完全準同型暗号の実装

中村 蓉子　(指導教員: 金子 晃)

1　はじめに

完全準同型暗号とは, 任意の論理ゲートあるいは環の多項式演算を, 平文に対してと同様に暗号文だけを用いて実行できるという暗号である.

現在, クラウドコンピューティングが普及し, 自分の情報を相手に知られることなく計算だけ行ってもらうという秘密計算の必要性が高まっている. その秘密計算を行うに当たって完全準同型暗号は有効な手段になるものであると期待されている.

本研究では, 完全準同型暗号の仕組みを解説し, そして, その実装を試みる.

2　準同型暗号とは

安全性を満足した最もシンプルな暗号スキームの一つとして, 以下の様な共通鍵暗号スキームが挙げられる.

KeyGen: 鍵は, η について予め設定された区間 $[2^{\eta-1}, 2^{\eta}]$ からランダムに選ばれた奇数整数 p である.

Encrypt(p, m): ビット $m \in \{0, 1\}$ を暗号化する. 暗号文 c を p の剰余が平文と同じ偶奇性を持つ整数とする. すなわち, $c = pq + 2r + m$ とする. そこでは, 整数 q, r は他の予め設定された区間の中からランダムに選び, $2r$ が絶対値 $p/2$ より小さいようにする.

Decrypt(p, c): $(c \bmod p) \bmod 2$ を出力する.

このシンプルなスキームは, 加算と乗算の両方, あるいは, それらを少数組み合わせたゲートで準同型となる. (すなわち, "いくらか準同型" 性を持つ.) しかし, 深いゲートに対しては, ノイズ (r の累積) が p を超えてしまうため正しく復号できず, 準同型性が崩れてしまう. すなわち, "完全準同型" とはなっていない.

また, p を相手に渡すのは実用的ではない. つまり, 公開鍵暗号化が望まれる.

Gentry[1] は "いくらか準同型な暗号" を "完全準同型暗号" に変換する仕組みを発見し, さらに, Marten van Dijk, Craig Gentry, Shai Halevi, Vinod Vaikuntanathan[2] が, 完全準同型性を持つ整数環上の具体的な公開暗号系を提案した.

3　準同型暗号の厳密な定義

定義 3.1 (適切な準同型復号) スキーム $\mathcal{E} = (\mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt}, \mathbf{Evaluate})$ が t 個の入力を持つ与えられた回路 C に対して "適切である" とは, $\mathbf{KeyGen}(\lambda)$ によるどんな鍵のペア (sk,pk) の出力や, どんな t 個の平文ビット $m_1, \ldots, m_t$ や, $c_i \leftarrow \mathbf{Encrypt}_{\mathcal{E}}(\mathrm{pk}, m_i)$ によるどんな暗号文 $\vec{c} = \langle c_1, \ldots, c_t \rangle$ に対しても, $\mathbf{Decrypt}(\mathrm{sk}, \mathbf{Evaluate}(\mathrm{pk}, C, \vec{c})) = C(m_1, \ldots, m_t)$ が成り立つことである.

定義 3.2 (準同型暗号) $\mathcal{C}$ をある回路のクラスとする. スキーム $\mathcal{E} = (\mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt}, \mathbf{Evaluate})$ は, $C \in \mathcal{C}$ となる全ての回路で適切であるとき, $\mathcal{C}$ に対して "準同型" であるという. さらに, 全ての boolean 回路に対して適切であるならば, $\mathcal{E}$ は "完全準同型" であるという.

以下, スキーム $\mathcal{E}$ に対して準同型となる回路の集合を $\mathcal{C}_{\mathcal{E}}$ と記す.

4　ブートストラップ可能な暗号化

それ自身の復号回路よりもほんの少しだけ大きな回路を計算することができる回路から, あらゆる深さの回路に対する準同型暗号を構成する.

定義 3.3 (増強された復号回路) $\mathcal{E} = (\mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt}, \mathbf{Evaluate})$ を暗号スキームとせよ. そこでは復号はセキュリティパラメータのみに依存する回路によって, 実行される.

セキュリティパラメータ λ が与えられたとき, 増強された復号回路の集合は二つの回路からなる. どちらも入力として秘密鍵と, 二つの暗号文をとる. 一つ目の回路は, 二つの暗号文を復号し, 得られた平文を mod 2 で加算する. もう一つの回路は, 二つの暗号文を復号し, 得られた平文を mod 2 で乗算する. この集合を $D_{\mathcal{E}}(\lambda)$ と記す.

定義 3.4 (ブートストラップ可能な暗号化) $\mathcal{E} = (\mathbf{KeyGen}, \mathbf{Encrypt}, \mathbf{Decrypt}, \mathbf{Evaluate})$ を準同型暗号スキームとせよ. そして, 各セキュリティパラメータ λ に対して, $\mathcal{C}_{\mathcal{E}}(\lambda)$ を $\mathcal{E}$ が適切な回路の集合とせよ.

どんな λ に対しても, $D_{\mathcal{E}}(\lambda) \subseteq \mathcal{C}_{\mathcal{E}}(\lambda)$ が成り立つとき, $\mathcal{E}$ は "ブートストラップ可能" であるという.

