Secure, Fast, and Loss-Tolerant Communication with Hill Cipher and Network Coding

Hiroshi Nishida
ASUSA Corporation, Salem Oregon, USA

***Abstract*—This paper presents a novel approach that combines a modified Hill Cipher (HC) with Random Network Coding (RNC) to achieve secure and loss-tolerant one-to-one communication. Both algorithms operate by encrypting/encoding data through a system of linear equations and decrypting/decoding it by solving these equations. However, they differ in the following features: modified HC ensures high security and RNC prevents data loss. Our approach unifies and harnesses these techniques, leveraging a high-speed Galois Field arithmetic library to expedite encryption and decryption processes. Our results show that our approach outperforms hardware-accelerated AES-256 encryption and decryption in terms of speed. Implementation of our algorithms within the QUIC protocol demonstrates significant throughput improvements in both lossy and non-lossy communication scenarios, promising higher reliability and increased speed for critical wireless communication.**

***Index Terms*—Cryptography, Network Coding, Packet Loss, Galois Field, QUIC, Security**

I. Introduction

With the spread of devices with communication functions in recent years, secure and smooth wireless communication has become essential. Furthermore, since those devices often have limited resources, it is crucial to develop efficient algorithms in many areas. In this paper, we propose a cipher algorithm that achieves smooth communication by minimizing the need for retransmission of lost packets. Implemented with our newly developed Galois Field (GF) arithmetic technique, the proposed algorithm performs faster encryption and decryption than AES-NI accelerated AES-256.

The Hill Cipher (HC) [1], introduced in 1929, is a classic symmetric cipher based on the following linear algebra:

$$\begin{aligned} c &= Kp \quad \text{(Encryption/Encoding)} \\ p &= K^{-1}c \quad \text{(Decryption/Decoding)}, \end{aligned} \tag{1}$$

where

p	Plaintext
c	Ciphertext
K	Shared key matrix
R	Rank of K

While this cipher is intuitive and straightforward, it is known to be vulnerable to the known plaintext attack. As a result, various modifications to HC have been proposed to address this vulnerability [2]–[7]. While some of these modifications remain vulnerable, others offer a sufficient level of security.

Random Network Coding (RNC) [8]–[13] operates with the same encoding and decoding principles as described in (1). However, it introduces redundancy to minimize the need for data retransmission in scenarios with lossy communication. For example, assuming $R = 3$, K is a 3×3 matrix and $|p| = |c| = 3$, the encoding part of (1) can be expressed as:

$$\begin{cases} c_0 = k_{0,0}p_0 + k_{0,1}p_1 + k_{0,2}p_2 \\ c_1 = k_{1,0}p_0 + k_{1,1}p_1 + k_{1,2}p_2 \\ c_2 = k_{2,0}p_0 + k_{2,1}p_1 + k_{2,2}p_2. \end{cases} \tag{2}$$

RNC, for example, augments the matrix K by adding a redundant row, resulting in the creation of four linear equations as outlined below:

$$\begin{cases} c_0 = k_{0,0}p_0 + k_{0,1}p_1 + k_{0,2}p_2 \\ c_1 = k_{1,0}p_0 + k_{1,1}p_1 + k_{1,2}p_2 \\ c_2 = k_{2,0}p_0 + k_{2,1}p_1 + k_{2,2}p_2 \\ c_3 = k_{3,0}p_0 + k_{3,1}p_1 + k_{3,2}p_3 \end{cases} \tag{3}$$

The values, $c_0 \cdots c_3$, are then sent to the receiver. Even if one of the values $c_0 \cdots c_3$ is lost, the receiver can still reconstruct $p_0 \cdots p_2$, as long as the rank of the linear equations remains at 3. For example, suppose c_1 was lost, and we have:

$$\bar{K} = \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} \\ k_{2,0} & k_{2,1} & k_{2,2} \\ k_{3,0} & k_{3,1} & k_{3,2} \end{pmatrix}. \tag{4}$$

If $\bar{K}$ is invertible, then the receiver can recover p by

$$\begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix} = \bar{K}^{-1} \begin{pmatrix} c_0 \\ c_2 \\ c_3 \end{pmatrix}. \tag{5}$$

Thus, RNC reduces packet retransmissions by transmitting redundant data. Data loss significantly impacts throughput due to frequent retransmission requests, but RNC greatly improves it.

One drawback of RNC is the cost associated with encoding and decoding. Integrating RNC into non-cryptographic communication results in additional CPU usage and communication delays. However, if the data is supposed to be encrypted, and RNC also serves as the encryption method, it can overcome this drawback. This inspired us to develop our approach.

The calculation of encryption and decryption, such as (1), (2), and (3), is generally performed in GF, and therefore, the processing speed in GF significantly impacts practical performance. We developed a new technique for region data multiplication in $GF(2^{16})$ and $GF(2^8)$ [14], which we implemented in our cipher algorithms. As a result, our algorithms achieve faster encryption and decryption speeds compared to

OpenSSL's AES-256, even when accelerated by AES-NI, as demonstrated in Section V.

Further details on the modified HC and RNC can be found in Sections II and III. Section IV provides an illustration of our algorithms, while benchmark results for throughput and processing speed are presented in Section V.

