準同型暗号による統計解析のアウトソーシング I: 記述等計量

陸 文杰 †　　川崎 将平 †　　佐久間 淳 ‡

† 筑波大学 大学院 システム情報工学研究科
305-8577 茨城県つくば市天王台 1 丁目 1-1
{kawasaki, riku}@mdl.cs.tsukuba.ac.jp
‡ 筑波大学 大学院 システム情報工学研究科 / JST CREST
305-8577 茨城県つくば市天王台 1 丁目 1-1
jun@cs.tsukuba.ac.jp

あらまし　近年クラウドへの計算タスクの委託が増加しつつある．Gentry らの完全準同型暗号 (FHE) に続き, 安全かつ有用な計算タスクのアウトソースに関する研究が盛んにおこなわれている．本稿の目的は大規模データを用いる統計解析において, 暗号理論的に安全かつ実用的な枠組み”CODA” を提案することである．CODA では平均, 共分散, 最頻値と分位点等の重要な統計量の計算を安全にアウトソースすることができる．また計算効率化のためのプリミティブとして, 我々は非対話的な greater-than プロトコルを提案する．このシステムの有用性を示すため, 我々は 14 属性の 30k レコードを含む Adult データセット上で CODA を実行し, 実験によりの統計量を実用的な時間で計算できることを示した.

Cryptographically-secure Outsorcing of statistical Data Analysis I: Descriptive Statistics

Wen-jie Lu†　　Shohei Kawasaki†　　Jun Sakuma‡

†Graduate School of SIE, University of Tsukuba.
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8577, JAPAN
{kawasaki, riku}@mdl.cs.tsukuba.ac.jp
‡Graduate School of SIE, University of Tsukuba/ JST CREST
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8577, JAPAN
jun@cs.tsukuba.ac.jp

Abstract　In recent years, there has been a growing trend towards outsourcing of computational tasks with the development of cloud services. The Gentry ’s pioneering work of fully homomorphic encryption (FHE) and successive works have opened a new vista for secure and practical outsourcing of computational tasks. The main aim of this study is to build a practical framework for cryptographically secure outsourcing of statistical data analysis (CODA) that works with large-scale data sets. To improve the computational efficiency of CODA, we propose a building block: a noninteractive greater-than protocol. CODA supports secure outsourcing of major descriptive statistics, including mean, covariance, mode and quantile. To demonstrate the effectness of our system, we ran experiments with a real data set that contains about 30k records of 14 (6 numerical) attributes (Adult). The results showed that all the descriptive statistics supported by CODA were calculated within a practical computation time.

1 はじめに

準同型暗号はプラベートなデータの処理する際にプライバシを犠牲にせず, 処理を行うことができる重要な技術と考えられている [1]. Gentry らのブレークスルー [4] に続き, いくつかの研究 [10, 2] は計算タスクの安全なアウトソーシングにおいて, 新たな展望を切り開いている. クラウドに計算タスクをアウトソースすると, 安価な計算資源により利用者のコストを削減することができるが, データの安全性の保障は難しくなる. そのため, クラウド環境における秘密計算は準同型暗号の一つの有望な応用である.

本稿では完全準同型暗号を用いた暗号理論的に安全に統計量をアウトソーシングする枠組み”CODA”(Crtptographically-secure Outsorcing of statistical Data Analysis) を提案する. 我々の枠組みは完全準同型暗号の実装 [9] を利用し, 平均, 分散, 共分散, 最頻値と分位点 (50 分位点は中央値), rank, max, min などほぼ全ての記述統計量をサポートする. 本稿では紙数の都合上 rank, max, min は省略する.

我々の枠組みでは, データ提供者, クラウド, 分析者の三パーティが登場する. 分析者はデータ提供者からデータを収集し, 統計量を計算したいパーティである. データ提供者はプラベートなデータを提供する個人や組織などのパーティである. クラウドは計算資源とストレージを提供するパーティである. ここで, 我々はクラウドは単一なパーティであることを仮定する.

CODA は様々なアウトソーシングのモデルにも適応することができる. 例えば一人のデータ提供者からなるモデルや複数の組織内でデータを水平にあるいは垂直にシェアするモデルにも適応することができる. 本稿では議論を簡単にするため, 複数のデータ提供者からデータを水平的に共有するモデルのみを扱う.

CODA は以下の三つの段階から構成される.

