Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

Cryptanalysis of Augmented Hill Cipher

Rania Saeed[1], **Hatem M. Bahig**[1,2], **Dieaa I. Nassr**[1*]

[1] Department of Mathematics, Faculty of Science, Ain Shams University, Cairo, Egypt
[2] College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 13318, Saudi Arabia
E-mail: dieaa.nassr@sci.asu.edu.eg

Received: 1 September 2025; **Revised:** 7 October 2025; **Accepted:** 15 October 2025

Abstract: The Augmented Hill Cipher (AHC) is a modified Hill Cipher that uses three secret keys and semi-cipher block chaining as its mode of operation. AHC is claimed to be secure. Recently, it has been used to obtain secure, fast, and loss-tolerant communication. In this paper, we present the first cryptanalysis of AHC. We show that AHC is vulnerable to two attacks: known-plaintext and chosen-plaintext attacks. The attacks are performed by introducing a new method to solve non-linear equations obtained by analyzing AHC. The attacks take place in polynomial time.

***Keywords*:** cryptanalysis, Hill Cipher, Augmented Hill Cipher (AHC), known-plaintext attack, chosen-plaintext attack

MSC: 94A60, 68P25, 68M25, 11T71

1. Introduction

A Symmetric-Key Cryptosystem (SKC) [1] is a cryptographic primitive that uses the same key for both encryption and decryption. It plays an important role in ensuring sensitive information's confidentiality and integrity across digital platforms. There are two categories of SKC: block and stream ciphers. In block ciphers, the plaintext is divided into a sequence of blocks, each of the same bit size. Each block is then encrypted with one of the modes of operations, such as the electronic codebook, cipher block chaining, or counter. There are two primitive steps included in the construction of modern block ciphers: (i) substitution (called S-box) and (ii) permutation (called P-box). Substitution techniques involve replacing plaintext elements (characters or bits) with different elements according to a rule, typically determined by a key. This process obscures the original message to create ciphertext, which can be decrypted back into plaintext using the same key or algorithm. A permutation (or transposition) technique rearranges the elements (or bits) of the message. Substitution techniques can be classified into two types: (i) monoalphabetic substitution, where each character (or bit) in the plaintext is replaced by another character (or bit) according to a fixed rule consistently throughout the message; and (ii) polyalphabetic substitution, where each occurrence of a character (or element) in the plaintext may have a different substitute. Modern block ciphers, such as Advanced Encryption Standard (AES) [2] and Data Encryption Standard (DES) [3], are iterated block ciphers where each iteration (round) applies the concepts of confusion (using S-box) and diffusion (using P-box) and key round.

The Hill Cipher is a polygraphic classical substitution cipher that uses square matrices. It is devised by [4, 5]. It uses matrix multiplication, making it resistant to frequency analysis, i.e., statistical properties of the English language. The

DOI: https://doi.org/10.37256/cm.6620258475

plaintext P is represented by $m \times m$ matrix (or a vector of m tuples) over $\mathbb{Z}_n$. The ciphertext C is a multiplication of P by the secret key K, where K is an invertible $m \times m$ matrix over $\mathbb{Z}_n$. Therefore, the Hill Cipher can be described as $C = PK \pmod{n}$ for encryption, and $P = CK^{-1} \pmod{n}$, for decryption.

Matrix operations are one of the most common and important operations used in the design of many ciphers. Although the Hill Cipher hides frequency information for plaintext components (i.e., diffusion property), it is insecure due to known-plaintext attacks.

A known-plaintext attack [6] is a type of cryptographic attack where an adversary possesses both the ciphertext and the corresponding plaintext. By analyzing the relationship between the known plaintext and its corresponding ciphertext, the attacker attempts to deduce the encryption key or the plaintext corresponding to a newly received ciphertext obtained by the same key. This type of attack exploits weaknesses in the encryption algorithm or key creation to compromise the security of the system. Defending against known plaintext attacks involves the use of strong encryption algorithms and ensuring secure key management practices.

Many enhancements to the Hill Cipher have been proposed and subjected to extensive cryptanalysis-particularly to counter known-plaintext attacks-but most have proved insecure [7–18].

Ismail et al. [8] improved the Hill Cipher by dynamically generating a new key matrix for each plaintext block to reduce susceptibility to known-plaintext attacks. Rangel et al. [19] found the improvement of Ismail et al. [8] still vulnerable to known-plaintext attack since the first block uses the original Hill key matrix. They also noted image-encryption issues with all-zero blocks. Li et al. [13] concluded that Hill-based image encryption is unsuitable due to weak diffusion. Toorani and Falahati [20, 21] proposed uses of the hash function SHA-1 to modify each encryption step, and claimed its security against ciphertext-only, known-plaintext, chosen-plaintext, and chosen-ciphertext attacks.

Keliher et al. [9] found that Falahati [20, 21] still vulnerable to a chosen-plaintext attack. In particular, encrypting chosen all-zero blocks leaked key information. Sastry et al. [22] proposed iterated Hill Ciphers with multiple rounds of key-based transformations, bit-level operations, and permutations to obscure linear dependencies. Keliher and Thibodeau [10] showed that the improvements of Sastry et al. [22] are weak against slide attacks, so additional rounds did not improve security. Sazaki et al. [17] combined the Hill Cipher with affine permutations to encrypt images. They concluded that affine permutations enhance diffusion and resistance to statistical and differential attacks for multimedia. Khazaei et al. [11] reduced the complexity of cipher-only attack of the original Hill Cipher $\mathcal{O}(m^3 26^{m^2})$ to $\mathcal{O}(m26^m)$, where $n = 26$, i.e., for the English language. They also showed that, using the Chinese Remainder Theorem, the attack can be reduced down to $\mathcal{O}(m13^m)$.

Paragas et al. [15] proposed a simplified variant using XOR, shifts, and Radix64, reporting a 53% avalanche effect and improved diffusion over the classical Hill Cipher. Qobbi et al. [16] combined chaotic substitution with affine transforms to encrypt images, demonstrating stronger diffusion and resistance to differential attack.

In [14], the authors analyze the vulnerabilities of [23], which employ elliptic curve cryptography and the Hill Cipher. The authors proposed an improvement of [23] by adding 3D chaotic map to encrypt images. Kumar et al. [12] proposed a Modified Hill Cipher using Radix 64 conversion. Their method introduces a multi-stage process: converting plaintext into binary, trimming redundant bits, encoding using Radix 64, and finally performing matrix-based encryption under modulo 64. Chen et al. [7] improved the Hill Cipher by randomly generating a high-order Hill key matrix based on the modular multiplicative inverse of a triangular matrix in a finite field and a rational number field.

The Augmented Hill Cipher (AHC) [24] is a variant of the Hill Cipher, reporting improved resistance to known or chosen plaintext attacks with good efficiency. Its design aims to enhance the versatility and robustness of the original Hill Cipher while maintaining its fundamental encryption principles based on matrix operations. It uses three secret keys and semi-cipher block chaining as its mode of operation. The security of the encryption and decryption processes is enhanced by performing the XOR operation and matrix operations over $\mathbb{Z}_n$. The size of the key space is large

$$p^{(k-1)m^2} \prod_{i=0}^{m-1} (p^m - p^i),$$

where $n = p^k$, p is a prime.

The security analysis of AHC in [24] claims that AHC is secure against attacks, particularly known plaintext attacks. To avoid a brute-force attack, one can choose $m = 8$ and $n = 2^{16}$. Recently, Nishida [25] used AHC (with random network coding) to obtain secure, fast, and loss-tolerant communication. His technique is faster than hardware-accelerated AES-256. Therefore, we are interested in studying the security of AHC. We subject AHC to two types of attacks: known-plaintext, and chosen-plaintext attacks.

