mathematics

MDPI

Article

Minimal-Polynomial-Based Lightweight Cryptography for IoT and Wireless Hardware: Security Analysis and FPGA Implementation

Parisa Ebrahimpour Moghaddam Tasouj [1,*] and Orhan Gazi [2]

[1] Biomedical Device Technology, Vocational School of Health Services, Ankara Medipol University, Ankara 06050, Türkiye
[2] Department of Electrical and Electronics Engineering, Ankara Medipol University, Ankara 06050, Türkiye; orhan.gazi@ankaramedipol.edu.tr
* Correspondence: parisa.ebrahimpour@ankaramedipol.edu.tr

Abstract

This paper proposes a lightweight polynomial-based encryption architecture for resource-constrained IoT and wireless communication hardware. Instead of substitution boxes, the method uses key-dependent polynomial multiplication with minimal polynomials over finite fields. Plaintext rows are XORed with secret keys and a nonce-dependent pseudorandom mask and multiplied by a minimal polynomial; multiplication by a complementary factor recovers the message exactly. A security analysis shows that, because the transformation is linear over $GF(2)$, reuse of the same key and polynomial allows for known- and chosen-plaintext key recovery; per-message nonce masking is therefore required. Software experiments (four message lengths, 100 trials) verified lossless decryption. The encryption time of the key-assisted method was comparable to that of SPECK64/128 at the same implementation level, whereas AES-128 was faster for long messages, and the avalanche ratio of the proposed method (0.070) was well below the value of about 0.5 obtained by the reference ciphers. On a Xilinx Artix-7 FPGA, the proposed core used 1211 LUTs and 205 registers without block RAM or DSPs and reached 154.1 MHz, compared with 2305 LUTs and 2733 registers for an iterative AES-128 core. The method is a compact, S-box-free polynomial processing layer rather than a stand-alone replacement for standard ciphers.

Keywords: lightweight encryption; FPGA implementation; wireless communication security; polynomial-based cryptography; IoT security

MSC: 94A60

Check for updates

Academic Editors: Cheng-Chi Lee and Lingfeng Liu

Received: 20 September 2026
Revised: 22 October 2026
Accepted: 2 November 2026
Published: 6 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rapid proliferation of wireless communication networks, cloud computing platforms, and blockchain-based systems has created an urgent and sustained demand for robust cryptographic mechanisms that can guarantee data confidentiality, integrity, and authenticity. Modern digital ecosystems rely on the secure transmission and storage of increasingly sensitive data, spanning personal health records, financial transactions, and authentication credentials, across heterogeneous networks exposed to a growing variety of adversarial threats [1]. As the volume and sensitivity of transmitted data continue to grow, so does the sophistication of attacks targeting cryptographic implementations, making the

design and analysis of new encryption methods a matter of critical importance to both academia and industry.

The Advanced Encryption Standard (AES), standardized by the National Institute of Standards and Technology (NIST) in 2001, has served as the dominant symmetric encryption algorithm across virtually every security-sensitive domain, including Wi-Fi (IEEE 802.11) [2], 4G/5G mobile networks, virtual private networks (VPNs), cloud storage, and blockchain applications [3]. The AES operates on 128-bit data blocks using key sizes of 128, 192, or 256 bits, and its design is fundamentally grounded in arithmetic within the binary extension field $\mathrm{GF}(2^8)$. The SubBytes transformation, in particular, relies on a fixed substitution box (S-box) constructed using multiplicative inverses over $\mathrm{GF}(2^8)$. At the same time, the MixColumns step treats each column of the state matrix as a polynomial over $\mathrm{GF}(2^8)$ and multiplies it modulo an irreducible polynomial [4].

Despite its continued prominence and resilience against classical cryptanalytic attacks, the emerging threat of quantum computing, and specifically the threat posed by quantum algorithms such as Grover's algorithm, which effectively halves AES key-security levels, has motivated an active search for new cryptographic paradigms that are either quantum-resistant or offer alternative structural properties that complement existing standards [5].

Extended Galois fields, denoted by $\mathrm{GF}(2^m)$, occupy a central role in modern cryptographic engineering. Their algebraic structure, characterized by a finite set of elements closed under addition and multiplication modulo an irreducible polynomial, provides a mathematically tractable framework for designing operations with guaranteed invertibility and strong diffusion properties [6]. Beyond the AES, Galois fields are foundational to elliptic curve cryptography (ECC), error-correcting codes such as Reed–Solomon and BCH codes, and the Galois/Counter Mode (GCM) of authenticated encryption, which is widely deployed in TLS, IPsec, and IEEE 802.11 wireless security protocols [7]. Their computational efficiency in hardware, especially for binary extension fields where addition reduces to XOR, makes them particularly attractive for resource-constrained environments such as IoT edge devices and wireless sensor nodes [8].

Blockchain networks present a distinct and increasingly important cryptographic context. Unlike traditional client–server architectures, blockchain systems require encryption schemes that can be efficiently executed across distributed, decentralized nodes with heterogeneous computational resources, while still providing strong guarantees against tampering and unauthorized disclosure [9]. Recent research has explored the application of dynamic AES variants, hybrid AES-ECC frameworks, and Galois field-based dynamic encryption algorithms (DGFEAs) for securing data in blockchain and 6G-enabled IoT ecosystems [10,11]. These investigations underscore the need for flexible encryption constructions that can scale across different Galois field orders, accommodate varying data frame sizes, and resist both classical and emerging quantum adversaries.

Minimal polynomials of a Galois field $\mathrm{GF}(2^m)$ are the monic irreducible polynomials over $\mathrm{GF}(2)$ whose roots are elements of $\mathrm{GF}(2^m)$. Every element of the field is a root of exactly one minimal polynomial, and the roots of each minimal polynomial form a conjugacy class; these roots need not be primitive elements. Minimal polynomials play a structural role in defining the factorization of $x^n + 1$. While their role in coding theory, particularly in the construction of cyclic and BCH codes, is well established [6], their direct application as encryption kernels has received little attention. The present work investigates whether multiplication by selected minimal polynomials can serve as a compact, S-box-free processing layer for resource-constrained hardware, and evaluates its correctness, security limitations, and implementation cost.

https://doi.org/10.3390/math14214082

2. Related Work

2.1. Galois Field Theory in Cryptographic Applications

The mathematical foundations of Galois field-based cryptography trace back to the standardization of the AES (Rijndael), which established $GF(2^8)$ arithmetic as the bedrock of modern symmetric block cipher design. Subsequent research has extended these foundations in multiple directions. A comprehensive survey published in the International Journal of Information Security (2025) [3] provides a panoramic review of the AES, from its algebraic underpinnings in Galois field arithmetic through its resistance to differential and linear cryptanalysis, while also mapping the trajectory toward post-quantum hybrid extensions. In the domain of IoT, a 2025 survey of lightweight cryptographic schemes published in Discover Computing analyzed 77 peer-reviewed studies and found that finite-field-based constructions, particularly ECC and AES-GCM, consistently offer the best trade-offs between computational overhead and security guarantees in resource-constrained wireless sensor environments [12].

A novel cipher-based encryption scheme exploiting Galois field properties was proposed and evaluated by Hazzazi et al. [13] in the journal Sensors. Their approach encoded plaintext data as Galois vectors and applied field-theoretic mathematical operations, including multiplicative inverses, to achieve confusion and diffusion, demonstrating competitive performance against standard AES on image encryption benchmarks. Complementarily, work exploring Galois field arithmetic in chaotic and transform-domain image encryption reinforces the versatility of GF-based operations as encryption primitives [14]. These studies collectively affirm that Galois field operations can serve not only as components within the AES but as standalone encryption operators when appropriately structured.