1. upload フェーズ: データ提供者から暗号文をアップロードする段階. データ提供者は自らのプライベートなデータを暗号化し, クラウドに送信する. その後データ提供者は offline となる.
2. evaluation フェーズ: クラウドが統計解析計算を行う段階. 分析者から統計解析のリクエストを受ける前に実行可能な preprocessing と, リクエストを受けたあとに実行する postprocessing が含まれる.
3. dowoload フェーズ: 分析者がクラウドから結果の暗号文をダウンロードし，復号する段階.

CODA はなるべく非対話的なプロトコルで構成されている. ここで, 非対話的とは evaluation フェーズの計算が全てクラウドで完結するプロトコルであることを示す. 分析者は秘密鍵・公開鍵ペアを保持し, データ提供者は公開鍵を保持するものとする. 分析者は秘密鍵・公開鍵ペアを保持し, データ提供者は公開鍵を保持するものとする. 鍵生成とその配布の方式はここでは特に議論しないが, 分析者が鍵ペアを生成し, 公開鍵をデータ提供者に配布する典型的な運用の場合には, man-in-the-middle 攻撃やその他の良く知られた脅威を回避するために, 安全な通信路や PKI を利用することが前提となる. **関連研究:** 近年プライバシーを保護した統計解析において多くの研究が報告されている. この分野では, 主に garbled circuit, 秘密分散に基づいた秘密計算, 準同型暗号の三種類の研究がある.

秘密分析に基づいたマルチパーティーの秘密計算についての研究も盛んである. 統計解析における秘密分散は三パーティ以上かつお互いに共謀しないサーバーが必要である [**?**, **?**]. そのため, アウトソーシングにおいて, 秘密分散の適切なシナリオが制限される. 我々が提案する CODA は単一なサーバーの存在を仮定するため, CODA は Amazon の EC2 や Microsoft の Azure などの単一の商用パブリッククラウドサービス上で動作できる.

近年の完全準同型暗号の発展によって, 完全準同型暗号を用いて統計解析を安全に外部に委託する研究がいくつか報告されている. Naehrig らは完全準同型暗号におけるメッセージエンコーディング方法を提案した [8]. このエンコーディングでは準同型性を保ちつつ, 複数のビットあるいは大きな整数を一つの暗号文に暗号化することができる. 統計解析への応用として, 彼ら

は平均, 標準偏差とロジスティック回帰を提案したが, モデル学習の計算は議論されていない.

貢献: 我々は暗号理論的に安全な統計解析のアウトソーシングの枠組みを提案した. 我々の枠組みは典型的な記述統計量をサポートする. 主な統計解析は行列式や不等式の計算で表現ことに着目し, 我々は暗号文上の行列のプリミティブと暗号化された数値の大小比較プロトコル (greater-than プロトコル) を設計した. greater-than プロトコルは二つの暗号文を入力として, 非対話的にこの二つ暗号文の大小関係を計算する. このプロトコルを利用することにより, CODA は最頻値や分位点, rank, max, min などの統計量を非会話的に計算することができる.

また, 我々の提案法では多項式の中国剰余定理 (CRT) に基づくパッキング手法 [11] を用いて, 整数のベクトルを一つの暗号文に暗号化し計算の効率化をしている. 実験では, 実際に記述統計量のアウトソーシングの枠組みのプログラムを実行し計算時間を計測した. 実験結果から, CODA は主要な記述等計量の多くを効率的かつ非対話的ににアウトソースできることを示した.

2 (Leveled) 完全準同型暗号

完全準同型暗号 (FHE) は秘密鍵を知ることなく, 暗号文上の任意の演算を可能にする暗号である. この準同型性を用いて, 秘密情報を犠牲にせずに安全に計算をアウトソースできる. 我々は Brakerski–Gentry–Vaikuntanathan の RLWE に基づいた完全準同型スキーム [2] を利用する. 実験では open-source のライブラリである HElib [9, 6] を利用する.

2.1 BGV の FHE スキーム

HElib は BGV のスキームを実装した open-source なライブラリである. このスキームは多項式環 $\mathbb{A} := \mathbb{Z}[x]/\Phi_m(x)$ 上で定義される. ここで, $\Phi_m(x)$ は m 番目の円分多項式である. 平文空間は多項式環 $\mathbb{A}_t := \mathbb{A}/t\mathbb{A}$ で定義され, この法 t は素数の冪数に設定する. BGV のスキームのパラメータは複数の法 $q_0 > q_1 > \cdots > q_L$ も含む. $q_0 > q_1 > \cdots > q_L$ は暗号文空間 $\mathbb{A}_{q_i}$ のためのパラメータであり, L は評価可能な準同型演算の最大の深さを示す. Level-i の暗号文は空間 $\mathbb{A}_{q_i} := \mathbb{A}/q_i\mathbb{A}$ の元となる. HElib で実装されている最適化方法 [2] では, 法を q_i から q_{i+1} へ置き換えると, 暗号文内のノイズを q_{i+1}/q_i の比によって減らすことができる.