II. Related Work

In response to vulnerabilities, numerous modifications of the HC have been proposed. The Augmented Hill Cipher (AHC) [2] stands out by employing three key matrices and iteratively XORing newly encrypted data with previously generated encrypted data. AHC also enhances security by integrating both addition and XOR operations in encryption and decryption processes. It is proven to be resistant to various common attacks, including known-plaintext, chosen-plaintext, ciphertext-only, dictionary, brute-force, and fault-analysis attacks. For further details, please refer to Section III. To the best of our knowledge, no vulnerabilities have been identified in this method, and we adopt its algorithm for the cipher component as a representative modified HC algorithm (see Section IV-B).

Another approach, presented in [7], employs an invertible matrix as a public key and a circulant matrix as a secret key for encryption and decryption. However, it has been reported that multiple vulnerabilities exist in this method, as indicated in [2]. Some methods combine the affine cipher with HC, such as in [15] and [3], where data undergo monoalphabetic substitution using a simple mathematical function. [16] proposes an affine HC algorithm with matrices that have a Fibonacci sequences feature. Additionally, [17] introduces a non-square key matrix and adds redundancy to the encrypted data, but unlike RNC, it cannot be used for communication redundancy.

To ensure security of IoT devices in which resources are limited, many researches in recent years have focused on developing lightweight cipher algorithms. Both [18] and [19] propose algorithms that improve the efficiency of encryption and decryption based on ARX (Addition or AND, Rotation, XOR) with a generalized Feistel structure. An algorithm introduced in [20] uses a technique named Matrix Encryption Walks that makes use of coding structures referred to as two-dimensional lattices. A decentralized approach for secure communication between IoT devices is explained in [21]. All of these algorithms perform fast encryption and decryption in resource constrained environment, while maintaining a sufficient level of security.

The efficiency of RNC in handling packet loss is discussed in [8] and particularly in [9], which implements RNC into TCP and showcases remarkable throughput improvements in lossy networks. Other works like [22], [23], [24], and [25] delve into secure communication using RNC, but they often assume networks consisting of many distributed nodes that deliver packets through multiple routes. Consequently, they do not guarantee security in one-to-one communication. In our investigation, we found no prior research on the proactive use of RNC for secure one-to-one communication.

QUIC is a recently developed network protocol aimed at resolving inherent issues in TCP [26]. Built on UDP, QUIC is designed to achieve low latency and more efficient congestion and packet loss control compared to TCP. It also serves as the official network protocol in HTTP/3. To showcase the advantages of our approach, we have implemented our algorithms in QUIC and measured their throughput.

III. Augmented Hill Cipher

In this section, we explain AHC using the following notations:

R	Rank of matrix
n	Modulus
P	$R \times R$ matrix form of p (plaintext)
C	$R \times R$ matrix form of c (ciphertext)
K	$R \times R$ shared key matrix

A. Algorithms

AHC uses three shared key matrices, denoted as K_0, K_1, K_2, and performs plaintext encryption as illustrated in Algo. 1 and Fig. 1. In this process, the ith plaintext P_i is first multiplied by $K_{i \bmod 3}$ and $K_{(i+1) \bmod 3}$ is then added to the result, yielding C'_i. To obtain ciphertext C_i, we XOR C'_i with C'_{i-1}.

Algorithm 1 Encryption in Augmented Hill Cipher

$C'_{-1} = K_2$
for $i \leftarrow 0$ to $N-1$ **do**
  $C'_i \leftarrow (K_{i \bmod 3} P_i + K_{(i+1) \bmod 3}) \bmod n$
  $C_i \leftarrow C'_i \oplus C'_{i-1}$
end for

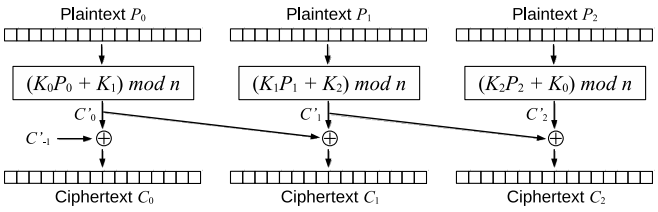

Fig. 1: Encryption in Augmented Hill Cipher.

Algorithm 2 Decryption in Augmented Hill Cipher

$C'_{-1} = K_2$
for $i \leftarrow 0$ to $N-1$ **do**
  $C'_i \leftarrow C_i \oplus C'_{i-1}$
  $P_i \leftarrow (K^{-1}_{i \bmod 3}(C'_i - K_{(i+1) \bmod 3})) \bmod n$
end for

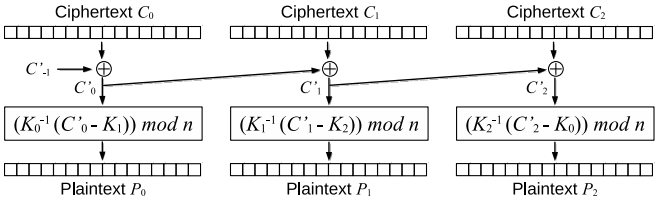

Fig. 2: Decryption in Augmented Hill Cipher.

