第 4 回 京大ミートアップ supported by GMO
準同型暗号の研究動向

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

CC BY SA

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/
  - 今年のセキュリティ・キャンプ応募は 6 月 2 日 (月) まで
- 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

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

- 暗号に関する学問
  - eg. エニグマ (換字) 暗号, AES, RSA 暗号, etc.

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

- 暗号に関する学問
  - eg. エニグマ (換字) 暗号, AES, RSA 暗号, etc.
- 認証に関する学問
  - eg. 本人認証, データ完全性認証, 送信元認証, 否認防止 [Bar20], etc.

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

- 暗号に関する学問
  - eg. エニグマ (換字) 暗号, AES, RSA 暗号, etc.
- 認証に関する学問
  - eg. 本人認証, データ完全性認証, 送信元認証, 否認防止 [Bar20], etc.
- 数学の一分野 [Bar20]
  - eg. 計算量理論, ゲーム変換, 確率論, 群論, 楕円曲線論, etc.

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

- 暗号に関する学問
  - eg. エニグマ (換字) 暗号, AES, RSA 暗号, etc.
- 認証に関する学問
  - eg. 本人認証, データ完全性認証, 送信元認証, 否認防止 [Bar20], etc.
- 数学の一分野 [Bar20]
  - eg. 計算量理論, ゲーム変換, 確率論, 群論, 楕円曲線論, etc.
- あらゆる悪用に耐えるスキームの構築に関する学問 [Gol04]
  - eg. 盗聴, なりすまし, サイドチャネル, etc.

CC BY SA

暗号学とは何か?(私見)

どんな定義をイメージしますか？

- 暗号に関する学問
  - eg. エニグマ (換字) 暗号, AES, RSA 暗号, etc.
- 認証に関する学問
  - eg. 本人認証, データ完全性認証, 送信元認証, 否認防止 [Bar20], etc.
- 数学の一分野 [Bar20]
  - eg. 計算量理論, ゲーム変換, 確率論, 群論, 楕円曲線論, etc.
- あらゆる悪用に耐えるスキームの構築に関する学問 [Gol04]
  - eg. 盗聴, なりすまし, サイドチャネル, etc.

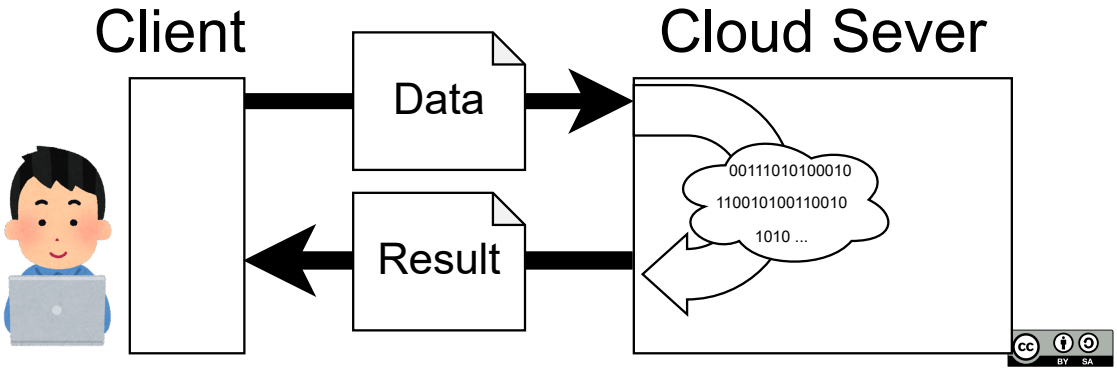
"**信用**" に関する学問

- 暗号学は"**信用**" が不要な代替手段を提供="**信用**" の価値を代替コストで測る
- eg. HTTPS, 暗号資産, etc.

CC BY SA

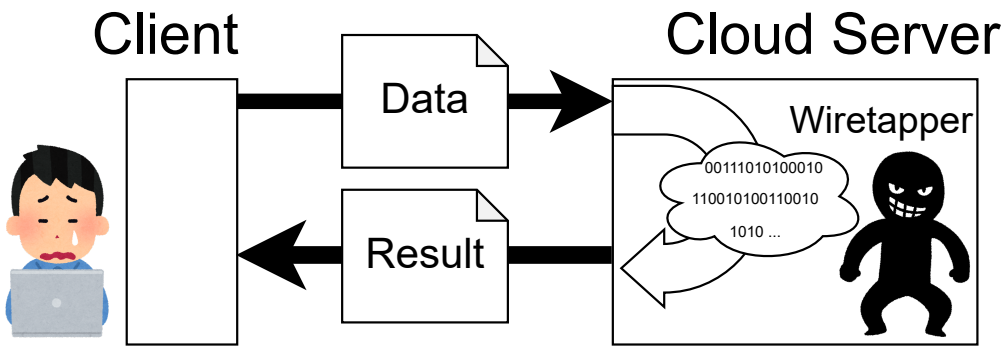
主な研究対象: 計算処理の委託

- "**信用**" 全体を直接扱うのは難しすぎる
  - 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