パラメーター m は以下の式を満たせば κ-bit セキュリティを保つ.

$$\phi(m) > \frac{(L(\log \phi(m) + 23) - 8.5)(\kappa + 110)}{7.2}.$$

ここでは, $\phi(\cdot)$ は Euler 関数であり, L は level パラメーターである. 詳細なセキュリティ解析は Gentry らの研究 [3] に示されている.

2.2 パッキング手法

効率的な計算を達成するために, 我々はパッキング手法を利用する. Smart らは [11] において, 多項式の Chinese-Remainder-Theorem を利用して, 平文空間 $\mathbb{A}_t$ を複数の “スロット” に分割できることを示した.

円分多項式 $\Phi_m(x)$ がある法 t において, ℓ 個の規約な因子に $\Phi_m(x) = \prod_{j=1}^{\ell} F_j(x) \mod t$ のように分解することができるとする. ここで, $F_j(x)$ の次数は $\phi(m)/\ell$ であり, スロットと呼ばれる. ある整数のベクトルが与えられた際に, CRT パッキングはこのベクトルの各要素を各スロットに組み込む. これにより, 我々は一つの暗号文に整数のベクトルを埋め込むことができる. さらに, ベクトル間の加算や乗算はそれぞれ一回の準同型加算や乗算で計算することができる.

CRT における加算, 乗算を用いて, 我々は暗号文上の効率的な計算を行うことができる. CODA において, 我々は多くの部分で CRT-パッキングを利用し, 暗号文上の計算を高速化する.

3 データの表現

CODA では数値属性, カテゴリ属性と順序属性を扱う. 本節ではそれぞれのデータの表現について述べる.

3.1 ノーテーション

行列はボールドな大文字で表す (e.g.,$\boldsymbol{A}$). $\boldsymbol{a}_i^{\mathrm{T}}$ を行列 $\boldsymbol{A}$ の i 行目として表記する. ベクトル $\boldsymbol{v}$ は列ベクトルとし, 行ベクトルは転置記号を用いて表現する (e.g.,$\boldsymbol{v}^{\mathrm{T}}$). 行列の各要素は a_{ij} のように表す. $\mathbb{R}$ は多項式環を表す. また, 集合 $\mathcal{D}$ のサイズを $|\mathcal{D}|$ と表記する. メッセージ x の暗号文は $\mathrm{Enc}(x)$ と明示的に表記するか, あるいは小文字のゴシックで $\mathfrak{x}$ のように表記する. 回転のオペレーターを $\ll$ で表し, 準同型加算と乗算オペレーターをそれぞれを $\oplus$ と $\odot$ で表記する.

数値属性: 一般的に数値属性は有限な浮動小数で近似される. 従って, 適切な拡大係数を選択すれば, 数値属性は整数として扱える. 例えば, ある実数 $x \in \mathbb{R}$ と拡大係数 $M \in \mathbb{Z}$ とすると, $\tilde{x} = \lfloor Mx \rceil \in \mathbb{Z}$ と整数化できる. ここで, $\lfloor \cdot \rceil$ は入力された実数に最も近い整数を返す関数である. 以降では実数値は適切な拡大係数がかけられていることとする.

数値属性の次元数を d_n とし, ベクトル $\boldsymbol{x}_i \in \mathbb{Z}_t^{d_n}$ を i 番目のレコードの数値データとする. また, ベクトル $\boldsymbol{x}_i$ の j 番目の値は j 番目の数値属性を表す. n 個の数値データを並べた行列は $\boldsymbol{X}^{\mathrm{T}} = (\boldsymbol{x}_1 \boldsymbol{x}_2 \cdots \boldsymbol{x}_n) \in \mathbb{Z}_t^{n \times d_n}$ と表記する.

カテゴリ属性と順序属性: カテゴリ属性の値は非数量的かつ無順序な属性である. カテゴリ属性の次元を d_c とし, 各属性のドメインを $\mathcal{C}_j = \{s_1^j, s_2^j, \cdots, s_{|\mathcal{C}_j|}^j\}$, $1 \le j \le d_c$ で表す. ここで, s_k^j は j 番目の属性 $\mathcal{C}_j$ の k 番目の値を示す. したがって, d_c 個カテゴリ属性の全ドメインは直積で表記できる ($\mathcal{C} := \mathcal{C}_1 \times \cdots \times \mathcal{C}_{d_c}$).