In [15], the security of a video-encryption scheme is examined from a cryptanalysis-oriented perspective, including known-plaintext, chosen-plaintext, and chosen-ciphertext attack scenarios, highlighting the importance of key-dependent and dynamically updated encryption mechanisms. In [16], authors consider nonlinear and dynamically evolving transformations in the context of image encryption, illustrating the relevance of confusion, diffusion, and dynamic state behavior for high-redundancy multimedia data. The cryptanalysis-driven perspectives in [15,16] provide useful complementary evaluation criteria for the proposed lightweight architecture, although these works are not treated as direct performance baselines because their cryptographic structures, application domains, and implementation assumptions differ from those of the present method.

2.2. Encryption in Wireless and Blockchain Networks

Secure data transmission in wireless networks has been extensively studied, with AES in Galois/Counter Mode (GCM) emerging as the de facto standard for authenticated encryption in IEEE 802.11 (Wi-Fi) [2], IEEE 802.15.4 [17], LTE, and 5G protocols. Bille and Tischhauser (2024) analyzed three authenticated-encryption algorithm families, proposed as alternatives to GCM specifically for wireless and real-time systems, focusing on reducing finite field multiplication overhead while preserving security [18]. Their cryptanalytic findings highlight the delicacy of constructing GF-based encryption modes that avoid vulnerability to forgery attacks, underscoring the importance of rigorous algebraic design.

Blockchain networks impose unique cryptographic requirements: encryption must be computationally tractable for nodes with heterogeneous resources, while providing strong assurances against both classical and quantum adversaries. A 2024 IEEE Access paper on dynamic AES encryption with blockchain key management proposed a three-layer security framework combining AES, ECC public-key encryption, and blockchain-based key lifecycle management to protect cloud-stored data at rest and in transit [9]. Separately, a hybrid ECC-AES framework (SymECCipher) reported in Scientific Reports [19] (2025) achieved

a 25–40% reduction in computational overhead compared to traditional cryptosystems, while remaining blockchain-compatible and quantum-resistant.

In the context of 6G-enabled smart city infrastructure, a 2025 study published in the Journal of Supercomputing proposed a dynamic Galois field expansion algorithm (DGFEA) combined with Reed–Solomon error correction and bio-inspired optimization for securing IoT data transmission across quantum-threat-aware networks [11]. This work demonstrates the practicality of extending standard Galois field constructions to dynamic, adaptive cryptographic systems. The growing interest in blockchain-integrated encryption for sectors including healthcare, finance, and critical infrastructure further motivates the need for encryption schemes that are both algebraically principled and computationally scalable across diverse Galois field orders.

2.3. Research Gap and Motivation

Despite the extensive literature on Galois field-based cryptography, the use of minimal polynomials as direct encryption operators, exploiting the factorization property $x^n + 1 = \prod_i m_i(x)$ to enable exact decryption via polynomial multiplication, has not been previously studied as an independent encryption paradigm. Existing schemes that employ polynomial operations over Galois fields do so primarily within the fixed structure of AES or ECC, without exploring the flexibility afforded by choosing distinct minimal polynomials of different orders over multiple fields to encrypt distinct data frames. This paper addresses this gap by proposing a symmetric encryption method based on minimal polynomial arithmetic over extended Galois fields, by establishing a theorem on the inclusion of minimal polynomial sets under field extensions, and by analyzing the correctness, security limitations, and software and FPGA implementation cost of the method.

The remainder of this paper is organized as follows. Section 2 reviews the related literature. Section 3 introduces the proposed minimal polynomial-based encryption scheme, its theoretical foundation, and its security analysis. Section 4 describes the software implementation and compares the proposed method with AES-128 and SPECK64/128. In Section 5, the proposed core and a standard AES-128 core are implemented and compared on an FPGA. Finally, Section 6 concludes the paper and discusses potential directions for future work.

3. Minimal Polynomial-Based Encryption

In this section, we explain our minimal polynomial-based novel encryption–decryption method. Applications requiring secure data transmission and storage widely use the Advanced Encryption Standard (AES). For instance, to prevent unauthorized access and to guarantee the security of data in wireless networks, such as Wi-Fi networks, the AES algorithm is used.

The AES can also be used to encrypt sensitive data stored in databases containing personal information, financial records, bank account numbers, and telephone numbers, among others. It can likewise be used to encrypt sensitive data stored on hard drives, USB drives, and other storage media; passwords can be encrypted by the AES before they are stored, and in VPN protocols, the AES secures communication between a client and a server. The AES is also used in cloud storage to encrypt files and folders in order to protect sensitive data stored on devices.

In blockchain and wireless communication networks, extended Galois fields, i.e., $\mathrm{GF}(2^m)$, are used. Consider the extended Galois field

$$\mathrm{GF}(2^m) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^r\}, \qquad r = 2^m - 2. \tag{1}$$

We can write

$$x^n + 1 = \prod_{i \geq 1} m_i(x), \qquad n = r + 1, \tag{2}$$

where $m_i(x)$ are the minimal polynomials. For instance, in GF(2^3), we have

$$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1). \tag{3}$$

3.1. Data Representation

To encrypt a binary string, we first divide the binary string into frames and write the frames into the rows of a matrix. Let D be the data matrix and $d_1, d_2, \ldots, d_k$ be the rows of the matrix; that is,

$$D = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix}. \tag{4}$$

Assume that GF(2^m) is employed for the design. Binary vectors can be represented by polynomials having binary coefficients. The rows have polynomial representations $d_1(x), d_2(x), \ldots, d_k(x)$; that is,

$$D(x) = \begin{bmatrix} d_1(x) \\ d_2(x) \\ \vdots \\ d_k(x) \end{bmatrix}. \tag{5}$$

3.2. Fixed-Length Encryption Using Minimal Polynomials

A row polynomial can be encrypted by multiplication with a selected minimal polynomial. Let $d_1(x)$ denote the first row polynomial. If $m_3(x)$ is selected, then encryption is defined as

$$e_1(x) = d_1(x) m_3(x), \tag{6}$$

which corresponds to binary convolution of the coefficient vectors, i.e.,

$$e_1 = d_1 * m_3. \tag{7}$$

For improved confidentiality, an additional key polynomial may be introduced:

$$e_1(x) = \left[d_1(x) + k_1(x) \right] m_3(x), \tag{8}$$

where $k_1(x)$ is a secret key polynomial known to the receiver.

3.3. Decryption Using Minimal Polynomials

For the Galois field

$$\mathrm{GF}(2^m) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^r\}, \qquad r = 2^m - 2, \tag{9}$$

we have Equation (2), from which we can obtain

$$m_j(x) = \frac{x^n + 1}{\prod_{i \neq j,\ i \geq 1} m_i(x)}, \tag{10}$$

which can be written as

$$x^n + 1 = m_j(x) \prod_{i \neq j,\ i \geq 1} m_i(x). \tag{11}$$

For instance, for GF(2^6),

$$x^{63}+1 = m_3(x) \prod_{i\neq 3,\ i\geq 1} m_i(x). \tag{12}$$

For the encryption polynomial

$$e_1(x) = d_1(x)m_3(x) \tag{13}$$

obtained in the previous section, the decryption operation can be achieved as

$$r_1(x) = e_1(x) \prod_{i\neq 3,\ i\geq 1} m_i(x), \tag{14}$$

$$r_1(x) = d_1(x)\, m_3(x) \prod_{i\neq 3,\ i\geq 1} m_i(x), \tag{15}$$

$$r_1(x) = d_1(x)(x^n+1), \tag{16}$$

$$r_1(x) = d_1(x)x^n + d_1(x), \tag{17}$$

where $d_1(x)x^n$ is the n-bit left-shifted polynomial version of d_1, and by properly designing the parameters it is possible to obtain non-overlapping binary vector versions of $d_1(x)$ and $d_1(x)x^n$.

