Hill Cipherとネットワークコーディングによる安全、高速かつパケットロスに耐性のある通信

西田博史
ASUSA Corporation, Salem Oregon, USA

概要—本稿は改良型 Hill Cipher（HC）とランダムネットワークコーディング（RNC）を組み合わせることにより、安全かつパケットロスに耐性のある通信が同時に実現できることについて解説する。HC と RNC の両者は、ともに一次連立方程式を作ることによってデータを暗号化／エンコードし、それを解くことによって復号化／デコードする。本方法はそれら二つのアルゴリズムを融合させ、効率的に両者の特性を引き出す。さらに新たに開発した高速ガロア体演算アルゴリズムを用いることにより、ハードウェアアクセラレーションの効いた AES-256 よりも高速な暗号化・復号化を実現する。本アルゴリズムを実装した QUIC プロトコルは、パケットロスが起こる通信状態でスループットを大きく改善する。

I. 導入

1929 年に発表された Hill Cipher (HC) [1] は古典的な対称型暗号化アルゴリズムで、以下のような連立方程式に基づいてデータの暗号化・復号化を行う。

$$\begin{aligned} c &= Kp \quad \text{(暗号化)} \\ p &= K^{-1}c \quad \text{(復号化)} \end{aligned} \tag{1}$$

$\boldsymbol{p}$　平文
$\boldsymbol{c}$　暗号文
$\boldsymbol{K}$　共有鍵行列
$\boldsymbol{R}$　行列 K の階数

この方式はシンプルで直感的であるが、既知平文攻撃に脆弱であることが知られており、そのため数多くの改良型が提唱されてきた [2]–[7]。これらの中には依然として脆弱なものもあるが、現時点で高い安全性を保持してると考えられているものも存在する。

ランダムネットワークコーディング (RNC) [8]–[13] は基本的に (1) と同様の方法でデータをエンコード・デコードするが、伝達ロスの大きい通信において、データの再送を減らすために冗長性を用いる。例えば (1) において $R = 3$, K を 3×3 の行列、$|p| = |c| = 3$ とすると、(1) における暗号化部分は

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

のように表される。RNC は例えば以下のように (2) にもう一つの方程式を足し、

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

のようにし、$c_0 \cdots c_2$ に加え c_3 も生成し、これら全てを受信側に送る。すると例え $c_0 \cdots c_3$ の 1 つを失ったとしても、残り 3 つの c_i から $p_0 \cdots p_2$ を復号することができる。例えば c_1 が失われたとし、

$$\hat{K} = \begin{pmatrix} k_{0,0} & k_{0,1} & k_{0,2} \\ k_{2,0} & k_{2,1} & k_{2,2} \\ k_{3,0} & k_{3,1} & k_{3,2} \end{pmatrix},$$

とする。すると受信者は

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

のようにデコードすることによって、元の平文 p を復元することができる。このようにして RNC はデータの冗長性と引き換えに、伝達ロスによるパケットの再送頻度を減少させる。通信時におけるパケットロスは、繰り返し行われるデータ再送要求のやり取りのため、著しくスループットを落とす原因となり、円滑な通信の妨げとなる。RNC はそれを大きく改善する役割を果たす。

RNC の欠点の 1 つにエンコード・デコードにかかるコストが高いということが挙げられる。平文の通信に RNC を用いることは、余分な CPU の使用とメッセージ伝達の遅延をもたらすことになる。しかし元のデータが暗号化されるという前提の場合、もし RNC がその暗号化を兼ねるとすれば、その欠点が打ち消される。これが本通信法の狙いである。

通常 (1) (2) (3) のような暗号化・復号化における計算はガロア体で行われる。そのためガロア体における計算速度が暗号化・復号化の実速度を大きく左右する。実際に我々のアルゴリズムは、使用するガロア体演算ライブラリーによって大きく処理速度が異なることを検証結果より得ている。我々は以前に開発した高速ガロア体演算ライブラリー [14] に新たなアルゴリズムを追加し、それを使用することによって、アルゴリズム内の暗号化・復号化部分の高速化に成功している。第 V 章で AES-NI によるハードウェアアクセラレーションの効いた OpenSSL の AES-256 と比較しているが、実際にそれよりも高速な暗号化・復号化を実現していることが示されている。

本稿では改良型 HC と RNC の詳細について第 II、III 章で述べ、第 IV 章で我々のアルゴリズムについて解説している。そして通信時における我々のアルゴリズムのスループットおよび処理速度の測定結果を第 V 章に記している。

