圧縮センシングによる準同型暗号文の通信量削減

泉 湖雪　(指導教員：小口 正人)

1　はじめに

スマートフォンなどの IoT デバイスが普及したことで，クラウドなどのサーバに個人情報を含むデータが蓄積され，データ分析に利用されるようになった．しかし，多くの個人情報を含むため，外部からの攻撃などによるデータの漏洩に備える必要がある．そこで，暗号文同士の加算や乗算が可能な準同型暗号を用いることで，盗聴のリスクを減らせるだけでなく，暗号状態のままデータ分析を行うことが可能となる．しかし，準同型暗号は暗号文のサイズが大きく通信量が大きくなってしまうという問題点がある．通信量を削減する方法の一つとして Key Switching による手法 [1] が挙げられる．本研究では，圧縮センシングを応用した通信量の削減を提案する．

圧縮センシングは，スパース性を利用することにより，少ないサンプル数からもとのデータを復元する技術である．圧縮行列 A を掛けることで低次元データを得て，得られた低次元データと圧縮行列 A から推定することで，元の多次元データを復元する．この技術を用いて圧縮されたスパースな多次元データを準同型暗号化してクライアントからサーバに送信することで，クライアントとサーバ間の通信量の削減が期待できる．

提案手法では，圧縮センシングによる圧縮と準同型暗号ライブラリの Microsoft SEAL[1]を使った平文の準同型暗号化をクライアント側で行い，圧縮センシングの復元をサーバ側で行う．代表的な再構成アルゴリズムとして Orthogonal Matching Pursuit(以下 OMP) があるが，OMP で復元する際には比較演算を行う．暗号化された状態では比較演算が困難であるため，比較演算を使用しないような復元アルゴリズムを用いる必要がある．そこで，今回は一般逆行列による L_2 ノルムの最小化によって圧縮前のデータを推定する．実験では，乱数により生成したスパースな多次元データを用いて，Baseline と提案手法で，クライアントからサーバに送信する準同型暗号文のサイズ，サーバでの圧縮センシングによる復元結果の精度，実行時間の比較を行う．

実験の結果，圧縮率を 50%に設定した場合は提案手法は圧縮せずに送信した場合に比べて 50%通信量を削減できることを示した．このとき，復元後のベクトル $\hat{\boldsymbol{x}}$ の RMSE はおよそ 0.15 となった．

2　提案手法

2.1　概要

ここでは単純に，スパースな多次元データをクライアントで準同型暗号化して，圧縮せずにサーバに送信する方法を Baseline とする．提案手法は図 1 のように，

[1]https://github.com/microsoft/SEAL

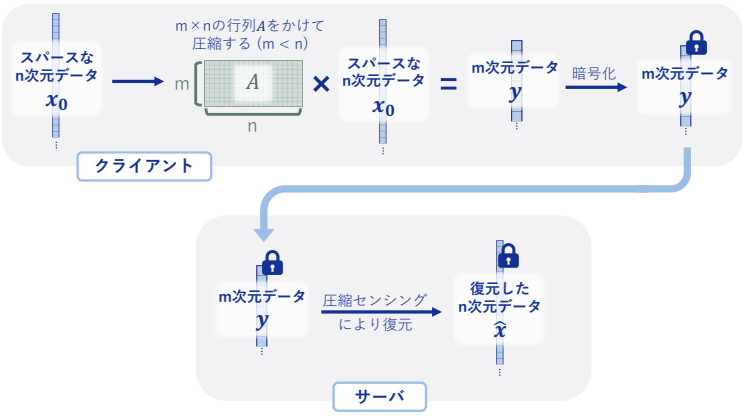


図 1: 提案手法．クライアント側は圧縮センシングによる平文の圧縮と準同型暗号化を行い，サーバ側は圧縮センシングの復元を行う．

以下の手順で行う．ここで m, n は $m < n$ であるものとする．

クライアント

1. スパースな n 次元データ $\boldsymbol{x_0}$ に，m 行 n 列の圧縮行列 A をかけて圧縮する
   $\boldsymbol{y} = A\boldsymbol{x_0}$
2. 得られた m 次元データ $\boldsymbol{y}$ を暗号化する
   $Encrypt(\boldsymbol{y})$
3. 暗号化状態の $\boldsymbol{y}$ をサーバに送信する

サーバ

4. 暗号化状態の $\boldsymbol{y}$ を受け取る
5. 再構成アルゴリズムにより復元した n 次元データ $\hat{\boldsymbol{x}}$ を得る
   $\hat{\boldsymbol{x}} = Reconstruction(Encrypt(\boldsymbol{y}))$

Baseline に比べて，クライアントからサーバに送信する暗号文のサイズは n 次元から m 次元に減少するため，クライアントとサーバ間の通信量の削減が可能となる．

2.2　提案手法で使用した再構成アルゴリズム

代表的な再構成アルゴリズムの 1 つである OMP のアルゴリズム [2] では，残差と最も内積の絶対値が大きい A の列ベクトルがどれかを見つける際に比較演算が必要となる．しかし，準同型暗号での比較演算は非常に難しい．そのため提案手法では，圧縮センシングの再構成アルゴリズム $Reconstruction(\cdot)$ において，一般逆行列によって最小ノルム解，つまり L_2 ノルムが最小となる推定結果を得る．すなわち $encrypt(\hat{\boldsymbol{x}}) = (A^TA)^{-1}A^T \times encrypt(\boldsymbol{y})$ のように一般逆行列と暗号化された低次元行列との積を求める．