同様に, 順序属性の次元数は d_o とし, 各順序属性のドメインを $\mathcal{O}_j$ とする. 順序属性も非数量であるが, カテゴリ属性と異なり次のように順序関係が成り立つ: $s_1^j \preceq s_2^j \preceq \cdots \preceq s_{|\mathcal{O}_j|}^j$. 全順序属性のドメインも同様に直積で表すことができる ($\mathcal{O} := \mathcal{O}_1 \times \cdots \times \mathcal{O}_{d_o}$).

$\boldsymbol{c}_i \in \mathcal{C}$ と $\boldsymbol{o}_i \in \mathcal{O}$ をそれぞれ i 番目のカテゴリデータと順序データとすると, n 人のデータ提供者から収集したカテゴリデータ, 順序データは次のように表せる.

$$\boldsymbol{C}^{\mathrm{T}} = \Big(\boldsymbol{c}_1 \boldsymbol{c}_2 \cdots \boldsymbol{c}_n\Big) \in \mathcal{C}^n$$
$$\boldsymbol{O}^{\mathrm{T}} = \Big(\boldsymbol{o}_1 \boldsymbol{o}_2 \cdots \boldsymbol{o}_n\Big) \in \mathcal{O}^n.$$

3.2 データエンコーディング

エンコーディングの選択は計算効率を左右する. そのため, 我々は提案プロトコルでは, 問題に合わせていくつかの異なるエンコーディングを用いる.

Indicator エンコーディング: Indicator エンコーディング $\mathcal{E}_{\mathrm{id}} : \mathcal{C}_j \to \mathbb{Z}_2^{|\mathcal{C}_k|}$ はあるカテゴリ属性の値 $s_k^j \in \mathcal{C}_j$ を入力として, k 番目の要素が 1 であり他の全ての要素が 0 であるベクトルを出力する. この indicator エンコーディングは順序属性にも適用できる.

Staircase エンコーディング: Staircase エンコーディング $\mathcal{E}_{\mathrm{st}} : \mathcal{O}_j \to \mathbb{Z}_2^{|\mathcal{O}_j|}$ はある順序属性の値 $s_k^j \in \mathcal{O}_j$ を入力として, 長さ $|\mathcal{O}_j|$ のバイナリベクトルを出力とする. この staircase エンコーディングの出力ベクトルでは 1 番目から $(k-1)$ 番目の要素は 0 であり, それ以降の要素は 1 である.

CRT-パッキング: CRT-パッキングについて詳細は 2 章に記載している. ここでは CRT-パッキングを用いて数値やカテゴリや順序属性データをエンコーディングする操作を定義する. CRT-パッキング $\mathcal{E}_{\mathrm{crt}} : \mathbb{Z}_t^{\ell} \to \mathbb{A}_t$ は, 整数のベクトル $\boldsymbol{u}$ を入力として環 $\mathbb{A}_t$ の元である多項式に写像する.

行列 $\boldsymbol{X}$ を扱う場合, $\mathcal{E}_{\mathrm{crt}}(\boldsymbol{X})$ は各行ベクトルに $\mathcal{E}_{\mathrm{crt}}$ を適応することを意味する. CODA では行列やベクトルに対してはすべて CRT-パッキングを利用する. 表記を簡潔にするため, 我々は特に指定が無い限り $\mathrm{Enc}(\mathcal{E}_{\mathrm{crt}}(\boldsymbol{x}))$ を $\mathrm{Enc}(\boldsymbol{x})$ と表記する.

4 プリミティブ

我々のプロトコルでは, いくつかのプリミティブを利用している. 利用するプリミティブは, HElib が提供する暗号文のプリミティブと我々

が設計したプリミティブに分けられる. 4.2 節では我々の提案したプロトコル GreaterThan プロトコルについて議論する. このプロトコルは Golle らの手法 [5] を, BGV スキームのために拡張したプロトコルである.

4.1 HElib によるプリミティブ

HElib [9] は BGV スキームを実装し, CRT-パッキングをサポートしている open-source ライブラリである. ここで, 暗号文 $\mathfrak{a}$ はベクトル $\boldsymbol{a} := [a_1, \cdots, a_\ell]$ をパッキングしている暗号文とする. CRT-パッキングの要素ごとの加算と乗算の他に, 我々は以下に示すプリミティブ利用する.

Replicate. replicate の操作はある暗号文 $\mathfrak{a}$ とインデックス $k \leq \ell$ を入力とする. その出力の暗号文は k 番目の要素を ℓ 個並べたベクトルの暗号文である. $[a_k, \cdots, a_k]$.