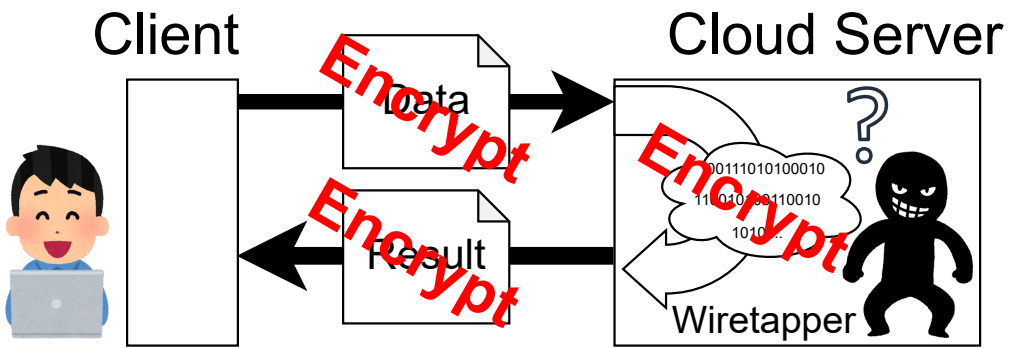
- "**計算処理の委託**" における "**信用**" はさらにいくつかに細分化できる
- 最も代表的なのは "**データの秘匿性**"
  - もしサーバに物理的アクセスを持つ盗聴者がいるとデータを守るのは現状難しい
- 他の "**信用**" としては "**計算結果の真正性**" がある
  - これは Verifiable Computation の話になるので今回は扱わない



CC BY SA

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



CC BY SA

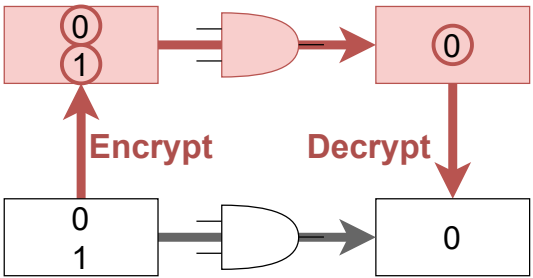
秘匿 (秘密) 計算の分類

- Secure Computation (SC) or Privacy-Preserving Computation (PPC) [CHP21]
- 現状大きく 4 種類に大別され, 下記では上に行くほど安全性の仮定が強い
  - " 理論的には" 脆弱だが効率は良い
  1. Trusted Execution Environment (TEE) [SSY+24]
     - CPU 内の隔離領域で計算を実施することで耐タンパ性を確保する方式
     - e.g: Intel SGX, AMD SEV, ARM TrustZone, RISC-V KeyStone
  2. Garbled Circuit (GC) [Yao82]
     - 演算を暗号文のテーブルの復号に置き換える方式
  3. Secure Multi-Party Computation (SMPC) [KPR17]
     - 日本語で秘密計算とだけ言った場合これを指す場合がある
     - 秘密分散 (Secret Sharing) をベースとした方式
  4. Homomorphic Encryption (HE) [RAD78]
     - 準同型暗号, 本日の主題
     - 暗号のまま計算ができる暗号

CC BY SA

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

CC BY SA

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

CC BY SA

準同型暗号の分類

基本的に下に行くほどできることが多くなるが計算は重くなる

1. 部分準同型暗号: PHE, Partially Homomorphic Encryption
   - または GHE: Group Homomorphic Encryption
   - 加算または乗算のいずれか片方だけをサポートする暗号
   - eg. RSA 暗号, ElGamal 暗号 [Elg85]
2. Somewhat 準同型暗号: SHE, Somewhat Homomorphic Encryption
   - 多くの場合加算は無限回できるが, 乗算回数に**暗号形式に依存した**上限があるもの
     - 例えば楕円曲線のペアリングで乗算を実装すると乗算後の暗号の形が変わってしまう
   - eg. Lifted ElGamal [AHM+18]
3. Leveled 準同型暗号: LHE, Leveled Homomorphic Encryption
   - 乗算回数に事前に決める**パラメータに依存した**上限があるもの
   - これも FHE として一緒くたに扱う場合もある
     - 理論上はパラメータを十分大きくすれば任意の関数が実行できるため
     - ただパラメータを大きくするとどんどん計算が重くなるので非現実的
4. 完全準同型暗号: FHE, Fully Homomorphic Encryption
   - **任意の**演算ができる方式, 今日主に扱う種別

CC BY SA

完全準同型暗号とは?