The decryption process in AHC is the reverse of encryption, as demonstrated in Algo. 2 and Fig. 2. In this process, we start with $C'_{-1} = K_2$ and then XOR the ciphertext C_i with C'_{i-1} to obtain C'_i. By subtracting $K_{(i+1) \bmod 3}$ from C'_i and then multiplying by $K^{-1}_{i \bmod 3}$, we retrieve plaintext P_i.

B. Security Analysis

In the encryption process, AHC initially XORs C'_{-1} $(= K_2)$ with $(K_{0 \bmod 3} P_0 + K_{(1) \bmod 3})$ mod n, as shown in the leftmost part of Fig. 1. This step ensures that P_0 (the beginning of the original data) remains concealed. Subsequently, newly encrypted data is XORed with previously generated encrypted data, providing protection for the succeeding data. This combination of operations, addition, and XOR, enhances security.

Regarding key length, the number of invertible $R \times R$ matrices over $n = 2^\alpha$ can be calculated as [27]:

$$l = 2^{(\alpha-1)R^2} \prod_{k=0}^{R-1} (2^R - 2^k).$$

This formula determines the key length for each of the Ks in AHC. For instance, if $n = 2^{16}$ and $R = 8$, we have $l \approx 2^{1022}$, and the total key length of AHC is $2^{1022 \times 3} = 2^{3066}$, which is more than sufficient. AHC can further extend its key length by increasing the values of n, R, or the number of Ks. As mentioned in Section II and Section 6 of [2], AHC is considered highly resistant to most common attacks.

However, it is essential to acknowledge the possibility of future vulnerabilities. It is important to note that our cipher algorithms are based on AHC, but we are prepared to adopt a more secure modified HC if any vulnerabilities are discovered. For the purposes of this paper, AHC is assumed to be secure, and it serves as an illustrative example of a modified HC.

IV. Hill Cipher with Network Coding

This section presents the algorithms for our Hill Cipher with Network Coding *HNC* approach, using the following notations:

R	Rank of matrix
n	Size of each data or size of GF (= 2^{16} or 2^8)
M	# of data to process determined by SIMD (= 32)
P	$R \times M$ matrix form of p (plaintext)
C	$R \times M$ matrix form of c (ciphertext)
K	$R \times R$ shared key matrix
K_β	$R \times M$ shared key matrix
$K_{C'_{-1}}$	$R \times M$ shared key matrix

A. Implementing Augmented Hill Cipher

To optimize the practical speeds of encryption and decryption, we made slight modifications to the original AHC implementation. In particular, we utilize GF for matrix multiplication and observed that the processing speed is highly dependent on the efficiency of the GF arithmetic library used. We developed *gf-nishida-16* [14], which leverages lookup instructions in SIMD, such as AVX VPSHUFB or NEON VTBL. To the best of our knowledge, this library is the fastest GF arithmetic library for region multiplication in $GF(2^{16})$ and $GF(2^8)$. It is also open-sourced at [28].

SIMD instructions like AVX, SSE, or NEON typically process data in chunks of 512, 256, or 128 bits. For matrix multiplication, where matrix A is $R \times R$ and matrix B is $R \times M$, gf-nishida-16 requires each row of matrix B to have a total size of 512 bits for $GF(2^{16})$ and 256 bits for $GF(2^8)$. With $512/16 = 256/8 = 32$, we set $M = 32$ for our matrix multiplication. Consequently, the size of the plaintext matrix P and the ciphertext matrix C must both be $R \times 32$. This adjustment also affects the sizes of the shared key matrices, as explained below.

B. Algorithms of HNC with No Redundancy

Algorithm 3 Encryption in HNC with no redundancy

1: $C'_{-1} = K_{C'_{-1}}$
2: **for** $i \leftarrow 0$ to $N - 1$ **do**
3: $\quad C'_i \leftarrow (K_{i \bmod 3} P_i + K_{\beta\, i \bmod 3})$ mod n
4: $\quad C_i \leftarrow C'_i \oplus C'_{i-1}$
5: **end for**

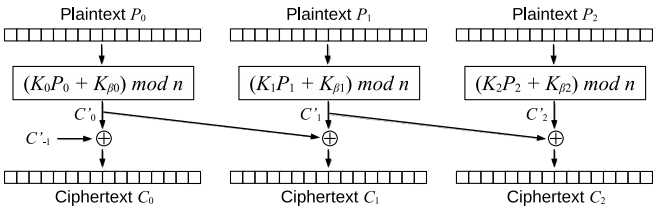

Fig. 3: Encryption in HNC with no redundancy.

Algorithm 4 Decryption in HNC with no redundancy

1: $C'_{-1} = K_{C'_{-1}}$
2: **for** $i \leftarrow 0$ to $N - 1$ **do**
3: $\quad C'_i \leftarrow C_i \oplus C'_{i-1}$
4: $\quad P_i \leftarrow (K^{-1}_{i \bmod 3}(C'_i - K_{\beta\, i \bmod 3}))$ mod n
5: **end for**

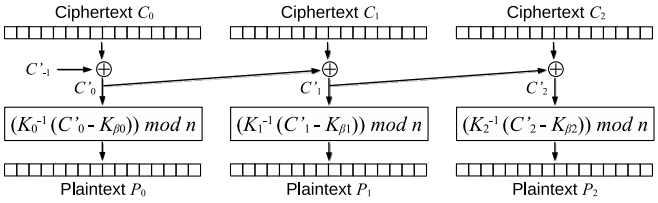

Fig. 4: Decryption in HNC with no redundancy.

