修士学位論文要約（令和４年３月）

確率的演算に基づく暗号化データ処理に関する研究

小関　隆介
指導教員：本間　尚文,　　研究指導教員：上野　嶺

Encrypted Data Processing based on Stochastic Computing

Ryusuke KOSEKI
Supervisor: Naofumi HOMMA,　Research Advisor: Rei UENO

How to construct HESC (HE for SC: Homomorphic Encryption for Stochastic Computing) that can evaluate SC on encrypted data has been investigated. The basic HESC integrates the additive or multiplicative HE (or Somewhat HE) with SC and can homomorphically evaluate both stochastic addition and multiplication without any bootstrapping. This implies that the HESC can be implemented with a low computational cost, equivalent or comparable to that of the combined additive/multiplicative HE or SHE, while presenting a higher arithmetic flexibility. We show the applications of HESC to secure computation of polynomial functions and to secure inference with a neural network. Through the evaluation results, the effectiveness of HESC compared to conventional HE in terms of accuracy and execution time will be demonstrated.

1.　はじめに

現在、プライバシーを保護したビックデータの利活用や委託計算を実現するために、データの暗号化状態を保持したまま処理が可能な準同型暗号が注目されている。準同型暗号は許容する演算によってその実装コストが大きく変化することが知られており、演算の自由度と効率性を両立する準同型暗号の開発は大きな課題である。そこで本研究では計算精度をある程度犠牲にしながらも計算効率の優れた演算方法であるストカスティック演算[1)]に着目し、ストカスティック演算を暗号化した状態で評価可能な暗号構成（以下、HESC と呼称）について、その構成方法を提案し、複数の応用例を用いて従来手法との比較評価を行った。また、ベースとなる準同型暗号に格子暗号を用いた HESC 向けの最適化手法を提案した。

2.　HESC の概要

ストカスティック演算は、バイナリ数をストカスティック数値と呼ばれる０，１の系列に変換することで、加算をビットパラレルな乱択、乗算をビットパラレルな XOR で実現することが可能である（IBP 表現されたものと仮定）。HESC の基本コンセプトはストカスティック数値をビット毎に準同型暗号で暗号化することで、これらの演算を暗号化したまま行うということである。アルゴリズム 1-4 に HESC の基本アルゴリズムを示す。なお、準同型暗号を HE と呼称し、HE は Enc（暗号化）、DEC（復号）、Eval（演算評価）の３つのアルゴリズムのタプルとみなしている。また、B2S、S2B はそれぞれバイナリ―ストカスティック数変換、ストカスティック―バイナリ数変換である。加算の際には暗号文を乱択すればよく、乗算の際には暗号文を足し合わせて復号後に２で剰余をとることで XOR を疑似的に再現する。すなわち評価の際に行っている演算は実質的に暗号文同士の加算のみである。よって加算だけが評価可能な、すなわち許容する演算の自由度が低い準同型暗号を用いて HESC を構成することが可能である。通常、準同型暗号を運用する際には演算の自由度を高めるために鍵長などのパラメータを大きくする、または、ブートストラップと呼ばれる演算コストが非常に大きな処理を必要とするが、HESC の場合は不要である。

Algorithm 1 HESC Encryption
Input: Plaintext P, Public key pk
Output: Ciphertext $C = (c_1, c_2, \ldots, c_L)$
function HESC.$\mathrm{ENC}_{\mathsf{pk}}(P)$
　$M \leftarrow \mathsf{B2S}(P)$; $\triangleright$ $M = (m_1, m_2, ..., m_L)$
　for i from 1 to L **do**
　　$c_i \leftarrow \mathsf{HE.Enc}_{\mathsf{pk}}(m_i)$;
　end for
　Return $C = (c_1, c_2, \ldots, c_L)$;
end function

Algorithm 2 HESC Decryption
Input: Ciphertext $C = (c_1, c_2, \ldots, c_L)$, Secret key sk
Output: Plaintext P
function HESC.$\mathrm{DEC}_{\mathsf{sk}}(C)$
　for i from 1 to L **do**
　　$m_i \leftarrow \mathsf{HE.Dec}_{\mathsf{sk}}(c_i)$;
　end for
　$M \leftarrow (m_1, m_2, ..., m_L)$; $P \leftarrow \mathsf{S2B}(M)$;
　Return P;
end function

Algorithm 3 HESC Multiplication
Input: Ciphertexts $C = (c_1, c_2, \ldots, c_L)$, $C' = (c'_1, c'_2, \ldots, c'_L)$
Output: $R = (r_1, r_2, \ldots, r_L)$
function HESC.$\mathrm{MULT}_{\mathsf{sto}}(C, C')$
　for i from 1 to L **do**
　　$r_i \leftarrow \mathsf{HE.Eval}(c_i, c'_i)$;
　end for
　Return $R = (r_1, r_2, \ldots, r_L)$;
end function

Algorithm 4 HESC Addition
Input: Ciphertexts $C = (c_1, c_2, \ldots, c_L)$, $C' = (c'_1, c'_2, \ldots, c'_L)$, Selection signal A
Output: $T = (t_1, t_2, \ldots, t_L)$
function HESC.$\mathrm{ADD}_{\mathsf{sto}}(C, C', A)$
　$S \leftarrow \mathsf{B2S}(A)$;　$\triangleright$ $S = (s_1, s_2, \ldots, s_L)$
　for i from 1 to L **do**
　　$t_i = s_i \cdot c_i + (1 - s_i) \cdot c'_i$
　end for
　Return $T = (t_1, t_2, \ldots, t_L)$;
end function