- 知る限りすべての LHE は FHE でもある
  - 違いは *Bootstrapping* と呼ばれる操作を使うかどうか
  - なので実質的には暗号形式というよりは実質的には運用に対する言葉
- 現状の LHE 以上の暗号はすべて何かしらの**ノイズ**が暗号文に含まれている
  - 演算を実施するたびに**ノイズ**が増加していく
  - **ノイズ**が大きすぎると復号が失敗する
- **ノイズ**をリセットするのが *Bootstrapping* という操作
  - 基本的なアイデアは**暗号上で復号関数を実行する**というもの [Gen09]
- *Bootstrapping* の具体的構成は触れない
  - セキュリティキャンプの資料で TFHE のものは説明している [2]
  - BFV 向けの Lifting を利用したものも比較的単純 [HS21]

[2] https://nindanaoto.github.io/

CC BY SA

Learning With Errors (LWE) 暗号

- 準同型暗号に利用される, **ノイズ**を含む代表的な暗号が LWE 暗号
  - これの一般形である Module Ring LWE 暗号が Kyber に採用されている [BDK+18]
  - Kyber は NIST の耐量子暗号として規格化されている (FIPS204)
- LWE 暗号は LWE 問題と呼ばれる格子 (Latiice) [3]上の NP 問題に基づいた暗号
  - ほぼ全ての (例外[4] [JLS22])FHE は LWE などを経由して Lattice Problem に帰着
- ここでは最も一般的な Integer LWE を説明する
- $m \in \mathbb{Z}_q$ が平文, $\mathbf{s} \in \mathbb{Z}_q^n$ が秘密鍵, $\mathbf{a} \in \mathbb{Z}_q^n$ は nonce[5]で一様乱数からとられる
- $e \in \mathbb{Z}_q$ がノイズ[6]で, 離散正規分布や Centered Binomial Distribution から
- 暗号化: $\mathbb{Z}_q \ni b = \mathbf{a} \cdot \mathbf{s} + m + e$ として $(\mathbf{a}, b)$ が暗号文
- 復号は多くの場合 $m >> e$ であることを仮定した一種の rounding で実装される
  - ノイズが増えると成り立たない (復号が失敗する)

[3]曖昧さ回避: `https://ja.wikipedia.org/wiki/格子_(数学)`

[4]Leraning Parity With Noise は LWE とよく似ているが帰着がしられていてない `https://crypto.stackexchange.com/questions/65999/are-lpn-and-lwe-problems-equivalent`

[5]暗号文ごとに事なるランダムな値. これがあるので同じ平文でも違う暗号文になる

[6]LWE とあるように Error と呼ぶこともあるので e が使われるのが一般的

CC BY SA

LWE の加法性

- 具体的な準同型暗号の例として LWE 暗号の加法を見る

$$(b_1 - \mathbf{a_1} \cdot \mathbf{s}) + (b_2 - \mathbf{a}_2 \cdot \mathbf{s}) = m_1 + e_1 + m_2 + e_2 \quad (1)$$

$$(b_1 + b_2) - (\mathbf{a}_1 + \mathbf{a}_2) \cdot \mathbf{s} = (m_1 + m_2) + (e_1 + e_2) \quad (2)$$

- **ノイズ**も加算されているため増加している
- 最小限としてはこれと平文との乗算 (定数倍) で FHE([GSW13] とか) は構成可能
  - 一般的には非常に非効率なので RLWR に拡張して考えたほうが良い
- 余談: もし秘密鍵ではなく nonce($\mathbf{a}$) を共通化すると Key Homomorphic になる
  - たまにそっちの性質を使うこともある

CC BY SA

GSW における乗算のアイデア

- **注意**: Decomposition を省略したためこのままだとノイズが大きくなりすぎる
- GSW では LWE 暗号を複数集めて作る新しい暗号を考える
- 以下では $(\mathbf{a}_i, b_i), i \in [1, n+1]$ は平文が 0 の LWE 暗号文とする
- $(\mathbf{a}, b)$ は被乗数となる LWE 暗号であり 0 とは限らない平文を持つ
- $\mu \in \mathbb{Z}$ は乗数となる平文,$\mathbf{I} \in \mathbb{Z}^{(n+1)\times(n+1)}$ は単位行列
- 下記の式は暗号文の μ 倍を計算する
- 0 の暗号文を被乗数の暗号文の係数で定数倍したものの和の平文も 0 なので結果の平文には影響しない
  - ノイズは大きく増幅されるので実際は Decomposition という工夫が必要
- [] の中は各行が μ または μ と秘密鍵の 1 係数の積を平文とする LWE 暗号

$$(\mathbf{a}, b) \cdot [\mu \cdot \mathbf{I} + \begin{pmatrix} \mathbf{a}_1 & b_1 \\ \vdots & \vdots \\ \mathbf{a}_{n+1} & b_{n+1} \end{pmatrix}] \tag{3}$$