II. 関連研究

既知平文攻撃に対処するため、多くの改良型 HC が提唱されてきた。Augmented Hill Cipher [2] は 3 つの鍵行列を用い、新しく暗号化されたデータと一つ前の暗号化時に生成

されたデータとの排他的論理和 ($\oplus$ または XOR) を繰り返すことによって既知平文攻撃を始めとした攻撃に対処している。またさらに $+$ と $\oplus$ の演算を組み合わせることによって、様々な攻撃に対する耐性を高めており、 既知平文攻撃を始め、選択平文攻撃、暗号文単独攻撃、辞書攻撃、総当たり攻撃、フォールト攻撃の全てに対して安全であると結論づけている。詳細は第 III 章を参照していただきたい。この方法は現時点で我々の知る限り脆弱性が見つかっておらず、我々の提唱するアルゴリズムの暗号化・復号化の部分にこのアルゴリズムを拡張したものを用いている。

[7] は可逆行列を公開鍵、巡回行列を秘密鍵にした改良型 HC を紹介しているが、[2] において複数の脆弱性が指摘されている。その他の改良型 HC には、アフィン暗号化と組み合わせたものがいくつか存在する [3], [15]。これらは平文を構成するアルファベットをシンプルな関数で変換することによって暗号化・復号化を行う。また [16] は非正方な鍵行列を用いて暗号化されたデータに冗長性を持たせている。しかしこれは安全性のためであり、RNC のように通信における冗長性には使用されていない。

[8], [9] は RNC によりどれほどパケットロスが効率的に対処されるかを説明している。特に [9] は RNC を TCP に実装し、伝達ロスが起こる通信でスループットが劇的に改善されることを示している。[17]–[20] は RNC を用いた安全な通信について述べているが、これらは多数の中間ノードが存在するネットワークで複数の経路を通ってデータが伝達されることを前提としており、一対一通信での安全性を保証していない。我々の調査では、RNC を安全な一対一の通信に特化して利用した研究は見つかっていない。

QUIC [21] は TCP で見られる様々な問題を解決するために新たに開発されたネットワークプロトコルであり、HTTP/3 で採用される唯一のプロトコルである。QUIC は UDP 上に実装され、TCP よりも低レイテンシーで、トラフィックの混雑やパケットロスを効率的に制御できるように設計されている。本稿では我々のアルゴリズムの有効性を実証するために、QUIC にアルゴリズムを実装し、スループットの測定を行っている。

III. Augmented Hill Cipher

本章では [2] で紹介されている Augmented Hill Cipher (AHC) について以下の表記を用いて解説する。

$\boldsymbol{R}$ 行列の階数
$\boldsymbol{n}$ モジュラス
$\boldsymbol{P}$ p（平文）の $R \times R$ 行列型
$\boldsymbol{C}$ c（暗号文）の $R \times R$ 行列型
$\boldsymbol{K}$ $R \times R$ な共有鍵行列

A. アルゴリズム

AHC は 3 つの共有鍵行列 K_0, K_1, K_2 を用い、アルゴリズム 1 や図 1 のように平文を暗号化する。 アルゴリズム 1 の 3 行目に見られるように、まず K_i と i 番目の平文 P_i の積を取り、それに $K_{(i+1) \bmod 3}$ を足す。その結果は C'_i として C_{i+1} を求める際のために保存しておく。そして C'_i と C'_{i-1} の排他的論理和が i 番目の暗号文 C_i となる。

復号化は暗号化と全く逆の手順でアルゴリズム 2 や図 2 のように行われる。まず暗号文 C_i と C'_{i-1} の排他的論理和

Algorithm 1 Augmented Hill Cipher における暗号化

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

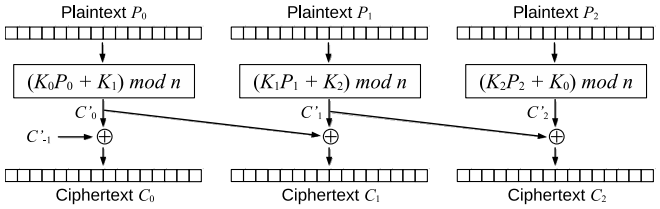

Fig. 1: Augmented Hill Cipher における暗号化

を C'_i として保存し、C'_i から $K_{(i+1) \bmod 3}$ を引いたものに $K^{-1}_{i \bmod 3}$ を掛けることによって平文 P_i を復号させる。