4.2 Greater Than プロトコル

我々は新たに暗号化された数値の大小比較プロトコル greater-than を提案する. 我々の greater-than プロトコルは Golle らのプロトコル [5] に基づている. 彼らのプロトコルは,「メイン $\mathcal{D} \subset \mathbb{Z}$ が与えられたとき, $x_1, x_2 \in \mathcal{D}$ のとき $x_1 > x_2$ が成り立つならば, $x_1 - x_2 - i = 0$ を満たす $i \in \mathcal{D}/\{0\}$ が存在する」という事実を利用している.

彼らのプロトコルでは暗号化を $O(|\mathcal{D}|)$ 回, 準同型加算を $O(|\mathcal{D}|)$ 回計算する必要がある. この要求はドメインの大きな数値属性の統計解析においてボトルネックになり得る. そこで, 我々は BGV のスキームと CRT-パッキングを利用することで, 新たなな greater-than プロトコルを設計した. 提案プロトコルでは複数の要素が一つの暗号にパックできることを利用する. ドメインサイズ $|\mathcal{D}|$ がスロット数 ℓ より小さい場合, 一度の準同型加算により, パックされたすべての要素に対して同時に加算操作を実現できる. 我々のプロトコルではすべての $i \in \mathcal{D}/\{0\}$ に対して, $x_1 - x_2 - i$ を一つの暗号文にパックする. そして, 一回の準同型性操作によりすべての i に対して $x_1 - x_2 - i$ を計算する. 全ての計算は CRT-パッキングを用いた準同型加算と乗算で実現できる. よって $|\mathcal{D}| < \ell$ の場合, 時間計算量もラウンド数も $O(1)$ である. ステップ 3 からステップ 7 では, すべての $i \in \mathcal{D}/\{0\}$ について, $r_i(x_1 - x_2 - i)$ を評価している. 結果を返す前に, 全ての要素 $r_i(x_1 - x_2 - i)$ はランダムにシャッフルされているので, 復号結果から Bob は x_1 と x_2 に関して何も情報を得ないことに注意されたい.

Algorithm 1 GreaterThan Protocol.

- **Input of Alice:** $\mathrm{Enc}(\boldsymbol{x}_1), \mathrm{Enc}(\boldsymbol{x}_2)$
- **Input of Bob: sk**
- **Output of Alice:** $\emptyset$
- **Output of Bob:** $\gamma = 1\{x_1 > x_2\}$

1: **procedure** GreaterThan($\mathrm{Enc}(\boldsymbol{x}_1), \mathrm{Enc}(\boldsymbol{x}_2)$)
2: **Alice**:
3: prepares a vector $\boldsymbol{i} = [1, 2, \cdots, |\mathcal{D}|]^{\mathrm{T}}$
4: randomly permutes vector $\boldsymbol{i}$ by random permutation π. Let $\tilde{\boldsymbol{i}}$ be the permutated $\boldsymbol{i}$ by π.
5: computes ciphertext $\mathfrak{a}$ as

$$\mathfrak{a} = \mathrm{Enc}(\boldsymbol{x}_1) - \mathrm{Enc}(\boldsymbol{x}_2) - \mathcal{E}_{\mathrm{crt}}(\tilde{\boldsymbol{i}}).$$

6: uniformly choose a length-$|\mathcal{D}|$ vector $\boldsymbol{r}$ over $\mathbb{Z}_t/\{0\}$
7: obfuscates each component of $\mathfrak{a}$ as $\tilde{\mathfrak{a}} \leftarrow \mathfrak{a} \otimes \mathcal{E}_{\mathrm{crt}}(\boldsymbol{r})$.
8: sends $\tilde{\mathfrak{a}}$ to Bob
9: **Bob**:
10: decrypts $\tilde{\mathfrak{a}}$ as $\boldsymbol{a} = \mathrm{Dec}(\tilde{\mathfrak{a}})$
11: gets $\gamma = 1\{x_1 > x_2\}$ as

$$\gamma = \begin{cases} 1 & (\exists i \Rightarrow a_i = 0) \\ 0 & \text{o.w.} \end{cases}$$

12: **endProcedure**

5 記述統計量のアウトソーシング

本節では記述統計量を安全にアウトソーシングするプロトコルについて述べる.

平均, 分散, 共分散: 平均と分散は数値データにおいて, 最も基本な記述統計量である. 平均と分散の安全なアウトソーシングは簡単に実現できる. i 番目のデータ提供者は upload の段階で自らのデータ $\boldsymbol{x}_i$ を CRT-パッキングを用いて暗号化した暗号文 $\mathfrak{x}_i := \mathrm{Enc}(\boldsymbol{x}_i)$ をクラウドに送信する. クラウドでは, データ提供者から暗号文を収集した後に平均, 分散と共分散を計

Algorithm 2 Mode Protocol.