In this paper, we show that AHC is insecure using (1) the known-plaintext attack under some conditions; and (2) the chosen-plaintext attack. To the best of our knowledge, this is the first cryptanalysis of AHC. The technique used for cryptanalysis of AHC is new and based on finding a relationship between known plaintexts, ciphertexts, and unknown keys. We also show that both attacks take $\mathcal{O}((l^2 + \log n)m^3)$, where l denotes the number of known pairs of plaintexts and ciphertexts.

The paper is organized as follows. Section 2 consists of two subsections. The first subsection presents a brief description of AHC. The other subsection reviews and introduces new binary operations used throughout the paper. Section 3 presents the first proposed attack: known-plaintext attack. The polynomial time complexity of the proposed attack is also presented in this section. Section 4 presents the chosen-plaintext attack and its polynomial time complexity. Section 5 presents illustrative examples of the known-plaintext and chosen-plaintext attacks. Finally, section 6 presents the conclusion.

2. Preliminaries

We provide a brief description of AHC [24]. Then, we review and introduce new binary operations used throughout the paper.

2.1 *AHC*

Before communication, the two parties agree on three different random secret keys K_0, K_1, and K_2 where K_0, K_1, and K_2 are invertible $m \times m$ matrices over $\mathbb{Z}_n$. The two positive integers m and n are security parameters.

2.1.1 *Encryption*

The plaintext P is divided into, say $l + 1$ blocks: $P_0, P_1, \ldots, P_l$, where each block P_i $(0 \le i \le l)$ is an $m \times m$ matrix over $\mathbf{Z_n}$. Each matrix P_i can be considered as a block of size $m^2 \times \lceil \log n \rceil$ bits.

If the length P is not a multiple of m^2, we add additional random characters (or bits) at the end of the last block P_l. The ciphertext $C : C_0, C_1, \ldots, C_l$ is computed as follows:

$$C_i' = \left(K_{i \pmod 3}\, P_i + K_{(i+1) \pmod 3}\right) \pmod n,\ 0 \le i \le l$$

$$C_i = \begin{cases} C_i' \oplus K_2 & \text{if } i = 0; \\ C_i' \oplus C_{i-1}' & \text{if } 1 \le i \le l. \end{cases}$$

Algorithm 1 describes the AHC encryption process.

Algorithm 1 Encryption.
Require: $K_0, K_1, K_2, P_0, P_1, \ldots, P_l$.
Ensure: $C_0, C_1, \ldots, C_l$.
1: $C'_0 = (K_0 P_0 + K_1) \pmod n$
2: $C_0 = C'_0 \oplus K_2$
3: $i = 1$
4: **while** $i \leq l$ **do**
5: $\quad C'_i = (K_{i \pmod 3} P_i + K_{(i+1) \pmod 3}) \pmod n$
6: $\quad C_i = C'_i \oplus C'_{i-1}$
7: $\quad i = i + 1$
8: **end while.**

2.1.2 *Decryption*

The plaintext P_i can be recovered from the ciphertext C_i, $0 \leq i \leq l$, as follows:

$$C'_i = \begin{cases} C_i \oplus K_2 & \text{if } i = 0, \\ C_i \oplus C'_{i-1} & \text{if } 1 \leq i \leq l \end{cases}$$

$$P_i = K_{i \pmod 3}^{-1} \left(C'_i - K_{(i+1) \pmod 3}\right) \pmod n, \quad 0 \leq i \leq l.$$

Algorithm 2 describes the AHC decryption process.
Algorithm 2 Decryption.
Require: $K_0, K_1, K_2, C_0, C_1, \ldots, C_l$.
Ensure: $P_0, P_1, \ldots, P_l$.
1: $C'_0 = (C_0 \oplus K_2)$
2: $P_0 = K_0^{-1} \times (C'_0 - K_1) \mod n$
3: $i = 1$
4: **while** $i \leq l$ **do**
5: $\quad C'_i = C_i \oplus C'_{i-1}$
6: $\quad P_i = K_{i \pmod 3}^{-1} \left(C'_i - K_{(i+1) \pmod 3}\right) \pmod n$
7: $\quad i = i + 1$
8: **end while.**

2.2 *Bitwise operations*

This section briefly reviews and introduces new rules on the bitwise operations we will use in our result.

Let x be a positive integer of size k-bit. The binary representation of x is written as $x = x_{k-1}x_{k-2}\cdots x_0$, where $x_i \in \{0, 1\}$, $0 \leq i \leq k-1$.

Define $x^{(i)}$ as $x \pmod{2^i}$, that is, $x^{(i)} = x_{i-1}x_{i-2}\cdots x_0$. Then

1. $x^{(i)} = x^{(i-1)} + 2^{i-1}x_{i-1}$.
2. $x^{(i)} = x^{(i-1)} \oplus 2^{i-1}x_{i-1}$.
3. If $b, b' \in \{0, 1\}$, then $(b2^{i-1} + b'2^{i-1}) \pmod{2^i} = (b2^{i-1} \oplus b'2^{i-1}) \pmod{2^i} = (b \oplus b')2^{i-1} \pmod{2^i}$.
4. If $b \in \{0, 1\}$, then

$$(x^{(i)} + 2^{i-1}b) \pmod{2^i} = (x^{(i-1)} + 2^{i-1}(x_{i-1} + b)) \pmod{2^i}$$

$$= x^{(i-1)} \oplus 2^{i-1}(x_{i-1} \oplus b).$$

5. If $y \oplus z$ is of size k-bits, then the ones' complement of $y \oplus z$ is given by

$$2^k - 1 - (y \oplus z) = (2^k - 1) \oplus (y \oplus z),$$

i.e., flipping all the bits of $y \oplus z$. Therefore,

$$(2^k - 1) - (y \oplus z) \pmod{2^k} = (2^k - 1) \oplus (y \oplus z) \pmod{2^k}$$

$$= (2^k - 1 - y) \oplus z \pmod{2^k}.$$

We can also define these operations on $m \times m$ matrices over $\mathbb{Z}_{2^k}$.

Let $X = (x_{pq})_{1 \le p,\, q \le m}$ be an $m \times m$ matrix over $\mathbb{Z}_{2^k}$, where each element x_{pq} is a positive integer of size k-bit. Let $x_{pq} = \left(x_{pq,\, k-1},\ x_{pq,\, k-2} \cdots x_{pq,\, 1},\ x_{pq,\, 0}\right)$ be the binary representation of x_{pq}. Now, we introduce the following operations.

1. $X^{(i)} = \left(x_{pq}^{(i)}\right)_{1 \le p,\, q \le m}$ (or equivalently, $X^{(i)} = \left(x_{pq} \pmod{2^i}\right)_{1 \le p,\, q \le m}$) is the matrix obtained from X by replacing each element x_{pq} with $x_{pq}^{(i)}$.

For example, let $k = 5,\ m = 2$ and

$$X = \begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix}, \tag{1}$$

where each element of X is a 5-bit positive integer, i.e.,

$$X = \begin{pmatrix} (00011)_2 & (00101)_2 \\ (00110)_2 & (00111)_2 \end{pmatrix}.$$

Then

$$X^{(1)} = \begin{pmatrix} (1)_2 & (1)_2 \\ (0)_2 & (1)_2 \end{pmatrix},$$

while

$$X^{(2)} = \begin{pmatrix} (11)_2 & (01)_2 \\ (10)_2 & (11)_2 \end{pmatrix}.$$

2. Define $X_i = (x_{pq,\,i})_{1\le p,\,q\le m}$ be the matrix obtained from X by extracting the i^{th} bit from each element x_{pq} of the matrix X. For example, let X be as in Equation (1),

Then:

$$X_0 = \begin{pmatrix} (1)_2 & (1)_2 \\ (0)_2 & (1)_2 \end{pmatrix},$$

while

$$X_1 = \begin{pmatrix} (1)_2 & (0)_2 \\ (1)_2 & (1)_2 \end{pmatrix}.$$

3. Note that $X_0 = X^{(1)}$.

3. Known-plaintext attack

This section presents the first cryptanalysis of the AHC using a known-plaintext attack. It is organized into two subsections. The first section presents the idea of cryptanalysis. The other section provides the detailed cryptanalysis and derives its time complexity.