For better data hiding, the encryption operation can also be written as

$$[d_1(x) + \text{key}_1(x)]\, m_3(x), \tag{18}$$

where $\text{key}_1(x)$ is a known key sequence that can be removed from the decrypted sequence at the receiver side. $\text{key}_1(x)$ can also be selected from other minimal polynomials.

3.4. Illustrative Example

Assume that we have a 224-bit information vector placed into a matrix as

$$D = \begin{bmatrix} 00 & 11 & 22 & 33 & 44 & 55 & 66 \\ 77 & 88 & 99 & AA & BB & CC & DD \\ EE & FF & 12 & 23 & 34 & 45 & 56 \\ 67 & 78 & 89 & 9A & A0 & B2 & C2 \end{bmatrix}, \tag{19}$$

where each row contains 56 bits and the matrix contains $4 \times 56 = 224$ bits.

Consider GF(2^6); the minimal polynomials for GF(2^6) can be tabulated as

$$\begin{aligned} &x, \quad x+1, \\ &x^2+x+1, \\ &x^3+x+1, \quad x^3+x^2+1, \\ &x^6+x+1, \quad x^6+x^3+1, \quad x^6+x^4+x^2+x+1, \\ &x^6+x^4+x^3+x+1, \quad x^6+x^5+1, \quad x^6+x^5+x^2+x+1, \\ &x^6+x^5+x^3+x^2+1, \quad x^6+x^5+x^4+x+1, \quad x^6+x^5+x^4+x^2+1. \end{aligned} \tag{20}$$

The first polynomial, x, is the minimal polynomial of the zero element and does not divide $x^{63}+1$; it is therefore not used for encryption. The remaining 13 polynomials are the irreducible factors of $x^{63}+1$ and form the set of admissible encryption polynomials.

Minimal polynomial encryption can be applied to the rows of the data matrix; in principle, the same operation could also be applied to the columns, but this extension is not evaluated in this work. For simplicity, only row encryption is considered here.

In Equation (19), each row can be expressed as a polynomial, giving row polynomials

$$d_1(x),\ d_2(x),\ d_3(x),\ \text{and}\ d_4(x). \tag{21}$$

Let the minimal polynomials from GF(2^6) chosen to encrypt the rows be

$$\begin{aligned} m_1(x) &= x^6 + x + 1, \\ m_2(x) &= x^6 + x^3 + 1, \\ m_3(x) &= x^6 + x^4 + x^3 + x + 1, \\ m_4(x) &= x^6 + x^4 + x^3 + x + 1. \end{aligned} \tag{22}$$

The rows are encrypted as

$$\begin{aligned} e_1(x) &= d_1(x)m_1(x), \\ e_2(x) &= d_2(x)m_2(x), \\ e_3(x) &= d_3(x)m_3(x), \\ e_4(x) &= d_4(x)m_4(x). \end{aligned} \tag{23}$$

The rows can also be encrypted as

$$\begin{aligned} e_1(x) &= [d_1(x) + k_1(x)]m_1(x), \\ e_2(x) &= [d_2(x) + k_2(x)]m_2(x), \\ e_3(x) &= [d_3(x) + k_3(x)]m_3(x), \\ e_4(x) &= [d_4(x) + k_4(x)]m_4(x), \end{aligned} \tag{24}$$

where $k_i(x)$, $i = 1, \ldots, 4$, are secret key polynomials that may also be chosen from other minimal polynomials.

At the receiver side, the decryption operation can be performed using the equality

$$x^n + 1 = \prod_{i \geq 1} m_i(x), \qquad n = 63. \tag{25}$$

For instance,

$$e_1(x) = d_1(x)\, m_1(x) \tag{26}$$

can be decrypted as

$$r_1(x) = e_1(x) \prod_{i \neq 1,\ i \geq 1} m_i(x), \tag{27}$$

leading to

$$r_1(x) = d_1(x)(x^{63} + 1), \tag{28}$$

$$r_1(x) = d_1(x)x^{63} + d_1(x), \tag{29}$$

where the term $d_1(x)x^{63}$ corresponds to the n-bit left-shifted sequence and can be separated from $d_1(x)$, since $n = 63$, each row contains 56 bits, and the data and left-shifted bit sequences do not overlap.

We now state an important theorem that can be used for the construction of encryption algorithms employing minimal polynomials.

3.5. Theorem on Field Extensions

Theorem 1. *Let S_i denote the set of minimal polynomials over* GF(2) *associated with the finite field* GF(2^{m_i}), *for $i = 1, \ldots, r$. Let*

$$n = \operatorname{lcm}(m_1, m_2, \ldots, m_r). \tag{30}$$

If S denotes the set of minimal polynomials over GF(2) *associated with* GF(2^n), *then*

$$\bigcup_{i=1}^{r} S_i \subseteq S. \tag{31}$$

Proof. The minimal polynomials of elements of GF(2^m) over GF(2) are precisely the monic irreducible polynomials $f(x) \in \mathrm{GF}(2)[x]$ whose degrees divide m. Hence, for each $i = 1, \ldots, r$, we may write

$$S_i = \{f(x) \in \mathrm{GF}(2)[x] : f(x) \text{ is monic and irreducible, and } \deg(f) \mid m_i\}. \tag{32}$$

Similarly, the set of minimal polynomials associated with GF(2^n) is

$$S = \{f(x) \in \mathrm{GF}(2)[x] : f(x) \text{ is monic and irreducible, and } \deg(f) \mid n\}. \tag{33}$$

Let $f(x) \in S_i$ for some $i \in \{1, \ldots, r\}$. Then $\deg(f) \mid m_i$. Since $n = \operatorname{lcm}(m_1, m_2, \ldots, m_r)$, we have $m_i \mid n$. Therefore, $\deg(f) \mid n$. Thus $f(x)$ is also a monic irreducible polynomial whose degree divides n, and hence $f(x) \in S$.

Since every polynomial in each S_i also belongs to S, we conclude that $S_1 \cup S_2 \cup \cdots \cup S_r \subseteq S$. Therefore,

$$\bigcup_{i=1}^{r} S_i \subseteq S \tag{34}$$

□

Lemma 1. *Let S_i and S be defined as in the theorem above. For each $i = 1, \ldots, r$, the polynomial $x^{2^{m_i}-1} + 1$ can be factored into a product of minimal polynomials belonging to S; that is,*

$$x^{2^{m_i}-1} + 1 = \prod_{j_i \geq 1} m_{j_i}(x), \tag{35}$$

where each $m_{j_i}(x)$ is a minimal polynomial contained in S.

Proof. Over the field GF(2^{m_i}), every nonzero element satisfies $\alpha^{2^{m_i}-1} = 1$. Therefore, the roots of $x^{2^{m_i}-1} + 1$ are precisely the nonzero elements of GF(2^{m_i}).