CC BY SA

完全準同型暗号の世代

1. 第一世代: 理論的構成[7]
   - Gentrty さんの博士論文 [Gen09] で提案された理論的な構成
   - メモリが 1TB とか必要だと試算されており実装は存在しない
2. 第二世代: 整数多項式剰余環上演算
   - 平文が多項式であるため, 複数の整数を並列に処理できる
     - Canonical Embedding(要は FFT ないし NTT) を使えば乗算も並列化可能
   - eg. BGV [BGV12], BFV [Bra12, FV12]
   - Apple は BFV を使っている
3. 第三世代: 論理演算
   - eg. TFHE [CGGI16], FINAL [BIP+22], FHEW [DM15], GSW [GSW13]
   - *Bootstrapping* が特に高速 (低レイテンシ) であることが特徴
   - 論理回路向けの発想が適用しやすい
   - 第三世代はスカラ演算 (CPU), 第二, 四世代は SIMD 演算 (GPU) と比喩される
4. 第四世代: 固定小数点多項式剰余環上演算
   - 現状 CKKS [CKKS17] だけのことを指す
   - Microsoft は CKKS を使っている
   - 第三世代よりは第二世代と近しく, 平文の表現の仕方が少し違うだけ
   - 機械学習が表現しやすいのもあって一番応用研究が多い (個人の認識)

[7] https://en.wikipedia.org/wiki/Homomorphic_encryption#History

CC BY SA

完全準同型暗号における問題点

最後の一つは理論的すぎて実用の話からは遠いので詳細は省略

1. 高速化
   - 平文の計算に対して 10^8 程度のオーダで遅い[8]
     - 原付と光の速度くらいの差
   - 実用的には計算コストが大きすぎるとアプリケーションが限られる
2. 利便性
   - 準同型暗号特有の事情を考慮してプログラミングをしないといけない
   - アプリケーションのコストは開発コストも含まれる
3. 安全性・ノイズ評価
   - 速度を上げるためにセキュリティのマージンを限界まで切り詰めがち
   - Circular Security に関してはわかっていることが少ない
   - RLWR の安全性も不十分 (Kyber のおかげで研究は進む?

[8] これは CPU での TFHE の *Bootstrapping* と物理的な論理ゲートの速度の比率

CC BY SA

高速化の阻害要因

FHE の高速化を阻む要因は大きく分けて二つある

1. 多項式乗算が重すぎる
   - 実際の準同型暗号では $\mathbb{Z}_q$ ではなく $\mathbb{Z}_q[X] \bmod X^N + 1$ がスカラー
   - MRLWE と LWE の違いはこれで N は $2^9 \sim 2^{13}$ くらいが typical
   - 代わりに n は $1 \sim 3$ とかになる
2. 評価鍵のサイズが大きすぎる
   - ここでいう評価鍵といえば"**暗号化された秘密鍵**"のこと
   - *Boostraping* は暗号上で復号を評価する
     - 何らかの形で秘密鍵の情報を暗号上評価を実施するパーティに渡す必要がある
     - 評価鍵と呼ぶより Bootstrapping Key のほうが呼称としては一般的
   - 一般的に TFHE でも MB 単位, BFV や CKKS だと GB を超える
     - メモリ帯域が足りない

CC BY SA

高速化の試み: 多項式乗算のアルゴリズム的アプローチ

- 多項式乗算はよく知られているように畳み込み定理によって高速化できる
- 畳み込み定理が成り立つ変換としては以下のものが代表的

1. 高速フーリエ変換 (FFT): 複素正弦波を回転因子とするよく知られた離散変換
   - pros: CPU は浮動小数点演算は速度が出しやすく, HPC でよく研究されている
   - cons: GPU をはじめとして CPU 以外では浮動小数点演算はコストが大きい
2. 固定小数点 FFT: FFT の複素数を固定小数点で表現する
   - pros: DPU など, 組み込みの文脈でよく研究されていて浮動小数点もいらない
   - cons: 計算途中でオーバフローを起こす可能性があり, q をあまり大きくできない
3. 数論変換 (NTT): 整数の剰余環上の原始根を回転因子とするような変換
   - pros: 演算誤差がなく, 中国剰余定理によって問題を分割することもできる
   - cons: 剰余演算を頻繁に実施する必要がありその計算量が重い
4. Fermat Number Transform(FNT): NTT の法をフェルマー数に限定した変換
   - pros: 回転因子が 2 のべき乗になるので計算が容易
   - cons: オーバフローを避けるのに多項式の長さ自体が数倍になる [KMM+24]

CC BY SA

高速化の試み: 多項式乗算の CPU 向けソフトウェア実装

Engineering の成果として乗算の実際の実装をあげておく

- Intel HEXL [BKS+21]: おそらく DARPA の DPRIVE プロジェクト関連
  - AVX512 に追加された倍精度演算器を整数乗算に転用する命令を使っている
  - NTT の Harvey のアルゴリズムが実装されている
- IBM の実装 [BDH21]
  - 実際に使われてるのは見たことがない (内部向け?)
  - IBM Z14 の AVX512 相当の命令を使っている
- Number Theory Library [9]
  - 準同型暗号とは関係なく昔から使われているライブラリ
  - Harvey のアルゴリズムの前身である Shoup のアルゴリズムが実装
- SPQLIOS: TFHE のために作られた FFT のアセンブラ実装 [10]
  - AVX2 向けだったが MOSFHET で AVX512 向けに拡張 [11]