3.1 *Outline of the attack*

From Algorithm 1, we have

$$C_0' = (K_0P_0 + K_1) \pmod n$$

$$C_0 = C_0' \oplus K_2 = (K_0P_0 + K_1) \pmod n \oplus K_2$$

$$C_1' = (K_1P_1 + K_2) \pmod n$$

$$C_1 = C_1' \oplus C_0' = (K_1P_1 + K_2) \pmod n \oplus (K_0P_0 + K_1) \pmod n$$

$$C_2' = (K_2P_2 + K_0) \pmod n$$

$$C_2 = C_2' \oplus C_1' = (K_2P_2 + K_0) \pmod n \oplus (K_1P_1 + K_2) \pmod n$$

$$C'_3 = (K_0P_3 + K_1) \pmod{n}$$

$$C_3 = C'_3 \oplus C'_2 = (K_0P_3 + K_1) \pmod{n} \oplus (K_2P_2 + K_0) \pmod{n}$$

$$C'_4 = (K_1P_4 + K_2) \pmod{n}$$

$$C_4 = C'_4 \oplus C'_3 = (K_1P_4 + K_2) \pmod{n} \oplus (K_0P_3 + K_1) \pmod{n}$$

$$C'_5 = (K_2P_5 + K_0) \pmod{n}$$

$$C_5 = C'_5 \oplus C'_4 = (K_2P_5 + K_0) \pmod{n} \oplus (K_1P_4 + K_2) \pmod{n}$$

$$C'_6 = (K_0P_6 + K_1) \pmod{n}$$

$$C_6 = C'_6 \oplus C'_5 = (K_0P_6 + K_1) \pmod{n} \oplus (K_2P_5 + K_0) \pmod{n}$$

$$\vdots$$

$$C'_i = (K_{i \pmod 3}P_i + K_{(i+1) \pmod 3}) \pmod{n}$$

$$C_i = C'_i \oplus C'_{i-1}$$

Therefore, we have

$$\bigoplus_{i=0}^{0} C_i = (K_0P_0 + K_1) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{1} C_i = (K_1P_1 + K_2) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{2} C_i = (K_2P_2 + K_0) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{3} C_i = (K_0P_3 + K_1) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{4} C_i = (K_1P_4 + K_2) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{5} C_i = (K_2P_5 + K_0) \pmod{n} \oplus K_2,$$

$$\bigoplus_{i=0}^{6} C_i = (K_0P_6 + K_1) \pmod{n} \oplus K_2,$$

Thus, we can deduce that

$$\bigoplus_{i=0}^{l} C_i = \begin{cases} (K_0P_l + K_1) \pmod{n} \oplus K_2, & \text{if } l \equiv 0 \pmod{3} \\ (K_1P_l + K_2) \pmod{n} \oplus K_2, & \text{if } l \equiv 1 \pmod{3} \\ (K_2P_l + K_0) \pmod{n} \oplus K_2, & \text{if } l \equiv 2 \pmod{3} \end{cases} \tag{2}$$

From Equation (2), we can estimate that if we have the ciphertext and the corresponding plaintext, then we can find the key $K_0,\ K_1,\ K_2$.

The road map of the proposed attack is

1. Find K_2.
2. Find K_1 since, from Equation (2), we have

$$K_1 = \left(\left(\bigoplus_{i=0}^{l} C_i \oplus K_2 \right) - K_2 \right) \times P_l^{-1} \mod n.$$

3. Find K_0 since, from Equation (2), we have

$$K_0 = \left(\left(\bigoplus_{i=0}^{l} C_i \oplus K_2 \right) - K_1 \right) \times P_l^{-1} \mod n.$$

The main task is to find K_2.

3.2 *The attack: Known plaintext*

Suppose that we have a plaintext $P : P_0,\ P_1,\ \ldots,\ P_l$, and the corresponding Ciphertext $C : C_0,\ C_1,\ \ldots,\ C_l$ obtained by the AHC using the key $K = (K_0,\ K_1,\ K_2)$. We show that we can recover the secret key K under the following conditions:

• There exist two (nonnegative) integers l_0 and l_1 such that

$$l_0 \equiv 0 \pmod{3},\ \gcd(\det(P_{l_0}),\ n) = 1.$$

and

$$l_1 \equiv 1 \pmod{3},\ \gcd(\det(P_{l_1}),\ n) = 1.$$

• T (or K_2) satisfies

$$K_2T \pmod{n} = K_2T,$$

where

$$T = P_{l_2} - P_{l_2'} \pmod{n}$$

for two positive integers $l_2 \neq l_2'$ such that

$$l_2 \equiv l_2' \equiv 2 \pmod{3}.$$

and the determinant of T, denoted by $\det(T)$, is odd.

Note that the condition "$\det(T)$ is odd" guarantees the existence of $T^{-1} \pmod{2^i}$, $1 \leq i \leq k$ since $T \pmod{2^i} = T^{(i)}$. Thus, $(T^{(i)})^{-1} \pmod{2^i}$ exists.

Now, let X, Y, and Z refer to the unknown keys K_0, K_1, and K_2 respectively. In the following, we explain how to find $Z = K_2$, $Y = K_1$, and $X = K_0$ respectively.

3.2.1 *Finding K_2*

Let $A = \bigoplus_{i=0}^{l_2} C_i$ and $A' = \bigoplus_{i=0}^{l_2'} C_i$. Then

$$A \oplus Z - A' \oplus Z = Z(P_l - P_l') \pmod{n} = ZT.$$

Since the operation $\oplus$ is performed on $\mathbb{Z}_{2^k}$, $k = [\log n]$ we have

$$-A' \oplus Z \pmod{2^k} = 2^k - A' \oplus Z = (2^k - 1) - \left(A' \oplus Z\right) + 1 = (2^k - 1) \oplus A' \oplus Z + 1,$$

Therefore,

$$A \oplus Z - A' \oplus Z = A \oplus Z + (2^k - 1) \oplus A' \oplus Z + 1 = ZT$$

This imply that,

$$A \oplus Z - A' \oplus Z - 1 = A \oplus Z + (2^k - 1) \oplus A' \oplus Z = ZT - 1$$

Let $G = A = \bigoplus_{i=0}^{l_2} C_i$ and $G' = (2^k - 1) \oplus A' = (2^k - 1) \bigoplus_{i=0}^{l_2'} C_i$. Then,

$$(G \oplus Z) + (G' \oplus Z) \pmod{2^k} = ZT - 1 \tag{3}$$

Now, we find a new strategy to solve Equation (3).

Let $H = ZT - 1$. Theorem 1 shows how to construct H from G, and G' using Equation (3). Theorem 1 is important to explain the relationship between $H^{(i)}$, $H^{(i-1)}$, G_{i-1}, and G'_{i-1}, $1 \leq i \leq k$. This relationship is the principal idea used in Algorithm 3 to obtain Z repetitively by knowing G, G', and T.

Theorem 1 Let $G,\ G',\ Z$ be matrices over $\mathbf{Z_{2^k}}$, and $H = (G \oplus Z) + (G' \oplus Z) \pmod{2^k}$. Then,

$$H^{(i)} = \begin{cases} G_0 \oplus G'_0 & \text{if } i = 1, \\ 2^{i-1}(G_{i-1} \oplus G'_{i-1} \oplus H_{i-1}) \oplus H^{(i-1)} \pmod{2^i} & \text{if } 2 \leq i \leq k \end{cases}$$

where, H_{i-1} is the i-th bit of $(G^{(i-1)} \oplus Z^{(i-1)}) + (G'^{(i-1)} \oplus Z^{(i-1)}) \pmod{2^i}$.