Since the minimal polynomial of each element of GF(2^{m_i}) over GF(2) belongs to S_i, the polynomial $x^{2^{m_i}-1} + 1$ can be decomposed as a product of minimal polynomials from S_i. Thus,

$$x^{2^{m_i}-1} + 1 = \prod_{j_i \geq 1} m_{j_i}(x), \qquad m_{j_i}(x) \in S_i. \tag{36}$$

By the theorem, $S_i \subseteq S$. Hence, each minimal polynomial appearing in the factorization above is also an element of S. Therefore, for each $i = 1, \ldots, r$, there exists a subset of minimal polynomials of S such that $x^{2^{m_i}-1} + 1 = \prod_{j_i} m_{j_i}(x)$, which proves the desired factorization. □

3.6. Security of the Proposed Encryption Algorithm

The total number of minimal polynomials for GF(2^n) is calculated as

$$N = \sum_{d|n} \frac{1}{d} \sum_{k|d} \mu(k)\, 2^{d/k}, \tag{37}$$

where $\mu(k)$ is the Möbius function, defined as

$$\mu(k) = \begin{cases} 1, & \text{if } k = 1, \\ 0, & \text{if } k \text{ is divisible by a square (e.g., } p^2), \\ (-1)^l, & \text{if } k \text{ is a product of } l \text{ distinct primes.} \end{cases} \tag{38}$$

For prime n, the total number of minimal polynomials of GF(2^n) can be calculated as

$$N = 2 + \frac{1}{n}(2^n - 2), \tag{39}$$

which grows approximately as $2^n/n$.

As an example, since $n = 100$ is not prime, the total number of minimal polynomials of GF(2^{100}) is obtained from Equation (37) as

$$12{,}676{,}506{,}002{,}282{,}305{,}273{,}966{,}813{,}560, \tag{40}$$

which can be considered approximately 10^{28}. The number of admissible encryption polynomials is one less than this value, since $m_0(x) = x$ is excluded. The largest degree of the minimal polynomials is 100.

If four minimal polynomials are chosen from the total set, the number of possible different choices is nearly

$$(10^{28})^4 = 10^{112}. \tag{41}$$

This number determines the cost of an exhaustive search over the polynomial selections. However, as shown in Section 3.8, it is not by itself a measure of security, since structural attacks do not require searching this space.

If fields GF(2^p) with smaller prime p are employed for the encryption operation, for instance GF(2^{13}), GF(2^{17}), GF(2^{19}), and so on, a subset of the total minimal polynomials can be chosen and used for the encryption process. As the number of such fields increases, the number of minimal polynomials also increases, which enlarges the polynomial-selection space.

Per-Message Randomization

To prevent deterministic reuse of the same masked polynomial input, a fresh nonce N_j is introduced for each message or frame j. A nonce-dependent pseudorandom mask is generated as

$$r_{i,j}(x) = \mathrm{PRF}_{K_R}(N_j \,\|\, i), \tag{42}$$

where K_R is a secret key associated with the mask-generation function, i identifies the row, and $\|$ denotes concatenation. The revised encryption operation is therefore

$$e_{i,j}(x) = \left[d_{i,j}(x) + k_i(x) + r_{i,j}(x)\right] m_i(x), \tag{43}$$

which replaces the original deterministic transformation.

The nonce does not need to be secret, but it must not be reused with the same masking key. The receiver uses the received nonce and the shared secret K_R to regenerate $r_{i,j}(x)$

during decryption. In this work, PRF is instantiated with AES-128 in counter (CTR) mode, and its output is truncated to L bits for each row.

3.7. Encryption for Stream Transmission

The minimal polynomial-based encryption method proposed in this paper can also be applied to stream broadcasting communication. In stream transmission, a continuous stream of bits is transmitted without interruption, and the convolution operation can be performed between a finite-length sequence and an infinite-length sequence in a continuous manner. For stream transmission, a multiplexing operation between minimal polynomials can be performed, as shown in Figure 1; this mode is not evaluated experimentally in this work.

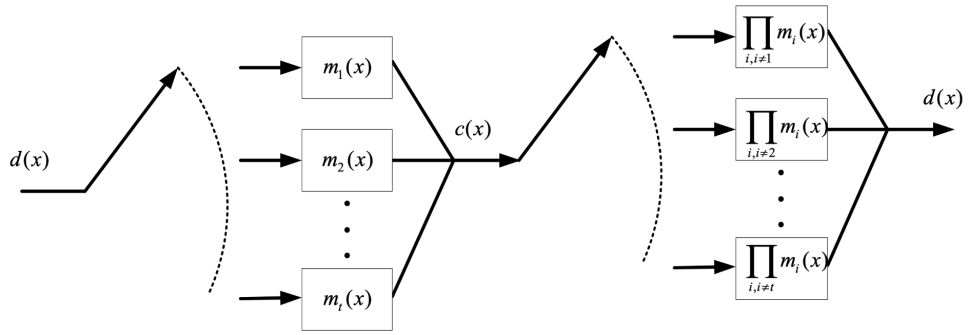


Figure 1. Block diagram of the proposed minimal-polynomial multiplexing scheme for stream encryption. The dotted arcs denote the switches that select, on the transmitter side, the minimal polynomial $m_i(x)$ and, on the receiver side, the corresponding product $\prod_{i \neq j} m_i(x)$; the arrows indicate the direction of data flow. $d(x)$: data sequence; $c(x)$: encrypted sequence.

3.8. Security Analysis Against Practical Attack Models

The exact reconstruction demonstrated in Sections 3.3 and 3.4 establishes the correctness and reversibility of the transformation; however, reversibility alone does not establish confidentiality. Decryption correctness and confidentiality are therefore treated as separate properties. The analysis assumes that the adversary knows the structure of the algorithm and can observe ciphertexts; depending on the scenario, the adversary may also possess plaintext–ciphertext pairs or choose plaintexts and observe the corresponding ciphertexts. The secret information consists of the key polynomials $k_i(x)$, the selected polynomials $m_i(x)$, and the mask-generation key K_R. Because XOR and binary polynomial multiplication are both linear over GF(2), the number of possible polynomial selections describes only the cost of exhaustive enumeration; structural attacks may exploit algebraic relations without searching this space.

3.8.1. Known-Plaintext Analysis

In a known-plaintext attack (KPA), the adversary has access to plaintext–ciphertext pairs. For the deterministic key-assisted transformation $e_i(x) = [d_i(x) + k_i(x)]\, m_i(x)$, once $m_i(x)$ is identified, polynomial division gives

$$k_i(x) = \frac{e_i(x)}{m_i(x)} + d_i(x). \tag{44}$$

Since $m_i(x)$ divides $e_i(x)$ and the admissible set for GF(2^6) contains only 13 polynomials, $m_i(x)$ can be identified from a single pair by trial division. Reuse of the same key and polynomial across messages therefore allows for key recovery without exhaustive search. With the nonce-dependent construction in Equation (43), a known pair reveals only the

combined quantity $k_i(x) + r_{i,j}(x)$; since $r_{i,j}(x)$ changes with the nonce, this quantity cannot be reused to decrypt another message.

3.8.2. Chosen-Plaintext Analysis

In a chosen-plaintext attack (CPA), the adversary selects plaintexts and observes the ciphertexts. For two plaintexts encrypted with the deterministic construction and the same $k_i(x)$ and $m_i(x)$, addition over GF(2) cancels the key:

$$e_{i,1}(x) + e_{i,2}(x) = [d_{i,1}(x) + d_{i,2}(x)]m_i(x). \tag{45}$$

If the plaintexts are chosen such that $d_{i,1}(x) + d_{i,2}(x) = 1$, the ciphertext difference equals $m_i(x)$, and the encryption polynomial is exposed directly; the key then follows from Equation (44). With nonce-dependent masking, the difference becomes $[1 + r_{i,1}(x) + r_{i,2}(x)]\, m_i(x)$ instead of $m_i(x)$, so this direct cancellation attack is prevented when fresh masks are used. This addresses the deterministic parameter-reuse vulnerability, but it is not a formal proof of security against chosen-plaintext attacks, which would require a security reduction to the pseudorandom function.

