Post-print of Hill Cipher Derivative with Tall Matrix as Key over Galois Field GF(2^8)

Authors: Liu Haifeng, Lu Kaiyi, Liang Xingliang

Date: 2018-06-19T00:00:00+00:00

Abstract

Addressing the limitation that the traditional Hill encryption algorithm merely utilizes an invertible numerical square matrix over the Galois field GF(p) as the key matrix to perform modulo-p multiplication with the plaintext vector for encryption operations, this paper proposes a novel Hill encryption derivative algorithm that employs polynomial matrices as the key matrix over the Galois field GF(2)[x]/p(x). In the Hill encryption derivative algorithm, the plaintext vector is a polynomial vector composed of polynomials corresponding to plaintext characters, a column of the key matrix is randomly selected as the shift increment during encryption, multiplication and addition of the key matrix and plaintext vector modulo the 8th-degree irreducible polynomial p(x) are performed over GF(2)[x]/p(x), and a ciphertext vector whose elements are polynomials is subsequently obtained, thereby achieving plaintext information encryption. When an attacker attempts to crack the ciphertext to obtain the plaintext without knowledge of p(x), the key matrix, and the randomly selected shift vector, the difficulty is significantly greater, thereby enhancing the attack resistance capability of the Hill encryption derivative algorithm over the Galois field GF(2)[x]/p(x).

Full Text

Hill Encryption Derivative Algorithm in Galois Field GF(2^8) with High-Matrix as Key Matrix

Liu Haifeng^{a,b}, **Lu Kaiyi**^{a}, **Liang Xingliang**^{b} ^{a}College of Electrical & Information Engineering, ^{b}College of Arts & Sciences, Shaanxi University of Science & Technology, Xi' an 710021, China

Abstract

The traditional Hill encryption algorithm employs an invertible numerical square matrix over Galois field GF(p) as the key matrix, performing modulo-p multiplication with plaintext vectors to compute ciphertext. This paper proposes a novel Hill encryption derivative algorithm that utilizes polynomial high-matrices as key matrices in the Galois field GF(2)[x]/p(x). In this derivative algorithm, the plaintext vector consists of polynomials derived from corresponding plaintext characters. A randomly selected column of the key matrix serves as a translation increment. Encryption is achieved through multiplication and addition of the key matrix and plaintext vector modulo an 8th-degree irreducible polynomial p(x) in GF(2)[x]/p(x), yielding a ciphertext vector whose elements are polynomials. This approach significantly increases the difficulty for attackers to recover plaintext from ciphertext without knowledge of p(x), the key matrix, and the randomly selected translation vector, thereby enhancing the attack resistance of the Hill encryption derivative algorithm over GF(2)[x]/p(x).

Keywords: Galois field GF(2)[x]/p(x); Hill encryption; polynomial high-matrix; irreducible polynomial

0 Introduction

Galois fields, also known as finite fields, contain a finite number of elements [1,2]. Traditional Hill encryption algorithms construct a coded character set from English letters, digits, and common symbols. When the cardinality of this set is not prime [3], special care must be taken to ensure the determinant of the encryption matrix has a multiplicative inverse modulo p. References [4,5] present the requirements for numerical square matrices as key matrices under modulo 26 and provide methods for selecting such matrices. Reference [6] addresses the issue where inverse matrices under modulo 26 may contain fractions and proposes improvements through row and column transformations. When p is prime, a Galois field GF(p) with p elements can be obtained, yet the Hill encryption key matrix remains vulnerable.

The field GF(2^8) is a special finite field [7] containing 256 elements, rather than requiring p elements where p must be prime. Each element in GF(2^8) can be represented as an 8-bit binary number and uniquely mapped to a univariate polynomial of degree less than 8 with coefficients 0 or 1. The polynomial addition and multiplication operations in this finite field are closed, making it an important mathematical tool in cryptography and information coding [8]. The arithmetic operations in finite fields possess certain complexities and peculiarities. Compared to traditional GF(p) Hill encryption, GF(2^8) substantially improves algorithm security.

This paper discusses Hill encryption over the finite field GF(2^8) as a derivative of traditional Hill encryption, employing a pair of pseudo-inverse matrices over

GF(2^8) as encryption and decryption keys, which satisfies the fundamental conditions of secure cryptosystems. We select a space character and 94 distinct visible characters for character encoding, as shown in Table 1 , and encode characters in row-major order.