To find $H^{(i)}$ in Theorem 1, we prove, in Proposition 1, how to find the value of an element $h^{(i)}$ in the matrix $H^{(i)}$. Thus, the proof of Theorem 1 is a straightforward generalization of Proposition 1.

Proposition 1 Let $g,\ g',\ z \in \mathbf{Z_{2^k}}$ and $h = (g \oplus z) + (g' \oplus z) \pmod{2^k}$. Then

$$h^{(i)} = \begin{cases} g_0 \oplus g'_0 & \text{if } i = 1, \\ 2^{i-1}(g_{i-1} \oplus g'_{i-1} \oplus h_{i-1}) \oplus h^{(i-1)} \pmod{2^i} & \text{if } 2 \leq i \leq k \end{cases}$$

where h_{i-1} is the i-th bit of $(g^{(i-1)} \oplus z^{(i-1)}) + (g'^{(i-1)} \oplus z^{(i-1)}) \pmod{2^i}$.

Proof. Since $h = (g \oplus z) + (g' \oplus z) \mod 2^k$.

For $i = 1$, we have:

$$\begin{aligned} h^{(1)} &= (g_0 \oplus z_0) + (g'_0 \oplus z_0) \mod 2 \\ &= (g_0 \oplus z_0) \oplus (g'_0 \oplus z_0) \\ &= g_0 \oplus g'_0 \end{aligned}$$

For $i = 2, \ldots, k$,

$$\begin{aligned} h^{(i)} &= (g^{(i)} \oplus z^{(i)}) + (g'^{(i)} \oplus z^{(i)}) \pmod{2^i} \\ &= (2^{i-1}g_{i-1} \oplus g^{(i-1)} \oplus 2^{i-1}z_{i-1} \oplus z^{(i-1)}) + (2^{i-1}g'_{i-1} \oplus g'^{(i-1)} \oplus 2^{i-1}z_{i-1} \oplus z^{(i-1)}) \pmod{2^i} \\ &= 2^{i-1}(g_{i-1} \oplus z_{i-1}) \oplus (g^{(i-1)} \oplus z^{(i-1)}) + 2^{i-1}(g'_{i-1} \oplus z_{i-1}) \oplus (g'^{(i-1)} \oplus z^{(i-1)}) \pmod{2^i} \\ &= 2^{i-1}(g_{i-1} \oplus z_{i-1} \oplus g'_{i-1} \oplus z_{i-1}) + (g^{(i-1)} \oplus z^{(i-1)}) + (g'^{(i-1)} \oplus z^{(i-1)}) \pmod{2^i} \end{aligned}$$

$$= 2^{i-1}(g_{i-1} \oplus g'_{i-1}) + (g^{(i-1)} \oplus z^{(i-1)}) + (g'^{(i-1)} \oplus z^{(i-1)}) \pmod{2^i}$$

If h_{i-1} is the i-th bit of $(g^{(i-1)} \oplus z^{(i-1)}) + (g'^{(i-1)} \oplus z^{(i-1)}) \pmod{2^i}$, then

$$h^{(i)} = 2^{i-1}(g_{i-1} \oplus g'_{i-1}) + 2^{i-1}h_{i-1} \oplus h^{(i-1)} \pmod{2^i}$$

$$= 2^{i-1}(g_{i-1} \oplus g'_{i-1} \oplus h_{i-1}) + h^{(i-1)} \pmod{2^i}$$

$$= 2^{i-1}(g_{i-1} \oplus g'_{i-1} \oplus h_{i-1}) \oplus h^{(i-1)} \pmod{2^i}$$ □

Algorithm 3 describes how to find Z that satisfies Equation (3) using G, G', and T. Thus, obtaining $Z = K_2$ in Equation (2) is equivalent to obtaining Z in Equation (3).

Algorithm 3 describes how to compute Z satisfying Equation (3) from the inputs G, G', and T. Hence, obtaining $Z = K_2$ in Equation (2) is equivalent to obtaining Z from Equation (3). The following theorem shows the time complexity of Algorithm 3.

Theorem 2 The time complexity of Algorithm 3 is $\mathcal{O}(k\ m^3)$.

Proof. (i) matrix addition and XOR cost $\mathcal{O}(m^2)$; (ii) scalar-shift for elements of matrix costs $\mathcal{O}(m^2)$; (iii) matrix multiplication and inversion cost $\mathcal{O}(m^3)$ using the standard algorithm for matrix multiplication. (iv) Steps 4-6 are repeated $k-1$ times. Thus, Steps 3-7 cost $\mathcal{O}(k\ m^3)$.

Therefore, the overall time complexity of the algorithm is $\mathcal{O}(k\ m^3)$. □

Algorithm 3 Solving Equation (3).
Require: T, G, G' are matrices over $\mathbf{Z_{2^k}}$, where $\det(T)$ is odd.
Ensure: Z satisfying Equation (3).
1: set $H_0 = G_0 \oplus G'_0 \pmod 2$, $Z_0 = (G_0 \oplus G'_0 + 1)T_0^{-1} \pmod 2$
2: $H^{(1)} = H_0$, $Z^{(1)} = Z_0$, $G^{(1)} = G_0$, $G'^{(1)} = G'_0$,
3: **for** $i = 2$ to k **do**
4: H_{i-1} is the i-th bit of $(G^{(i-1)} \oplus Z^{(i-1)}) + (G'^{(i-1)} \oplus Z^{(i-1)}) \pmod{2^i}$
5: $H^{(i)} = 2^{i-1}(G_{i-1} \oplus G'_{i-1} \oplus H_{i-1}) \oplus H^{(i-1)} \pmod{2^i}$
6: $Z^{(i)} = (H^{(i)} + 1)(T^{(i)})^{-1} \pmod{2^i}$.
7: **end for**
8: return $Z^{(k)}$.

3.2.2 *Finding K_1*

We now show that knowing $Z = K_2$ in Equation (2) leads to knowing $Y = K_1$ such that we have $l_1 \equiv 1 \pmod 3$ and $\gcd(\det(P_{l_1}),\ n) = 1$. This can be done easily using the second case of Equation (2) as follows.

$$Y = \left(\left(\bigoplus_{i=0}^{l_1} C_i \right) \oplus Z - Z \right) \times P_{l_1}^{-1} \pmod{n}.$$

3.2.3 *Finding K_0*

Now, after obtaining Z and Y, we can easily compute $X = K_0$ from the first case in Equation (2) as follows:

$$X = \left(\left(\bigoplus_{i=0}^{l_0} C_i \right) \oplus Z - Y \right) \times P_{l_0}^{-1} \pmod{n},$$

where $l_0 \equiv 0 \pmod 3$ and $\gcd(\det(P_{l_0}),\ n) = 1$.

3.2.4 *The algorithm*

We have shown that obtaining the values $X = K_0, Y = K_1$, and $Z = K_2$ of the secret key in AHC is the same as solving the following system of three equations with unknowns X, Y, and Z :

$$E = (XR + Y) \pmod{n} \oplus Z \pmod{2^k} \tag{4}$$

$$F = (YS + Z) \pmod{n} \oplus Z \pmod{2^k} \tag{5}$$

$$ZT - 1 = (G \oplus Z) + (G' \oplus Z) \pmod{2^k} \tag{6}$$

In Proposition 1, we study the case in which Equation (6) can be solved and the steps to obtain Z are presented in Algorithm 3. We present steps to solve the system of Eqs. (4)-(6) in Algorithm 4. The following theorem shows the time complexity of Algorithm 4.

