［安全なデータ活用を実現する秘密計算技術］

⑤準同型暗号を用いた秘密計算技術と実用化に向けた活動

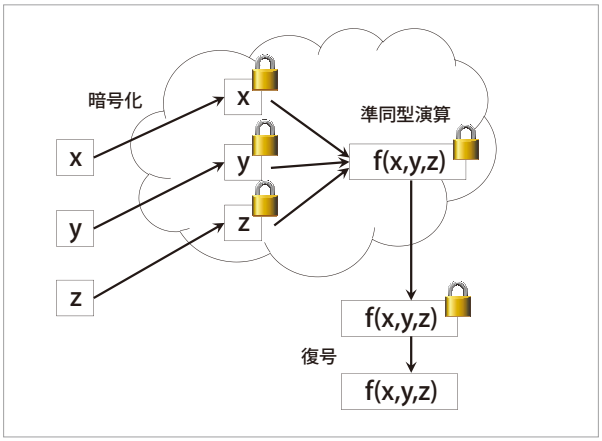
佐久間淳 筑波大学／理化学研究所
陸　文傑 筑波大学

データ解析におけるプライバシ

計算資源のクラウドへの集約が進むにつれ，機密情報やパーソナルデータのクラウドにおける保管とその活用は，重要な関心事になりつつある．データベースのオーナーは，必要な情報を単純に暗号化してクラウドに保管すれば，プライバシの問題は解決するように見える．データを利用するたびに暗号化データをダウンロードし，オーナーと手元でデータを復号化することにすれば，暗号化状態にないデータがクラウド上に保管されることは一切ないからである．しかしユーザの過去のメールから適切な広告を表示する，患者の過去の医療履歴から将来の健康上の問題を予測するなど，データを活用した計算を伴う処理をクラウドに丸ごと委託するためには，その処理の途中において暗号化された情報をクラウド上で復号する必要がある．これでは，データを利用するたびに暗号化状態にないデータがクラウド上に存在することになり，その都度プライバシが危険にさらされる．

準同型暗号は，データを暗号化したときに，その暗号化データを復号せずに暗号化したまま計算を実行できる性質を持つ暗号系である．究極的には，データベースのオーナーがそのデータベースを準同型暗号で丸ごと暗号化しクラウドに設置すれば，クラウドは要求された情報処理を end-to-end で暗号化したまま実行することができる．情報処理の結果もまた暗号化状態のまま，オーナーに届けられ，暗号を復号できる鍵情報を持つオーナーのみがその処理結果を知ることができる．この一連の過程において，すべての情報は暗号化状態のまま処理されるため，オーナー以外のすべての者に，元のデータについて情報が知られることはない（図 -1）．

準同型暗号は秘密データの委託におけるプライバシ保護の問題を理想に近い形で解決することができるアプローチとして期待されている．登場当初の準同型暗号は計算効率性の悪さや消費する計算資源が膨大であることから，実用性に乏しい理論上の存在と考えられていたが，ここ 10 年の研究の発展により，その実用性は徐々に高まりつつあるといえる．本稿では，準同型暗号の最近の発展を紹介するとともに，その応用可能性を紹介する．



■図 -1　準同型暗号を用いたクラウド上での計算

完全準同型暗号とは

公開鍵暗号系の公開鍵および秘密鍵（復号鍵）のペアを（pk, sk）とする．復号鍵 sk で復号できるようなメッセージ m の暗号化を $\mathrm{ctx}_{\mathrm{sk}}(m)$，その暗号文を復号する関数を $m=\mathcal{F}(\mathrm{ctx}_{\mathrm{sk}}(m), \mathrm{sk})$ と書くことにする．

この暗号系が準同型性を持つとき，2 つのメッセージに関する暗号文 $\mathrm{ctx}_{\mathrm{sk}}(m_1)$, $\mathrm{ctx}_{\mathrm{sk}}(m_2)$ から，そのメッセージや秘密鍵に関係する情報なしに，加法 m_1+m_2 に対応する暗号文 $\mathrm{ctx}_{\mathrm{sk}}(m_1+m_2)$ や乗法 $m_1\times m_2$ に対応する暗号文 $\mathrm{ctx}_{\mathrm{sk}}(m_1\times m_2)$ を得ることができる（図 -2）．ここでは暗号文同士の加法を

