1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

効率的な暗号処理に向けたFHE暗号方式・ライブラリの比較

Comparison of FHE Schemes and Librariesfor Efficient Cryptographic Processing

山本 藤也（Touya Yamamoto）
u220067@st.pu-toyama.ac.jp

富山県立大学 情報システム工学科

January 21, 2025

1. 背景と課題

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

背景

- クラウド利用が拡大し、大量データを分析する機会が増加。
- 既存の暗号技術ではデータの復号が必要 → セキュリティリスク。
- 完全準同型暗号（FHE）：暗号化したまま任意の演算が可能。

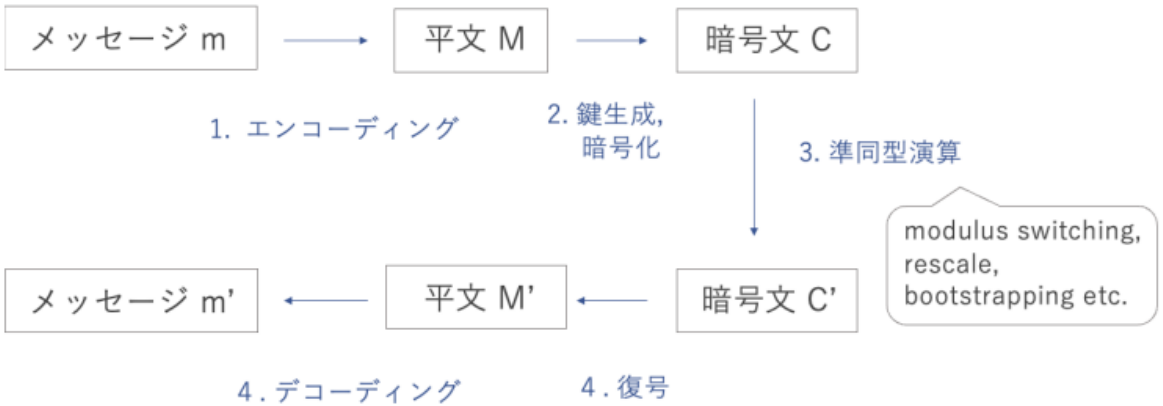


図 1　FHE の暗号処理の流れ

1. 背景と課題

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

課題

- 暗号方式やライブラリの選択が複雑。
- クラウド環境ではリソース制限（DRAM 容量）も課題。

2. 研究目的

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

本研究の目的

- 暗号方式（BFV, BGV, CKKS, TFHE 亜種）の比較評価。
- 複数のライブラリ（OpenFHE, Lattigo, TFHEpp）間の性能比較。
- 実アプリケーションでの課題抽出（ゲノム秘匿検索を対象）。

3. 暗号方式とライブラリの概要

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

暗号方式の特徴

- 算術演算/論理演算の対応有無
- パッキングサイズの違い（整数や浮動小数点）
- 使用する数学的背景（RLWE 問題、トーラスなど）

表 1　FHE 暗号方式の比較

Table 1　Features of FHE schemes.

FHE 暗号方式	特徴			
	算術演算	論理演算	ドメイン	パッキングサイズ
BFV	✓		integer	N
BGV	✓		integer	N
CKKS	✓		floating-point	$N/2$
TFHE		✓	binary	1
TFHE 亜種	✓	✓	floating-point	1

3. 暗号方式とライブラリの概要

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

ライブラリの対応

- 各暗号方式がサポートされているライブラリ一覧

表 2　FHE 暗号方式が実装されているライブラリ

Table 2　Libraries in which the FHE encryption schemes are implemented.

