第 16 回!w 会
20 分でわからせる BFV

松岡 航太郎
京都大学情報学研究科情報学専攻通信情報システムコース
佐藤 高史 研究室 博士課程 3 年

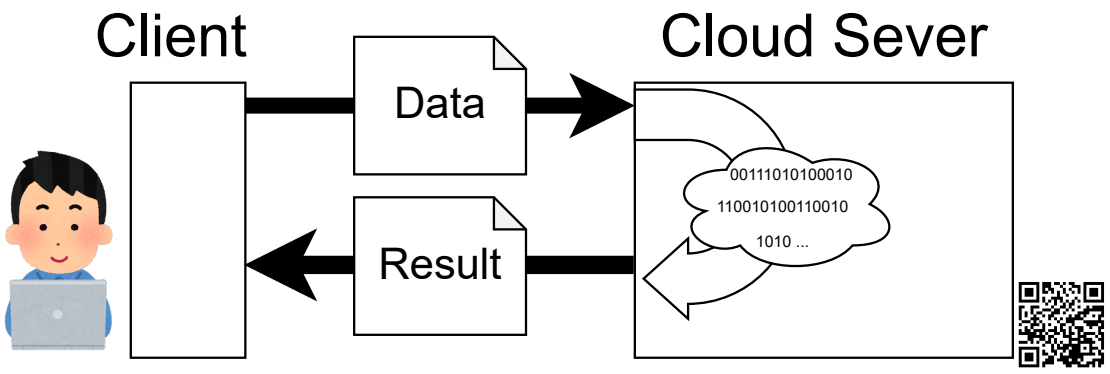
May. 30th 2025

自己紹介

- 松岡 航太郎
- 京都大学大学院情報学研究科通信情報システムコース 佐藤研究室 博士課程 3 年
  - 集積回路の研究室
  - 端っこで準同型暗号を専門に生きている
- セキュリティ・キャンプ 開発コースプロデューサ
  - 去年まで準同型暗号を教えてた (資料:https://nindanaoto.github.io/
- 日本学術振興会特別研究員 DC1
  - 申請書: https://github.com/nindanaoto/DC1proposal
- NHK 学生ロボコン 2019 優勝
- 2019 年度未踏スーパクリエータ
  - 応募資料&成果報告資料: https://github.com/virtualsecureplatform/MitouDocument

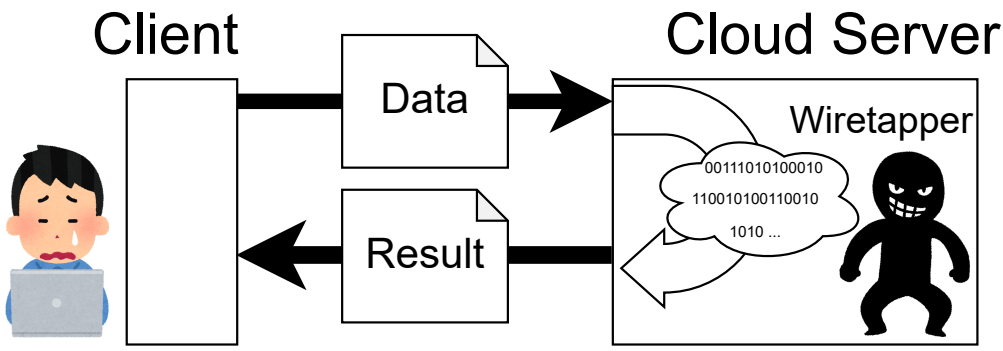
主な研究対象: 計算処理の委託

- 私の研究では主に”**計算処理の委託**”に関する”**信用**”が対象
- 最もわかりやすい例は”**クラウドコンピューティング**”
- ”**計算処理の委託**”は”**サービス提供者**”への”**信用**”に依存
  - e.g., Amazon Web Service (AWS), Google, Microsoft, Oracle, etc.