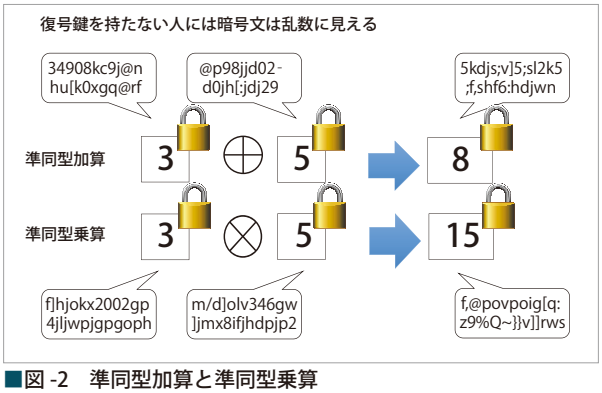
$$\mathrm{ctx}_{\mathrm{sk}}(m_1)\oplus\mathrm{ctx}_{\mathrm{sk}}(m_2)=\mathrm{ctx}_{\mathrm{sk}}(m_1+m_2), \quad (1)$$

ここでは暗号文同士の乗法を

$$\mathrm{ctx}_{\mathrm{sk}}(m_1)\otimes\mathrm{ctx}_{\mathrm{sk}}(m_2)=\mathrm{ctx}_{\mathrm{sk}}(m_1\times m_2), \quad (2)$$

と書くことにする．

加法的準同型性の性質を持つ準同型暗号を加法準同型暗号と呼ぶ．また乗法的準同型性の性質を持つ準同型暗号を乗法準同型暗号と呼ぶ．加法準同型性と乗法準同型性の両方の性質を持つ準同型暗号を完全準同型暗号 (Fully Homomorphic Encryption, FHE) と呼ぶ．乗法準同型性の性質を持つ準同型暗号は数多く知られていた．また加法準同型性を持つ公開鍵暗号系は比較的限られるが，Paillier 暗号がよく知られている．一方，FHE の実現は長年の未解決問題であった．FHE はメッセージを暗号化したまま任意の論理回路を評価できるという優れた性質[☆1]を持つことから，長年に渡り精力的に研究が続けられ，2009 年に Gentry によって FHE を実現する方法が初めて示された[1]．



■図 -2　準同型加算と準同型乗算

完全準同型暗号のアイディア

FHE の実現にあたり，Gentry は FHE の構成を 2 つの部分問題に分解した．1 つ目の問題は FHE を構成するかわりに，「まあまあ」準同型な暗号（Somewhat Homomorphic Encryption, SwHE）を構成することである．ここでいう「まあまあ」とは，事実上任意の回数の加法準同型演算が実行できるが，乗法準同型演算はある限られた回数しか実行できないような準同型暗号を意味している．ここでは乗法準同型演算が実行可能な回数を L と表すことにする．SwHE から FHE を構築する上で重要な性質として，SwHE の復号関数 D_{sk} が，ある低次元の多項式で表現される必要がある．具体的には，復号関数の多項式 $\mathcal{F}_{\mathrm{dec}}$ について，その次数は $\mathrm{degree}(\mathcal{F}_{\mathrm{dec}})+1\leq L$ であるものとする．なぜこの性質が必要なのかは，2 番目の問題の構成にかかわっている．

2 番目のステップは，***bootstrapping*** と呼ばれるアルゴリズムの構成である．bootstrapping を説明するために，暗号文を“物体が中に入った鍵のかかった箱”に例えよう．この箱は，箱を開ける鍵を持つ者のみがその中身を取り出すことができる．SwHE は，鍵を開けて箱の中の物体を取り出さずとも中の物体を操作できる魔法の箱に例えることができる．ここでいう操作とは，加法や乗法などの演算に対応する．このとき，魔法の箱 A を開ける鍵を，別の魔法の箱 B に入れたらどうなるだろうか．魔法の箱 B は中に入っている物体を，箱 B の鍵を開けることなく操作することができるため，箱