- **Input of the i-th contributor:** ordinal data o_{ij}
- **Input of the analyst: sk**
- **Output of the analyst:** $\mathrm{Mode}(o_{1j}, \ldots, o_{nj})$

1: **Upload**: The i-th contributor submits the ciphertext of his ordinal data $\mathrm{Enc}(\mathcal{E}_{\mathrm{id}}(o_{ij}))$ to the cloud.
2: **Evaluation**: The cloud firstly computes $\mathfrak{o} = \bigoplus_i \mathrm{Enc}(\mathcal{E}_{\mathrm{id}}(o_{ij}))$.
3: The cloud calls $\mathfrak{o}_k := \mathrm{replicate}(\mathfrak{o}, k)$ for all $1 \le k \le |\mathcal{O}_j|$.
4: The cloud generates a random permutation θ.
5: For all $k \neq k'$, the cloud and analyst call:

$$(\gamma_{\theta(k)\theta(k')}, \emptyset) \leftarrow \textsc{GreaterThan}((\mathfrak{o}_{\theta(k)}, \mathfrak{o}_{\theta(k')}), \mathbf{sk}).$$

6: The analyst obtains $\mathbf{\Gamma} = (\gamma_{\theta(k)\theta(k')})$ for all $k \neq k'$.
7: **Download**: The analyst finds the row of $\mathbf{\Gamma}$ that includes the most entries of 1. Let k^* be the index of that row.
8: The analyst privately downloads $\theta(k^*)$ from the cloud and outputs it as $\mathrm{Mode}(o_{1j}, \ldots, o_{nj})$.

Algorithm 3 k-Percentile Protocol.

- **Input of the i-th data contributor:** the j-th ordinal attribute o_{ij}
- **Input of the analyst: sk**
- **Output of the analyst:** $\textsc{Percentile}_k(o_{1j}, \ldots, o_{nj})$

1: **Upload:** The i-th contributor submits $\mathfrak{o}_{ij} := \mathrm{Enc}(\mathcal{E}_{\mathrm{st}}(o_{ij}))$ to the cloud.
2: **Request:** The analyst requests that the cloud evaluate the k-percentile of the j-th attribute.
3: **Evaluation**: The cloud joins the collected ciphertexts as $\mathfrak{o} = \bigoplus_{i=1}^{n} \mathfrak{o}_{ij}$.
4: The cloud evaluates $\mathfrak{k} := \mathrm{Enc}(\lfloor kn/100 \rfloor \cdot \mathbf{1})$.
5: For $p = 1, \cdots, |\mathcal{O}_j|$, the cloud and analyst call:

$$(\gamma_p, \emptyset) \leftarrow \textsc{GreaterThan}((\mathfrak{k}, \mathrm{replicate}(\mathfrak{o}, p)), \mathbf{sk}).$$

6: The analyst obtains $\boldsymbol{\gamma} = (\gamma_p)$ for all $p = 1, \cdots, |\mathcal{O}_j|$.
7: **Download:** The analyst finds p^* s.t. $\gamma_{p^*} = 1 \wedge \gamma_{p^*+1} = 0$, and outputs it as $\textsc{Percentile}_k(o_{1j}, \ldots, o_{nj})$.

表 1: BGV スキームのパラメーターセット

	t	L	ℓ	m	κ
(I)	2^{59}	13	36	13423	90
(II)	72727	5	871	5227	80
(III)	67499	5	1742	5227	80

算する．全ての計算が行列の加算, 乗算で実現されるため, これらの統計量の評価は, 非対話的な 1-round プロトコルである．

最頻値: 最頻値を求めるプロトコルでは greater-than プロトコルを利用する．各データ提供者は j 番目の順序データを indicator エンコーディングを適用してから暗号化する (line 1)．クラウドでは, 暗号文を受け取った後, これらの暗号文の集約を行う (line 2)．次にクラウドは replicate を呼び出す．最後に greater-than を複数回呼び出すことにより, 解析者は $|\mathcal{C}_j| \times |\mathcal{C}_j|$ の行列 $\mathbf{\Gamma}$ を得る．その行列から解析者は j 番目の順序属性の中間値 $\theta(k^*)$ をクラウドに問い合わせることで最頻値を得ることができる．この問い合わせは, クラウドに中間値 $\theta(k^*)$ の位置を知らせてしまう．これを防ぐためには, 一般的な PIR が利用できる。詳細は省略するが, 実験ではこの操作も BGV スキームを用いて実現している．GreaterThan は非対話的なプロトコルであるが, 中間値 $\theta(k^*)$ の取得が必要であるため, アルゴリズム 2 は, 2-round プロトコルである．また, max, min, rank もほぼ同じプロトコルで実現可能である．