Theorem 3 The time complexity of Algorithm 4 is $\mathcal{O}((\log n)\ m^3)$.

Proof. (i) Step 1 executes Algorithm 3 that takes $\mathcal{O}(km^3)$, where $k = \lceil \log n \rceil$; (ii) Steps 2-3 take $\mathcal{O}(m^3)$ since they require matrix multiplication and inversion with costs $\mathcal{O}(m^3)$.

Therefore, the overall time complexity of the algorithm is $\mathcal{O}((\log n)\ m^3)$. □

Algorithm 4 Solving Equations (4)-(6).

Require: T, G, G', E, F, R, S, n where $\gcd(\det(T),\ 2^i) = 1$ and $\gcd(\det(R),\ n) = \gcd(\det(S),\ n) = 1$.

Ensure: $X,\ Y,\ Z$ matrices $m \times m$ over $\mathbf{Z_{2^k}}$ satisfying Eqs. (4)-(6).

1: using Algorithm 3 to obtain Z from $ZT - 1 = (G \oplus Z) + (G' \oplus Z) \pmod{2^k}$, $k = \lceil \log n \rceil$.

2: set $Y = ((F \oplus Z \pmod{2^k}) - Z)S^{-1} \pmod n$

3: $X = ((E \oplus Z \pmod{2^k}) - Y)R^{-1} \pmod n$

4: return $X,\ Y$ and Z.

Now, we give a general algorithm to conclude and implement our known-plaintext attack on the AHC. Theorem 3 shows the polynomial time complexity of Algorithm 5.

Theorem 4 The time complexity of Algorithm 5 is $\mathcal{O}((\log n + l^2)m^3)$.

Proof. 1. Step 1 involves identifying indices l_2 and l_2' such that $l_2 \equiv l_2' \equiv 2 \pmod 3$, and the determinant of $T = P_{l_2} - P_{l_2'}$ is odd. Identifying l_2 and l_2' requires at most $l(l-1)/2$ iterations. More precisely, at most $\frac{l}{3}\left(\frac{l}{3} - 1\right)/2$ iterations. While the determinant of T costs $\mathcal{O}(m^3)$. Thus, Step 1 costs $\mathcal{O}(l^2 m^3)$.

2. Step 3 costs $\mathcal{O}(lm^2)$ since $l_2,\ l_2' \le l$.

3. Similarly, Steps 5 and 7 cost $\mathcal{O}(lm^3)$.

4. Step 4 costs $\mathcal{O}(km^3)$ using Algorithm 3, where $k = \lceil \log n \rceil$.

5. Steps 6 and 8 cost $\mathcal{O}(km^3)$ using Agorithm 4.

Hence, the overall time complexity of Algorithm 5 is $\mathcal{O}((k + l^2)m^3)$ or $\mathcal{O}((\log n + l^2)m^3)$. □

Remark 1 Using [26], matrix multiplication can be done in $\mathcal{O}(m^{\omega})$, with $\omega < 2.37286$. Thus, the time complexity of the Algorithm 5 reduces to $\mathcal{O}((\log n + l^2)m^{\omega})$.

Algorithm 5 Known plaintext attack.

Require: $n,\ m$, and $l-$pairs of ($m \times m$ matrix over Z_n) plaintext/ciphertext $(P_0,\ C_0),\ (P_1,\ C_1),\ \ldots,\ (P_l,\ C_l)$.

Ensure: $X = K_0,\ Y = K_1,\ Z = K_2$ the secret key of the AHC.

1: find l_2 and l_2' where $l_2 \equiv l_2' \equiv 2 \pmod 3$ and $\det(T = P_{l_2} - P_{l_2'})$ is odd

2: set $k = \lceil \log n \rceil$

3: set $G = \bigoplus_{i=0}^{l_2} C_i$ and $G' = (2^k - 1)\bigoplus_{i=0}^{l_2'} C_i$,

4: use Algorithm 3 to obtain Z satisfying $ZT - 1 = (G \oplus Z) + (G' \oplus Z) \pmod{2^k}$

5: find $l_1 \equiv 1 \pmod 3$ such that $\gcd(\det(P_{l_1}),\ n) = 1$ and compute $F = \bigoplus_{i=0}^{l_1} C_i$

6: set $Y = ((F \oplus Z \pmod{2^k}) - Z)P_{l_1}^{-1} \pmod n$ ▷ Step 2 in Algorithm 4

7: find $l_0 \equiv 0 \pmod 3$ such that $\gcd(\det(P_{l_0}),\ n) = 1$ and compute $E = \bigoplus_{i=0}^{l_0} C_i$

8: $X = ((E \oplus Z \pmod{2^k}) - Y)P_{l_0}^{-1} \pmod n$ ▷ step 3 in Algorithm 4

9: return $X = K_0$, $Y = K_1$ and $Z = K_2$.

4. Chosen-plaintext attack

Suppose that an attacker chooses a plaintext $P : P_0,\ P_1,\ \ldots,\ P_l$ of his/her choice, and observes the corresponding ciphertext $C : C_0,\ C_1,\ \ldots,\ C_l$ produced by the AHC using the key $K = (K_0,\ K_1,\ K_2)$.

We show that the secret key K can be recovered in the following three steps:

Step 1: An attacker chooses, for example, $l_0 = 0$, and $l_1 = 1$. Then he/she chooses plaintexts P_{l_0} and P_{l_1} as $m \times m$ matrix $[p_{i,\ j}]_{m\times m}$, where

$$p_{i,\ j} = \begin{cases} \text{odd number and has no prime factor with } n, & \text{if } i = j \\ 0, & \text{otherwise} \end{cases}$$

i.e, they have the structure:

$$\begin{bmatrix} p_{1,\ 1} & 0 & 0 & \cdots & 0 \\ 0 & p_{2,\ 2} & 0 & \cdots & 0 \\ 0 & 0 & p_{3,\ 3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & p_{m,\ m} \end{bmatrix}.$$

where $p_{i,\ i}$ is odd number and has no prime factor with n.

Clear that, $\gcd(\det(P_{l_0}),\ n) = 1$, and $\gcd(\det(P_{l_1}),\ n) = 1$.

Step 2: An attacker chooses $l_2 \neq l_2'$ such that $l_2 \equiv l_2' \equiv 2 \pmod 3$, for example, set $l_2 = 2$, and $l_2' = 5$. Then he/she chooses plaintexts P_{l_2} and P_{l_2} as $m \times m$ matrix óf the form:

$$P_{l_2} = [p_{i,\ j}]_{m\times m} \quad \text{where} \quad p_{i,\ j} = \begin{cases} \alpha_i + 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \quad \text{with } \alpha_i \in \mathbb{N},$$

and

$$P_{l_2'} = [p_{i,\ j}]_{m\times m} \quad \text{where} \quad p_{i,\ j} = \begin{cases} \alpha_i, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \quad \text{with } \alpha_i \in \mathbb{N},$$

i.e., P_{l_2} has the structure:

$$P_{l_2} = \begin{bmatrix} \alpha_1+1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2+1 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3+1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_m+1 \end{bmatrix},$$

and $P_{l_2'}$ has the structure:

$$P_{l_2'} = \begin{bmatrix} \alpha_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_m \end{bmatrix}.$$

Thus, we have

$$K_2T \pmod n = K_2T, \qquad \text{where } T = P_{l_2} - P_{l_2'} \pmod n \quad \text{and } \det(T) \text{ is odd.}$$

Step 3: Now, the attacker can apply the known-plaintext attack presented in section 3 to recover the key. The following remark shows the possibility to choose different forms of plaintext in Step 2.