In the AHC encryption process, both $K_{i \bmod 3} P_i$ and $K_{(i+1) \bmod 3}$ are $R \times R$ matrices (refer to Line 3 in Algo. 1). However, in the context of HNC with no redundancy (HNCn), $K_{i \bmod 3} P_i$ is $R \times 32$, while $K_{(i+1) \bmod 3}$ remains $R \times R$, resulting in a matrix size mismatch. To address this, we introduce new $R \times 32$ key matrices: $K_{\beta 0}, K_{\beta 1}, K_{\beta 2}$ to replace $K_{(i+1) \bmod 3}$ in Line 3 of Algo. 1, and an $R \times 32$ matrix $K_{C'_{-1}}$ to replace K_2 in Line 1 of Algo. 1. The encryption algorithm for HNCn is presented in Algo. 3 and Fig. 3, while the decryption algorithm is detailed in Algo. 4 and Fig. 4. It is important to

TABLE I: Key lengths (bit) in HNC with no redundancy

	$n = 2^{16}$ $R = 4$	$n = 2^{16}$ $R = 6$	$n = 2^8$ $R = 4$	$n = 2^8$ $R = 6$
Each of Ks	255.99997	575.99997	127.994	287.994
Each of K_βs	2048	3072	1024	1536
$K_{C'_{-1}}$	2048	3072	1024	1536
Total	8959.99991	14015.99991	4479.982	7007.982

note that these new keys not only accommodate the matrix size differences but also contribute to an increase in the total key length of HNC, thereby enhancing its security.

C. Security Analysis

First, as HNCn shares the same algorithm with AHC except for the newly added shared key matrices (K_βs, $K_{C'_{-1}}$) and the extended sizes of matrices, we can consider HNCn to be at least as secure as AHC from an algorithmic perspective. Next, calculate the key length of HNCn. While K_0, K_1, and K_2 are $R \times R$ matrices, they have a constraint that they must be invertible. The number of invertible $R \times R$ matrices in $GF(2^\alpha)$ is given by [29]:

$$l = \prod_{k=0}^{R-1} (2^{\alpha R} - 2^{\alpha k}). \tag{6}$$

As a result, we can determine the key lengths for each of K_0, K_1, and K_2 for different values of n (2^{16} and 2^8) and R (4 and 6), as shown in the second row of Table I. Since K_βs and $K_{C'_{-1}}$ are $R \times 32$ matrices over 2^{16} or 2^8 and have no constraints, we can determine the key lengths for each of them, as shown in the third and fourth rows of Table I. The total key length can be calculated as:

$$\{\prod_{i=0}^{2}(\text{Key len of } K_i)(\text{Key len of } K_{\beta i})\}(\text{Key len of } K_{C'_{-1}}). \tag{7}$$

Using this formula, we obtain the key lengths for HNCn for different values of n (2^{16} and 2^8) and R (4, 6), as indicated in the last row of Table I. Furthermore, it is worth noting that HNCn exhibits a 50% Avalanche Effect for all combinations of n (2^{16}, 2^8) and R (4, 6, 8). As a result, HNCn can be considered secure, as long as AHC is secure. The complexity of the key length for $n = 2^\alpha$ and R is $O(n^{R^2})$.

D. Computational Complexity and Configuration of Algorithms

The most expensive calculation in Algos. 3 and 4 is matrix multiplication which costs $O(R^3)$. Hence, the difference in R values of 4 and 6 can greatly affect the encryption and decryption speeds. Regarding $n = 2^\alpha$, where $\alpha = 8, 16, 32, \ldots$, many GF multiplication algorithms including gf-nishida-16 exhibit a time complexity of $O(\alpha)$ for processing data in $GF(2^\alpha)$. In practical terms, this means that multiplication in $GF(2^{16})$ costs almost twice as much as in $GF(2^8)$. Consequently, we have an overall time complexity of $O((\log n)R^3)$ for HNCn.

While it is true that using larger GFs and R hugely increases the key length and thereby strengthens security, as indicated in Table I, HNCns for R equal to 4, especially when coupled with $n = 2^{16}$, still provide sufficiently long key lengths, and therefore we recommend using R equal to 4, especially when paired with $n = 2^{16}$, as the preferred configuration for our algorithms. To substantiate the computational cost in practical communication, our benchmark was made with the combinations of n (2^{16}, 2^8) and R (4, 6) (see Section V).

E. Implementing Random Network Coding

In this section, we explain how to incorporate RNC into HNC to create HNC with redundancy (HNCr). As mentioned in Section I, adding a row to each key matrix introduces communication redundancy, reducing the need for retransmission of lost packets. We denote these extended key matrices as follows:

$\hat{K}$	$(R+1) \times R$ shared key matrix
$\hat{K}_\beta$	$(R+1) \times 32$ shared key matrix
$\hat{K}_{\hat{C}'_{-1}}$	$(R+1) \times 32$ shared key matrix
$\hat{C}$	$(R+1) \times 32$ matrix form of c (ciphertext)
$A^{(r)}$	Matrix without rth row of A
L	Packet loss rate $(0 \le L \le 1)$

Algos. 3 and 4 are updated to Algos. 5 and 6, respectively, to accommodate this redundancy.