☆1　任意の論理関数を表現できる少数の基本論理関数の集合を完備集合と呼ぶ．mod 2 において加算，乗算が評価できる FHE は完備集合に含まれる基本論理関数の集合を評価することができるため，メッセージを暗号化したまま，任意の論理回路を評価できるといえる．

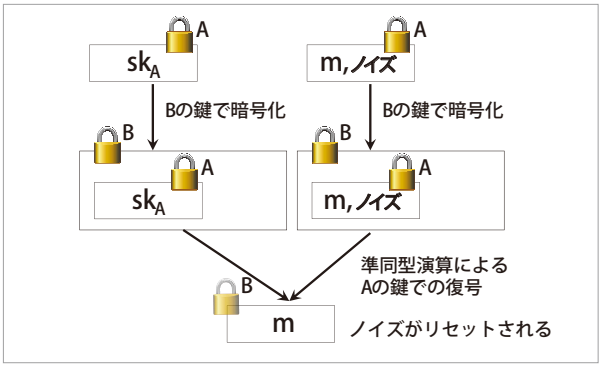
B の中に，鍵のかかった箱 A と箱 A の鍵を入れておけば，箱 B の中に箱 A を入れたまま，箱 A の鍵を開け中のものを取り出すことができる．このときに，箱 B を操作している人は，1）箱 A の中にどんなものが入っていたか，2）箱 A を開ける鍵は何か，3）箱 B を開ける鍵は何か，を一切知ることなく，箱 B の内部で箱 A の鍵を開けることができることに注意されたい．

Bootstrapping はこの「箱の中の箱」のアイディアを用いたアルゴリズムである（図 -3）．復号鍵 sk_A で復号できるメッセージ m の暗号文を $\mathsf{ctx}_{\mathsf{sk}_A}(m)$ とする．Bootstrapping では，まずこれを別の復号鍵 sk_B に対応した暗号鍵で暗号化し，$\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{ctx}_{\mathsf{sk}_A}(m))$ を得る．さらに，暗号化された復号鍵 $\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{sk}_A)$ を用いて，復号のための多項式 $\mathcal{F}_{\mathsf{dec}}$ を $\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{ctx}_{\mathsf{sk}_A}(m))$ および $\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{sk}_A)$ を準同型演算を用いて評価する：

$$\mathsf{Bootstrapping}\,(\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{ctx}_{\mathsf{sk}_A}(m)),\,\mathsf{ctx}_{\mathsf{sk}_B}(\mathsf{sk}_A)) \quad (3)$$
$$\rightarrow \mathsf{ctx}_{\mathsf{sk}_B}(\mathcal{F}_{\mathsf{dec}}(\mathsf{ctx}_{\mathsf{sk}_A}(m),\,\mathsf{sk}_A)) = \mathsf{ctx}_{\mathsf{sk}_B}(m).$$

ここで，sk_B に対応した SwHE の暗号文を復号せずに，高々 L 回の準同型演算のみで sk_B に対応する暗号文を復号するために，SwHE の暗号化の復号関数が次数 L 以下であるという性質が必要になる．これによって，メッセージ m の暗号文 $\mathsf{ctx}_{\mathsf{sk}_A}(m)$ の鍵を別の鍵を用いた暗号文 $\mathsf{ctx}_{\mathsf{sk}_B}(m)$ にスイッチすることができた．

Gentry は Bootstrapping を用いて SwHE から FHE を実現できることを示した．直感的な言い方をすると，Gentry らが示した方法では，乗法準同型演算を行うたびに暗号文にノイズ（図 -3 の m の暗号文に含まれるノイズに相当）が蓄積され，ノイズの影響が限度を越えると，その暗号文を正しく復号することができなくなる☆2．ただし，Bootstrapping を用いて鍵をスイッチすることでこのノイズの影響をリセットすることができる．SwHE の暗号文に対して乗算を多数回実行する場合，ノイズの影響で正しく復号ができなくなる前に，Bootstrapping でノイズの影響をリセットすることで，事実上任意の回数の乗算ができることになる．これは FHE の実現にほかならない．



