrated based on representative corpus data prior to deployment. Furthermore, SFCC-S provides only lightweight payload obfuscation and should not be considered as a cryptographic protection mechanism; full confidentiality requires dedicated encryption such as LoRaWAN AES-128 session keys.

Future work will extend the evaluation to physical LoRaWAN hardware (ESP32, STM32) to measure actual energy consumption and radio-on time savings. Additional directions include adaptive alphabet extension using the K = 0.031 Kraft reserve, evaluation on larger sensor alphabets including alphanumeric IoT protocols, and integration of SFCC-S with LoRaWAN session key management for coordinated key rotation.

AUTHOR CONTRIBUTIONS

O.P. – conceptualization, methodology, supervision, investigation, writing – review and editing; I.H. – software, validation, visualization, writing – original draft.

COMPETING INTERESTS

The authors declare no competing interests.

REFERENCES

[1] O. Väänänen and T. Hämäläinen, “Efficiency of temporal sensor data compression methods to reduce LoRa-based sensor node energy consumption,” *Sensor Review*, vol. 42, no. 5, pp. 503–516, 2022. doi: 10.1108/SR-10-2021-0360.

[2] M. A. Almuhaya, W. A. Jabbar, N. Sulaiman, and S. Abdulmalek, “A survey on LoRaWAN technology: Recent trends, opportunities, simulation tools and future directions,” *Electronics*, vol. 11, no. 1, Art. no. 164, 2022. doi: 10.3390/electronics11010164.

[3] J. A. Oliveira Júnior, E. T. de Camargo, and M. S. Oyamada, “Data compression in LoRa networks: A compromise between performance and energy consumption,” *Journal of Internet Services and Applications*, vol. 14, no. 1, pp. 95–106, 2023. doi: 10.5753/jisa.2023.3000.

[4] R. L. Dias, E. C. Vilas Boas, F. A. P. de Figueiredo, S. B. Mafra, and M. Ahmed Ouameur, “Data compression in LoRa networks: Performance and energy trade-offs of classical and cutting-edge compression algorithms,” *Sensors*, vol. 26, no. 5, Art. no. 1414, 2026. doi: 10.3390/s26051414.

[5] R. S. Sinha, Y. Wei, and S. H. Hwang, “A survey on LPWA technology: LoRa and NB-IoT,” *ICT Express*, vol. 3, no. 1, pp. 14–21, 2017. doi: 10.1016/j.icte.2017.03.004.
[6] LoRa Alliance, “LoRaWAN Regional Parameters, RP002-1.0.3,” 2021. [Online]. Available: https://lora-alliance.org/wp-content/uploads/2021/05/RP002-1.0.3-FINAL-1.pdf
[7] P. Gkotsiopoulos, D. Zorbas, and C. Douligeris, “Performance determinants in LoRa networks: A literature review,” *IEEE Communications Surveys & Tutorials*, vol. 23, no. 3, pp. 1721–1758, 2021. doi: 10.1109/COMST.2021.3090409.
[8] K. L. Ketshabetswe, A. M. Zungeru, B. Mtengi, C. K. Lebekwe, and S. Prabaharan, “Data compression algorithms for wireless sensor networks: A review and comparison,” *IEEE Access*, vol. 9, pp. 136872–136891, 2021. doi: 10.1109/ACCESS.2021.3116311.
[9] O. Gimenez and I. Petrov, “Static context header compression and fragmentation (SCHC) over LoRaWAN,” *IETF RFC 9011*, 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9011
[10] J. Sanchez-Gomez, J. Gallego-Madrid, R. Sanchez-Iborra, J. Santa, and A. F. Skarmeta, “Impact of SCHC compression and fragmentation in LPWAN: A case study with LoRaWAN,” *Sensors*, vol. 20, no. 1, Art. no. 280, 2020. doi: 10.3390/s20010280.
[11] T. M. Cover and J. A. Thomas, *Elements of Information Theory*, 2nd ed. Hoboken, NJ, USA: Wiley-Interscience, 2006. doi: 10.1002/047174882X.
[12] K. Sayood, *Introduction to Data Compression*, 5th ed. Morgan Kaufmann, 2017.
[13] A. Moffat, “Huffman coding,” *ACM Computing Surveys*, vol. 52, no. 4, Art. no. 85, 2019. doi: 10.1145/3342555.
[14] D. Salomon, *Data Compression: The Complete Reference*, 4th ed. Springer, 2007. doi: 10.1007/978-1-84628-603-2.