Remark 2 In Step 2, an attacker can easily choose different forms of P_{l_2}, and $P_{l_2'}$, such as

$$P_{l_2} = \begin{bmatrix} 0 & \alpha_1+1 & 0 & \cdots & 0 \\ \alpha_2+1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3+1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_m+1 \end{bmatrix},$$

and

$$P_{l_2'} = \begin{bmatrix} 0 & \alpha_1 & 0 & \cdots & 0 \\ \alpha_2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \alpha_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \alpha_m \end{bmatrix},$$

where $\alpha_i \in \mathbb{N}$.

Now, we give an example of choosing P_{l_0} and P_{l_1}, in Step 1, when n is a power of 2.

The attacker chooses plaintexts P_{l_0} and P_{l_1} of the form:

$$P = [p_{i,\,j}]_{m\times m} \quad \text{where} \quad p_{i,\,j} = \begin{cases} 2o_i + 1, & \text{if } i = j \\ 0, & \text{if } i \neq j \end{cases} \quad \text{for some } o_i \in \mathbb{N},$$

i.e, P_{l_0} and P_{l_1} has the structure:

$$\begin{bmatrix} 2o_1+1 & 0 & 0 & \cdots & 0 \\ 0 & 2o_2+1 & 0 & \cdots & 0 \\ 0 & 0 & 2o_3+1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 2o_m+1 \end{bmatrix}.$$

Clear that, $\gcd(\det(P_{l_0}),\, n) = 1$, and $\gcd(\det(P_{l_1}),\, n) = 1$.

As a result, in a chosen-plaintext attack, an attacker requests the ciphertext corresponding to adaptively chosen plaintexts. Constructing such adaptive plaintexts to meet the criteria of a known-plaintext attack is relatively straightforward. Thus, the AHC is also considered insecure under Chosen-plaintext attack. Clearly, the time complexity of the attack takes $\mathcal{O}((\log n + l^2)m^3)$.

5. Numerical examples

In this section, we present numerical examples for known-plaintext and chosen-plaintext attacks (sections 3 and 4). These examples were generated by implementing Algorithm 5 in MATLAB [27], which provides well-optimized functions for matrix operations. In addition to basic matrix operations (addition, subtraction, multiplication, inverse, determinant), MATLAB provides a lot of useful functions for implementation. Examples of such functions are: (1) $bitget(X,\, i)$ returns the bit value at position i in every element of X; (2) $bitxor(X,\, Y)$ returns the bit-wise XOR of X and Y; (3) $bitshift(X,\, i)$ and $bitand(X,\, Y)$ return X shifted to the left by i bits, and return the bit-wise AND of X and Y, respectively. Together, both functions are used to extract the first i bits to the right of each element in a matrix X; (4) $mod(X,\, n)$ returns the remainder after division of (each element of) X by n.

5.1 *Example demonstrating a known-plaintext attack*

Let $n = 32$ and we have a list of plaintexts $P_0,\, P_1,\, \ldots,\, P_8$ with their corresponding ciphertexts $C_0,\, C_1,\, \ldots,\, C_8$ as shown in Table 1.

Table 1. Pairs of plaintexts and corresponding ciphertexts

P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8
$\begin{bmatrix}4 & 19\\ 1 & 6\end{bmatrix}$	$\begin{bmatrix}8 & 5\\ 5 & 4\end{bmatrix}$	$\begin{bmatrix}3 & 5\\ 2 & 7\end{bmatrix}$	$\begin{bmatrix}5 & 2\\ 5 & 3\end{bmatrix}$	$\begin{bmatrix}4 & 3\\ 7 & 7\end{bmatrix}$	$\begin{bmatrix}8 & 7\\ 3 & 4\end{bmatrix}$	$\begin{bmatrix}11 & 2\\ 8 & 3\end{bmatrix}$	$\begin{bmatrix}4 & 17\\ 1 & 10\end{bmatrix}$	$\begin{bmatrix}9 & 8\\ 5 & 5\end{bmatrix}$
C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
$\begin{bmatrix}26 & 21\\ 7 & 22\end{bmatrix}$	$\begin{bmatrix}29 & 13\\ 7 & 7\end{bmatrix}$	$\begin{bmatrix}11 & 12\\ 13 & 3\end{bmatrix}$	$\begin{bmatrix}12 & 8\\ 29 & 4\end{bmatrix}$	$\begin{bmatrix}29 & 3\\ 28 & 21\end{bmatrix}$	$\begin{bmatrix}12 & 5\\ 3 & 0\end{bmatrix}$	$\begin{bmatrix}24 & 6\\ 2 & 21\end{bmatrix}$	$\begin{bmatrix}18 & 10\\ 17 & 19\end{bmatrix}$	$\begin{bmatrix}17 & 31\\ 24 & 20\end{bmatrix}$

In the following, we apply Algorithm 5 to the plaintexts P_0, P_1, …, P_8 and the corresponding ciphertexts C_0, C_1, …, C_8 to find the secret keys. Note that the known-plaintext attack succeeds only if (P_i, C_i) satisfies the condition of the attack as stated in Section 3.2.

1. Set $l_2 = 8$, $l_2' = 5$ and $T = P_8 - P_5 = \begin{bmatrix}1 & 1\\ 2 & 1\end{bmatrix} = \begin{bmatrix}(00001)_2 & (00001)_2\\ (00010)_2 & (00001)_2\end{bmatrix}$.

2. $k = 5$.

3. $G = \bigoplus_{i=0}^{8} C_i = \begin{bmatrix}10 & 9\\ 4 & 17\end{bmatrix} = \begin{bmatrix}(01010)_2 & (01001)_2\\ (00100)_2 & (10001)_2\end{bmatrix}$, $G' = (2^k - 1)\bigoplus_{i=0}^{5} C_i = \begin{bmatrix}14 & 5\\ 16 & 28\end{bmatrix} = \begin{bmatrix}(01110)_2 & (00101)_2\\ (10000)_2 & (11100)_2\end{bmatrix}$.

4. The steps to obtain $Z = K_2$, according to Algorithm 3, are

4.1 $H_0 = G_0 \bigoplus G_0' = \begin{bmatrix}(0)_2 & (0)_2\\ (0)_2 & (1)_2\end{bmatrix}$, $T_0^{-1} \pmod 2 = \begin{bmatrix}(1)_2 & (1)_2\\ (0)_2 & (1)_2\end{bmatrix}$, and $Z_0 = \left(G_0 \oplus G_0' + \begin{bmatrix}(1)_2 & (1)_2\\ (1)_2 & (1)_2\end{bmatrix}\right) T_0^{-1} \pmod 2 = \begin{bmatrix}(1)_2 & (0)_2\\ (1)_2 & (1)_2\end{bmatrix}$.

4.2 (At $i = 1$) $H^{(1)} = H_0$, $Z^{(1)} = Z_0$, $G^{(1)} = G_0 = \begin{bmatrix}(0)_2 & (1)_2\\ (0)_2 & (1)_2\end{bmatrix}$, $G'^{(1)} = G_0' = \begin{bmatrix}(0)_2 & (1)_2\\ (0)_2 & (0)_2\end{bmatrix}$.

4.3 $i = 2$.

4.4 H_1 is the second bit of $(G^{(1)} \oplus Z^{(1)}) + (G'^{(1)} \oplus Z^{(1)}) \pmod{2^2} = \begin{bmatrix}(1)_2 & (1)_2\\ (1)_2 & (0)_2\end{bmatrix}$.

4.5 $H^{(2)} = 2^1(G_1 \oplus G_1' \oplus H_1) \oplus H^{(1)} \pmod{2^2} = \begin{bmatrix}(10)_2 & (10)_2\\ (10)_2 & (01)_2\end{bmatrix}$.

4.6 $Z^{(2)} = (H^{(2)} + 1)(T^{(2)})^{-1} \pmod{2^2} = \begin{bmatrix}(11)_2 & (00)_2\\ (01)_2 & (01)_2\end{bmatrix}$.