FHE 暗号方式	ライブラリ		
	OpenFHE	Lattigo	TFHEpp
BFV	✓	✓	
BGV	✓	✓	
CKKS	✓	✓	
TFHE	✓		✓
TFHE 亜種	✓		✓

4. 評価方法

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

評価対象と環境

- 比較対象：BFV, BGV, CKKS, TFHE 亜種方式
- 使用ライブラリ：OpenFHE, Lattigo, TFHEpp
- 実験環境：
  CPU: AMD Ryzen 7 5700G @ 3.8GHz
  DRAM: 16GB
  SSD: SK hynix PC711 (238GB)

4. 評価方法

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

評価基準

- 時間計算量と空間計算量の測定
  m, n : 暗号化される値
  式 (1): 加算の準同型性　式 (2): 乗算の準同型性

$$Decrypt(Encrypt(m) \oplus Encrypt(n)) = Decrypt(Encrypt(m+n)) \quad (1)$$

$$Decrypt(Encrypt(m) \otimes Encrypt(n)) = Decrypt(Encrypt(m \times n)) \quad (2)$$

4. 評価方法

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

比較で使用するパラメータ

- t: 平文の空間を定義するモジュラス。
  N: 暗号文の多項式次数。
  batch: 1 つの暗号文にパッキングできるメッセージ数。
  q: 乗算深度ごとに変化する暗号文のモジュラス。

表 3 BFV，BGV，CKKS 方式の比較で使用するパラメータ
Table 3 Parameters used to compare BFV, BGV, and CKKS schemes.

(a) Parameters

暗号方式	t	N	batch
BFV	65,537	32,768	32,768
BGV	65,537	32,768	32,768
CKKS	65,537	65,536	32,768

(b) Ciphertext modulus q

暗号方式	ライブラリ	乗算深度				
		2	4	6	8	10
BFV	Lattigo	118	177	235	294	352
	OpenFHE	118	177	236	295	354
BGV	Lattigo	118	235	352	410	583
	OpenFHE	118	236	354	472	590
CKKS	Lattigo	146	236	326	416	506
	OpenFHE	145	235	325	415	505

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

目的

- 暗号方式とライブラリの性能比較
  →暗号方式とライブラリの時間・空間計算量を比較。
  →暗号化された状態での処理効率を明らかにする。

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

暗号方式とライブラリの性能比較

- BGV, BFV, CKKS の順に高速

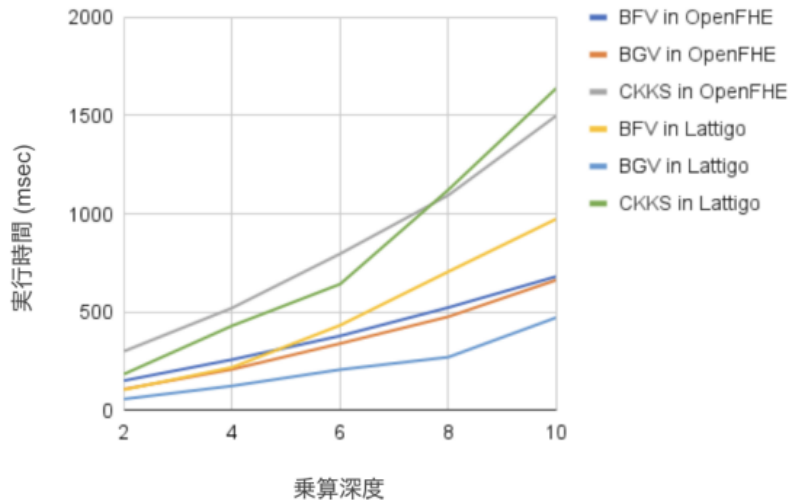


図 2 OpenFHE と Lattigo に実装された BFV, BGV, および CKKS 方式の乗算深度を変化させた際の実行時間の比較

Fig. 2 Comparison of the execution time of BFV, BGV, and CKKS schemes using openFHE and Lattigo libraries.