3.8.3. Algebraic and Equivalent-Key Recovery

When the same polynomial $m_i(x)$ is used for several messages, every ciphertext row has the form $e_{i,j}(x) = u_{i,j}(x)\, m_i(x)$, where $u_{i,j}(x)$ is the key-mixed or masked data polynomial. The polynomial $m_i(x)$ is therefore a common factor of all such ciphertexts and can be recovered, for example, by a greatest-common-divisor computation, independently of the masks. Nonce masking protects the data polynomial $u_{i,j}(x)$ but does not hide $m_i(x)$. Consequently, the secrecy of the polynomial selection should not be regarded as a security parameter, and the confidentiality of the nonce-masked construction relies on the secrecy of K_R and on the pseudorandom function.

3.8.4. Statistical Distinguishability and Ciphertext Predictability

With fixed parameters, identical plaintexts produce identical ciphertexts, and related plaintexts produce algebraically related ciphertexts, which may reveal repetitions and structural patterns in correlated data such as sensor measurements, images, or video. The nonce-dependent mask changes the input of the polynomial multiplication for each message, so that repeated plaintext blocks encrypted under different nonces produce different ciphertexts. The diffusion of the transformation was quantified with the avalanche ratio (Section 4.4), which confirmed that a single-bit change affects only $\mathrm{wt}(m_i)$ bits of one ciphertext row. A comprehensive empirical evaluation of statistical indistinguishability, for example with standard randomness test suites, was not performed in this work and remains a subject for future investigation.

3.8.5. Scope and Limitations of the Security Claims

The security claims of the proposed architecture are limited to the properties evaluated in this work. A large polynomial-selection space increases the cost of exhaustive enumeration, but it does not establish semantic security or indistinguishability under chosen-plaintext attack. Exact reconstruction demonstrates correctness rather than confidentiality, and avalanche measurements characterize diffusion rather than providing a security proof. The proposed architecture is therefore investigated as a lightweight polynomial processing layer for resource-constrained hardware, whose secure use requires avoiding deterministic parameter reuse and applying message-dependent masking. Formal security reductions, authenticated-encryption extensions, side-channel and fault-attack analysis, and comprehensive cryptanalysis remain subjects for further investigation.

3.9. Description of the Proposed Encryption and Decryption Algorithms

Algorithm 1 presents the proposed polynomial-based encryption method over GF(2), where each plaintext row is first combined with a secret key sequence and a nonce-dependent mask using an XOR operation and then multiplied by a row-specific irreducible polynomial. The pre-computation stage determines the complementary factors required for exact decryption.

For a selected polynomial $m_i(x)$, the complementary factor is $q_i(x) = (x^n + 1)/m_i(x)$, i.e., the product of the remaining minimal polynomials, as in Equation (14), so that $m_i(x)\, q_i(x) = x^n + 1$. It should be distinguished from a multiplicative inverse—$m_i(x)\, q_i(x) \neq 1$—and the reconstruction does not require polynomial inversion, but only the known factorization of $x^n + 1$; equivalently, the key-mixed row can be obtained by dividing the ciphertext polynomial by $m_i(x)$. Exact reconstruction holds for every admissible polynomial $m_i(x)$, that is, for every irreducible factor of $x^n + 1$, provided that the row length satisfies $L \leq n$, so that the shifted and unshifted copies of the row do not overlap (here, $L = 56 \leq n = 63$).

Algorithm 2 describes the corresponding decryption procedure. Each received ciphertext polynomial is multiplied by its associated complementary factor to recover the key-mixed data sequence. The original plaintext is then reconstructed by applying XOR with the same secret key matrix and the regenerated mask. Owing to the algebraic factorization property over GF(2), the proposed scheme enables exact and computationally efficient data recovery.

Algorithm 1 Proposed polynomial-based encryption over GF(2)

Input: $D[1..N][1..L]$ plaintext matrix (N rows, L bits each)
Input: $K[1..N][1..L]$ secret key matrix
Input: K_R secret mask-generation key
Input: $M = \{m_1, m_2, \ldots, m_N\}$ row-specific irreducible polynomials over GF(2)
Output: $E[1..N]$ ciphertext polynomials, nonce N_j

Pre-computation: complementary factors
for $i = 1$ to N **do**
  $q_i \Leftarrow \text{PolyDiv}_{GF2}(x^{63} + 1, m_i)$
  $m_i(x) \cdot q_i(x) = x^{63} + 1$
end for

Nonce generation: generate a fresh nonce N_j for message j

Encryption
for $i = 1$ to N **do**
  Step 1: Mask generation
  $R_{i,j}(x) \Leftarrow \text{PRF}_{K_R}(N_j \,\|\, i)$
  Step 2: Key and mask mixing (XOR over GF(2))
  $dk_{i,j}(x) \Leftarrow D_j[i](x) \oplus K[i](x) \oplus R_{i,j}(x)$
  Step 3: Polynomial multiplication
  $E[i](x) \Leftarrow dk_{i,j}(x) \cdot m_i(x)$
  $e_{i,j}(x) = [d_{i,j}(x) + k_i(x) + r_{i,j}(x)]\, m_i(x)$
  Step 4: Hardware metrics
  $XOR_count \Leftarrow XOR_count + L + \deg(E[i])$
  $AND_count \Leftarrow AND_count + L \times \deg(m_i)$
end for
return E, N_j

Algorithm 2 Proposed polynomial-based decryption over GF(2)

Input: $E[1..N]$ ciphertext polynomials, received nonce N_j
Input: $K[1..N][1..L]$ secret key matrix
Input: K_R secret mask-generation key
Input: $Q[1..N]$ complementary factors
Output: $D_{rec}[1..N][1..L]$ recovered plaintext

Decryption
for $i = 1$ to N **do**
Step 1: Mask regeneration
$R_{i,j}(x) \Leftarrow \mathrm{PRF}_{K_R}(N_j \,\|\, i)$
Step 2: Multiply by complementary factor
$R_i(x) \Leftarrow E[i](x) \cdot q_i(x)$
$R_i(x) = dk_{i,j}(x)(x^{63} + 1)$
Step 3: Extract first L coefficients
$dk_{i,j}^{rec} \Leftarrow R_i[1{:}L]$
Step 4: Remove key and mask
$D_{rec}[i] \Leftarrow dk_{i,j}^{rec} \oplus K[i] \oplus R_{i,j}$
Step 5: Verification
if $D_{rec}[i] \neq D[i]$ **then**
Decryption failed for row i.
end if
end for
return D_{rec}

4. Software Implementation and Performance Evaluation

4.1. Experimental Setup

All software experiments were carried out in MATLAB R2023a (The MathWorks, Inc., Natick, MA, USA) on a MacBook Pro (13-inch, 2019; Apple Inc., Cupertino, CA, USA) with a 1.4 GHz quad-core Intel Core i5 processor and 8 GB of 2133 MHz LPDDR3 memory, running macOS Sequoia 15.7.9. Three variants of the proposed method were evaluated: the keyless form (Equation (6)), the key-assisted form (Equation (24)), and the key-assisted form with nonce-dependent masking (Equation (43)). In all variants, plaintext was divided into rows of $L = 56$ bits, and each row was multiplied by a polynomial selected uniformly at random from the nine degree-6 irreducible factors of $x^{63} + 1$. The complementary factors $q_i(x)$ were computed once in a pre-computation stage and were not included in the timing. In the nonce-masked variant, the pseudorandom function in Equation (42) was instantiated with AES-128 in counter (CTR) mode using a 128-bit nonce, and the cost of mask generation was included in both encryption and decryption times.