4.7 Repeat the steps from item 4.3 to 4.6 until $i = 5$, then we obtain $Z^{(5)} = \begin{bmatrix}(00111)_2 & (01000)_2\\ (00101)_2 & (01001)_2\end{bmatrix}$. Therefore, $Z = K_2 = \begin{bmatrix}7 & 8\\ 5 & 9\end{bmatrix}$.

5. Set $l_1 = 1$ and $F = \bigoplus_{i=0}^{l_1} C_i = \begin{bmatrix}7 & 24\\ 0 & 17\end{bmatrix}$.

6. We have $P_{l_1} = \begin{bmatrix}8 & 5\\ 5 & 4\end{bmatrix}$ and $P_{l_1}^{-1} \pmod{32} = \begin{bmatrix}28 & 13\\ 13 & 24\end{bmatrix}$. Therefore, we get $Y = K_1 = \begin{bmatrix}4 & 5\\ 3 & 8\end{bmatrix}$.

7. Set $l_0 = 0$ and $E = \bigoplus_{i=0}^{l_0} C_i = \begin{bmatrix}26 & 21\\ 7 & 22\end{bmatrix}$.

8. We have $P_0 = \begin{bmatrix}4 & 19\\ 1 & 6\end{bmatrix}$ and $P_0^{-1} \pmod{32} = \begin{bmatrix}14 & 9\\ 19 & 20\end{bmatrix}$. Therefore, we get $X = K_0 = \begin{bmatrix}6 & 1\\ 7 & 3\end{bmatrix}$.

9. We get $X = K_0$, $Y = K_1$, and $Z = K_2$.

5.2 *Example demonstrating a chosen-plaintext attack*

Assume that an adversary has an ability to encrypt arbitrary plaintexts of his/her choosing. Let the block size be $n = 32$. The attacker selects a set of plaintexts: $P_0, P_1, \ldots, P_5$ which are submitted to the encryption algorithm, which returns the corresponding ciphertexts: $C_0, C_1, \ldots, C_5$ as follows:

An attacker chooses P as shown in Table 2.

Table 2. Chosen plaintexts

P_0	P_1	P_2	P_3	P_4	P_5
$\begin{bmatrix} 3 & 0 \\ 0 & 5 \end{bmatrix}$	$\begin{bmatrix} 7 & 0 \\ 0 & 3 \end{bmatrix}$	$\begin{bmatrix} 5 & 0 \\ 0 & 7 \end{bmatrix}$	$\begin{bmatrix} 9 & 0 \\ 0 & 3 \end{bmatrix}$	$\begin{bmatrix} 3 & 0 \\ 0 & 11 \end{bmatrix}$	$\begin{bmatrix} 4 & 0 \\ 0 & 6 \end{bmatrix}$

Note that the required conditions to satisfy the attack hold since

• Set $l_0 = 0,\ l_1 = 1$, where $\gcd(\det(P_0),\ n) = \gcd(\det(P_1),\ n) = 1$.

• $l_2 = 2, l_2' = 5$ and $T = P_2 - P_5 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (00001)_2 & (00000)_2 \\ (00000)_2 & (00001)_2 \end{bmatrix}$, $\det(T)$ is odd, and $K_2 T \pmod n = K_2 T$ for any subkey K_2.

Then, the attacker obtains the corresponding ciphertext as shown in Table 3.

Table 3. The ciphertexts corresponding to the plaintexts in Table 2

C_0	C_1	C_2	C_3	C_4	C_5
$\begin{bmatrix} 17 & 2 \\ 29 & 30 \end{bmatrix}$	$\begin{bmatrix} 21 & 29 \\ 2 & 22 \end{bmatrix}$	$\begin{bmatrix} 10 & 14 \\ 26 & 3 \end{bmatrix}$	$\begin{bmatrix} 19 & 17 \\ 2 & 19 \end{bmatrix}$	$\begin{bmatrix} 9 & 23 \\ 12 & 16 \end{bmatrix}$	$\begin{bmatrix} 17 & 14 \\ 21 & 24 \end{bmatrix}$

Now, the attacker can we apply Algorithm 5 to the plaintexts $P_0, P_1, \ldots, P_5$ and the corresponding ciphertexts $C_0, C_1, \ldots, C_5$ to find the secret keys.

1. l_2, and l_2' are already taken by 2, and 5 respectively.

2. $k = 5$.

3. $G = \bigoplus_{i=0}^{2} C_i = \begin{bmatrix} 14 & 17 \\ 5 & 11 \end{bmatrix} = \begin{bmatrix} (01110)_2 & (10001)_2 \\ (00101)_2 & (01011)_2 \end{bmatrix}$, $G' = (2^k - 1) \bigoplus_{i=0}^{5} C_i = \begin{bmatrix} 26 & 6 \\ 1 & 15 \end{bmatrix} = \begin{bmatrix} (11010)_2 & (00110)_2 \\ (00001)_2 & (01111)_2 \end{bmatrix}$.

4. The steps to obtain $Z = K_2$, according to Algorithm 3, are

4.1 $H_0 = G_0 \bigoplus G_0' = \begin{bmatrix} (0)_2 & (0)_2 \\ (1)_2 & (1)_2 \end{bmatrix}$, $T_0^{-1} \pmod 2 = \begin{bmatrix} (1)_2 & (0)_2 \\ (0)_2 & (1)_2 \end{bmatrix}$, and $Z_0 = \left(G_0 \oplus G_0' + \begin{bmatrix} (1)_2 & (1)_2 \\ (1)_2 & (1)_2 \end{bmatrix} \right) T_0^{-1} \pmod 2 = \begin{bmatrix} (1)_2 & (0)_2 \\ (1)_2 & (1)_2 \end{bmatrix}$.

4.2 (At $i = 1$) $H^{(1)} = H_0, Z^{(1)} = Z_0, G^{(1)} = G_0 = \begin{bmatrix} (0)_2 & (1)_2 \\ (0)_2 & (1)_2 \end{bmatrix}$, $G'^{(1)} = G_0' = \begin{bmatrix} (0)_2 & (1)_2 \\ (1)_2 & (1)_2 \end{bmatrix}$.

4.3 $i = 2$.

4.4 H_1 is the second bit of $(G^{(1)} \oplus Z^{(1)}) + (G'^{(1)} \oplus Z^{(1)}) \pmod{2^2} = \begin{bmatrix} (1)_2 & (0)_2 \\ (0)_2 & (0)_2 \end{bmatrix}$.

4.5 $H^{(2)} = 2^1 (G_1 \oplus G_1' \oplus H_1) \oplus H^{(1)} \pmod{2^2} = \begin{bmatrix} (10)_2 & (11)_2 \\ (00)_2 & (00)_2 \end{bmatrix}$.

4.6 $Z^{(2)} = (H^{(2)} + 1)(T^{(2)})^{-1} \pmod{2^2} = \begin{bmatrix} (11)_2 & (00)_2 \\ (01)_2 & (01)_2 \end{bmatrix}$.

4.7 Repeat the steps from item 4.3 to 4.6 until $i = 5$, then we obtain $Z^{(5)} = \begin{bmatrix} (00111)_2 & (01000)_2 \\ (00101)_2 & (01001)_2 \end{bmatrix}$. Therefore, $Z = K_2 = \begin{bmatrix} 7 & 8 \\ 5 & 9 \end{bmatrix}$.

5. l_1 is already taken by 1. Thus, $F = \bigoplus_{i=0}^{l_1} C_i = \begin{bmatrix} 4 & 31 \\ 31 & 8 \end{bmatrix}$.

6. We have $P_{l_1} = \begin{bmatrix} 7 & 0 \\ 0 & 3 \end{bmatrix}$, and $P_{l_1}^{-1} \pmod{32} = \begin{bmatrix} 23 & 0 \\ 0 & 11 \end{bmatrix}$. Therefore, we get $Y = K_1 = \begin{bmatrix} 4 & 5 \\ 3 & 8 \end{bmatrix}$.