Algorithm 2 Augmented Hill Cipher における復号化

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

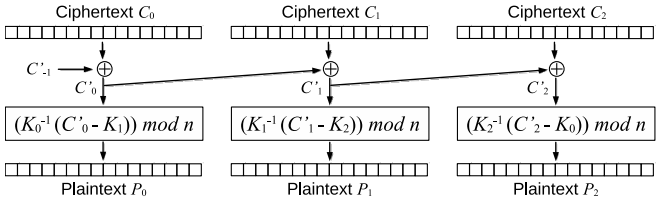

Fig. 2: Augmented Hill Cipher における復号化

B. 安全性の分析

暗号化時、AHC は初めに $(K_{0 \bmod 3} P_0 + K_{(1) \bmod 3}) \bmod n$ と $C'_{-1} (= K_2)$ の排他論理和を取り（図 1 の一番左参照）、元のデータの先頭部分である P_0 が 容易に判明されないようにしている。その後、新しく暗号化されたデータと 1 つ前に暗号化されたデータの排他論理和を取ることにより、後続データの安全性を確保している。また $+$ と $\oplus$ の演算を組み合わせることで、より安全性を高めている。

鍵長に関しては、$n = 2^\alpha$ のモジュラスで構成される $R \times R$ の正則行列の数は

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

であり [22]、これが AHC における K_i $(0 \le i \le 2)$ の各々の鍵長となる。ここで $n = 2^{16}$、$R = 8$ ならば $l \approx 2^{1022}$ となり、AHC の全鍵長は $2^{1022 \times 3} = 2^{3066}$ となる。これは鍵長として充分な長さで、また AHC は n, R の値や K の数を

増やすことにより、さらに鍵長を伸ばすことも可能である。これらにより、第 II 章と [2] の第 6 章に記述されてる通り、AHC は一般的な攻撃に対しての耐性を保持しているものと考えられる。

とは言え、将来 AHC に脆弱性が見つかることは大いにあり得る。現在我々のアルゴリズムは AHC をベースにしているが、AHC に脆弱性が見つかれば他の改良型 HC へ移行する。本稿では AHC は現時点では安全だと仮定し、改良型 HC の一例として用いる。

IV. Hill Cipher + ネットワークコーディング

この章では本法である *HNC* (= Hill Cipher + Network Coding) の詳細について以下の表記を用いて解説する。

$\boldsymbol{R}$	行列の階数
$\boldsymbol{n}$	各データのサイズまたはガロア体の大きさ (= 2^{16} または 2^8)
$\boldsymbol{M}$	SIMD によって決定される処理されるデータ数 (= 32)
$\boldsymbol{P}$	p（平文）の $R \times M$ 行列型
$\boldsymbol{C}$	c（暗号文）の $R \times M$ 行列型
$\boldsymbol{K}$	$R \times R$ な共有鍵行列
$\boldsymbol{K_\beta}$	$R \times M$ な共有鍵行列
$\boldsymbol{K_{C'_{-1}}}$	$R \times M$ な共有鍵行列

A. *Augmented Hill Cipher* の実装

暗号化と復号化の実処理速度を最大限にするために、我々は実装において AHC に若干手を加える必要があった。まず我々は $K_{i \bmod 3} P_i$ といった行列の積にガロア体（GF）での演算を用いるが、検証時に実際の演算速度は、使用するガロア体演算ライブラリーに大きく左右することが分かった。そのため、我々は以前に開発した gf-nishida-16 [14] に新たに AVX や NEON といった SIMD を使用するアルゴリズムを加え、$GF(2^{16})$、$GF(2^8)$ における現時点で知りうる限り最も高速な領域積演算を実現し、それを採用した。ガロア体の大きさについては、$GF(2^{32})$, $GF(2^{64})$, ... と指数関数的に大きくなるほど共有鍵長も指数関数的に長くなりより安全度を増すが、多くの GF ライブラリーにおいて $GF(2^\alpha)$ $(\alpha = 8, 16, 32, 64, \cdots)$ のデータを処理するためにかかる計算量は $O(\alpha)$ であり、例えば $GF(2^{32})$ での計算量は $GF(2^{16})$ の約倍となる。そのため $\alpha \geq 32$ ではコストが高すぎると判断し、$\alpha = 8, 16$ を用い、代わりに新たなる共有鍵である 3 つの K_β と $K_{C'_{-1}}$ を導入することとした。