[9] https://libntl.org/

[10] https://github.com/tfhe/tfhe/tree/master/src/libtfhe/fft_processors/spqlios

[11] https://github.com/antoniocgj/MOSFHET/tree/main/src/fft/spqlios

CC BY SA

高速化の試み: FPGA による高速化

- FPGA: Field Programmable Gate Array
  - 論理回路を動的に構成可能なデバイス

1. HEAX [RLPD20]: Microsoft の人がかかわっている実装
   - CKKS の LHE モードを対象とした実装, 評価鍵はすべてオンチップ
2. HEAWS [TRV20]: KU Leuven の実装 [12]
   - AWS 上の F1 インスタンスで動く FV 向けの実装
3. FPT [VBDTV23]: KU Leuven の実装
   - TFHE 向け, 現状では最高速の実装, 最新の CPU よりも 3,4 倍くらいは早い
   - 評価鍵は High Band-width Memory(HBM) で転送
4. HOGE [MBS24]: 私が作ったやつ
   - FPT よりは 2 倍くらい遅い.....(実装の方針が高精度向けだったのはそう
   - 評価鍵は High Band-width Memory(HBM) で転送
5. HPU: Zama 社の TFHE 向け FPGA 実装 [13]
   - 存在を昨日知ったのでまだ性能評価などしていない

[12] https://github.com/KULeuven-COSIC/HEAT/tree/master/AWSF1

[13] https://github.com/zama-ai/hpu_fpga

CC BY SA

高速化の試み: ASIC

- ASIC: Application Specific Integrated Circuit
- 要は普通に専用の半導体デバイスとして作られたもの
- 実際に製造するのは非常にコストがかかるためほとんどがシミュレーション
  - 多くの設計は 100MiB オーダの SRAM を想定しているので面積が大きい

1. TREBUCHET [CPB+23]: DARPA DPRIVE 計画の CKKS 向け
   - 128-bit 512 要素のベクトル演算器を複数並べるアーキテクチャ
   - SPIRAL ベースの専用言語で制御
   - GF12LP で 150mm^2 らしい (本当に作ったら製造費だけで数億円するはず
2. ISSCC 2024 16.1 [LKL24]: 最初の製造された CKKS 向けアクセラレータ
   - 28nm で 11.3mm^2 と比較的小さい
   - どちらかといえば省電力小面積が優先されている
   - メモリが足りないので多分 *Bootstrapping* は動かない?
3. YATA: 私が作ったやつ, 最初の製造された TFHE 向けアクセラレータ
   - TSMC22nm^{2}ULP, 6mm^2
   - メモリが足りなかったのでベンチマークは乱数生成器のダミーも使った
   - FPT より倍くらい速いところまでは可能

CC BY SA

高速化の試み: その他のハードウェア高速化

- DRAM ベース Compute In Memory(CIM) [NJOP25]
  - メモリアクセスが重いならメモリのそばで計算すればいいということ自体は妥当
  - このアイデアでいくつか CIM で準同型暗号をやろうという研究がある
- 光コンピュータ [ZCL+24, ZLC+23]
  - 畳み込み演算なら光コンピュータが強いのは原理的にそう
  - Optalysys っていうベンチャー企業がやっているらしい [14]
- TPU [KSX+24]: Google の TPU を使って準同型暗号を評価
  - メモリアクセスが問題になるのは深層学習も同じ
  - 深層学習アクセラレータを転用する試みとして TPU は既に動くらしい

[14] https://optalysys.com/

CC BY SA

高速化の試み: まとめ

- 企業レベルではこのレイヤは Microsoft と Intel(DARPA) が強い印象
- 多項式乗算関連のアルゴリズムに関しても毎年新しいものが出る
  - 2025 年にもなってまだ進展するとは思わなかったというのが正直な感想
- ハードウェア高速化についてはまだまだ進展の余地がある
  - 省略したがもちろん GPU での高速化もいろいろ研究がある
    - TFHE だと私のメンテしてる実装 [15] と Zama 社の実装 [16] がある
  - アーキテクチャも結構バラバラ
    - どこまで汎用性を持たせるかが難しいところ
    - 暗号方式間の違い, セキュリティ評価の変化への対応