7. l_0 is already taken by $l_0 = 0$. Thus, $E = \bigoplus_{i=0}^{l_0} C_i = \begin{bmatrix} 17 & 2 \\ 29 & 30 \end{bmatrix}$.

8. We have $P_0 = \begin{bmatrix} 3 & 0 \\ 0 & 5 \end{bmatrix}$, and $P_0^{-1} \pmod{32} = \begin{bmatrix} 11 & 0 \\ 0 & 13 \end{bmatrix}$. Therefore, we get $X = K_0 = \begin{bmatrix} 6 & 1 \\ 7 & 3 \end{bmatrix}$.

9. Return the secret key $K = (K_0, K_1, K_2)$.

6. Conclusion and future work

In this work, we have shown that the AHC algorithm is weak against both known-plaintext and chosen-plaintext attacks. Our analysis introduces a new method based on solving equations from the internal structure of the cipher. This method allows an attacker to break the cipher by using either known or carefully chosen plaintexts. These results confirm that AHC is not secure under standard cryptographic attack models. The attacks can be performed in polynomial time. Hence, we cannot use it in practice since the proposed chosen-plaintext attack is constructive.

In the future, we are interested in applying our strategy or a similar one to analyze other ciphers, such as [28, 29].

Acknowledgment

The authors are grateful to the referees for their valuable comments, which helped to improve the article.

Conflict of interest

The authors declare no conflict of interest.

References

[1] Rubinstein-Salzedo S. *Cryptography*. Vol. 260. Cham, Switzerland: Springer International Publishing; 2018.

[2] Daemen J, Rijmen V. *The Design of Rijndael: The Advanced Encryption Standard (AES)*. 2nd ed. Berlin: Springer-Verlag; 2020.

[3] Asseisah MS, Bahig H, Daoud SS. Interactive visualization system for DES. In: An A, Lingras P, Petty S, Huang R. (eds.) *Active Media Technology. AMT 2010*. Berlin, Heidelberg: Springer; 2010. p.18-25.

[4] Asseisah M, Bahig H. Visual exploration of classical encryption on the web. In: *The Ninth IASTED International Conference on Web-based Education*. Anaheim, CA: ACTA Press; 2010. p.15-17.

[5] Hill LS. Cryptography in an algebraic alphabet. *The American Mathematical Monthly*. 1929; 36(6): 306-312.

[6] Cortez DMA, Barrieta RG, Canlas AS, Mata KE, Blanco MCR, Regala RC. Cryptanalysis of the modified Hill Cipher algorithm using Myszkowski transposition. In: *Proceedings of the 2nd International Conference in Information and Computing Research (iCORE)*. Cebu, Philippines: IEEE; 2022. p.127-132.

[7] Chen Y, Xie R, Zhang H, Li D, Lin W. Generation of high-order random key matrix for Hill Cipher encryption using the modular multiplicative inverse of triangular matrices. *Wireless Networks*. 2023; 30: 5697-5707.

[8] Ismail IA, Amin M, Diab H. How to repair the Hill Cipher. *Journal of Zhejiang University-Science A*. 2006; 7(12): 2022-2030.

[9] Keliher L, Delaney AZ. Cryptanalysis of the Toorani-Falahati Hill Ciphers. In: *2013 IEEE Symposium on Computers and Communications (ISCC)*. Split, Croatia: IEEE; 2013. p.436-440.

[10] Keliher L, Thibodeau S. Slide attacks against iterated Hill Ciphers. In: *Security in Computing and Communications: International Symposium*. Berlin, Heidelberg: Springer; 2013. p.179-190.

[11] Khazaei S, Ahmadi S. Ciphertext-only attack on $d \times d$ Hill in $O(d13^d)$. *Information Processing Letters*. 2017; 118: 25-29.

[12] Kumar AA, Kiran S, Reddy DS. Modified Hill Cipher with invertible key matrix using Radix 64 conversion. In: *Futuristic Communication and Network Technologies*. Singapore: Springer; 2023. p.175-184.

[13] Li C, Zhang D, Chen G. Cryptanalysis of an image encryption scheme based on the Hill Cipher. *Journal of Zhejiang University-Science A*. 2008; 9: 1118-1123.

[14] Lone PN, Singh D, Stoffová V, Mishra DC, Mir UH, Kumar N. Cryptanalysis and improved image encryption scheme using elliptic curve and affine Hill Cipher. *Mathematics*. 2022; 10(20): 3878.

[15] Paragas JR, Sison AM, Medina RP. Hill Cipher modification: A simplified approach. In: *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*. Chongqing, China: IEEE; 2019. p.821-825.

[16] Qobbi Y, Jarjar A, Essaid M, Benazzi A. New image encryption scheme based on dynamic substitution and Hill Cipher. In: *WITS 2020: Proceedings of the 6th International Conference on Wireless Technologies, Embedded, and Intelligent Systems*. Singapore: Springer; 2022. p.797-808.

[17] Sazaki Y, Putra R. Implementation of affine transform method and advanced Hill Cipher for securing digital images. In: *IEEE 10th International Conference on Telecommunication Systems Services and Applications (TSSA)*. Denpasar, Bali, Indonesia: IEEE; 2016. p.1-5.

[18] Sharma A, Singh A, Kumar A. Encryption and decryption of marker-based 3-dimensional augmented reality image using modified Hill Cipher technique for secure transfer. In: *IEEE 2nd International Conference on Computer Communication and Artificial Intelligence (CCAI)*. Indonesia: IEEE; 2022. p.155-159.

[19] Rangel-Romero Y, Vega-García R, Menchaca-Méndez A, Acoltzi-Cervantes D, Martínez-Ramos L, Mecate-Zambrano M, et al. Comments on "How to repair the Hill Cipher". *Journal of Zhejiang University Science A*. 2008; 9(2): 211-214. Available from: https://doi.org/10.1631/jzus.A072143.

[20] Toorani M, Falahati A. A secure variant of the Hill Cipher. In: *Proceedings of the 14th IEEE Symposium on Computers and Communications (ISCC 2009)*. Sousse, Tunisia: IEEE; 2009. p.313-316.

[21] Toorani M, Falahati A. A secure cryptosystem based on affine transformation. *Security and Communication Networks*. 2011; 4(2): 207-215.

[22] Sastry VU, Janaki V. A Modified Hill Cipher with multiple keys. *International Journal of Computational Science*. 2008; 2(6): 815-826.

[23] Dawahdeh ZE, Yaakob SN, Othman RR. A new image encryption technique combining Elliptic Curve Cryptosystem with Hill Cipher. *Journal of King Saud University-Computer and Information Sciences*. 2018; 30(3): 349-355.

[24] ElHabshy AA. Augmented Hill Cipher. *International Journal of Network Security*. 2019; 21(5): 812-818.

[25] Nishida H. Secure, fast, and loss-tolerant communication with Hill Cipher and network coding. In: *IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*. USA: IEEE; 2023. p.120-126.

[26] Alman J, Vassilevska Williams V. A refined laser method and faster matrix multiplication. *TheoretiCS*. 2024; 3(21): 1-32.

[27] MATLAB. *MATLAB ver 940*. Available from: https://www.mathworks.com/products/matlab.html [Accessed 23th April 2025].

[28] Prasad K, Mahato H. Cryptography using generalized Fibonacci matrices with Affine-Hill Cipher. *Journal of Discrete Mathematical Sciences and Cryptography*. 2021; 25(8): 2341-2352.

[29] Sundarayya P, Vara Prasad G. A public key cryptosystem using Affine Hill Cipher under modulation of prime number. *Journal of Information and Optimization Sciences*. 2019; 40(4): 919-930.