AVX, SSE や NEON といった SIMD は基本的に 512, 256 または 128bit 単位でデータを処理する。A が $R \times R$ で B が $R \times M$ のような行列同士の積 AB に gf-nishida-16 を使用する場合、B の各行のデータの合計サイズは $GF(2^{16})$ では 512bit、$GF(2^8)$ では 256bit である必要がある。ここで $512/16 = 256/8 = 32$ であるため、$M = 32$ となり、結果的に平文行列 P と暗号文行列 C のサイズは $R \times 32$ となる。これにより次章に記すように、AHC の暗号化・復号化アルゴリズムに変更を加える必要があった。

B. 冗長性のない *HNC* のアルゴリズム

AHC の暗号化では $K_{i \bmod 3} P_i$ と $K_{(i+1) \bmod 3}$ の両者はともに $R \times R$ の行列であった（アルゴリズム 1 の 3 行目参照）。しかし冗長性のない HNC (HNCn) では $K_{i \bmod 3} P_i$ は $R \times 32$ であるのに対して、$K_{(i+1) \bmod 3}$ は $R \times R$ であるため、行列の大きさに差異が生じる。そのため我々は新しい $R \times 32$ な鍵行列 $K_{\beta 0}, K_{\beta 1}, K_{\beta 2}$ と $K_{C'_{-1}}$ を導入する。これにより HNCn の暗号化アルゴリズムはアルゴリズム 3 と図 3 のようになり、復号化アルゴリズムはアルゴリズム 4 と図 4 のようになる。なおこれら新しい鍵行列は鍵長をさらに長くするため、HNC の安全性向上につながる。

Algorithm 3 冗長性のない HNC における暗号化

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**

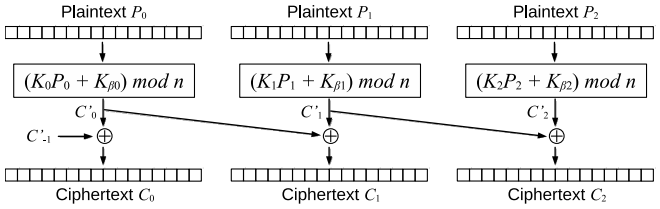

Fig. 3: 冗長性のない HNC における暗号化

Algorithm 4 冗長性のない HNC における復号化

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**

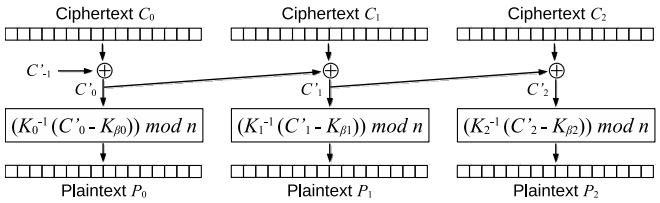

Fig. 4: 冗長性のない HNC における復号化

C. 安全性の分析

まず、HNCn は新しく追加された鍵行列（K_βs, $K_{C'_{-1}}$）と拡張された行列の大きさを除けば、AHC と同じアルゴリズムを使用する。そのためアルゴリズムの観点から、HNCn は AHC 以上に安全だと言える。次に HNC の鍵長を求める。K_0, K_1, K_2 は $R \times R$ 行列であるが、正則行列でなくてはいけない。$GF(2^\alpha)$ な要素で構成される正則な $R \times R$ 行列の数は

$$l = \prod_{k=0}^{R-1} (2^{\alpha R} - 2^{\alpha k}) \tag{4}$$

である [23]。従って $n = 2^{16}, 2^8$ と $R = 4, 6$ に対する K_0, K_1, K_2 の各々の鍵長は表 I の 2 行目のようになる。また

	$n=2^{16}$ $R=4$	$n=2^{16}$ $R=6$	$n=2^{8}$ $R=4$	$n=2^{8}$ $R=6$
各 K	255.99997	575.99997	127.994	287.994
各 K_β	2048	3072	1024	1536
$K_{C'_{-1}}$	2048	3072	1024	1536
合計	8959.99991	14015.99991	4479.982	7007.982

TABLE I: HNC における鍵長 (bit)