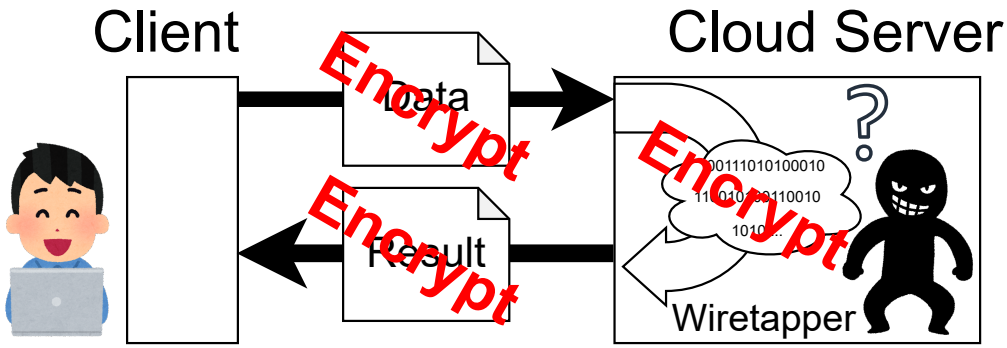
計算処理の委託における信用

- "**計算処理の委託**" における "**信用**" はさらにいくつかに細分化できる
- 最も代表的なのは "**データの秘匿性**"
  - もしサーバに物理的アクセスを持つ盗聴者がいるとデータを守るのは現状難しい
- 他の "**信用**" としては "**計算結果の真正性**" がある
  - これは Verifiable Computation の話になるので今回は扱わない



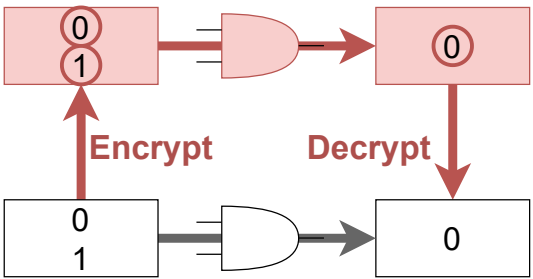
解決策: 秘匿 (秘密) 計算

- 実行時の盗聴を防ぐ最もシンプルな方法は常に暗号化したまま計算すること
  - 盗聴しても暗号文しか見えない
- これを達成する手段を総称して, "**秘匿計算**" と呼ぶ
  - その一つが準同型暗号



準同型暗号とは何か？

- 名前の通り暗号文に関して準同型性が成り立つ
  - 平文での操作に対応するような暗号文に対する演算が定義できる
- ここでいう”演算”の種類は暗号形式による
  - 下の図では AND だが整数の加算や乗算のことの方が多い
- 暗号化以外の機能も持つため”**高機能暗号**”(Advanced Cryptography) でもある
  - e.g., Functional Encryption (FE), Searchable Encryption (SE), etc.
  - 余談: FE と HE の違いは未解決, SE は Mongo DB に入っている[1]



[1] https://www.mongodb.com/blog/post/mongodb-announces-queryable-encryption

準同型暗号の実応用

現状今すぐにでも使える応用は以下のものだけと認識

1. Microsoft の Password Monitor
   - パスワードが漏洩しているかどうかを判定
   - Edge 経由で使えるらしい
2. Apple の Caller ID, Business Services in Mail, Enhanced Visual Search
   - Caller ID: 迷惑電話として知られている番号かどうかを判定 [BTR24]
   - Business Services in Mail: メールの送信元がスパムとして知られているかを判定
   - Enhanced Visual Search: 画像からどこでとられた写真家推定する
     - On-Device ML や Differential Privacy なども使われているので HE だけではない