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

暗号方式とライブラリの性能比較

- CKKS 方式は他の方式より多くのメモリを使用

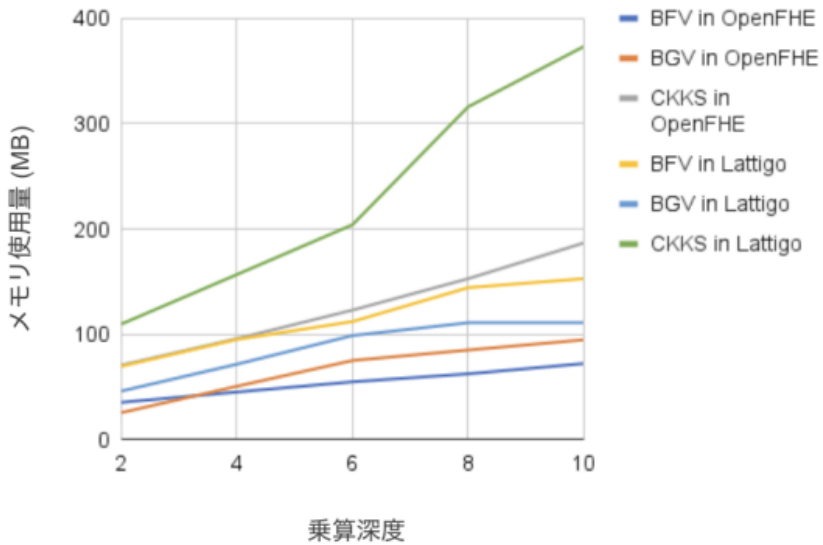


図 3　OpenFHE と Lattigo に実装された BFV，BGV，および CKKS 方式の乗算深度を変化させた際のメモリ使用量の比較

Fig. 3　Comparison of memory usage of BFV, BGV, and CKKS schemes using OpenFHE and Lattigo libraries.

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

目的

- CKKS 方式と TFHE 亜種方式の比較
  CKKS 方式と TFHE 亜種方式の特性を比較し、適用範囲を明確化。

設定

- TFHE 亜種方式のパラメータ

表 4　TFHE 亜種方式の評価で使用するパラメータ

Table 4　Parameters used to evaluate Zama's variant of TFHE scheme.

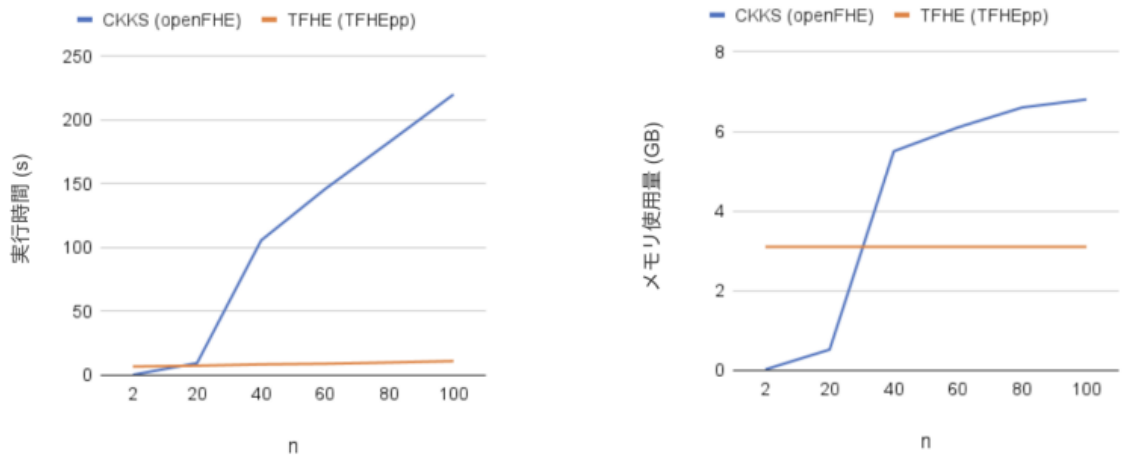
ライブラリ	N	n	q	gadgetBase	baseSK
OpenFHE	2,048	512	2^{27}	2^7	2^7
TFHEpp	2,048	500	2^{32}	2^6	2^4