Given the one-to-one correspondence and isomorphic properties between elements of GF(2^8) and their operations, the structures are completely equivalent. Therefore, the following discussion focuses on the Hill encryption derivative algorithm over GF(2^8). The corresponding operations over GF(2^8) are defined as follows:

---

1 Galois Field GF(2^8)

Let GF(2)[x] denote the set of all univariate polynomials over GF(2). For an 8th-degree irreducible polynomial p(x) over GF(2), the algebraic structure GF(2)[x]/p(x), +, *forms a finite field where: - The additive group is an Abelian group with the zero polynomial as identity - The multiplicative group of non-zero elements is an Abelian group - Operations satisfy the distributive law: a(x), b(x), c(x) S, (a(x) + b(x))* c(x) = a(x) * c(x) + b(x) * c(x) and c(x) * (a(x) + b(x)) = c(x) * a(x) + c(x) * b(x)

The set S = {a(x) | a(x) GF(2)[x], deg(a(x)) < 8} contains all polynomials of degree less than 8 over GF(2). For any a(x), b(x) S where a(x) = Σ_{i=0}^7 a_i x^i and b(x) = Σ_{i=0}^7 b_i x^i with a_i, b_i {0,1}: - Addition: a(x) + b(x) = Σ_{i=0}^7 (a_i b_i) x^i (mod 2) - Multiplication: a(x) * b(x) = Σ_{i=0}^7 Σ_{j=0}^7 (a_i × b_j) x^{i+j} mod p(x)

For polynomial inversion, given a(x) GF(2)[x]/p(x) with a(x) ≠ 0, since p(x) is irreducible, gcd(a(x), p(x)) = 1, guaranteeing a unique multiplicative inverse b(x) such that a(x) * b(x) 1 mod p(x). The inverse can be computed using the extended Euclidean algorithm:

a) Initialize: Let r_0 = p(x), r_1 = a(x), s_0 = 1, s_1 = 0, t_0 = 0, t_1 = 1
b) Iteration: While r_i ≠ 0, compute quotient q_i = r_{i-1} div r_i, then update: r_{i+1} = r_{i-1} - q_i * r_i s_{i+1} = s_{i-1} - q_i * s_i t_{i+1} = t_{i-1} - q_i * t_i
c) When r_k = 1, the inverse is b(x) = t_k mod p(x)

---

2 Polynomials and Polynomial Matrices over GF(2^8)

2.1 Polynomial Inversion over GF(2^8)

For a non-zero polynomial a(x) GF(2)[x]/p(x), there exists a unique multiplicative inverse polynomial b(x) GF(2)[x]/p(x) such that a(x) * b(x) 1 mod

p(x). The inversion process involves: 1. Computing the product a(x) * b(x) = c(x) where deg(c(x)) may exceed 7 2. Replacing terms x^m with m > 7 by their remainders modulo p(x) 3. Solving the resulting system of equations for the coefficients b_i of b(x) 4. Using Gaussian elimination under modulo 2 to solve for the inverse polynomial coefficients

2.2 Matrix Inversion over GF(2^8)

For an n×n polynomial matrix A over GF(2)[x]/p(x), if det(A) ≠ 0, the inverse matrix A^{-1} can be computed as: A^{-1} = (det(A))^{-1} * A^ *where A^* is the adjugate matrix whose elements are the cofactors of A. Each cofactor is a polynomial, and (det(A))^{-1} is the multiplicative inverse of the determinant polynomial computed using the method in Section 2.1.

2.3 Left Pseudo-Inverse of Polynomial High-Matrices

In linear algebra, a high-matrix refers to a matrix with linearly independent columns (column full-rank). Invertible square matrices are a special case of high-matrices.

A polynomial high-matrix over GF(2^8) is a column full-rank polynomial matrix where elements belong to GF(2)[x]/p(x). For an l×k high-matrix A with l > k, the left pseudo-inverse A_{pinv} can be computed as: A_{pinv} = (A^T * A)^{-1} * A^T

The computation involves: 1. Calculating A^T * A (a k×k square matrix) 2. Computing its determinant and verifying it is non-zero 3. Finding the inverse of A^T * A using the method in Section 2.2 4. Multiplying by A^T to obtain the left pseudo-inverse

---

3 Hill Encryption Derivative Algorithm over GF(2^8)

References [12-14] propose ideas for derivative Hill encryption and block encryption over finite fields, including combining finite field conic curve cryptosystems with Hill block encryption. This paper presents the following derivative of the general Hill encryption algorithm.