BFV とは

- Brakerski-Fan-Vercauteren [Bra12, FV12] の略
  - 著者の頭文字の結合だが Brakerski さんの論文とほか二人の論文で別れている
    - これを明示するために B/FV と書くこともある
- 準同型暗号の一種で有限体上での準同型演算をサポートする
  - 有限体の各係数は整数剰余環の元なので定数項だけ使えば整数演算として使える
  - もっと一般には数論変換を使って多項式の根に値を埋め込むことが多い
    - canonical embedding と呼ばれる
    - 加算も乗算も多項式の操作 1 回で複数行えるので SIMD ライク
  - Apple のアプリケーションで使われているらしい
  - これより後に提案された暗号形式もあるため最新ではない
    - CKKS [CKKS17]: 固定小数点多項式
    - TFHE [CGGI16]: 論理演算

Torus(トーラス, 円周群)

- Torus は" 角度" がなす群のこと
  - $\mathbb{R}/\mathbb{Z}$, つまり実数の少数点部分がなす群として表現できる
    - ラジアンを 2π で割ったと思えばいい
- 角度なので加算が定義できる ($0.3 + 0.3 = 0.6 \bmod 1$ or 30 度+30 度=60 度)
- 乗算は定義できない ($0.3 \cdot 0.3 = 0.09 \bmod 1$??? or 30 度 $\times$30 度=900 度 [2]???)
- 整数倍は定義できる ($0.3 \cdot 3 = 0.9 \bmod 1$ or 3$\times$30 度=90 度)
- 可除群なので[2]正整数での除算もできる (ので有理数倍は定義できる)
  - $0.3/3 = 1$ or 30 度/3=10 度
  - $\mathbb{T}$ の値域を $[0, 1)$ とするか $[-0.5, 0.5)$ と見るかで厳密には挙動が変わる気はする
    - BFV の範囲だとどっちでもいい気もするが
- 整数への rounding による写像が定義できる ($\lceil 6 \cdot 0.3 \rfloor = \lceil 1.8 \rfloor = 2$)
  - 与えられた整数 p を法とした剰余環上の整数への写像という方が厳密
  - 実数としての掛け算を避けるなら $1/p \in \mathbb{T}$ の倍数で最も近いものに丸めている

[2]正確には可除群はある種の除算が定義可能という話であって正整数での除算とは限らないのだが

多項式剰余環

- 環の元を係数とする多項式を多項式で割ったあまりの成す環
  - ただ今回の説明では係数が $\mathbb{T}$ の場合も含むので多項式剰余群な気もする
- 出てくるのは $X^N + 1$ で割った余りの成すもの
- $\sum_{i=0}^{2N-1} a_i X^i \bmod X^N + 1 \equiv \sum_{i=0}^{N-1} (a_i - a_{i+N}) X^i$
  - 計算途中で N 次を超えたら N 以下の係数に上位の係数に加減算すれば良い
- N は常に 2 冪だと思って良い
  - 細かい話をすると暗号としての都合で既約多項式でないといけない
  - 円分多項式 [3]は規約である

[3] https://ja.wikipedia.org/wiki/円分多項式

TRLWE 暗号

- RLWE は Ring Learning With Errors [LPR13] の略
  - 一般には整数剰余環から係数を取るがここでは Torsu 版 [CGGI16] を使う
  - Torus は丸めれば任意の整数剰余環にできるが逆はロスがある
- LWE 暗号 [Reg09] の多項式環版
- $\Delta \in \mathbb{Z}^+$ をスケールといい, 平文空間を制限するパラメータである
- $m[X] \in (\mathbb{Z} \bmod \Delta)[X]/X^N + 1$ を平文とする
- $a[X] \in \mathbb{T}[X]/X^N + 1$ を nonce(暗号文ごとに異なる乱数) とする
- $s[X] \in \mathbb{Z}[X]/X^N + 1$ を秘密鍵とする
- $e[X] \in \mathbb{T}[X]/X^N + 1$ をノイズ (暗号文ごとに異なる乱数) とする
- 暗号化: $b[X] = a[X] \cdot s[X] + m[X]/\Delta + e[X]$ として $(a[X], b[X])$ が暗号文
- 復号: $\lceil \Delta \cdot (b[X] - a[X] \cdot s[X]) \rfloor = \lceil m[X] + \Delta \cdot e[X] \rfloor \approx m[X]$
  - $|e[X]|_\infty \ll 1/\Delta$ である必要がある