Two reference ciphers were used. The first was the standardized AES-128 in CTR mode with a fresh 128-bit nonce for each message, provided by the Java Cryptography Extension (Java SE 1.8.0_202; Oracle Corporation, Austin, TX, USA) and called from MATLAB without modification. The second was the lightweight block cipher SPECK64/128 in CTR mode [20], implemented in MATLAB. Both reference implementations were verified against published test vectors (the example vector in Appendix C.1 of FIPS-197 [4] and the SPECK64/128 test vector in [20]) before the experiments.

Message lengths of 224, 1024, 8192, and 65,536 bits were tested. For each method and message length, 100 independent trials were performed after one warm-up run, and a new random plaintext and new random keys were generated in each trial. All random plaintexts, keys, and polynomial selections were generated with a fixed random seed (MATLAB `rng(2026)`), so that the experiments can be reproduced exactly with the provided script (File S1). Execution times were measured with the MATLAB `tic/toc` functions and are

reported as mean ± standard deviation. Exact plaintext recovery was verified in every trial for all methods.

4.2. Execution Time

Table 1 shows that the standardized AES-128 implementation was the fastest method for messages of 8192 bits and longer. Its execution time remained almost constant (approximately 0.13–0.23 ms) across all message lengths, which indicates that, at short message lengths, it is dominated by the fixed cost of initializing the cipher object rather than by the encryption itself. For this reason, the shorter execution times of the proposed method at 224 bits should not be interpreted as an algorithmic advantage over the AES. In addition, the AES implementation is a compiled library, whereas the proposed method and SPECK64/128 were executed as MATLAB code; the comparison with the AES therefore also reflects a difference in implementation level.

Table 1. Encryption and decryption times (ms, mean ± SD, 100 trials) for different message lengths.

Method	224 Bits	1024 Bits	8192 Bits	65,536 Bits
Encryption				
Proposed (keyless)	0.040 ± 0.175	0.051 ± 0.016	0.345 ± 0.054	2.557 ± 0.166
Proposed + key	0.043 ± 0.209	0.090 ± 0.032	0.538 ± 0.041	4.321 ± 0.362
Proposed + key + nonce mask	0.335 ± 0.268	0.354 ± 0.128	1.004 ± 0.100	6.714 ± 0.429
AES-128-CTR (Java JCE)	0.233 ± 0.087	0.181 ± 0.064	0.157 ± 0.032	0.207 ± 0.022
SPECK64/128-CTR (MATLAB)	0.060 ± 0.090	0.093 ± 0.025	0.564 ± 0.019	4.299 ± 0.340
Decryption				
Proposed (keyless)	0.046 ± 0.165	0.097 ± 0.021	0.694 ± 0.117	5.065 ± 0.367
Proposed + key	0.040 ± 0.015	0.171 ± 0.046	1.066 ± 0.058	8.386 ± 0.355
Proposed + key + nonce mask	0.363 ± 0.351	0.433 ± 0.169	1.543 ± 0.121	10.806 ± 0.362
AES-128-CTR (Java JCE)	0.229 ± 0.082	0.173 ± 0.070	0.131 ± 0.019	0.164 ± 0.025
SPECK64/128-CTR (MATLAB)	0.060 ± 0.096	0.091 ± 0.024	0.563 ± 0.023	4.285 ± 0.133

Under the same implementation level, the encryption time of the key-assisted proposed method was close to that of SPECK64/128 for all message lengths (e.g., 4.32 vs. 4.30 ms at 65,536 bits). Decryption of the proposed method was approximately twice as slow as its encryption, because reconstruction requires multiplication by the degree-57 complementary factor instead of a degree-6 polynomial. The nonce-masked variant was the slowest of the proposed variants, since its cost includes the generation of the pseudorandom mask. The large standard deviations observed at 224 bits are attributed to timing variability at sub-millisecond durations.

4.3. Ciphertext Expansion

Each 56-bit row of the proposed method produces a 62-bit ciphertext row, which leads to an expansion ratio of approximately 1.11 independently of the message length; small deviations are caused by zero-padding of the last row. In the nonce-masked variant and in AES-128-CTR, the transmitted 128-bit nonce adds a fixed overhead that is significant for short messages and becomes negligible for long messages (Table 2).

Table 2. Ciphertext expansion ratio (transmitted bits/plaintext bits), including the transmitted nonce where applicable.

Method	224 Bits	1024 Bits	8192 Bits	65,536 Bits
Proposed (keyless/+key)	1.107	1.150	1.113	1.108
Proposed + key + nonce mask	1.679	1.275	1.128	1.110
AES-128-CTR	1.571	1.125	1.016	1.002
SPECK64/128-CTR	1.143	1.031	1.004	1.001

4.4. Avalanche Behavior

The avalanche ratio is defined here as the fraction of ciphertext bits that change when a single, randomly selected plaintext bit is flipped while the key, the encryption polynomial, and the nonce are kept unchanged. For a cipher with good diffusion, this ratio is expected to be close to 0.5. Because a single-bit change in CTR mode changes only the corresponding ciphertext bit, the avalanche ratio of the reference ciphers was measured on a single block of the underlying block cipher. Each value in Table 3 is the mean ± SD of 1000 trials with random plaintexts and keys.

Table 3. Avalanche ratio for a single-bit plaintext change (mean ± SD, 1000 trials).

Method	Output Size (Bits)	Avalanche Ratio
AES-128 (single block)	128	0.501 ± 0.044
SPECK64/128 (single block)	64	0.498 ± 0.064
Proposed (single row)	62	0.070 ± 0.015
Proposed (224-bit message)	248	0.018 ± 0.004

Both reference ciphers reached the expected value of approximately 0.5, whereas the proposed method produced 0.070 for a single row and 0.018 for a 224-bit message. These low values are a direct consequence of the linearity of the proposed transformation over $\mathrm{GF}(2)$: flipping one plaintext bit adds a shifted copy of $m_i(x)$ to the ciphertext row, so exactly $\mathrm{wt}(m_i)$ bits of that row change (three to five bits for the polynomials used here), while the other rows are not affected. The proposed polynomial multiplication therefore provides only local diffusion and does not achieve the diffusion level of nonlinear block ciphers. This limitation is consistent with the security analysis in Section 3 and is reflected in the revised claims of this paper.

4.5. Storage Requirements

Table 4 lists the stored constants and key material required by each method. The values refer to a straightforward implementation in which all constants are stored explicitly, and key material is counted separately and in the same way for all methods. For the proposed method, the values correspond to the four-row (224-bit) frame of Section 3.4.

Table 4. Storage requirements of constants and key material (bits).