3.1 Encryption Algorithm over GF(2^8)

Assume a plaintext string M = M_1 M_2 ···M_m composed of characters from the character encoding set. To encrypt: 1. Select a column full-rank polynomial high-matrix eKey of size l×k over GF(2)[x]/p(x) as the encryption key matrix 2. Randomly select column t of eKey as the translation increment a_t(x) 3. Group plaintext into blocks of k characters: M = (M_1···M_k)(M_{k+1}··· M_{2k})···4. For each block, convert characters to index values from Table 1, then to binary polynomials

For the first block with characters C_1,⋯,C_k: - Convert each character C_j to its index Index_j - Represent Index_j as binary polynomial f_j(x) by substituting 2 for x in the binary representation - Form plaintext polynomial vector f(x) = [f_1(x), f_2(x), ⋯, f_k(x)]^T

The ciphertext vector e(x) is computed as: e(x) = eKey * f(x) + a_t(x) (mod p(x))

where a_t(x) is the t-th column of eKey. The resulting ciphertext string C has length l characters per k-character plaintext block.

3.2 Decryption Algorithm over GF(2^8)

Decryption requires: 1. Computing the left pseudo-inverse dKey of eKey over GF(2)[x]/p(x) 2. Grouping ciphertext into blocks of l characters 3. For each ciphertext block g(x) = [g_1(x), ⋯, g_l(x)]^T: - Subtract the translation increment: g' (x) = g(x) - a_t(x) - Multiply by the decryption matrix: f(x) = dKey * g' (x) (mod p(x)) - Convert polynomial vector f(x) back to characters by evaluating at x=2 and mapping to indices

The matrix representation is: f(x) = dKey * (e(x) - a_t(x)) (mod p(x))

where a_t(x) denotes the t-th column of eKey.

---

4 Irreducible Polynomials in Polynomial Ring GF(2)[x]

An 8th-degree irreducible polynomial over GF(2) has coefficients in {0,1} and cannot be factored into products of polynomials of degree $ $7. To find the set S of all 8th-degree irreducible polynomials: 1. Construct the set N of all 8th-degree polynomials over GF(2) 2. Compute the set B of reducible polynomials as: B = B_1 B_2 B_3 B_4 where: - B_1: products of any 1st-degree and any 7th-degree polynomial - B_2: products of any 2nd-degree and any 6th-degree polynomial - B_3: products of any 3rd-degree and any 5th-degree polynomial - B_4: products of any two 4th-degree polynomials 3. Compute S = N B

A Python program can implement this by enumerating all polynomials and performing difference set operations. Table 2 lists all 8th-degree irreducible polynomials as coefficient vectors.

---

5 Example Verification of Hill Encryption Derivative Algorithm over GF(2^8)

Consider encrypting the string: "hello, It' s nice to meet you" .

Select the 8th-degree irreducible polynomial: p(x) = x^8 + x^4 + x^3 + x + 1

Choose a 3$×$2 column full-rank polynomial high-matrix as the encryption key matrix: eKey = [[x^2+1, x^3+x2+x], [x^3+x2, x^5+x3+x^2+x], [x^5+x4+x^3+x2, x^6+x5+x^4+x3+x^2]]

Select column 2 as the translation increment a_2(x) = [x^3+x2+x, x^5+x3+x^2+x, x^6+x5+x^4+x3+x^2]^T.

Compute the decryption matrix dKey as the left pseudo-inverse of eKey: 1. Compute det(eKey) = x^6 + x^5 + x^4 + x^2 + x + 1 2. Find its inverse: (det(eKey))^{-1} = x^4 + x^2 + x + 1 (mod p(x)) 3. Compute dKey = (eKey^T * eKey)^{-1} * eKey^T

Resulting in: dKey = [[x^6+x5+x^4+x3+x^2+x+1, x^5+x4+x^3+x2+1, x^4+x3+x^2+x], [x^6+x5+x^4+x3+1, x^5+x4+x^3+1, x^4+x3+x^2]]

5.1 Encryption Phase

Step 1: Convert plaintext characters to index values using Table 1, yielding the index sequence shown in Table 3 .