TRLWE の加算

- ベクトルとして暗号文を足せば平文としても加算

$$
\begin{aligned}
(a_1[X], b_1[X]) + (a_2[X], b_2[X]) &= (a_1[X] + a_2[X], b_1[X] + b_2[X]) \\
b_1[X] + b_2[X] - (a_1[X] + a_2[X]) \cdot s[X] &= (b_1[X] - a_1[X] \cdot s[X]) + (b_1[X] - a_1[X] \cdot s[X]) \\
&= m_1[X]/\Delta + e_1[X] + m_2[X]/\Delta + e_2[X]
\end{aligned}
$$

BFV の乗算 (アイデア)

- TRLWE は Torus 係数なのでそのままでは乗算はできない
  - 片方を整数に丸めて計算を行う
- $Bg \in \mathbb{Z}^+$ は十分大きな正整数
  - ノイズの増加を抑えるために乗算ごとに違う値を使っても良い
- あくまでアイデアの説明なので式変形はかなりラフ
  - 特に $e_1[X] \cdot e_2[X]$ はありえないが, ノイズが 2 乗で増えることがわかれば良い
    - 実際は丸め誤差を考えたりすると $e_1[X]$ 側が整数の形で表現されるはず

$$
\begin{aligned}
&(\lceil Bg \cdot b_1[X] \rfloor - \lceil Bg \cdot a_1[X] \rfloor \cdot s[X]) \cdot [\frac{\Delta}{Bg} \cdot (b_2[X] - a_2[X] \cdot s[X])] \\
&\approx \lceil Bg \cdot (m_1[X]/\Delta + e_1[X]) \rfloor \cdot [\frac{\Delta}{Bg}(m_2[X]/\Delta + e_2[X])] \\
&\approx \Delta \cdot (m_1[X]/\Delta + e_1[X]) \cdot (m_2[X]/\Delta + e_2[X]) \\
&\approx (m_1[X] \cdot m_2[X])/\Delta + m_1[X] \cdot e_2[X] + m_2[X] \cdot e_1[X] + \Delta \cdot e_1[X] \cdot e_2[X]
\end{aligned}
$$

BFV の乗算 (実際の演算)

- 先程のアイデアを暗号文から暗号文への演算として定義する

$$
\begin{aligned}
&(\lceil Bg \cdot b_1[X] \rfloor - \lceil Bg \cdot a_1[X] \rfloor \cdot s[X]) \cdot [\frac{\Delta}{Bg} \cdot (b_2[X] - a_2[X] \cdot s[X])] \\
&\quad = (\lceil Bg \cdot b_1[X] \rfloor \cdot \frac{\Delta}{Bg} \cdot b_2[X]) \\
&\qquad - (\lceil Bg \cdot b_1[X] \rfloor \cdot \frac{\Delta}{Bg} \cdot a_2[X] + \lceil Bg \cdot a_1[X] \rfloor \cdot \frac{\Delta}{Bg} \cdot b_2[X]) \cdot s[X] \\
&\qquad + \lceil Bg \cdot a_1[X] \rfloor \cdot a_2[X] \cdot (s[X])^2 \\
&\quad = b_{mul}[X] - a_{mul}[X] \cdot s[X] + c_{mul}[X] \cdot (s[X])^2
\end{aligned}
$$

- よって, 乗算結果の暗号文は $(a_{mul}, b_{mul}, c_{mul})$ の 3 つの多項式の組になる
- このままだと乗算するたびにどんどん暗号文が伸びてしまう
  - これを解決するのが Relinealization

BFV の Relinealization

$$b_{mul}[X] - a_{mul}[X] \cdot s[X] + c_{mul}[X] \cdot (s[X])^2 = m_1[X] \cdot m_2[X]/\Delta + e_{mul}[X]$$

$$b_{mul}[X] - a_{mul}[X] \cdot s[X] = m_1[X] \cdot m_2[X]/\Delta - c_{mul}[X] \cdot (s[X])^2 + e_{mul}[X]$$