K_β の各々と $K_{C'_{-1}}$ は 2^{16} の要素より成る $R \times 32$ 行列であり、さらに制限はないためそれらの鍵長は表 I の 3 と 4 行目のようになる。HNC における鍵の全長は

$$\{\prod_{i=0}^{2}(\text{Key len of } K_i)(\text{Key len of } K_{\beta i})\}(\text{Key len of } K_{C'_{-1}}), \quad (5)$$

であるため、 $n = 2^{16}, 2^8$, $R = 4, 6$ における HNC の鍵長は表 I の最後の行のようになる。これらより、AHC が安全であれば HNCn は安全だと言える。

D. ランダムネットワークコーディングの実装

この章ではどのようにして RNC を HNC に組み込み、冗長性を持った HNC (HNCr) を作り出すかを解説する。第 I 章で述べたように、鍵行列に新たな行を足すことによって通信に冗長性がもたらされ、パケットロスに耐性ができる。これが RNC の原理であるが、これを前章で説明した HNC に導入する。本章で用いる表記は以下の通りである。なお冗長度は 1 とする。

$\hat{\boldsymbol{K}}$ $(R+1) \times R$ 共有鍵行列
$\hat{\boldsymbol{K}}_{\boldsymbol{\beta}}$ $(R+1) \times 32$ 共有鍵行列
$\hat{\boldsymbol{K}}_{\hat{\boldsymbol{C}}'_{-1}}$ $(R+1) \times 32$ 共有鍵行列
$\hat{\boldsymbol{C}}$ c の $(R+1) \times 32$ 行列型
$\boldsymbol{A^{(r)}}$ A から r 番目の行を取り除いた行列
$\boldsymbol{L}$ パケットロス率 $(0 \leq L \leq 1)$

アルゴリズム 3, 4 はそれぞれアルゴリズム 5, 6 に置き換えられる。

Algorithm 5 冗長性を持った HNC における暗号化

$\hat{C}'_{-1} = \hat{K}_{\hat{C}'_{-1}}$
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

送信者は $\hat{C}_i$ の各行を別々のパケットで送信する。$\hat{C}_i$ の r 番目 $(0 \leq r \leq R)$ の行のデータをパケット r で送信するとすると、たとえ $R+1$ 個あるパケットの 1 つが失われたとしても、受信者は残り R 個のパケットから平文 P_i を復号することができる。パケットロス率 $L = 0$ の時は冗長性のためスループットは $1/(R+1)$ 減少する。しかしそれ以外の際はパケットロスによるパケット再送回数を減らし、スループットを大きく改善する。冗長性がない場合、パケットロスが起こる度に送信者は失われたパケットを再送しなくてはならない。冗長度が 1 でパケットロスが起こった場合、$R+1$ 個ある同じ $\hat{C}_i$ 内の行のデータのうち、別の行のデータを含むパケットも失われた際にのみ送信者は再送を要求される。その確率は

$$1 - (1-L)^R \quad (6)$$

で、再送率は $(1-L)^R$ 減少することになる。これは例えば $L = 0.01, R = 4$ で $1 - (1-0.01)^4 = 0.04$ となり、パケットロス率が 1%ならば、失われたパケットの 4%にのみ再送を要求すればいいことを意味する。このように冗長性の実装は再送回数を激減させ、スループットの向上に寄与することが分かる。実際に $L = 0.01, R = 4$ でスループットは、第 V-C 章に見られるように 200%増加している。

安全性に関しては、HNCr をクラックする方法が存在すると仮定すると、HNCn は HNCr の特殊なケースとして捉えることができるので（例: 全ての K の R 番目の行が $R-1$ 番目の行と同一)、HNCn も同様に簡単にクラックすることができる。つまり HNCr の安全性は HNCn 以上であり、AHC が安全であれば HNCr も安全であると言える。

Algorithm 6 冗長性を持った HNC における復号化

$\hat{C}'_{-1} = \hat{K}_{\hat{C}'_{-1}}$
for $i \leftarrow 0$ to $N-1$ **do**
  $\hat{C}_i$ において、失われた行 $r \in [0, R]$ を特定する。
  if パケットロスが見当たらない **then**
    $r \leftarrow$ 任意の値 $\in [0, R]$.
    $PacketLoss \leftarrow False$
  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**
    アルゴリズム 5 を用いて、失われた $\hat{C}'_i$ の r 番目の行を P_i から作る
  end if
end for