Algorithm 5 Encryption in HNC with redundancy

$\hat{C}'_{-1} = \hat{K}_{\hat{C}'_{-1}}$
for $i \leftarrow 0$ to $N-1$ **do**
  $\hat{C}'_i \leftarrow (\hat{K}_{i \bmod 3} P_i + \hat{K}_{\beta\, i \bmod 3})\ mod\ n$
  $\hat{C}_i \leftarrow \hat{C}'_i \oplus \hat{C}'_{i-1}$
end for

Algorithm 6 Decryption in HNC with redundancy

$\hat{C}'_{-1} = \hat{K}_{\hat{C}'_{-1}}$
for $i \leftarrow 0$ to $N-1$ **do**
  Identify lost row $r \in [0, R]$ in $\hat{C}_i$.
  if no r found **then**
    $r \leftarrow$ any value $\in [0, R]$.
    $PacketLoss \leftarrow False$
  else
    $PacketLoss \leftarrow True$
  end if
  $\hat{C}'_i \leftarrow \hat{C}_i \oplus \hat{C}'_{i-1}$
  $P_i \leftarrow (\hat{K}^{(r)}_{i \bmod 3})^{-1}(\hat{C}'^{(r)}_i - \hat{K}^{(r)}_{\beta\, i \bmod 3}))\ mod\ n$
  if $PacketLoss$ **then**
    Create lost rth row of $\hat{C}'_i$ from P_i using Algo. 5.
  end if
end for

The sender transmits each row of $\hat{C}_i$ in separate packets. If we assume that the rth row of $\hat{C}_i$ $(0 \le r \le R)$ is sent in packet r, then even if one of these packets is lost during transmission, the receiver can still reconstruct the corresponding plaintext P_i using the remaining R packets (as described in (4) and (5)). When the packet loss rate $L = 0$, the communication throughput decreases by $1/(R+1)$ due to the redundant transmission. However, for $L \neq 0$, this redundancy technique significantly reduces the need for retransmissions and thereby

TABLE II: Processing speeds of HNC with no redundancy (HNCn) over OpenSSL ($S_{\text{HNCn}}/S_{\text{OpenSSL}}$) in local TCP communication. M1 shows lower values than Ryzen 7.

	HNCn-16bit-4	HNCn-16bit-6	HNCn-8bit-4	HNCn-8bit-6
Ryzen 7 5800X	3.215	2.229	4.387	2.699
Apple M1	1.225	0.930	1.948	1.133

improves overall throughput. In the absence of redundancy, the sender is required to retransmit the lost packet every time packet loss occurs. With a redundancy of 1, if a packet is lost, the sender only needs to retransmit the lost packet if one or more other packets are lost within the $R+1$ packets that contain the rows from the same $\hat{C}_i$. The probability of this happening is given by

$$1-(1-L)^R. \tag{8}$$

Hence, the retransmission rate decreases by $(1-L)^R$, leading to a significant improvement in throughput in lossy communication. For instance, when we have a packet loss rate of $L = 0.01$ and a rank set to $R = 4$, the calculation yields $1-(1-0.01)^4 = 0.04$. This indicates that retransmissions are required for only 4% of the lost packets when the packet loss rate is 1%. In our simulations, this translates to a remarkable 200% increase in throughput, as demonstrated in Section V-C.

Regarding security, if there were a method to crack HNCr, it would also apply to HNCn since HNCn can be seen as a special case of HNCr (e.g., when the Rth rows of K matrices are identical to their $(R-1)$th rows). This implies that HNCr is at least as secure as HNCn. Therefore, HNCr is secure as long as AHC is secure.

V. Benchmark Results

A. Throughput in TCP

In this section, we present a comparison of throughput and processing speed in local TCP client-server communication using OpenSSL (accelerated with AES-NI, utilizing the TLS_AES_256_GCM_SHA384 algorithm) and HNCn. We consider different values of n, specifically $n = 2^{16}$ and $n = 2^8$, as well as different ranks (R) for the matrices, with R taking values 4 and 6. These measurements were conducted on both AMD Ryzen 7 5800X and Apple M1 CPUs, and all communication and encryption/decryption processes were executed on a single thread.

As shown in Fig. 5, HNCn implementations, with the exception of HNCn-16bit-6, show higher throughput than OpenSSL, particularly noteworthy are HNCn-8bit-4 and HNCn-16bit-4 on Ryzen 7, which exhibit three times and two and a half times the throughput of OpenSSL, respectively.

The difference in results for R values of 4 and 6 is attributed to the computational cost of matrix multiplication, as described in Section IV-D. HNCn with R equal to 6 performs noticeably slower than the version with R equal to 4.

To assess the processing speed, which encompasses encryption, decryption, and any additional overhead introduced by

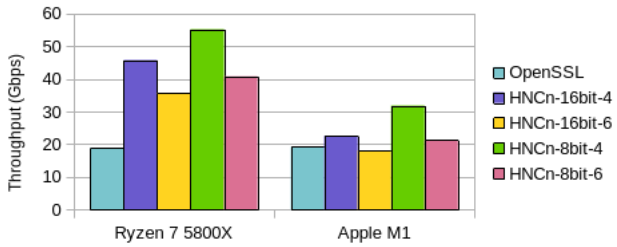

Fig. 5: Throughputs in local TCP communication with OpenSSL and HNC with no redundancy (HNCn). HNCns mostly show higher throughput.