■図 -3　bootstrapping の概念図

完全準同型暗号の発展

最初の完全準同型暗号は前章に示した文献 1）において示された．完全準同型暗号の具体的な実現方法を示したという意味で大きなブレイクスルーではあったが，1bit のメッセージ (i.e., $m \in \{0, 1\}$) の暗号文に対して，bootstrapping の計算時間は 30 分もかかることが報告されており，実用性には乏しいものであった．その後，完全準同型暗号の実用性を改善する多くの手法が提案された．ここでは，その中で主要な 3 つの FHE の最適化を紹介する．

1 つ目は modulus switching と呼ばれる手法である [2]．Gentry の当初のアイディア [1] では，次数 L の多項式に基づく SwHE の実現に，指数的に大きな個数のパラメータ $\mathcal{O}(\exp(L))$ を持つ暗号文を必要とした．modulus switching は，これよりはるかに少ない多項式個のパラメータ $\mathcal{O}(\mathrm{poly}(L))$ を持つ暗号文で SwHE を実現する手法である．これによって，より小さいサイズの暗号文で実行可能な準同型暗号演算の数を増やすことができるようになった．

2 つ目は Ring Learning-With-Error（RLWE）仮定 [3] に基づく SwHE の実現である．文献 3）以前の準同

☆2　加法準同型演算でもノイズは蓄積されるが，その影響は乗法に比べ非常に小さく事実上無視できる．

型暗号は Learning-With-Error（LWE）仮定[4]に基づくものであった．LWE 仮定における SwHE の構成は行列ベクトル演算に基づいており，その時間計算量は $\mathcal{O}(n^2)$ となる．一方，RLWE 仮定における SwHE の構成は多項式の積に基づく．多項式積の計算は高速フーリエ変換を用いることで $\mathcal{O}(n \log n)$ で実行することができ，計算効率性が優れている．

3 つ目はパッキングあるいはバッチングと呼ばれる RLWE に基づく準同型暗号のためのテクニックである．パッキングは ℓ 個のメッセージ（たとえば ℓ =4096）を 1 つの暗号文に埋め込む手法である．このとき，多数のメッセージをパッキングした暗号文に準同型演算を適用すれば，追加コストなしに個別のメッセージに独立に準同型演算を適用されるようにパッキングを設計することがポイントである．このように設計されたパッキングは個別の準同型演算を効率化することはできないが，多数のメッセージに対して準同型演算を適用する場合には，メッセージ 1 つあたりに必要な計算コストを削減することができることに注意されたい．文献 5）は，ℓ =960 個の 24-bit のメッセージの bootstrapping が 7 分で実行できることを報告している．これは 1bit あたり 18ms で bootstrapping ができることに相当する．

完全準同型暗号による計算

すでに述べたように，FHE は暗号文上での任意の論理回路評価を可能にする．しかし FHE はメッセージ空間に対して鍵のサイズや暗号文のサイズが大きいことが知られており，特に準同型乗算と bootstrapping を用いた演算には大きなオーバヘッドがかかる．このことから，準同型暗号を用いた，大規模入力に対する段数の多い論理回路の評価は，今のところあまり現実的ではない☆3．

一方で，対象の計算がベクトルや行列に対する算術演算で記述される場合には，パッキングの技術を用いることで比較的規模の大きな計算も現実的な計算資源で計算を行うことができる．そのような例の 1 つとして，2 つの変数に対するカイ二乗独立性検定を準同型暗号上で計算するアルゴリズムを取り上げる．

カイ二乗独立性検定の例として「ある遺伝子を持つ者は，肺がんに罹患するリスクが高い」という仮説が正しいかどうかについての疫学調査を取り上げる．ある母集団において「肺がんを罹患している群」（ケース群）と「肺がんを罹患していない群」（コントロール群）を考える．また肺がんの罹患に関連が深いと予想される要因として，個人ごとに異なる遺伝的特徴の有無を考える．この「肺がんの罹患の有無」と「ある遺伝的特徴の有無」が有意に関連しているかどうかを統計的検定により調査する．