- $c_{mul}[X] \cdot (s[X])^2$ を計算できればそれを $(a_{mul}[X], b_{mul}[X])$ に足すと長さが戻る
- $(s[X])^2$ を平文で渡すことはできないので暗号文として渡す必要がある
  - $m_[X]/\Delta$ の代わりに $(s[X])^2/Bg$ を入れた暗号文 (a_{relin}, b_{relin}) を使う

$$\lceil Bg \cdot c_{mul}[X] \rfloor \cdot (b_{relin} - a_{relin} \cdot s[X]) \approx c_{mul}[X] \cdot (s[X])^2 + \lceil Bg \cdot c_{mul}[X] \rfloor \cdot e_{relin}[X]$$

- Relinealization の具体的計算は以下のようなものになる
  - 実用的には Bg ではなく Bg^l 倍して l 桁に分解 (Decomposition) しノイズ増幅を抑えたりもする

$$(a_{mul}[X] + \lceil Bg \cdot c_{mul}[X] \rfloor \cdot a_{relin}[X], b_{mul}[X] + \lceil Bg \cdot c_{mul}[X] \rfloor \cdot b_{relin}[X]$$

Bootstrapping

- Bootstrapping [Gen09] はノイズをリセットする演算
  - 加算や乗算でノイズが増えるのでどこかで減らさないと演算が継続できなくなる
- Bootstrapping のアイデアは「暗号上で復号を評価する」こと
  - 復号には必ず暗号からノイズを取り除く演算が含まれる
- Bootstrapping をどうやって達成するかは暗号形式によって違う
  - 厳密に言えば同じ暗号形式でも複数の方法がある

BFV の Bootstrapping

- BFV の Bootstrapping の中でも特に簡単な HElib [4]で使われている方法 [HS21, ASP13] のアイデアを説明する
- $m[X]/\Delta$ の代わりに $s[X]/\Delta'$ を暗号化したものを $(a_{boot}[X], b_{boot}[X])$ とする
  - $\Delta \ll \Delta'$ である
- $(-\lceil \Delta' \cdot a[X] \rfloor \cdot a_{boot}[X], b[X] - \lceil \Delta' \cdot a[X] \rfloor \cdot b_{boot}[X])$ を計算する
- 平文は $\lceil \Delta' \cdot (m[X]/\Delta + e[X]) \rfloor / \Delta'$
  - ここではノイズも平文とみなしていることに注意
- この平文から $m[X]$ を取り出すということは係数の上位桁を取り出すということ
  - これを乗算と加減算, つまり多項式として表現する必要がある
  - そのような多項式の一つを Lifting と呼ぶ

[4]IBM のライブラリ `https://github.com/homenc/HElib`

Lifting

- 簡単のため Δ, Δ' は 2 冪であるとする[5]
- $\mathfrak{b} \in \mathbb{B}, \mathbb{Z}^+ \ni e \geq 1$ とする
- $z \equiv \mathfrak{b} \bmod 2^e \Rightarrow z^2 \equiv 1 \bmod 2^{e+1} \wedge 2^e | z - z^2 \wedge (z - z^2)/2^e \equiv (z - \mathfrak{b})/2^e \bmod 2$
  - $z = 2^e x + \mathfrak{b} \Rightarrow z^2 = 2^{2e}\mathfrak{b} + 2^{e+1} x\mathfrak{b} + \mathfrak{b}$
  - これを使って下の bit から順に値を確定させていく

,

- エッセンスを日本語で言い換えるとこう
  1. 有限の bit 幅であれば入力の 2 乗を取り続けると LSB がとれる
     - x^{2^e} で LSB 以外の下位 $e$bit が 0 にできる
  2. 入力の i-bit 目が LSB で以外下位 $(e - i)$bit が 0 のやつを作ればそれを 2^i 倍して入力から引けば下位 $e$bit を 0 にできる
     - 割れば最下位に e-bit 目を持ってこれる
     - べき乗すれば e-bit 目だけとれる