V. 測定結果

A. TCP におけるスループット

この章ではローカルホスト上の TCP 通信におけるスループットと処理速度を OpenSSL（AES-NI アクセラレーション付、TLS_AES_256_GCM_SHA384 アルゴリズム）と HNCn の $n = 2^{16}, 2^8$, $R = 4, 6$ とで比較する。CPU には AMD Ryzen 7 5800X と Apple M1 を用い、暗号化または復号化と通信処理はすべて同一のシングルスレッドで行っている。

結果は図 5 に見られるように、Apple M1 での $n = 2^{16}$, $R = 6$ (HNCn-16bit-6) 以外は OpenSSL よりも HNCn の方が高いスループットを記録している。特に Ryzen 7 での HNCn-8bit-4 と HNCn-16bit-4 のスループットは、それぞれ OpenSSL の 3 倍と 2.5 倍と特に高くなっている。

行列の積のコストが HNCn の $R = 4$ と 6 での結果の差として現れている。アルゴリズム 3、4 における行列の積の計算量は $O(R^3)$ で、$R = 6$ は明らかに $R = 4$ よりもスループットが低くなっている。表 I にあるように、$R = 4$（特

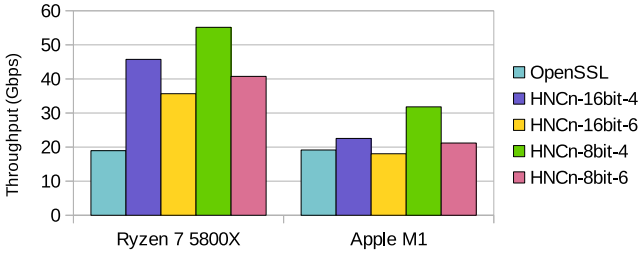

Fig. 5: ローカルホスト上での TCP 通信における OpenSSL と HNCn のスループット値の比較。HNCn の殆どが OpenSSL より高いスループット値を示している。

	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

TABLE II: ローカルホスト上での TCP 通信における HNCn の OpenSSL に対する処理速度（$S_{\text{HNCn}}/S_{\text{OpenSSL}}$）の比較。HNCn の殆どが OpenSSL よりも速いが、Ryzen 7 と M1 の間で差がある。

に $n = 2^{16}$）でも鍵長は充分な長さが保持されているので、HNC には $R = 4$ の方が望ましいと考えられる。

アルゴリズムによって暗号化や復号化の処理速度が変わるのはもちろんであるが、アルゴリズムによってもたらされるオーバーヘッドの量も異なってくる。ここで対象のアルゴリズム（Algo）における処理速度を測る基準値として以下のメトリックを用いる。

$$S_{\text{Algo}} = 1/\left(\frac{1}{TP_{\text{Algo}}} - \frac{1}{TP_{\text{Non-crypto}}}\right), \tag{7}$$

ここで TP はスループット値を表し、平文でのスループット値である $TP_{\text{Non-crypto}}$ は Ryzen 7 で 126.55Gbps、M1 で 105.58Gbps である。表 II はその $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ 値を表している。OpenSSL に対して HNCn は Ryzen 7 で 2.2 から 4.3 倍の処理速度を示しており、M1 では 0.9 から 1.9 倍の処理速度を示している。$R = 4$ の HNCn は OpenSSL よりも高速にデータを処理することが分かるが、M1 での値は Ryzen 7 での値よりも全般的に低くなっている。これは ARM NEON のレジスターのサイズによるものと考えられ、AVX の 256bit に対し NEON は半分の 128bi しかないことが原因だと思われる。実際に同じ 128bit のレジスターを持つ SSE は、AVX よりも処理速度が 30%落ちることが我々の実験で判明している。高速な暗号化・復号化のためにはレジスターのサイズが大きい SIMD を用いる方が望ましい。

B. *QUIC* におけるスループット

QUIC は暗号化された多重化接続と低遅延などを実現するプロトコルで、従来の HTTPS のトラフィックにおける問題点を様々な点で改善している [24]。QUIC ではデータが暗号化されることが前提となっているため、HNC を実装するには最適なプラットフォームであると考えられる。QUIC を実装するオープンソースプロジェクトは幾つか存在するが、Microsoft が主導する MsQuic [25] は最もコードのコミットが多いプロジェクトの一つで、Linux と MacOS では OpenSSL を暗号化アルゴリズムとして使用している。我々は HNCn を MsQuic に組み込み、前章と同じ条件でスループットと処理速度を測定した。