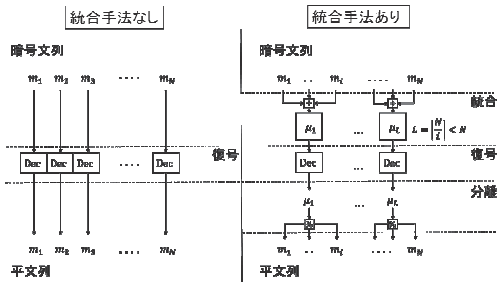

図１　暗号文統合手法の概要

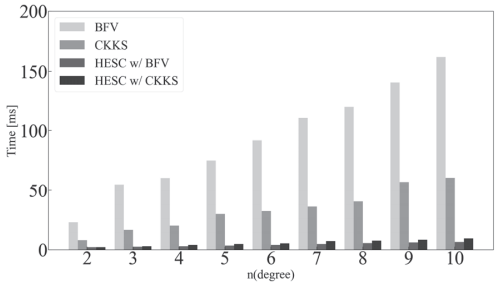

図２　多項式関数の評価結果

3.　格子暗号を用いた HESC

BFV やCKKS のような格子暗号に基づく HE は、パッキングと呼ばれるエンコード処理を行うことで複数の平文をまとめて暗号化することが可能である。また、複数の平文に対する準同型評価を１度の命令で実現可能である。これは SIMD 処理と呼ばれる。HESC で利用する HE に格子暗号を適用し、SIMD 処理を利用することで前述の基礎的構成よりも効率的なHESCの構成が可能である。一方で、複数のビットがまとめて暗号化されているため、アルゴリズム４のような暗号文列の乱択はビットの乱択に相当しない。すなわち従来のストカスティック加算であるビットの乱択に相当する演算を実現することは難しい。そこで、ストカスティック数値を連結することで加算を実現する連結型ストカスティック加算を提案する。連結型ストカスティック加算は従来の加算に比べて乱択による情報落ちがなく、従来問題視されてきた加算による精度の著しい劣化が存在しない。

しかし、加算の評価の際には語長（暗号文長）が延びていくことになるため、行列演算のような高次の演算を評価する際には暗号文数が著しく増大し、実行時間的観点から復号命令がボトルネックとなり得る。そこで増大した暗号文を統合することで復号命令の回数を削減する暗号文統合手法を提案する（図 1)。

連結型ストカスティック加算、暗号文統合手法という２つの最適化手法を導入することで、格子暗号を用いた HESC においても効率的にストカスティック加算・乗算の評価をすることを可能にしている。

4.　HESC の有効性

第一に、HESC の基礎的構成と格子暗号を用いた構成の比較を行った。結果を表１に示す。最左端

表１　１ビット当たりの実行時間　［μｓ］

	Packing	Encryption	Stochastic Add.	Stochastic Mult.	Decryption	Unpacking
GM	-	2.92×10^{1}	7.03×10^{-1}	5.73	4.14×10^{1}	-
ElGamal	-	1.40×10^{4}	1.40	1.19×10^{1}	1.42×10^{4}	-
ECElGamal	-	1.90×10^{3}	7.90×10^{-1}	1.21×10^{1}	1.94×10^{3}	-
BFV	1.63×10^{-2}	1.50×10^{-1}	5.79×10^{-4}	2.76×10^{-3}	6.53×10^{-2}	1.70×10^{-2}
CKKS	5.92×10^{-2}	1.30×10^{-1}	5.70×10^{-4}	2.46×10^{-3}	6.76×10^{-3}	5.69×10^{-2}

の列がベースとなる HE を表し、上側３つが基礎的構成、下側２つが格子暗号を用いた構成である。格子暗号を用いることで複数の平文を纏めて暗号化可能であるため、スループットが向上し、実行時間的に優位になる。

第二に、多項式関数の評価と、アヤメ推論（ニューラルネットワークを用いて解く分類問題の一種）の評価に HESC を応用し、従来の HE との比較評価を行った。まず多項式関数の評価についてだが、図２は 10 次関数までの実行時間を比較した結果である。横軸の値（次数）n に対し、0 からｎ次まですべての係数が１の多項式を評価したときの実行時間を表している。高次になるほど乗算の深さが必要であり、従来の HE の場合は巨大なパラメータが必要になるため HESC は実行時間的に有利である。次にアヤメ分類問題の秘匿推論を通して比較評価を行った。比較対象は図２同様の４つのスキームである。HESC に構成することで BFV 単体と比較して約 10.1 倍、CKKS 単体と比較して約2.4倍の高速化を達成していることが確認できた。最後に暗号文統合手法の検証を行った。演算回数自体は増えるためパラメータは大きくなるものの、実行時間は約 50 パーセントの削減を確認できた。パラメータを大きくすることで演算精度の向上も見込めるため、暗号文統合手法は演算精度と演算時間の２つに寄与している手法であるといえる。

5.　まとめ

確率的演算を暗号文上で評価可能な HESC を提案し、その関連手法を提案した。またそれらの評価を行った。評価結果により、従来の HE と比較して HESC の実行時間的な優位性を示した。

文献

1) B. R. Gaines, “Stochastic computing systems,” in Advances in information systems science. Springer, 1969, pp. 37-172.