表 1: 実験環境

(a) マシン性能

	Raspberry Pi	サーバ
OS	Raspbian 11	Ubuntu 22.04.1
CPU	ARM Cortex-A72	Intel(R) Xeon(R) Gold 5115 CPU
コア数	4	10
プロセッサ速度	600MHz	1GHz
RAM	4GB	192GB

(b) 圧縮センシングに関するパラメータ

k	元のベクトル $\boldsymbol{x_0}$ のスパース値	$n \times 0.1$
n	元のベクトル $\boldsymbol{x_0}$ の要素数	1024
m	圧縮後のベクトル $\boldsymbol{y}$ の要素数	512

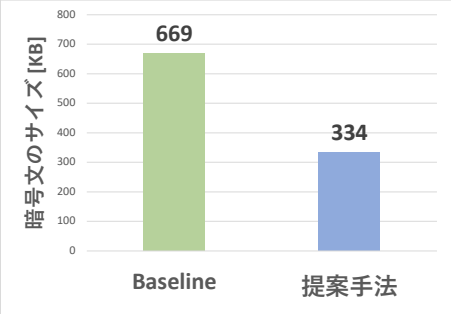


図 2: クライアントとサーバ間の通信量

3 実験

3.1 概要

Baseline と提案手法で，クライアントからサーバに送信する準同型暗号文のサイズ比較，圧縮センシングの再構成アルゴリズムによるサーバでの復元結果の精度測定，実行時間の測定を行った．その際，表 1(a) に示したマシン，表 1(b) のパラメータを使用した．精度の評価方法には二乗平均平方根誤差 (以下 RMSE: Root Mean Squared Error) を用いた．

3.2 実験結果

3.2.1 通信量

図 2 から，Baseline に比べ，提案手法では約半分の通信量に抑えられていることが分かる．これは，表 1(b) の n と m の比率に対応していることが分かる．

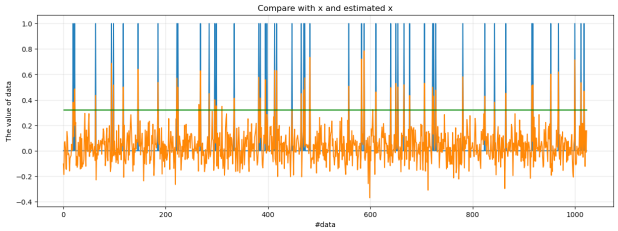
3.2.2 精度

提案手法における RMSE はおおよそ 0.15 となった．ここで，RMSE の値は小さいほど誤差の小さいモデルである．

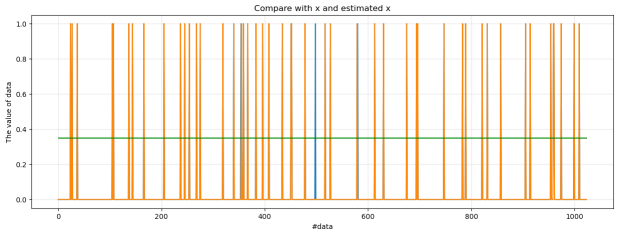
また，図 3(a) に元のベクトル $\boldsymbol{x_0}$ と再構成アルゴリズムにより復元した n 次元データ $\hat{\boldsymbol{x}}$ の比較を，図 3(b) に閾値により 0,1 の 2 値化を行った結果を示す．図 3(a), 図 3(b) ともに，青が元のベクトル $\boldsymbol{x_0}$，オレンジが復元後のベクトル $\hat{\boldsymbol{x}}$ を表し，緑は今回設定した閾値 0.32 を示す．このとき，1 である部分は 91%一致していた．

3.2.3 実行時間

Baseline と提案手法における実行時間を図 4 に示す．提案手法では，クライアントからサーバに送信する暗号文のサイズの減少に伴い，暗号化にかかる時間は減っているが，クライアントで圧縮させる際の行列計算や，特にサーバで復元する際の再構成アルゴリズムの計算に時間がかかってしまっていることが分かる．



(a) 元のベクトル $\boldsymbol{x_0}$ と復元後のベクトル $\hat{\boldsymbol{x}}$ の比較



(b) 図 3(a) の $\hat{\boldsymbol{x}}$ を閾値により 2 値化した結果

図 3: 精度に関する実験結果

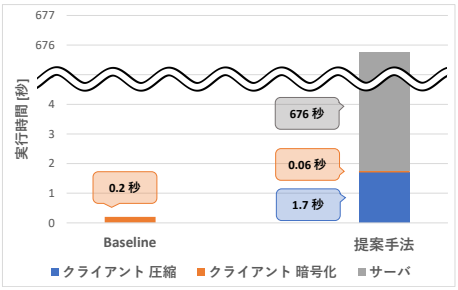


図 4: 実行時間

4 まとめと今後の課題

本研究では，圧縮センシングによるスパースな多次元データのサーバ送信時における通信量削減を提案した．実験の結果，通信量を約半分に減らすことができ，復元後のベクトル $\hat{\boldsymbol{x}}$ の RMSE はおよそ 0.15 であった．

今後は，スパースな多次元データの実数への対応や，他の再構成アルゴリズムの使用した精度の向上，実行時間の改善に取り組みたい．

参考文献

[1] Marin Matsumoto and Masato Oguchi. IoT Device Friendly Leveled Homomorphic Encryption Protocols. In *2022 IEEE International Conferences on Smart Data (SmartData)*, pp. 525–532. IEEE, 2022.

[2] 圧縮センシング: 疎情報の再構成とそのアルゴリズム． https://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1803-03.pdf.