k-分位点: アルゴリズム 3 は k-分位点 (特に $k = 50$ のとき, 中央値) を計算するプロトコルである．ある順序属性の値 $s_p^j \in \mathcal{O}_j$ の累積頻度を $F(s_p^j)$ としたとき, k-分位点アルゴリズムは $k \cdot n/100 > F(s_p^j)$ and $k \cdot n/100 \le F(s_{p+1}^j)$ を満たすような p を求める．

アルゴリズム 3 では, j 番目の順序属性 $\mathcal{O}_j$ について考えている．line 3 では staircase エンコーディングを利用している．そのため, クラウドが集約した暗号文 $\mathfrak{o}$ は累積頻度となっている (line 3)．暗号文を集約した後, クラウドは GreaterThan を複数回呼び出す (line 5)．この結果を用いて, 分析者 $\gamma_p^* = 1 \wedge \gamma_{p^*+1} = 0$ を満たす p^* を見つけることができる (line 7)．

ここで, ベクトル $\boldsymbol{\gamma}$ は p^* 以外の情報は漏らさない．また, GreaterThan は非対話的なプロトコルであるため, アルゴリズム 3 も非対話的なプロトコルである．GreaterThan は非対話的なプロトコルであるため, アルゴリズム 3 は非対話的な 1-round プロトコルである．

6 実験

本節では, CODA がサポートする統計量を計算するプロトコルの実験結果について議論する．CODA は様々な統計量をサポートするが紙数の

表 2: 平均, 分散と共分散のベンチマーク (sec). 結果は十回平均した値を示す.

	Mean			Variance			Covariance		
n	upload	eval.	download	upload	eval.	download	upload	eval.	download
10000	0.088	0.577	0.336	0.088	9.89	0.350	0.164	10.8	0.372
20000	0.088	1.12	0.346	0.088	19.5	0.349	0.164	12.2	0.357
25000	0.088	1.36	0.347	0.088	24.3	0.354	0.161	12.7	0.339
32561	0.088	1.80	0.327	0.088	31.8	0.338	0.166	14.2	0.338

都合上, 本稿では一部の統計量の実験結果を示す. 実験は実データを用いておこなった. 利用したデータセットは UCI [7] で提供されている *Adult* データセットである. *Adult* データセットには 32651 レコード含まれており, 属性は数値属性が 6 個, カテゴリ属性が 7 個, 順序属性が 1 個の合計 14 属性である.

6.1 ベンチマークステートメント

CODA では主に三つの段階があるため, 実験ではこの三つの段階を分けて, それぞれの計算時間を計測した. upload 段階には各データ提供者の計算コストを測り, eval 段階ではクラウドの統計量の評価コストを測る. 最後に解析者が暗号文をダウンロードする download 階段では, 解析者の暗号文を復号するコストを測る.

実験は CPU が Xeon 2.60 GHz, メモリが 32G RAM の計算機を用いた. プログラムは C++で実装され, 8 並列で実行した. 本実験では通信コストについては議論しないが, 暗号文のサイズとネットの通信スピードから見積ることができる. 表 1 は実験で用いた BGV スキームのパラメーターセットである. 表の各列はそれぞれ, 法パラメーター t, levels L, スロット数 ℓ, 円分多項式パラメーター m とセキュリティパラメーター κ を表す.

平均, 分散, 共分散の実験結果: 表 2 は平均, 分散と共分散の計算のベンチマークを示す. この実験はパラメーターセット (I) を用いておこなった. CODA は CRT-パッキングを利用するため, 全ての次元の平均, 分散と共分散の計算は同時おこなうことができる. 表 2 に示すように, upload 段階と download 段階の計算コストはデータ数に依存せず, eval 段階に計算コストは扱うデータ数 n に対して線形である. 共分散の計算では, データ提供者は二つの暗号文を生成するため, upload 段階の計算コストは平均・分散に比べ倍になっている.

表 4: k-分位点プロトコルのベンチマーク (sec). 数値は 10 回の平均値を 示す.