Step 2: Since eKey is 3$×2, $group plaintext into blocks of 2 characters. For the first block "he" : $-Indices : h→8, e→$5 - Binary: 8 = 1000_2 → x^3, 5 = 0101_2 → x^2 + 1 - Plaintext vector: f(x) = [x^3, x^2 + 1]^T

Compute ciphertext: e(x) = eKey * f(x) + a_2(x) (mod p(x)) = [x^3 + x^2 + x + 1, x^6 + x^5 + x^4 + x^3 + x^2 + x, x^6 + x^5 + x^4 + x^3 + x^2]^T

Convert each polynomial to binary coefficients, then to decimal indices: - x^3 + x^2 + x + 1 → 1111_2 = 15 - x^6 + x^5 + x^4 + x^3 + x^2 + x → 1111110_2 = 126 - x^6 + x^5 + x^4 + x^3 + x^2 → 1111100_2 = 124

Map to characters: 15→k, 126→+, 124→. (using Table 1), producing ciphertext "k.+" . Repeat for all blocks to obtain the full ciphertext shown in Table 4 .

Observation: The ciphertext length is 1.5 times the plaintext length (3 characters per 2-character block), a characteristic of high-matrix encryption where ciphertext and plaintext lengths need not match.

5.2 Decryption Phase

Step 1: Compute the left pseudo-inverse matrix dKey (already obtained above).

Step 2: Group ciphertext into blocks of 3 characters. For the first block "k.+": - Indices: k$→15, +→126, .→$124 - Convert to polynomials: g(x) = [x^3+x2+x+1, x^6+x5+x^4+x3+x^2+x, x^6+x5+x^4+x3+x^2]^T

Step 3: Decrypt: f(x) = dKey * (g(x) - a_2(x)) (mod p(x)) = [x^3, x^2 + 1]^T

Convert to indices: x^3$→8, $x^2 + 1$→$5, yielding plaintext characters "he" . Repeat for all blocks to recover the original message "hello, It' s nice to meet you" .

6 Security Analysis

Algorithm security depends on key security and attack complexity. The proposed scheme exhibits several security advantages:

6.1 One-Time Pad Communication

The key matrix is a randomly selected polynomial high-matrix over GF(2^8) that is inherently unpredictable, making both encryption and decryption functions secure. Using a fresh key matrix for each communication session achieves one-time pad encryption [16]. Shannon proved that one-time pads provide absolute security—previously cracked ciphertexts offer no assistance in decrypting subsequent messages, making the system theoretically unbreakable [16].

6.2 Avalanche Effect

Since Hill encryption mixes plaintext message blocks, minute changes in plaintext or key produce significant ciphertext alterations. With larger key matrices, a single-bit plaintext change affects an entire ciphertext block, and a single-bit key change impacts fixed positions across all blocks. This avalanche effect prevents attackers from analyzing ciphertext patterns to deduce the key matrix.

6.3 Large Key Space

The GF(2^8) Hill derivative algorithm significantly expands the key space. For conventional polynomial square key matrices, attackers who determine the matrix order through ciphertext length analysis can brute-force all matrices of that order. However, high-matrices have unequal row and column counts, making ciphertext and plaintext lengths inconsistent, which effectively resists ciphertext-only brute-force attacks. Higher-order key matrices exponentially increase the number of possible polynomial matrices, and different irreducible polynomials p(x) produce different finite fields, further complicating brute-force attempts.

6.4 Translation Increment

The algorithm introduces a translation increment a_t(x) randomly selected from the key matrix columns. Decryption requires subtracting this increment before applying the pseudo-inverse. Since attackers cannot determine whether a translation increment exists or which column was used, attack complexity increases substantially.

7 Security Model

Based on the security analysis, we evaluate attack resistance:

7.1 Ciphertext-Only Attack

In ciphertext-only attacks, adversaries possess minimal information. In our high-matrix-based scheme, attackers cannot determine: - The plaintext length from ciphertext - The irreducible polynomial p(x) defining the finite field - The key matrix dimensions - Whether the key is a special high-matrix (invertible square matrix) - The randomly selected translation increment column

Thus, the algorithm is secure against ciphertext-only attacks.

7.2 Known-Plaintext Attack

Known-plaintext attacks provide specific plaintext-ciphertext pairs. While adversaries can infer the ratio l:k of rows to columns in the high-matrix from length ratios, they cannot determine: - The exact matrix dimensions - The specific irreducible polynomial p(x) - The translation increment

Therefore, the algorithm remains secure against known-plaintext attacks.