[15] https://github.com/virtualsecureplatform/cuFHE.git

[16] https://github.com/zama-ai/tfhe-rs/tree/main/backends/tfhe-cuda-backend

CC BY SA

利便性の阻害要因

1. 暗号上で実施可能な演算が特殊かつ限られている
   - BFV や CKKS では加算と乗算しかない
     - 計算したい処理を多項式として表現しないといけない
     - 一般的には Arithemtic Circuit として表現
   - TFHE は論理演算なので論理合成が使える
     - しかし既存の論理合成だと TFHE の機能全てが使えるわけではない
     - 最低でも多入出力 LUT として扱うべき
2. タスクの並列実行
   - アプリケーションレベルになると, 複数の準同型演算を同時に実行する必要がある
   - 評価環境に合わせて適切にこの並列性を扱う必要がある
   - いくつか実装を書いているが実装より過ぎる話なので割愛 [17]

[17] https://github.com/virtualsecureplatform/Iyokan,https://github.com/virtualsecureplatform/Sudachi,https://github.com/virtualsecureplatform/Tangor

CC BY SA

利便性の改善: コンパイラ

- 通常のプログラミング言語に近い言語から準同型暗号の演算に変換する
  - 抽象化や最適化の方法はアプローチによってずいぶん異なる

1. HEIR[18]: Google のプロジェクト, LLVM MLIR ベース
   - BFV,CKKS,TFHE の複数の方式に対してそれぞれ中間言語を作る
   - 現状実装は対象となる方式ごとに分離されて要るっぽい
   - 噂だとフルタイムの人は二人くらいで作ってるらしい
2. Concrete [19]: Zama 社の TFHE 向けフレームワーク
   - Zama 社は TFHE の原著者がいるベンチャー企業
   - TFHE 向けの Python Wrapper
     - UInt8 の掛け算とかそのレベルまで抽象化してくれる
   - PyTorch のコードが一部そのまま動いたりもする
3. VSP [MBM+21]: 私の最初の研究
   - CPU の論理回路を暗号上で評価する
   - こうすると普通の CPU 向けのバイナリがそのまま動く
   - 当然上二つより遅くはなるので抽象化と速度はトレードオフ

[18] https://github.com/google/heir

[19] `https://github.com/zama-ai/concrete`

CC BY SA

利便性の改善: 論理合成

- TFHE の場合論理演算がベースなので通常の論理回路向けの方法が使える
- しかし TFHE は加算と CNOT が前後についた多入出力 LUT が基本単位
  - 通常の回路では一般的ではないので, 特別なフローが必要

1. 2 入力 1 出力+MUX に限定する [MBM+21, GT20]
   - 演算のごく限られたサブセットとしてよく知られたゲートだけを使う方法
2. 多入出力のゲートの Technology Mapping [MHSB21, YCTCDM24]
   - Full Adder など本当に一部に絞って拡張したのが私の研究
   - 去年もっと一般の 3 入力ゲートへの拡張が出た
   - 事前に使いたいゲートのライブラリを用意して合致するところ置換する
3. 局所的に全探索を実施する [GMZ+24]
   - 回路の一部に注目して TFHE の基本単位で表現できないか全探索で調べる

CC BY SA

利便性の改善: まとめ

- 現状はそもそもどういう表現で準同型暗号のプログラムを書くべきかが未定
  - 私は論理回路側の人間なので基本 HDL に近いほうがよいという認識
    - 論理合成の知見を持ち込みたい
    - 抽象化したいなら回路側で抽象化すればいい
  - Google は C++っぽく書きたい?
    - LLVM などコンパイラの知見を持ち込みたい
  - Zama は Rust のローレイヤ API と Python を両方提供という方針っぽい
    - 最適化より Python のエコシステムが欲しい？

CC BY SA

結び

- 駆け足に背景から応用, 実装の話まで軽く見た
  - 明示的に書いたものも含め割愛した内容は多い
  - 事前知識の要求が多いので理論の話は意図して優先的に省いた
- TFHE を教えようと思うと駆け足でも週 1 時間 2 ヵ月くらいの講義になる
  - 資料は公開しているので自学自習はできるはず [20]
- 応用は進みつつあるがまだまだ理論レベルでもやるべきことが多い
  - 第 5 世代というべき FHE がないのも懸念事項
  - Batch Bootstrapping [LW23] から生まれそうではある
- 社会実装には理論から実装, 応用まで一貫したプラットフォームが必要
  - できているのは Zama, Microsoft, Apple あたりだけか
  - それぞれ方向性が違うのでまだ勝負のついていない土俵

[20] https://nindanaoto.github.io/

CC BY SA

Reference I

[AHM+18] Nuttapong Attrapadung, Goichiro Hanaoka, Shigeo Mitsunari, Yusuke Sakai, Kana Shimizu, and Tadanori Teruya.
Efficient Two-level Homomorphic Encryption in Prime-order Bilinear Groups and A Fast Implementation in WebAssembly.
In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, pages 685–697, New York, NY, USA, 2018. Association for Computing Machinery.