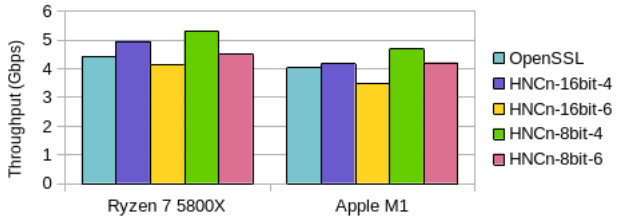

Fig. 6: Throughputs in local QUIC communication with OpenSSL and HNC with no redundancy (HNCn). HNCns show higher throughputs except for HNCn-16bit-6.

utilizing the target algorithm (Algo), we introduce the following metric:

$$S_{\text{Algo}} = 1/\left(1/TP_{\text{Algo}} - 1/TP_{\text{Non-crypto}}\right), \tag{9}$$

where TP represents throughput. It is important to note that the throughputs in non-cryptographic TCP communication ($TP_{\text{Non-crypto}}$) are 126.55 Gbps on Ryzen 7 and 105.58 Gbps on M1. Table II presents the values of $S_{\text{HNCn}}/S_{\text{OpenSSL}}$. On Ryzen 7, HNCn performs 2.2 to 4.3 times faster than OpenSSL, while on M1, it achieves processing speeds 0.9 to 1.9 times faster. Notably, HNCn with R equal to 4 outpaces OpenSSL in processing speed. However, it is worth mentioning that HNCn on M1 does not exhibit as high processing speed ratios as those on Ryzen 7. This discrepancy can be primarily attributed to the register size of ARM NEON, which is 128 bits—half that of AVX's 256 bits. In practice, our algorithms with SSE, which also has a 128-bit register size, are approximately 30% slower than their AVX counterparts. For efficient encryption and decryption, SIMD instructions with larger register sizes are preferable.

B. Throughput in QUIC

QUIC is an encrypted, multiplexed, and low-latency protocol designed to enhance transport performance for HTTPS traffic [30]. Because data encryption is an inherent feature of QUIC, it serves as an ideal platform for testing HNC. MsQuic [31], backed by Microsoft, is one of the most active open-source projects implementing QUIC. It utilizes OpenSSL as its cipher library on Linux and MacOS. We integrated HNCn into MsQuic and measured throughput and data processing speed using the same settings as those outlined in Section V-A.

TABLE III: Processing speeds of HNC with no redundancy (HNCn) over OpenSSL ($S_{\text{HNCn}}/S_{\text{OpenSSL}}$) in local QUIC communication. HNCn-8bit-4 on M1 shows an irregularly high value.

	HNCn-16bit-4	HNCn-16bit-6	HNCn-8bit-4	HNCn-8bit-6
Ryzen 7 5800X	1.566	0.816	2.214	1.076
Apple M1	1.244	0.494	7.922	1.352

The results are presented in Fig. 6 and Table III. The non-cryptographic throughputs ($TP_{\text{Non-crypto}}$) are 6.31Gbps for Ryzen 7 and 4.80Gbps for M1, respectively. With the exception of HNCn-16bit-6, HNCns generally exhibit higher throughputs compared to OpenSSL, although the differences are not particularly significant. Regarding processing speed, the values of $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ on Ryzen 7 are not as impressive as those observed in TCP scenarios. However, HNCns running on M1 demonstrate relatively strong performance, except for HNCn-16bit-6. It is worth noting that our implementation does not fully leverage packet length optimization, which can impact data processing efficiency in QUIC. QUIC operates on top of UDP, and the packet length depends on the Maximum Transmission Unit (MTU) size, which in turn determines the payload length in each packet. For an MTU size of 1500 bytes, QUIC packets generally have payloads exceeding 1450 bytes, as MsQuic tends to maximize payload utilization. However, the most efficient payload lengths for HNCn are given by $64 \times R \times N$ for $n = 2^{16}$ and $32 \times R \times N$ for $n = 2^8$, where N is an integer. With an MTU size of 1500 bytes and $R = 4$, these payload lengths have upper bounds of 1280 bytes and 1408 bytes, respectively. For $R = 6$, the upper bounds are 1152 bytes and 1344 bytes, respectively. Consequently, the ideal payload lengths for HNCn are generally shorter than typical QUIC payload lengths. This inefficiency in payload length contributes to HNCn's lower efficiency in QUIC compared to TCP. This results in lower $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ values on Ryzen 7 (when comparing the second rows of Table II and III). We have not yet identified the specific factors causing some high $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ values on M1, particularly for HNCn-8bit-4. Further investigation and payload length optimization will be necessary in the future.

C. Throughput in Lossy Communication

While redundancy is unnecessary for HNC when no packet loss occurs, it becomes crucial to introduce HNCr (HNC with redundancy) to enhance throughput when consistent packet loss is observed. In our experiment, we extended MsQuic to accommodate HNCr and conducted a throughput comparison between OpenSSL (without redundancy) and HNCr (with redundancy level set to 1) in a lossy communication environment. To simulate packet loss, we intentionally dropped received packets within MsQuic based on the designated loss rate. We also limited the available bandwidth to 60Mbps and introduced a 50ms network delay using Linux's tc command. This designated loss rate is denoted as L.