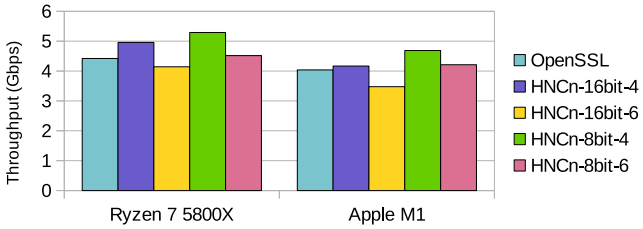

Fig. 6: ローカルホスト上での QUIC 通信における OpenSSL と HNCn のスループット値の比較。HNCn-16bit-6 を除く HNCn が OpenSSL より高いスループット値を示している。

	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

TABLE III: ローカルホスト上での QUIC 通信における HNCn の OpenSSL に対する処理速度（$S_{\text{HNCn}}/S_{\text{OpenSSL}}$）の比較。M1 の HNCn-8bit-4 で異常に高い値が見られる。

結果は図 6 と表 III に示される通りであり、$TP_{\text{Non-crypto}}$ は Ryzen 7 で 6.31Gbps、M1 で 4.80Gbps である。 HNCn-16bit-6 を除く HNCn が OpenSSL よりも高いスループット値を示しているが、TCP に比べると差は小さい。処理速度に関しては、$S_{\text{HNCn}}/S_{\text{OpenSSL}}$ の値が Ryzen 7 で TCP ほど高くなっていないが、M1 では HNCn-16bit-6 を除いて高くなっている。

注意して頂きたいのは、我々の実装は現時点でパケット長を有効に利用しておらず、QUIC においてあまり効率のいいデータ処理がなされていないという点である。QUIC は UDP 上に構築されたプロトコルであるため、各パケットとそれに含まれるペイロードのサイズは MTU のサイズによって決められる。1500byte の MTU に対して、QUIC における一般的なペイロード長は 1450byte 以上であり、MsQuic はそのペイロードになるべく沢山のデータを詰め込むように設計されている。しかしながら HNC はブロック暗号アルゴリズムであり、HNCn において最も理想的なペイロード長は、$n = 2^{16}$ で $64 \times R \times N$（N は自然数）、$n = 2^8$ で $32 \times R \times N$ である。MTU 長 1500byte、$R = 4$ におけるそれらの上限値は $n = 2^{16}$ で $64 \times 4 \times 5 = 1280$byte、$n = 2^8$ で $32 \times 4 \times 8 = 1408$byte であり、$R = 6$ ではそれぞれ $64 \times 6 \times 3 = 1152$byte、$32 \times 6 \times 7 = 1344$byte となる。このように HNCn における理想的なペイロード長は一般的な QUIC でのペイロード長よりも短く、これが TCP の際と比べも非効率になる原因となり、Ryzen 7 での測定結果で低い $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ 値として現れているものと考えられる。現時点でどうして HNCn-8bit-4 を除く M1 での $S_{\text{HNCn}}/S_{\text{OpenSSL}}$ 値が TCP 時よりも高いのか判明していない。これについての調査と、HNCn でのペイロード長の最適化が今後の課題である。

C. パケットロスが起こる通信でのスループット

パケットロスが見られない通信では冗長付きの HNC (HNCr) は必要ない。しかしパケットロスがわずかでも定常的に観測されるような環境であれば、スループット改善のため、HNCr に移行すべきである。我々は HNCr を扱えるように MsQuic に手を加え、パケットロスが起こる通信環境をエミュレートし、MsQuic のデフォルト暗号化アルゴリズム

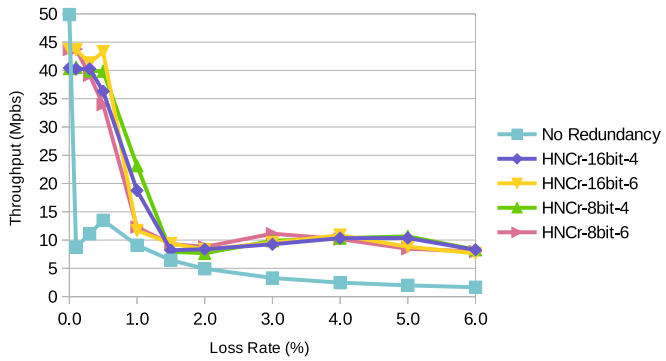

Fig. 7: パケットロスが起こる QUIC 通信での冗長付き HNC (HNCr) と OpenSSL (No Redundancy) でのスループット値の比較。HNCr が OpenSSL よりも全範囲で高いスループット値を示している。