定理 3.5 ブートストラップ可能なスキーム $\mathcal{E}$ や, パラメータ $d = d(\lambda)$ が与えられたとき, 次のような別の暗号スキーム $\mathcal{E}^{(d)}$ を出力する変換が存在する.

1. $\mathcal{E}^{(d)}$ は, コンパクトである. (特に, $\mathcal{E}^{(d)}$ の復号回路は, $\mathcal{E}$ の復号回路に等しい)
2. $\mathcal{E}^{(d)}$ は, d までの深さの全ての回路に対して, 準同型である.

もし, ブートストラップ可能なスキーム $\mathcal{E}$ が "circular secure" ならば, 一つのコンパクトな完全準同型暗号スキーム $\mathcal{E}'$ に変換できることが知られている.

5　暗号スキームの構成

5.1　いくらか準同型な暗号スキーム

【パラメータ】

γ: 公開鍵のビット長, η: 秘密鍵のビット長

ρ: ノイズのビット長, τ: 公開鍵の個数

正当性の条件から, パラメータの制限がつく. 例えば, 以下のように設定すれば, 条件は満たされる.

$$\rho = \lambda, \rho' = 2 * \lambda, \eta = \tilde{O}(\lambda^2), \gamma = \tilde{O}(\lambda^5), \tau = \gamma + \lambda$$

η-bit の正の奇数整数 p から, γ-bit 整数を生成するために次の分布を使う.

$$D_{\gamma,\rho}(p) = \{\text{choose } q \xleftarrow{R} \mathbf{Z} \cap [0, 2^{\gamma}/p), \\ r \xleftarrow{R} \mathbf{Z} \cap (-2^{\rho}, 2^{\rho}) : \text{output } x = pq + r\}$$

【構成】

KeyGen(λ). 秘密鍵は η-bit の奇数整数: $p \xleftarrow{R} (2\mathbf{Z} + 1) \cap [2^{\eta-1}, 2^{\eta})$. 公開鍵を生成するために, $i = 0, \ldots, \tau$ に対して, $x_i \xleftarrow{R} D_{\gamma,\rho}(p)$ を行う. そして, x_0 が最も大

きくなるように並べ替える. x_0 が奇数で, x_0 の p の剰余が偶数にならないならば, やり直す.

公開鍵は pk=$\langle x_0, x_1, \ldots, x_\tau \rangle$ である.

Encrypt(**pk**, $m \in \{0,1\}$). ランダムな部分集合 $S \subseteq \{1, 2, \ldots, \tau\}$ と, $(-2^{\rho'}, 2^{\rho'})$ 間のランダムな整数 r を選ぶ. そして, $c \leftarrow [m + 2r + 2\sum_{i \in S} x_i]_{x_0}$ を出力する. ただし, $[a]_b = a \bmod b$ である.

Evaluate(**pk**, $C, c_1, \ldots, c_t$). t 個の入力と t 個の暗号文 c_i で, (二進数の) 回路 $C \in \mathcal{C}_\mathcal{E}$ が与えられたとき, 暗号文に対して C の加算と乗算ゲートを適用し, 結果の整数を返す.

Decrypt(**sk**, c). $m' \leftarrow (c \bmod p) \bmod 2$ を出力する.

【正当性】

補題 4.1.1 (sk,pk) を **KeyGen**(λ) による出力とし, $c \xleftarrow{R}$ **Encrypt**(pk, m) for $m \in \{0,1\}$ とせよ. そのとき, $|2b + m| \leq \tau 2^{\rho+3} (<< p)$ となるある整数 a, b に対して, $c = a \cdot p + (2b + m)$ となる.

補題 4.1.2 (sk,pk) を **KeyGen** による出力とし, $C \in C_\mathcal{E}$ を t 入力, 1 出力の回路とせよ. また, $i \in \{1, \ldots, t\}$, $m_i \in \{0, 1\}$ のとき, $c_i \xleftarrow{R}$ **Encrypt** (pk, m_i) とせよ. さらに, $m \leftarrow C$ (m_1, ..., m_t), $c \leftarrow$ **Evaluate** (pk, C, c_1, ..., c_t) とせよ. そのとき, $|2b + m| < p/8$ となるある整数 a, b に関して, $c = a \cdot p + (2b + m)$ となる.

補題 4.1.3 上のスキームは $\mathcal{C}_\mathcal{E}$ に対して適切である.

補題 4.1.1 と 4.1.2 から補題 4.1.3 が直ちに得られる. つまり, $C_\mathcal{E}$ のどんな回路や, その回路への入力のどんな暗号化に対しても, **Evaluate** による整数の出力は, $2b + m \leq p/8$ で $c = a \cdot p + (2b + m)$ の形をしている. (m は, c に暗号化されると思われる平文である.) 従って, $[c]_p = 2b + m$ となり, ゆえに $m = \Big[[c]_p\Big]_2$ である.