N 人の被験者について，それぞれの被験者がケース群かコントロール群かを ℓ 次元のバイナリベクトル $\vec{u}$ で表現することにする（ケース群が 1）．同様に，それぞれの被験者がある遺伝的特徴を持っているか否かを ℓ 次元のバイナリベクトル $\vec{v}$ で表現することにする（ある遺伝的特徴を有すれば 1）．ここではカイ二乗独立性検定の内容には立ち入らないが，ケース群であり，かつ，ある遺伝的特徴を有する被験者の数を求めることができれば，カイ二乗独立性検定のための検定統計量を求めることができるものとする．この数は 2 つのベクトルの内積 $\vec{u} \cdot \vec{v}$ で求めることができる．

この内積計算をナイーブに準同型暗号上で実行する場合には，$u_i, v_i (i=1, \ldots \ell)$ それぞれの値について 2ℓ 個の暗号文を生成し，以下の計算を行う必要がある．

$$\mathrm{ctx}_{\mathrm{sk}}\left(\vec{u} \cdot \vec{v}\right) = \prod_{i=1}^{\ell} \mathrm{ctx}_{\mathrm{sk}}\left(u_i\right) \oplus \mathrm{ctx}_{\mathrm{sk}}\left(v_i\right) \tag{4}$$

ここで $\prod_{i=1}^{\ell}$ は準同型乗算 $\otimes$ について定義される繰り返し計算である．この方法では，2ℓ 個の暗号文について，ℓ 回の準同型乗算と準同型加算が必要になる．

☆3 ただし，次の章で説明するように論理回路評価に適した完全準同型暗号も発表されつつあり，今後の発展が期待される．

一方で，RLWE に基づく FHE において文献 6）で示されたパッキング手法を用いると，暗号文の個数と準同型演算の回数を削減することができる（図 -4）．RLWE に基づく FHE では，メッセージが多項式で表現される．このため，多項式の次元が ℓ より大きい場合には，1 つの暗号文にベクトルを丸ごと埋め込むことができる．ここでは x を変数とする以下のような 2 通りのパッキングを用いる．

$$\rho_{\mathrm{fw}}(\vec{u}) := \sum_{i=0}^{\ell-1} u_i x^i, \qquad \rho_{\mathrm{bw}}(\vec{v}) := -\sum_{j=0}^{\ell-1} v_j x^{n-j}. \tag{5}$$

$\vec{u}$ は 0 次から $\ell - 1$ 次に順に，$\vec{v}$ はその逆順に，多項式の係数に情報がパッキングされている．

この多項式表現されたベクトルの暗号文について，準同型乗算を行うことで以下を得る．

$$\begin{aligned}
&\mathrm{ctx}_{\mathrm{sk}}\left(\rho_{\mathrm{fw}}(\vec{u}) \otimes \mathrm{ctx}_{\mathrm{sk}}\rho_{\mathrm{bw}}(\vec{v})\right) \qquad (6)\\
&= \mathrm{ctx}_{\mathrm{sk}}\left(\sum_{i=0}^{\ell-1} u_i x^i \times \left(-\sum_{j=0}^{\ell-1} v_j x^{n-j}\right)\right) = \mathrm{ctx}_{\mathrm{sk}}\left(-\sum_{i=0}^{\ell-1}\sum_{j=0}^{\ell-1} u_i v_j x^{n+i-j}\right)\\
&= \mathrm{ctx}_{\mathrm{sk}}\left(\sum_{j=0}^{\ell-1} u_i v_j x^0 + \sum_{h=0}^{\ell-1}\sum_{j=0}^{j+h<\ell} u_{h+j} v_j x^h - \sum_{k=0}^{\ell-1}\sum_{j=0}^{j+k<\ell} u_j v_{j+k} x^{n-k}\right).
\end{aligned}$$