	Parameter Set (II)			Parameter Set (III)		
n	upload	eval.	down.	upload	eval.	down.
500	0.019	8.04	1.67	0.018	9.24	2.53
1000	0.018	9.11	2.78	0.018	9.28	2.59
10000	0.018	13.1	15.9	0.017	12.0	12.5
20000	0.017	18.0	31.0	0.018	14.1	25.1
32561	0.017	24.1	49.7	0.018	18.5	41.2

最頻値の実験結果: 表 3 は最頻値を計算するプロトコルの実験結果である. 表 3(a) と表 3(b) はそれぞれ workclass 属性 (ドメインサイズ 8) と education 属性 (ドメインサイズ 16) の実験結果である. この結果から見ると evaluation 段階の計算時間は扱う属性のドメインサイズに対して二次的に増加していることがわかる. 例えば, $n = 32561$ のときパラメータセット (II) では workclass 属性の評価時間は 7.11s であるが education 属性を扱う時 24.1s かかっている.

k-分位点の実験結果: アルゴリズム 3 の評価では $O(n)$ 回の準同型加算 (line 4) 以外に greather-than を $O(|\mathcal{O}_j|)$ 回呼び出す. また, 最後の download 段階に k-分位点を得るため, 解析者は $O(\lceil n/\ell \rceil)$ 個の暗号文を復号する.

表 4 は k 分位点アルゴリズムの実験結果を示す. この実験では age 属性 (ドメインは 0 から 99 を想定した) に対して分位点を計算した. $n \leq 1000$ では, パラメーターセット (II) と (III) の計算時間はほとんど差がなかったが, n の増加に伴って (III) の実験時間は (II) の下の実験時間を下回った. この理由としては n がある程度まで増加すると greater-than プロトコルの計算時間が k 分位点プロトコル全体の計算時間の大半を占めるからであると考えられる.

表 3: 最頻値プロトコルのベンチマーク (sec). 左半分は workclass 属性, 右半分は education 属性の計算結果であり, 数値は十回の実行の平均値を示す.

	Workclass 属性						Education 属性					
	Parameter Set (II)			Parameter Set (III)			Parameter Set (II)			Parameter Set (III)		
n	upload	eval.	down.	upload	eval.	down.	upload	eval.	down.	upload	eval.	down.
500	0.019	6.11	0.67	0.018	7.11	0.99	0.020	22.6	2.25	0.019	24.1	2.28
1000	0.018	6.37	0.80	0.019	7.14	1.01	0.019	24.4	3.09	0.019	23.6	3.46
10000	0.017	8.67	4.22	0.018	8.76	3.86	0.017	28.7	16.4	0.017	28.0	13.2
20000	0.018	11.0	8.86	0.018	10.8	7.43	0.017	35.2	32.6	0.017	31.4	25.9
32561	0.018	13.2	13.4	0.017	11.5	11.1	0.017	44.2	50.2	0.017	38.1	39.1

7 結論

本稿では統計量の計算を安全にアウトソースする枠組み”CODA” を提案した. 我々のシステムは複数なデータ提供者からデータを集合し, クラウド上で秘密計算をおこなう. CODA を実現するために, 我々はいくつかのプリミティブを提案した. これらのプリミティブの利用により, 共分散, 分位点といった重要な記述統計量の計算を非対話的かつ安全的にクラウドに委託することができる. 実験では, 大規模データに対しても CODA が実用的であることを示した.

謝辞

本研究は，JST CREST「ビッグデータ統合利活用のための次世代基盤技術の創出・体系化」領域におけるプロジェクトおよび科学研究費 24680015 の助成を受けました.

参考文献

[1] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *Financial Cryptography and Data Security*, pages 325–343. Springer, 2009.

[2] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[3] Craig Gentry, Shai Halevi, and Nigel P Smart. Homomorphic evaluation of the aes circuit. In *Advances in Cryptology–CRYPTO 2012*, pages 850–867. Springer, 2012.

[4] Craig Goentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[5] Philippe Golle. A private stable matching algorithm. In *Financial Cryptography and Data Security*, pages 65–80. Springer, 2006.

[6] Shai Halevi and Victor Shoup. Algorithms in helib. In *Advances in Cryptology–CRYPTO 2014*, pages 554–571. Springer, 2014.

[7] M. Lichman. UCI machine learning repository, 2013.

[8] Michael Naehrig, Kristin Lauter, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM CCSW 2011*, pages 113–124. ACM, 2011.

[9] Victor Shoup Shai Halevi. HELib. `http://shaih.github.io/HElib/index.html`. Accessed: 2014-12-10.

[10] Nigel P Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography–PKC 2010*, pages 420–443. Springer, 2010.

[11] Nigel P Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. *Designs, codes and cryptography*, 71(1):57–81, 2014.