Method	Constants	Key Material	Components
Proposed + key	260	224	m_i: 4×7; q_i: 4×58; K: 4×56
Proposed + key + nonce mask	260 + PRF	224 + 128	as above; K_R: 128; PRF: AES-128 in this work
AES-128-CTR	2048	128 + 1408	S-box: 256×8; round keys: 11×128
SPECK64/128-CTR	0	128 + 864	round keys: 27×32

For the proposed method, the 260 bits consist of the four encryption polynomials (7 coefficients each) and the four corresponding complementary factors (58 coefficients

each). If decryption is performed by polynomial division by $m_i(x)$ instead of multiplication by $q_i(x)$, the complementary factors need not be stored, and the constant storage reduces to 28 bits at the cost of a sequential division. Compared with a table-based AES-128 implementation, the proposed method requires less constant storage because it does not use a substitution box. However, the ARX-based SPECK64/128 cipher requires no constant tables at all, and the nonce-masked variant inherits the storage of the selected PRF. The storage advantage of the proposed method is therefore relative to table-based designs and should not be generalized to all lightweight ciphers.

5. FPGA Implementation and Evaluation

5.1. Hardware Architecture

The proposed method was implemented as a synthesizable VHDL core that processes one 56-bit row per clock cycle. In encryption mode, the core computes $e(x) = [d(x) + k(x) + r(x)]\, m_s(x)$, where $m_s(x)$ is selected by a 4-bit index from the nine degree-6 irreducible factors of $x^{63} + 1$, and $r(x)$ is the nonce-dependent mask. The 62-bit product is obtained with a parallel array of AND gates and XOR trees. In decryption mode, the core computes only the lowest 56 coefficients of $e(x)\, q_s(x)$, where $q_s(x) = (x^{63} + 1)/m_s(x)$ is the stored degree-57 complementary factor, and then removes $k(x) + r(x)$. The polynomials $m_s(x)$ and $q_s(x)$ are hard-wired constants, and no substitution table, block RAM, or DSP resource is used. Inputs and outputs are registered, so that the latency is two clock cycles and a new row can be accepted in every cycle. The pseudorandom function that generates $r(x)$ is not included in the core; the mask is supplied as an input.

As a reference, the standardized AES-128 was implemented using the open-source `secworks/aes` core [21], which contains the encipher, decipher, and key-expansion blocks. The core uses an iterative architecture with four shared S-boxes that process one 32-bit word per clock cycle, so that each round requires several cycles. The core was placed in a wrapper with the same input and output registers as the proposed core.

5.2. Implementation Setup and Functional Verification

Both designs were synthesized and implemented separately with Vivado 2025.2 (Advanced Micro Devices, Inc., Santa Clara, CA, USA) for the AMD Xilinx Artix-7 xc7a35tcpg236-1 device (Basys3 board; Digilent Inc., Pullman, WA, USA), using out-of-context synthesis, default synthesis and implementation strategies (`opt_design`, `place_design`, `phys_opt_design`, `route_design`), and the same 10 ns (100 MHz) clock constraint. All results were taken after place-and-route. The maximum operating frequency was computed as $f_{\max} = 1/(T - \mathrm{WNS})$, where T is the clock period and WNS is the worst negative slack. Power was estimated with the Vivado power analyzer using vectorless activity estimation (default toggle rate of 12.5% and static probability of 0.5) at 100 MHz; I/O power is not included because the designs were implemented out of context. These values are tool estimates and not board-level measurements. The HDL sources, testbenches, and Vivado script are provided in File S1.

Functional correctness was verified in the Vivado simulator. For the proposed core, 200 randomly generated rows with random keys, masks, and polynomial selections were encrypted and decrypted, and all ciphertexts and recovered plaintexts matched the reference software model without error. The AES-128 implementation reproduced the example vector in Appendix C.1 of FIPS-197 [4] for both encryption and decryption. The latency and the number of clock cycles per block were measured in the same simulations.

5.3. Implementation Results

Table 5 summarizes the post-place-and-route results. The proposed core used 1211 slice LUTs and 205 registers, compared with 2305 LUTs and 2733 registers for the AES-128 core, and neither design used block RAM or DSP resources. Most of the logic of the proposed core is used by the decryption multiplier with the degree-57 complementary factor and by the selection among the nine polynomials. The proposed core met timing with a slack of 3.509 ns and reached an estimated maximum frequency of 154.1 MHz; its critical path runs from the polynomial-selection register to the output register through four logic levels, and 80% of the delay is routing delay.

Table 5. Post-place-and-route results on Artix-7 xc7a35tcpg236-1 (Vivado 2025.2, 100 MHz constraint).

Metric	Proposed Core	AES-128 (Iterative)
Slice LUTs (utilization)	1211 (5.82%)	2305 (11.08%)
Slice registers	205	2733
Block RAM/DSP	0/0	0/0
Data block	56-bit row	128-bit block
WNS at 10 ns (ns)	3.509	0.419
Maximum frequency (MHz)	154.1	104.4
Critical-path delay (ns), logic levels	6.52, 4	9.53, 7
Latency (clock cycles)	2	55
Throughput at $f_{\max}$ (Mbit/s)	8630	243
Throughput per LUT (Mbit/s/LUT)	7.13	0.105
Dynamic/static power (mW)	16/68	19/68
Total on-chip power (mW)	85	88
Energy per bit at 100 MHz (pJ/bit)	2.9	81.6

Throughput is given in plaintext bits. Energy per bit is the dynamic power divided by the throughput at 100 MHz. The AES latency was measured in simulation and includes the wrapper registers and handshake; key expansion requires 17 additional cycles.

Because the proposed core processes one 56-bit row per clock cycle, whereas the AES core requires 55 cycles per 128-bit block, the proposed core achieved a substantially higher throughput and a lower energy per bit. This difference is mainly a consequence of the different architectures (a fully parallel linear datapath versus an iterative round-based cipher) and should not be interpreted as an equivalent level of security: as shown in Sections 3 and 4, the proposed transformation is linear over $\mathrm{GF}(2)$ and provides only local diffusion. The dynamic power of the two designs was similar (16 and 19 mW), and the total on-chip power was dominated by the device static power in both cases. Finally, the nonce-masked variant requires a pseudorandom function; if this function is realized with an AES core, as in the software experiments, its area adds to that of the proposed core. The hardware advantage of the proposed method is therefore limited to the polynomial processing layer itself, and a lightweight mask generator would be required to preserve it in a complete implementation.

6. Conclusions

This paper introduced a polynomial-based encryption framework for resource-constrained IoT and wireless communication hardware by exploiting minimal polynomials over extended Galois fields. Unlike conventional AES-type structures that rely on fixed nonlinear mappings, the proposed method performs encryption and decryption directly through polynomial convolution and complementary factorization over $\mathrm{GF}(2)$, without substitution tables. Two practical variants of the proposed method were investigated, and a nonce-dependent masking stage was added to the key-assisted variant.

The security analysis showed that exact reconstruction establishes correctness but not confidentiality. Because the transformation is linear over GF(2), deterministic reuse of the same key and polynomial allows the encryption polynomial and the key to be recovered from known or chosen plaintexts without exhaustive search. The size of the polynomial-selection space alone is therefore not a measure of security. Per-message nonce masking prevents this direct cancellation attack, although it does not constitute a formal proof of chosen-plaintext security.

Software experiments with four message lengths and 100 repeated trials verified lossless decryption for all variants. At the same implementation level, the encryption time of the key-assisted method was comparable to that of SPECK64/128, whereas the standardized AES-128 implementation was faster for messages of 8192 bits and longer. The ciphertext expansion was approximately 11%, independently of the message length. The avalanche ratio of the proposed method (0.070 per row) was well below the value of approximately 0.5 obtained by AES-128 and SPECK64/128, showing that polynomial multiplication provides only local diffusion. The constant storage was lower than that of a table-based AES-128 implementation but not lower than that of the table-free SPECK64/128.