[Bar20] Elaine Barker.
Guideline for Using Cryptographic Standards in the Federal Government: Cryptographic Mechanisms.
Technical Report NIST Special Publication (SP) 800-175B Rev. 1, National Institute of Standards and Technology, March 2020.

[BDH21] Jonathan Bradbury, Nir Drucker, and Marius Hillenbrand.
NTT software optimization using an extended Harvey butterfly, 2021.
Publication info: Preprint.

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle.
CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM.
In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, April 2018.

Reference II

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan.
(Leveled) fully homomorphic encryption without bootstrapping.
In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, pages 309–325, New York, NY, USA, January 2012. Association for Computing Machinery.

[BIP+22] Charlotte Bonte, Ilia Iliashenko, Jeongeun Park, Hilder V. L. Pereira, and Nigel P. Smart.
FINAL: Faster FHE Instantiated with NTRU and LWE.
In *Advances in Cryptology – ASIACRYPT 2022*, pages 188–215. Springer, Cham, 2022.
ISSN: 1611-3349.

[BKS+21] Fabian Boemer, Sejun Kim, Gelila Seifu, Fillipe D.M. de Souza, and Vinodh Gopal.
Intel HEXL: Accelerating Homomorphic Encryption with Intel AVX512-IFMA52.
In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '21, pages 57–62, New York, NY, USA, 2021. Association for Computing Machinery.

[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.

CC BY SA

Reference III

[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.

[CHP21] José Cabrero-Holgueras and Sergio Pastrana.
SoK: Privacy-Preserving Computation Techniques for Deep Learning.
Proceedings on Privacy Enhancing Technologies, 2021(4):139–162, October 2021.

[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.

CC BY SA

Reference IV

[CPB+23] David Bruce Cousins, Yuriy Polyakov, Ahmad Al Badawi, Matthew French, Andrew Schmidt, Ajey Jacob, Benedict Reynwar, Kellie Canida, Akhilesh Jaiswal, Clynn Mathew, Homer Gamil, Negar Neda, Deepraj Soni, Michail Maniatakos, Brandon Reagen, Naifeng Zhang, Franz Franchetti, Patrick Brinich, Jeremy Johnson, Patrick Broderick, Mike Franusich, Bo Zhang, Zeming Cheng, and Massoud Pedram.
TREBUCHET: Fully Homomorphic Encryption Accelerator for Deep Computation, 2023.
Publication info: Preprint.

[DM15] Léo Ducas and Daniele Micciancio.
FHEW: Bootstrapping Homomorphic Encryption in Less Than a Second.
In *Advances in Cryptology – EUROCRYPT 2015*, pages 617–640. Springer, Berlin, Heidelberg, 2015.
ISSN: 1611-3349.

[Elg85] T. Elgamal.
A public key cryptosystem and a signature scheme based on discrete logarithms.
IEEE Transactions on Information Theory, 31(4):469–472, July 1985.

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

CC BY SA

Reference V

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

[GMZ+24] Zhenyu Guan, Ran Mao, Qianyun Zhang, Zhou Zhang, Zian Zhao, and Song Bian.
AutoHoG: Automating Homomorphic Gate Design for Large-Scale Logic Circuit Evaluation.
IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 43(7):1971–1983, July 2024.
Conference Name: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems.

[Gol04] Oded Goldreich.
Foundations of Cryptography.
Cambridge University Press, 2004.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters.
Homomorphic Encryption from Learning with Errors: Conceptually-Simpler, Asymptotically-Faster, Attribute-Based.
In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 75–92, Berlin, Heidelberg, 2013. Springer.

CC BY SA

Reference VI

[GT20] Charles Gouert and Nektarios Georgios Tsoutsos.
Romeo: Conversion and Evaluation of HDL Designs in the Encrypted Domain.
In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6, July 2020.
ISSN: 0738-100X.

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

[JLS22] Aayush Jain, Huijia Lin, and Amit Sahai.
Indistinguishability Obfuscation from LPN over $$\mathbb {F}_p$$, DLIN, and PRGs in NC ^{0}.
In *Advances in Cryptology – EUROCRYPT 2022*, pages 670–699. Springer, Cham, 2022.
ISSN: 1611-3349.

[KMM+24] Andrey Kim, Ahmet Can Mert, Anisha Mukherjee, Aikata Aikata, Maxim Deryabin, Sunmin Kwon, Hyung Chul Kang, and Sujoy Sinha Roy.
Exploring the Advantages and Challenges of Fermat NTT in FHE Acceleration.
In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 76–106, Cham, 2024. Springer Nature Switzerland.

[KPR17] Marcel Keller, Valerio Pastro, and Dragos Rotaru.
Overdrive: Making SPDZ Great Again, 2017.
Publication info: A minor revision of an IACR publication in EUROCRYPT 2018.