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

CKKS 方式と TFHE 亜種方式の比較

- n=40 より大きい値では TFHE 亜種方式が効率的。



(a) 実行時間　　(b) メモリ使用量

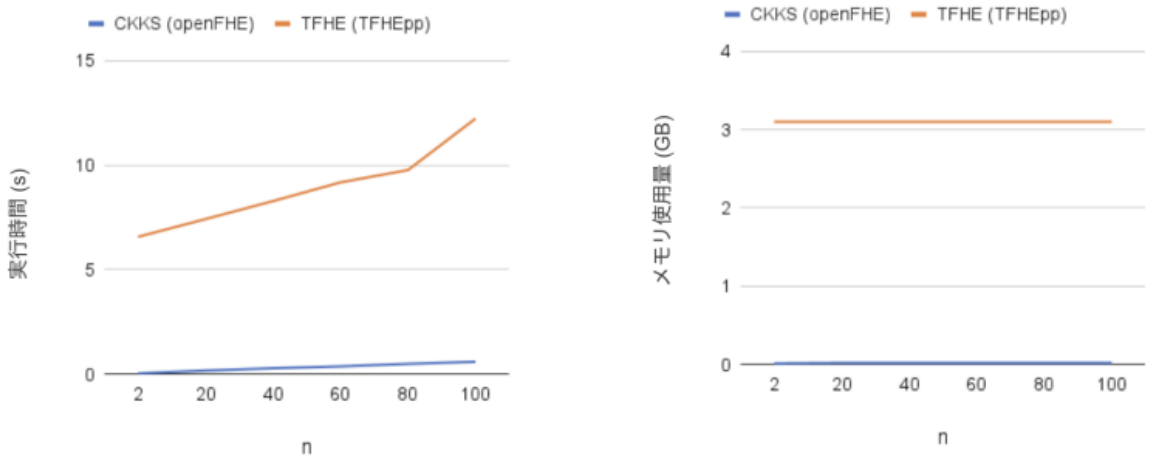
図 4　CKKS 方式と TFHE 亜種方式における式 (3) の時間空間計算量の比較

Fig. 4　Comparison of the execution status of Eq. (3) between CKKS and Zama's variant of TFHE.

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

CKKS 方式と TFHE 亜種方式の比較

- CKKS 方式は他の方式より多くのメモリを使用



(a) 実行時間　　(b) メモリ使用量

図 5　CKKS 方式と TFHE 亜種方式における式 (4) の時間空間計算量の比較

Fig. 5　Comparison of the execution status of Eq. (4) between CKKS and Zama's variant of TFHE.

5. 評価結果

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

目的

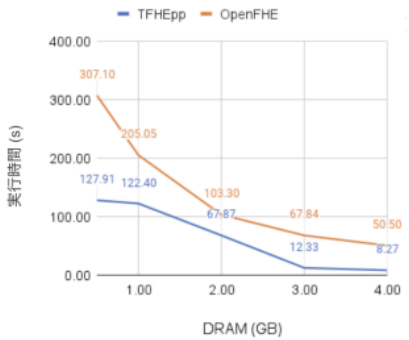
- DRAM 制限下の評価
  DRAM 容量制限が TFHE 亜種方式の性能に与える影響を評価。

5. 評価結果

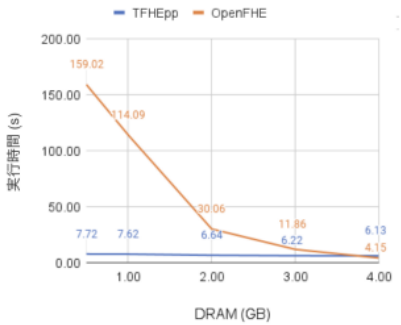
1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