The results, shown in Fig. 7, can be summarized as follows:

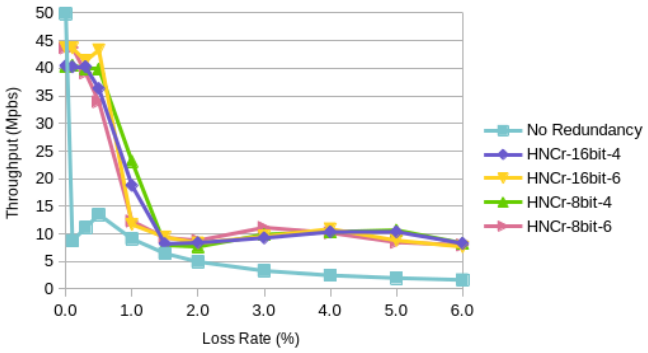

Fig. 7: Throughputs in local lossy QUIC communication with no redundancy (OpenSSL) and HNC with redundancy (HNCr). HNCrs generally show higher throughputs.

1) HNCrs consistently achieve higher throughputs than OpenSSL across the entire range. The overall average ratio of $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$ is 3.08.
2) HNCrs exhibit significantly higher throughputs than OpenSSL for $L = 0.1$–0.5%, with $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$ ratios ranging from 2.69 to 5.00. However, their throughputs become comparable at around $L = 1.5\%$.
3) Notable differences are not observed among different HNCr configurations, especially among those with the same R value.
4) OpenSSL experiences a significant drop in throughput between $L = 0$ and 0.1%. Although throughput slightly recovers beyond this point, it gradually declines after reaching $L = 0.5\%$.
5) HNCr–4 and HNCr–6 configurations exhibit sharp reductions in throughput between $L = 0.5$–1.5% and $L = 0.5$–1.0%, respectively, but subsequently maintain stable throughputs.

The advantage of HNCr is highlighted in point 1), but it is important to note that in point 2), $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$ is not consistent. While the reasons behind points 4) and 5) have not been confirmed, our speculation is that they may be attributed to the design and implementation of MsQuic. In summary, the redundancy (RNC) technique proves to be effective, and HNCr significantly enhances throughput in lossy communication.

Note that QUIC is used to incorporate Forward Error Correction (FEC) to recover a single packet loss within a group. However, this feature was eventually removed due to its limited effectiveness. It was also reported that FEC led to a slight increase in video latency and video rebuffering rates, without providing sufficient advantages in terms of bandwidth usage [30]. While we have not measured all the impacts of HNCr, it has the potential to serve as an alternative to FEC.

VI. Future Work

It is clear that HNC is better suited for wireless communication than wired connections. In particular, we believe it meets the requirements for secure and reliable satellite communication, which is essential for preventing eavesdropping

and mitigating data loss due to jamming. Suppose a satellite communication system has multiple channels denoted as $Ch_0, Ch_1, ..., Ch_R$, and each packet, labeled as r in Section IV-E, is transmitted through the channel $Ch_{r \bmod (R+1)}$. In this setup, even if one of the channels becomes jammed or disrupted, the receiver can still recover the original data using the packets received through the unaffected channels. Additionally, even in a scenario where an eavesdropper gains access to all the communication channels and intercepts all encrypted data, decrypting it would be an exceedingly difficult task. Thus, theoretically, HNC enables secure and reliable satellite communication. However, as described in Section V-B, our current implementation is not optimized for transmitting data in packets of a specific size. We need to explore more efficient methods for packaging encrypted data into packets. Additionally, it is crucial to maintain a vigilant focus on security analysis regarding the modified HC.

VII. Conclusion

In this paper, we have introduced an approach that securely encrypts and decrypts data at high speeds while demonstrating tolerance to packet loss. Our benchmark results illustrate that HNC with non-redundancy outperforms AES-256 in terms of encryption and decryption speed. Additionally, HNCr significantly reduces the need for packet retransmission in lossy communication scenarios. These results collectively manifest as an overall high throughput in our benchmarks, substantiating the advantages of HNC. We believe that our approach contributes to enhancing the reliability and smoothness of wireless communication, ultimately providing added convenience to Internet users.

References

[1] Lester S. Hill, "Cryptography in an algebraic alphabet," *The American Mathematical Monthly*, vol. 36, no. 6, pp. 306–312, 1929.

[2] Abdallah A. Alhabshy, "Augmented hill cipher," *International Journal of Network Security*, vol. 21, pp. 812–818, 2019.

[3] Mohsen Toorani and Abolfazl Falahati, "A secure variant of the hill cipher," in *2009 IEEE Symposium on Computers and Communications*, 2009, pp. 313–316.

[4] Narendra B. Parmar and Kiritkumar Ramanbhai Bhatt, "Hill cipher modifications: A detailed review," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 3, pp. 1467–1474, 2015.

[5] M.N.A. Rahman, A.F.A. Abidin, Kamir Yusof, and N.S.M. Usop, "Cryptography: A new approach of classical hill cipher," *International Journal of Security and its Applications*, vol. 7, pp. 179–190, 01 2013.