CC BY SA

Reference VII

[KSX+24] Rabimba Karanjai, Sangwon Shin, Wujie Xiong, Xinxin Fan, Lin Chen, Tianwei Zhang, Taeweon Suh, Weidong Shi, Veronika Kuchta, Francesco Sica, and Lei Xu.
TPU as Cryptographic Accelerator.
In *Proceedings of the International Workshop on Hardware and Architectural Support for Security and Privacy 2024*, HASP '24, pages 37–44, New York, NY, USA, 2024. Association for Computing Machinery.

[LKL24] Hyunhoon Lee, Hyeokjun Kwon, and Youngjoo Lee.
16.1 A 2.7-to-13.3 µ J/boot/slot Flexible RNS-CKKS Processor in 28nm CMOS Technology for FHE-Based Privacy-Preserving Computing.
In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 296–298, February 2024.
ISSN: 2376-8606.

[LW23] Feng-Hao Liu and Han Wang.
Batch Bootstrapping I:.
In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology – EUROCRYPT 2023*, pages 321–352, Cham, 2023. Springer Nature Switzerland.

[MBM+21] Kotaro Matsuoka, Ryotaro Banno, Naoki Matsumoto, Takashi Sato, and Song Bian.
Virtual Secure Platform: A {Five-Stage} Pipeline Processor over {TFHE}.
pages 4007–4024, 2021.

CC BY SA

Reference VIII

[MBS24] Kotaro Matsuoka, Song Bian, and Takashi Sato.
HOGE : Homomorphic Gate on an FPGA.
In *Proceedings of the 29th Asia and South Pacific Design Automation Conference*, ASPDAC '24, pages 325–332, Incheon, Republic of Korea, 2024. IEEE Press.

[MHSB21] Kotaro Matsuoka, Yusuke Hoshizuki, Takashi Sato, and Song Bian.
Towards Better Standard Cell Library: Optimizing Compound Logic Gates for TFHE.
In *Proceedings of the 9th on Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '21, pages 63–68, New York, NY, USA, 2021. Association for Computing Machinery.

[NJOP25] Kevin Nam, Heonhui Jung, Hyunyoung Oh, and Yunheung Paek.
Affinity-based Optimizations for TFHE on Processing-in-DRAM.
In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS '25, pages 16–31, New York, NY, USA, 2025. Association for Computing Machinery.

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos.
On Data Banks and Privacy Homomorphisms.
Foundations of Secure Computation, pages 169–179, 1978.

CC BY SA

Reference IX

[RLPD20] M. Sadegh Riazi, Kim Laine, Blake Pelton, and Wei Dai.
HEAX: An Architecture for Computing on Encrypted Data.
In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '20, pages 1295–1309, New York, NY, USA, 2020. Association for Computing Machinery.

[SSY+24] Stephan van Schaik, Alex Seto, Thomas Yurek, Adam Batori, Bader AlBassam, Daniel Genkin, Andrew Miller, Eyal Ronen, Yuval Yarom, and Christina Garman.
SoK: SGX.Fail: How Stuff Gets eXposed.
pages 248–248. IEEE Computer Society, May 2024.
ISSN: 2375-1207.

[TRV20] Furkan Turan, Sujoy Sinha Roy, and Ingrid Verbauwhede.
HEAWS: An Accelerator for Homomorphic Encryption on the Amazon AWS FPGA.
IEEE Transactions on Computers, 69(8):1185–1196, August 2020.
Conference Name: IEEE Transactions on Computers.

[VBDTV23] Michiel Van Beirendonck, Jan-Pieter D'Anvers, Furkan Turan, and Ingrid Verbauwhede.
FPT: A Fixed-Point Accelerator for Torus Fully Homomorphic Encryption.
In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, pages 741–755, New York, NY, USA, 2023. Association for Computing Machinery.

CC BY SA

Reference X

[Yao82] Andrew C. Yao.
Protocols for secure computations.
In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 160–164, November 1982.
ISSN: 0272-5428.

[YCTCDM24] Mingfei Yu, Sergiu Carpov, Alessandro Tempia Calvino, and Giovanni De Micheli.
On the Synthesis of High-performance Homomorphic Boolean Circuits.
In *Proceedings of the 12th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*, WAHC '24, pages 51–63, New York, NY, USA, November 2024. Association for Computing Machinery.

[ZCL+24] Mengxin Zheng, Cheng Chu, Qian Lou, Nathan Youngblood, Mo Li, Sajjad Moazeni, and Lei Jiang.
OFHE: An Electro-Optical Accelerator for Discretized TFHE, May 2024.
arXiv:2405.11607 [cs].

[ZLC+23] Mengxin Zheng, Qian Lou, Fan Chen, Lei Jiang, and Yongxin Zhu.
CryptoLight: An Electro-Optical Accelerator for Fully Homomorphic Encryption.
In *Proceedings of the 17th ACM International Symposium on Nanoscale Architectures*, NANOARCH '22, pages 1–2, New York, NY, USA, 2023. Association for Computing Machinery.

CC BY SA