On the Xilinx Artix-7 FPGA, the proposed core required 1211 LUTs, 205 registers, no block RAM, and no DSP slices, processed one 56-bit row per clock cycle, and reached a maximum frequency of 154.1 MHz. The iterative AES-128 core required 2305 LUTs and 2733 registers and reached 104.4 MHz. The higher throughput of the proposed core mainly reflects its fully parallel linear datapath, the dynamic power of the two designs was similar, and the pseudorandom mask generator was not included in the proposed core. These results indicate that the main strength of the method is its compact, S-box-free hardware structure. The method should therefore be used as a lightweight polynomial processing layer combined with nonce-dependent masking, rather than as a stand-alone replacement for standard ciphers.

Future work will focus on introducing a nonlinear stage to improve diffusion, implementing a lightweight mask generator in hardware, optimizing polynomial selection strategies, reducing ciphertext expansion, and performing comprehensive cryptanalytic analysis against linearand differential attack models, together with formal security analysis, authenticated-encryption extensions, and side-channel evaluation. Integration of the proposed framework into real-time wireless protocols also represents an important direction for practical deployment.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/math14214082/s1, File S1: Source code and test data, including the MATLAB benchmark script, the VHDL and Verilog sources of the proposed core and of the AES-128 reference design, the testbenches, the test vectors, and the Vivado implementation script.

Author Contributions: Conceptualization, O.G.; methodology, O.G.; software, O.G. and P.E.M.T.; validation, O.G. and P.E.M.T.; formal analysis, O.G.; investigation, O.G. and P.E.M.T.; resources, O.G.; data curation, P.E.M.T.; writing—original draft preparation, O.G.; writing—review and editing, O.G. and P.E.M.T.; visualization, P.E.M.T.; supervision, O.G.; project administration, O.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in this study are included in the article and in the Supplementary Materials (File S1). Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AES	Advanced Encryption Standard
NIST	National Institute of Standards and Technology
GF	Galois field
ECC	Elliptic curve cryptography
GCM	Galois/Counter Mode
VPN	Virtual private network
IoT	Internet of Things
FPGA	Field-programmable gate array
XOR	Exclusive-or (logic operation)
LUT	Lookup table
BRAM	Block random-access memory
DSP	Digital signal processing (slice)
CPA	Chosen-plaintext attack
CTR	Counter (mode of operation)
FF	Flip-flop
HDL	Hardware description language
KPA	Known-plaintext attack
PRF	Pseudorandom function
SD	Standard deviation
WNS	Worst negative slack

References

1. Desai, B.; Patil, K.; Mehta, I.; Patil, A. A secure communication framework for smart city infrastructure leveraging encryption, intrusion detection, and blockchain technology. *Adv. Comput. Sci.* **2024**, *7*, 1.
2. *IEEE Std 802.11-2020*; IEEE Standard for Information Technology—Telecommunications and Information Exchange between Systems—Local and Metropolitan Area Networks—Specific Requirements—Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE: Piscataway, NJ, USA, 2021.
3. Ganesh, R.; Khan, B.U.I.; Khan, A.R.; Kamsin, A.B. A panoramic survey of the Advanced Encryption Standard: From architecture to security analysis, key management, real-world applications, and post-quantum challenges. *Int. J. Inf. Secur.* **2025**, *24*, 216.
4. McKay, K.A.; Cooper, D.A. *Federal Information Processing Standard (FIPS) 197: Advanced Encryption Standard (AES)*; U.S. Department of Commerce: Washington, DC, USA, 2001.
5. National Institute of Standards and Technology. *Post-Quantum Cryptography Standards (FIPS 203, 204, 205)*; U.S. Department of Commerce: Washington, DC, USA, 2024.
6. Lidl, R.; Niederreiter, H. *Introduction to Finite Fields and Their Applications*, revised ed.; Cambridge University Press: Cambridge, UK, 1994.
7. Dworkin, M.J. *NIST SP 800-38D: Recommendation for Block Cipher Modes of Operation, Galois/Counter Mode (GCM) and GMAC*; NIST: Gaithersburg, MD, USA, 2007.
8. Ramakrishna, R.P.S.; Boppana, L. A high-throughput fully digit-serial polynomial basis finite field GF(2^m) multiplier for IoT applications. In Proceedings of the TENCON 2019–IEEE Region 10 Conference, Kochi, India, 17–20 October 2019; pp. 920–924.
9. Shakor, M.Y.; Khaleel, M.I.; Safran, M.; Alfarhood, S.; Zhu, M. Dynamic AES encryption and blockchain key management: A novel solution for cloud data security. *IEEE Access* **2024**, *12*, 26334–26343.
10. Kadhim, S.A.; Yas, R.M.; Abdual Rahman, S.A. Advancing IoT device security in smart cities: Through innovative key generation and distribution with D_F, Gf, and multi-order recursive sequences. *J. Cybersecur. Inf. Manag.* **2024**, *13*, 84–95.
11. Alyami, R.Y. Secure IoT transmission in 6G smart cities: A quantum-resilient hybrid Galois field and Reed–Solomon approach. *J. Supercomput.* **2025**, *81*, 576.
12. Ansari, S.A.; Siddiqui, A.S.; Khan, M.A.; Alzahrani, B.A.; Ahmad, S. A systematic review of lightweight cryptographic schemes for security and privacy in IoT. *Discov. Comput.* **2025**, *28*, 266.
13. Hazzazi, M.M.; Attuluri, S.; Bassfar, Z.; Joshi, K. A novel cipher-based data encryption with Galois field theory. *Sensors* **2023**, *23*, 3287.

14. Al-Maadeed, T.; Hussain, I.; Anees, A.; Mustafa, M.T. An image encryption algorithm based on chaotic Lorenz system and novel primitive polynomial S-boxes. *Multimed. Tools Appl.* **2021**, *80*, 24801–24822.
15. Lin, Y.; Wei, Y.; Chen, D.; Li, Y.; Erkan, U.; Toktaş, A.; Gao, S.; Zhang, Y. Cryptanalysis and Improvement of a Video Cryptosystem via Chaos and S-Box. *ACM Trans. Multimed. Comput. Commun. Appl.* **2026**, *22*, 173. https://doi.org/10.1145/3808699.
16. Lin, Y.; Liao, Y.; Zeng, W.; Wei, Y.; Chen, D.; Yuan, X.; Li, Y.; Erkan, U.; Toktaş, A.; Zhang, C.; et al. 3D Non-degenerate Hyperchaos: Design, Analysis, and Application in Image Encryption. *IEEE Trans. Consum. Electron.* **2026**, *72*, 5483–5494.
17. IEEE Std 802.15.4-2020; IEEE Standard for Low-Rate Wireless Networks. IEEE: Piscataway, NJ, USA, 2020.
18. Bille, A.; Tischhauser, E. Cryptanalysis of authenticated encryption modes for wireless and real-time systems. In *Advances in Information and Computer Security, IWSEC 2024*; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2024; Volume 14977.
19. Selvi, P.; Sakthivel, S. A hybrid ECC-AES encryption framework for secure and efficient cloud-based data protection. *Sci. Rep.* **2025**, *15*, 30867.
20. Beaulieu, R.; Shors, D.; Smith, J.; Treatman-Clark, S.; Weeks, B.; Wingers, L. The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Paper 2013/404. 2013. Available online: https://eprint.iacr.org/2013/404 (accessed on 13 October 2026).
21. Strömbergson, J. AES: Verilog Implementation of the AES Block Cipher. GitHub Repository, Commit 80dc471. Available online: https://github.com/secworks/aes (accessed on 22 October 2026).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.