5.2 完全準同型暗号スキーム

定義 3.4 に従って, "いくらか準同型な暗号スキーム" からブートストラップ可能な "完全準同型暗号スキーム" を作る. 復号アルゴリズムを計算することは, "いくらか準同型な暗号スキーム" が扱えるよりも深い boolean 回路が要求されるからである. 従って, "復号回路の squashing" という変換を用いる. この変換では, 公開鍵に秘密鍵に関するいくつかの付加情報を加える. そして, その付加情報を暗号文の "後処理" のために用いる. 後処理された暗号文は, 元の暗号文よりも効率的に復号できる. こうして, ブートストラップ可能なスキームを作る.

【パラメータ】 $\kappa = \gamma\eta/\rho'$, $\theta = \lambda$, $\Theta = \omega(\kappa \cdot \log \lambda)$

【構成】

KeyGen. sk*=p と pk*を以前と同様に生成する. $x_p \leftarrow \lfloor 2^\kappa / p \rceil$ とする. ただし, $\lfloor\ \rceil$ は四捨五入を表す. そして, ハミング重みが θ となるようなランダムな Θ-bit のベクトル $\vec{s} = \langle s_1, \ldots, s_\Theta \rangle$ を選ぶ. $S = \{i : s_i = 1\}$ とする.

$i = 1, \ldots, \Theta$ に対して、ランダムな整数 $u_i \in \mathbf{Z} \cap [0, 2^{\kappa+1})$ を選ぶ. ただし, $\sum_{i \in S} u_i = x_p (\mathrm{mod} 2^{\kappa+1})$ という条件に従う. $y_i = u_i / 2^\kappa, \vec{y} = \{y_1, \ldots, y_\Theta\}$ とする. それ故, それぞれの y_i は, 二進の小数点以下の精度が κ ビットとなる 2 以下の正の整数となる. また, $\sum_{i \in S} y_i = (1/p) - \Delta_p$ ($|\Delta_p| < 2^{-\kappa}$) となる.

秘密鍵 sk=(sk*, $\vec{s}$) と, 公開鍵 pk=(pk*, $\vec{y}$) を出力する.

Encrypt and **Evaluate**. 暗号文 c*を以前と同様に生成する. $i \in \{1, \ldots, \Theta\}$ に対して, $z_i \leftarrow [c^* \cdot y_i]$ とする. それぞれの z_i に対して, 2 進の小数点以下の精度の n=$\lceil \log \theta \rceil + 3$ ビットだけを保存する. c*と $\vec{z} = \langle z_1, \ldots, z_\Theta \rangle$ を出力する.

Decrypt. $m' \leftarrow [c^* - \lfloor \sum_i s_i z_i \rceil]_2$ を出力する.

定理 4.2.1 $\mathcal{E}$ を上記のスキームとせよ. さらに, $D_\mathcal{E}$ を増強され squash された復号回路の集合とせよ.

その時, $D_\mathcal{E} \subset C(\mathcal{P}_\mathcal{E})$ となる.

言い換えると, $\mathcal{E}$ は "ブートストラップ可能" である. 定理 3.5 から, どんな深さの回路に対しても準同型となる暗号スキームを得る.

6 実装とその結果

gmp(GNU 任意精度計算ライブラリ) を用いた C 言語で実装した. 回路 C は, 最初の演算が乗算で, 残りが加算の回路を用いた. 平文の個数は t=4 とした.

【開発環境】以下の二つで実験した.

(1)diamond:Opteron2431 2.4GHz 6 core×2,
メモリー 32GB, OS:CentOS 5.3

(2)tornado2:AMD Phenom 9950 2.6GHz 4 core,
メモリー 8GB, OS:Ubuntu10.04

メモリの減量化に努めた結果, 実行可能な λ の最大値と実行時間 (秒) は以下の様になった.

6.1 いくらか準同型な暗号スキーム (diamond)

【パラメータ】$\lambda = 14$ 【メモリ使用量】32GB
【公開鍵のビット長】537824 【公開鍵の個数】537838
【実行時間】[**Key**]290.99621820,[**Enc**]2711.85526084,
[**Eval**] 0.00911092,[**Dec**] 0.00003719

6.2 完全準同型暗号スキーム (tornado2)

配列をディスクに書き出して行った.
【パラメータ】$\lambda = 9$ 【メモリ使用量】1.6GB
【公開鍵】59049bit→59058 個, 531442bit→583846 個
【実行時間】[**Key**]3528.56734800,[**Dec**]0.13150716
[**Enc**&**Eval**]5335.13199997

7 まとめと今後の課題

"いくらか準同型な暗号スキーム" を実装し, それを修正して "完全準同型暗号スキーム" を実装した. 暗号化した結果を相手に渡し, 戻ってきた結果を復号すると, 正しい計算値になることがわかった. しかし, 現時点では λ の値が小さく, 実用的ではない.

今後の課題は, スキームの効率性を改善することである. 実用化されれば, クラウドコンピューティングなどで大きな応用が期待される.

参考文献

[1] Craig Gentry:"Fully Homomorphic Encryption Using Ideal Lattices",(June.2009).

[2] Marten van Dijk,Craig Gentry,Shai Halevi and Vinod Vaikuntanathan:"Fully Homomorphic Encryption over the Integers",(June.2010).