[5] 2 冪じゃないと LSB の取り出しの多項式が複雑に

Lifting の疑似コード

- ここでは e-bit 目だけを取り出すコードを示す
- Naïve にはこれを各 bit で頑張って暗号上でやる
  - 実際には 2 冪の時は全 bit をもうちょっと効率よく取り出せる
  - 最上位を取り出すときの途中結果が下の bit
  - 最上位の 1bit 下まで取り出せば最上位は計算できる

Require: z: input
Ensure: $w_{e,e}$: e-th bit

1: $w_{0,0} \leftarrow z$
2: **for** $k \leftarrow 0$ until e **do**
3: $\quad y \leftarrow z$
4: $\quad$ **for** $j \leftarrow 0$ to k **do**
5: $\quad\quad w_{j,k+1} \leftarrow w_{j,k}^2$
6: $\quad\quad y \leftarrow (y - w_{j,k+1})/2$
7: $\quad w_{k+1,k+1} \leftarrow y$
8: **return** $w_{e,e}$

結論

- 準同型暗号は『暗号のまま計算できる暗号』
  - すでに実用例もある
- BFV は Apple の実用例で使われている [6]
  - 整数多項式剰余環上の演算ができる
- 実装としては HElib 以外にも OpenFHE [7]でも実装されている
- BFV の進展としては LUT の評価 [LW24] や 2^N を超えるような整数が扱えるもの [CLPX17] がある

[6] https://github.com/apple/swift-homomorphic-encryption
[7] https://github.com/openfheorg/openfhe-development

Reference I

[ASP13] Jacob Alperin-Sheriff and Chris Peikert.
Practical Bootstrapping in Quasilinear Time.
In *Advances in Cryptology – CRYPTO 2013*, pages 1–20. Springer, Berlin, Heidelberg, 2013.
ISSN: 1611-3349.

[Bra12] Zvika Brakerski.
Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP.
In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pages 868–886, Berlin, Heidelberg, 2012. Springer.

[BTR24] Fabian Boemer, Karl Tarbe, and Rehan Rishi.
Announcing Swift homomorphic encryption, July 2024.
Swift.org blog accessed 2025-05-26.

[CGGI16] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène.
Faster Fully Homomorphic Encryption: Bootstrapping in Less Than 0.1 Seconds.
In Jung Hee Cheon and Tsuyoshi Takagi, editors, *Advances in Cryptology – ASIACRYPT 2016*, pages 3–33, Berlin, Heidelberg, 2016. Springer.

[CKKS17] Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song.
Homomorphic Encryption for Arithmetic of Approximate Numbers.
In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

Reference II

[CLPX17] Hao Chen, Kim Laine, Rachel Player, and Yuhou Xia.
High-Precision Arithmetic in Homomorphic Encryption, 2017.
Publication info: Preprint. MINOR revision.

[FV12] Junfeng Fan and Frederik Vercauteren.
Somewhat Practical Fully Homomorphic Encryption, 2012.
Publication info: Published elsewhere. Unknown where it was published.

[Gen09] Craig Gentry.
A fully homomorphic encryption scheme.
phd, Stanford University, Stanford, CA, USA, 2009.
AAI3382729 ISBN-13: 9781109444506.

[HS21] Shai Halevi and Victor Shoup.
Bootstrapping for HElib.
Journal of Cryptology, 34(1):7, January 2021.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev.
On Ideal Lattices and Learning with Errors over Rings.
J. ACM, 60(6):43:1–43:35, 2013.

[LW24] Zeyu Liu and Yunhao Wang.
Relaxed Functional Bootstrapping: A New Perspective on BGV/BFV Bootstrapping, 2024.
Publication info: Preprint.

Reference III

[Reg09] Oded Regev.
On lattices, learning with errors, random linear codes, and cryptography.
J. ACM, 56(6):34:1–34:40, September 2009.