7.3 Chosen Plaintext/Ciphertext Attack

These stronger attacks grant adversaries access to encryption/decryption oracles, enabling them to obtain ciphertexts (or plaintexts) for self-constructed inputs. If the key remains fixed across sessions, the system becomes vulnerable. Our scheme mitigates this through one-time pad communication, where each session uses a fresh key matrix, and translation increments are cyclically selected from key matrix columns per block. This ensures forward and backward security—compromise of current keys does not endanger past or future communications.

---

8 Conclusion

The Hill encryption concept is fundamental to cryptography education. This paper proposes a novel GF(2^8) Hill encryption derivative algorithm that advances classical Hill encryption. Even with a fixed key high-matrix, the scheme resists known-plaintext attacks on the key matrix, offering superior security compared to traditional Hill encryption. This work provides valuable insights for further research into the Hill cryptosystem.

References

[1] Feng Keqin. *Finite Fields* [M]. Changsha: Hunan Education Press, 1991: 24-57.

[2] Hu Guanzhang. *Applied Modern Algebra* [M]. Beijing: Tsinghua University Press, 1993: 171-193.

[3] Wan Fuyong, Dai Haohui. The design of dummy variable in hill2 coding system [J]. *Mathematics in Practice and Theory*, 2007 (8): 87-90.

[4] Yang Shuju. Method for encryption and decryption matrix of hill cipher [J]. *Value Engineering*, 2016 (26): 285-287.

[5] Xu Xiaohua, Li Minying. The solution of matrix of hill cipher in encryption and decryption [J]. *Computer and Information Technology*, 2010 (2): 31-33.

[6] Wang Rong, Liao Qunying, Wang Yunying, et al. Improvement of hill encryption algorithm [J]. *Journal of Sichuan Normal University: Natural Science*, 2015 (1): 8-14.

[7] Fu Weiping, Chen Jiye. An algorithm to implement division operation of finite field GF(2^n) [J]. *Journal of Shaoyang University: Natural Science Edition*, 2015 (2): 3-10.

[8] Pu Baoxing, Wang Weiping. Evaluation and analysis of the computation cost of linear network coding [J]. *Journal on Communications*, 2011 (5): 47-55.

[9] Jiao Zhanya, Zeng Yongying, Liu Haifeng. Design and realizing of a system of block cipher [J]. *Journal of Xi' an University of Science and Technology*, 2005 (4): 477-480.

[10] Xie Bangjie. *Linear Algebra* (First Edition) [M]. Beijing: Higher Education Press, 1978.

[11] Wang Songgui. *Generalized Inverses and Applications* [M]. Beijing: Beijing University of Technology Press, 2006.

[12] Zhang Yu' an, Feng Dengguo. A practical one-time-pad-like block cipher scheme [J]. *Journal of Beijing University of Posts and Telecommunications*, 2005 (2): 101-104.

[13] Liu Haifeng, Wu Peng, Ma Lingkun. Implementation of group encryption algorithm based on conic curve over finite fields [J]. *Journal of Jilin University: Science Edition*, 2012, 50 (1): 54-58.

[14] Lu Kaicheng. *Computer Cryptology—Data Privacy and Security in Computer Networks* (Third Edition) [M]. Beijing: Tsinghua University Press, 2003.

[15] Liu Haifeng, He Liyong, Guo Gaihui, et al. The dummy and encryption matrix in the Hill coding system [J]. *Journal of Southwest University: Natural Science Edition*, 2014, 36 (11): 138-142.

[16] Qin Sihan. *Cryptography and Computer Network Security* [M]. Beijing: Tsinghua University Press, 2001.

[17] Li Sujuan, Zhang Mingwu, Zhang Futai. Security analysis and improvement of CCA secure PKE with (continual) auxiliary input [J]. *Chinese Journal of Computers*, 2017: 1-13.

[18] Wang Xin, Xue Rui. Analysis of a new CCA-secure public-key cryptosystem [J]. *Journal of Cryptologic Research*, 2017 (2): 106-113.

[19] Wang Zhanjun, Ma Haiying, Wang Jinhua. Public key encryption scheme with auxiliary inputs based on indistinguishability under adaptive chosen ciphertext attack [J]. *Journal of Computer Applications*, 2014 (5): 1288-1291.

Note: Figure translations are in progress. See original paper for figures.