Oleksiy Polikarovskykh

In 2015,he received a Doctor of Science degree (D.Sc. in Engineering) at the State University of Intellectual Technologies and Communication (Odessa, Ukraine) in the field of signal synthesis in telecommunication systems. In 2019 Full Professor of the Department of Telecommunications Technologies, Khmelnitsky National University (Khmelnitsky, Ukraine). Currently, Full Professor of the Department of Technical Cybernetics and Information Technologies, Odessa National Maritime University (Odessa, Ukraine). Research includes issues related to the development of devices for signal synthesis, the theory of cybersecurity and Software Defined Radio.

ORCID ID: 0000-0002-1893-7390

Ihor Hula

In 2014, he received a PhD in Engineering from Vinnytsia National Technical University (Vinnytsia, Ukraine), specializing in radio measuring devices. In 2020, he became an Associate Professor at the Department of Physics and Electrical Engineering, Khmelnytskyi National University (Khmelnytskyi, Ukraine). He is currently an Associate Professor at the Department of Physics, Mathematics, and Informatics at Khmelnytskyi National University. His research interests include radio measurement, the development of signal synthesis devices, and software-defined radio.

ORCID ID: 0000-0002-4434-5794

Статично-частотне канонічне кодування як метод стиснення без накладних витрат і обфускації даних IoT-сенсорів у мережах LoRaWAN

Олексій Полікаровських[1,*], Ігор Гула[2]

[1]Кафедра технічної кібернетики та інформаційних технологій, Одеський національний морський університет, Одеса, Україна
[2]Кафедра фізики, математики та інформатики, Хмельницький національний університет, Хмельницький, Україна

*Автор-кореспондент (Електронна адреса: polalexey@gmail.com)

АНОТАЦІЯ У статті запропоновано метод Static Frequency-Based Canonical Coding (SFCC) та його розширення з функцією обфускації SFCC-S — легковагові методи безвтратного стиснення числових даних IoT-сенсорів, що передаються через мережі LoRaWAN. Класичні адаптивні алгоритми стиснення, такі як кодування Хаффмана, потребують передавання разом зі стисненим повідомленням таблиці частот або словника, що займає від 28 до 128 байт обмеженого корисного навантаження LoRaWAN, тоді як для пакета SF12 максимальний розмір корисного навантаження становить лише 51 байт. Запропонований підхід усуває ці накладні витрати шляхом попереднього спільного використання статичної канонічної кодової таблиці між сенсорним вузлом і сервером приймання даних, безпосередньо поширюючи концепцію статичного контексту стандарту SCHC IETF (RFC 9011) з мережевих заголовків на корисні дані прикладного рівня. Метод SFCC призначає двійкові кодові слова змінної довжини символам 14-елементного числового алфавіту сенсорних даних відповідно до очікуваних частот їх появи, досягаючи ефективності кодування 96,3 % відносно межі ентропії Шеннона. З урахуванням накладних витрат на передавання кодової книги SFCC забезпечує зменшення загальної кількості переданих бітів у 2,97 раза порівняно з динамічним кодуванням Хаффмана. Під час 24-годинного моделювання IoT-системи з частотою 60 вимірювань на годину метод SFCC скорочує кількість пакетів LoRaWAN на 52,4 %, що потенційно забезпечує приблизно двократне збільшення часу автономної

роботи батареї. Метод SFCC-S розширює базовий підхід за рахунок перестановки символів без додаткових накладних витрат, що знижує спостережуваність структури корисного навантаження; цей механізм не забезпечує криптографічного захисту, проте зменшує витік статистичної та форматно-структурної інформації в обмежених мережах LPWAN. Обидва методи потребують лише статичної таблиці пошуку з 32 записів, що робить їх придатними для реалізації на будь-яких пристроях класу MCU.

КЛЮЧОВІ СЛОВА стиснення даних, LoRaWAN, Інтернет речей, канонічне кодування, обфускація корисного навантаження.