である OpenSSL とのスループットを比較した。HNCr の冗長度は 1 である。パケットロスのエミュレーションは、パケットロス率 L に従って意図的に MsQuic 内部で受け取ったパケットが落とされるように処理し、また Linux の tc コマンドを用いて帯域を 60Mbps に、ネットワークの遅延を 50ms に設定した。

結果は図 7 および以下の通りである。

1) HNCr のスループットは全域に渡って OpenSSL よりも高くなっている。全体での平均 $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$ は 3.08 である。
2) HNCr は $L = 0.1$–0.5% で OpenSSL よりも遥かに高いスループットを記録している。そこでの $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$ は 2.69 から 5.00 である。しかしながら $L = 1.5\%$ 付近では HNCr と OpenSSL のスループット値が接近している。
3) HNCr 同士での違いはさほどないが、$L = 1.0\%$ で $R = 4$ と 6 間で若干差がある。
4) OpenSSL は $L = 0$ と 0.1% の間でスループットに大きな落ち込みが見られる。しかしその後 $L = 0.5\%$ までは若干値が回復し、それ以降は緩やかに下がっている。
5) HNCr-*-4 と HNCr-*-6 のスループットはそれぞれ $L = 0.5$–1.5% と $L = 0.5$–1.0% で著しく落ちている が、その後は安定している。

HNCr の長所は 1) で見られるが、2) は $TP_{\text{HNCr}}/TP_{\text{OpenSSL}}$s が一定でなく L によって大きく変わることを意味している。4) と 5) の原因については現時点では分かっていないが、おそらく MsQuic の設計と実装によるものだと思われる。

QUIC はかつてグループ内におけるパケットロスを補完するために前方誤り訂正（FEC）をサポートしていたが、効果が限定的であったため削除された。また FEC はわずかに動画再生の遅延と再バッファリング率を増加させることが報告されており、使用帯域の増加に見合った利点が得られないと結論付けられている [24]。我々は HNCr がもたらす全ての影響を測定したわけではないが、HNCr はこの FEC の代替となる可能性を持っている。

VI. 今後の課題

HNC が有線よりも無線通信に適していることは間違いない。我々はその中でも HNC は衛星通信と非常に親和性が高いと考えており、衛星通信が求める「盗聴と妨害電波に強く、安全で安定した通信」という条件を満たすことができるものと考えている。衛星通信が $Ch_0, Ch_1, ..., Ch_R$ の $R+1$ 個のチャンネルを持っているものとする。ここで第 IV-D 章で述べたパケット r を $Ch_{r \mod (R+1)}$ で送信するとすると、もしチャンネルの 1 つが妨害されたとしても、受信者は残りのチャンネルで送られてきたパケットより元のデータを復号することができる。またたとえ全チャンネルのパケットが盗聴されたとしても、HNC により暗号化されたデータを復号することはほぼ不可能である。このようにして HNC は安全で安定した衛星通信を実現する可能性があり、今後これについての研究を進めていくことを課題としている。また第 V-B 章で述べた通り、我々の MsQuic での実装は定められたパケット長に対して最適なデータ送信を実現できていない。これに対する最適化を検討する必要がある。AHC を始めとした改良型 HC の安全性にも注視しなくてはいけない。より安全な暗号化アルゴリズムを模索することは今後も不可欠であると思われる。

VII. 結論

本稿ではデータを安全かつ高速に暗号化・復号化し、パケットロスにも耐性のあるアルゴリズムを紹介した。我々の測定結果では、冗長性のない HNC は AES-256 よりも高速に暗号化・復号化を行い、冗長性を持った HNC はパケットロスが起こる環境で従来よりもスループットを大きく改善することが示された。我々は本方法が円滑で信頼性の高い無線通信の発展に貢献し、より安全で快適なインターネット通信をもたらすものと信じている。

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 web site," https://github.com/scopedog/gf-nishida-16/.

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

[16] 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.

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

[18] 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.

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

[20] Mohamed Amine Brahimi and Fatiha Merazka, "Data confidentiality-preserving schemes for random linear network coding-capable networks," *Journal of Information Security and Applications*, vol. 66, pp. 103136, 2022.

[21] 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.

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

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

[24] 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.

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