DRAM 制限下の評価

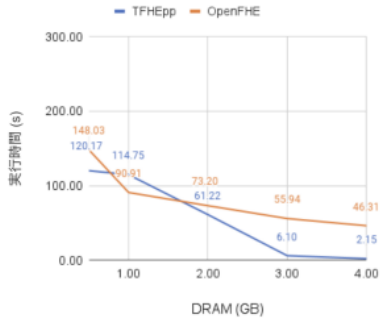
- TFHEpp は全ての DRAM 制限値で OpenFHE より高速。



(a) 総実行時間



(b) gate key の生成



(c) bootstrapping

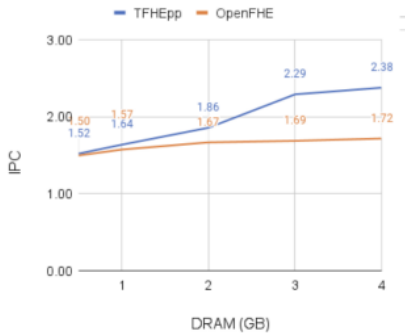
図 6 DRAM 容量が制限された際の TFHEpp および OpenFHE ライブラリにおける TFHE 亜種方式の実行時間

5. 評価結果

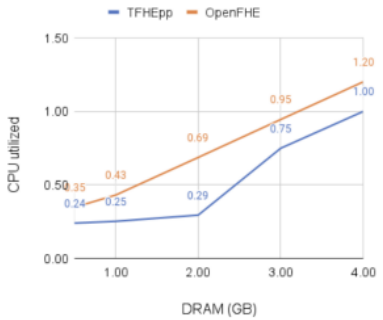
1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

DRAM 制限下の評価

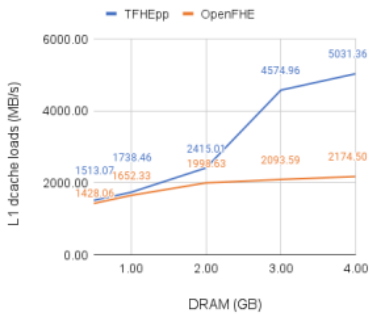
- DRAM 制限なしでは TFHEpp の効率が最も高い。



(a) IPC



(b) CPU utilized



(c) L1 Dcache load

図 7　DRAM 容量が制限された際の TFHEpp および OpenFHE ライブラリにおける TFHE 亜種方式の実装効率

6. 実アプリケーションでの検証

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

目的

- ゲノム秘匿検索
  実アプリケーションで FHE の性能を評価し、課題を特定。

手順

- 1. クライアントが検索クエリを暗号化し、サーバに送信。
- 2. サーバが暗号化された状態でデータベース検索を実施。
- 3. クライアントが結果を復号して取得。

6. 実アプリケーションでの検証

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

ゲノム秘匿検索

- Hyperthreading の有効/無効による影響

表 8　hyperthreading 有効時・無効時の比較

Table 8　Comparison between when hyperthreading is enabled and disabled.

	無効時（並列数 12）	有効時（並列数 12）	有効時（並列数 24）
総実行時間（s）	477.33	605.57	623.67
照合処理（s）	144.17	155.43	166.41
bootstrapping（s）	164.96	223.12	230.30
IPC	2.56	2.49	2.19

6. 実アプリケーションでの検証

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

ゲノム秘匿検索

- Rotation: 48.41%
  Bootstrapping: 35.43%
  → Rotation と Bootstrapping が全体の主要な負荷要因

表 8　hyperthreading 有効時・無効時の比較

Table 8　Comparison between when hyperthreading is enabled and disabled.

	無効時（並列数 12）	有効時（並列数 12）	有効時（並列数 24）
総実行時間（s）	477.33	605.57	623.67
照合処理（s）	144.17	155.43	166.41
bootstrapping（s）	164.96	223.12	230.30
IPC	2.56	2.49	2.19

6. 結論と今後の展望

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

結論

- 暗号方式とライブラリの比較により、最適な選択肢を提案。
- DRAM 制限下での TFHEpp の優位性を確認。
- 実アプリケーションでの課題を特定（Rotation と Bootstrapping が主な負荷要因）。

6. 結論と今後の展望

1. 背景と課題
2. 研究目的
3. 概要
4. 評価方法
5. 評価結果
6. 検証
7. まとめ

今後の展望

- ハードウェア最適化（GPU/FPGA）による性能向上の可能性。
- クラウド環境における FHE の実用化に向けた課題解決の必要性。