[6] Ahmed Mahmoud and Alexander Chefranov, "Secure hill cipher modifications and key exchange protocol," in *2010 IEEE International Conference on Automation, Quality and Testing, Robotics, AQTR 2010 - Proceedings*, 06 2010, vol. 2, pp. 1 – 6.

[7] K. Adinarayana Reddy, B. Vishnuvardhan, Madhuviswanatham, and A.V.N. Krishna, "A modified hill cipher based on circulant matrices," *Procedia Technology*, vol. 4, pp. 114–118, 2012, 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[8] Desmond S. Lun, Muriel Médard, Ralf Koetter, and Michelle Effros, "On coding for reliable communication over packet networks," *CoRR*, vol. abs/cs/0510070, 2005.

[9] J. K. Sundararajan, D. Shah, M. Medard, M. Mitzenmacher, and J. Barros, "Network coding meets tcp," in *IEEE INFOCOM 2009*, April 2009, pp. 280–288.

[10] B. Li and D. Niu, "Random network coding in peer-to-peer networks: From theory to practice," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 513–523, March 2011.

[11] Hiroshi Nishida and Thinh Nguyen, "Rncdds - random network coded distributed data system," in *2017 IEEE International Conference on Multimedia Expo Workshops (ICMEW)*, July 2017, pp. 297–302.

[12] Luísa Lima, Steluta Gheorghiu, João Barros, Muriel Médard, and Alberto Lopez Toledo, "Secure network coding for multi-resolution wireless video streaming," *IEEE J.Sel. A. Commun.*, vol. 28, no. 3, pp. 377–388, Apr. 2010.

[13] D. Vukobratović, C. Khirallah, V. Stanković, and J. S. Thompson, "Random network coding for multimedia delivery services in lte/lte-advanced," *IEEE Transactions on Multimedia*, vol. 16, no. 1, pp. 277–282, Jan 2014.

[14] Hiroshi Nishida, "gf-nishida-16: Simple and efficient $gf(2^{16})$ library," https://github.com/scopedog/gf-nishida-16/blob/master/gf-nishida-16.pdf.

[15] Mozhgan Mokhtari and Hassan Naraghi, "Analysis and design of affine and hill cipher," *Journal of Mathematics Research*, vol. 4, 01 2012.

[16] Vaishali Billore and Naresh Patel, "Cryptography utilizing the affine-hill cipher and extended generalized fibonacci matrices," *Electronic Journal of Mathematical Analysis and Applications*, vol. 11, pp. 1–11, 07 2023.

[17] Mohd Aziz UR Rehman, Hasan Raza, and Israr Akhter, "Security enhancement of hill cipher by using non-square matrix approach," *Proceedings of the 4th international conference on knowledge and innovation in Engineering, Science and Technology*, 2018.

[18] Y. Guo, L. Li, and B. Liu, "Shadow: A lightweight block cipher for iot nodes," *IEEE Internet of Things Journal*, vol. PP, pp. 1–1, 03 2021.

[19] Xing Zhang, Shaoyu Tang, Tianning Li, Xiaowei Li, and Changda Wang, "Gfrx: A new lightweight block cipher for resource-constrained iot nodes," *Electronics*, vol. 12, no. 2, 2023.

[20] A. Dunmore, J. Samandari, and J. Jang-Jaccard, "Matrix encryption walks for lightweight cryptography," *Cryptography*, vol. 7, no. 3, 2023.

[21] Muath A. Obaidat, Joseph Brown, and Abdullah Al Hayajneh, "A novel paradigm for access control trust in iot applications: A distributed cross-communication approach," in *2021 13th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2021, pp. 25–31.

[22] Luísa Lima, Muriel Médard, and João Barros, "Random linear network coding: A free cipher?," *CoRR*, vol. abs/0705.1789, 2007.

[23] Keesook Han, Tracey Ho, Ralf Koetter, Muriel Medard, and Fang Zhao, "On network coding for security," in *MILCOM 2007 - IEEE Military Communications Conference*, 2007, pp. 1–6.

[24] João P. Vilela, Luísa Lima, and João Barros, "Lightweight security for network coding," *CoRR*, vol. abs/0807.0610, 2008.

[25] M. A. Brahimi and F. Merazka, "Data confidentiality-preserving schemes for random linear network coding-capable networks," *Journal of Information Security and Applications*, vol. 66, pp. 103136, 2022.

[26] John Dellaverson, Tianxiang Li, Yanrong Wang, Jana Iyengar, Alexander Afanasyev, and Lixia Zhang, "A quick look at quic," *The Internet Protocol Journal*, pp. 2–12, 2019.

[27] Jeffrey Overbey, William Traves, and Jerzy Wojdylo, "On the keyspace of the hill cipher," *Cryptologia*, vol. 29, no. 1, pp. 59–72, 2005.

[28] Hiroshi Nishida, "gf-nishida-16 web site," https://github.com/scopedog/gf-nishida-16/.

[29] Gabe Cunningham, "The genral linear group," https://math.mit.edu/~dav/genlin.pdf.

[30] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. R. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and Z. Shi, "The quic transport protocol: Design and internet-scale deployment," *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017.

[31] Microsoft, "Msquic," https://github.com/microsoft/msquic.