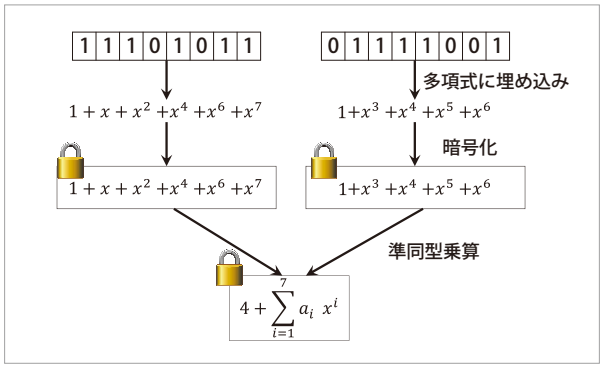
計算結果の暗号文を復号すれば，定数部分に内積 $\vec{u}\cdot\vec{v} = \sum_{j=0}^{\ell-1} u_j v_j$ を得ることができることが分かる[☆4]．この方法では，2 個の暗号文について，1 回の準同型乗算のみで計算を行うことができた．文献 7）では，実用的なゲノム疫学の規模として，10,000 人の被験者について 10^6 個の遺伝的要因について準同型暗号上でカイ二乗検定を行ったとき，ナイーブな方法で実行すると 2,000 日かかるところ，この方式で行うと 12 時間程度で計算が完了することを示した．また文献 8）では，異なるパッキング手法を用いることで，数万レコードに対する主成分分析や線形回帰などの統計解析が，数分程度で実行できることを報告している．このように，計算方法を工夫することで，FHE 上でも実用的な計算を実現することが可能になりつつある．

完全準同型暗号の実装

本章では主要な 3 つの完全準同型暗号の実装である，HElib[9]，SEAL[10] および TFHE[11] を紹介する．HElib は IBM の研究者らによって実装され 2013 年に公表された Apache-2.0 ライセンスに基づくオープンソースの C++ ライブラリである．HElib は bootstrapping を含む基本的な準同型暗号の実装に加え，パッキング，homomoprhic rotation，linear map など多くの機能が実装されており，一般に入手可能な実装の中では最も高機能な準同型暗号の実装といえる．

SEAL（Simple Encrypted Arithmetic Library）は Microsoft によって開発され，Microsoft Research ライセンスのもと公開された完全準同型暗号のライブラリである[12]．現在のところ，SEAL はパッキング，modulus switching，限定的な homomorphic rotation など，HELib が提供している機能の一部を提供している．

HElib および SEAL は RLWE に基づく準同型暗号方式の実装である．TFHE[11] は，LWE に基づく SwHE の実装である．TFHE の特徴は，0.1 秒以下の非常に高速な bootstrapping を実現している点にある．一方，TFHE が対応するメッセージは二値に限られており，パッキングがサポートされていない．TFHE ではメッセージを二値に変換した上でビット



■図 -4　多項式パッキングを用いた内積計算

☆4　実際には，定数項以外の部分から内積以外のメッセージにに関する情報が推測される恐れがあるため，定数項以外の部分にはランダムな係数を持つ多項式を加算しマスクする必要がある．

ごとに暗号化し，準同型演算も各ビットごとに行う必要がある．TFHE は bootstrapping を高速に実行できるため，バイナリ列を入力にとる多段の論理回路を評価する用途に向いているが，パッキングなどが利用できないため，多数の入力に対する算術演算を必要とする用途には向いていない．逆に，HELib や SEAL は，算術演算の評価に向いており，TFHE に比べ相対的に bootstrapping の実行速度が低速であるため，論理回路評価には向いていないといえる．

完全準同型暗号の今後

本稿では発展著しい完全準同型暗号の最近の動向を紹介するとともに，準同型暗号を用いた応用や実装について紹介した．秘密の情報を秘密にしたまま計算する技法には，準同型暗号のほかに Garbled circuit や秘密分散に基づく秘密計算などが知られている． これらの秘密計算手法の計算効率性も近年目覚ましく改善し，実応用に耐え得る性能を実現しつつある． これらの手法と準同型暗号（特に完全準同型暗号）の最大の違いは，通信における帯域の考え方にある．近年の Garbled circuit や秘密分散は秘密データを持つ主体同士が対話的に計算を進めるため，ネットワークの通信速度がきわめて高速である場合にその計算効率性が高くなる方向で発展が進んでいる．一方で，準同型暗号では，一度暗号化データを計算者（クラウド）に送信した後は計算を非対話的に実行するため，ネットワークの通信速度は計算効率性にはあまり関係せず，むしろ計算主体の CPU のスピードが重要になる．このように，秘密計算技術はそれぞれ想定する計算環境が異なることから，それぞれの長所を活かした発展が今後も続くと考えられる．

参考文献

1) Gentry, C. : Fully Homomorphic Encryption using Ideal Lattices, In 41st Annual ACM Symposium on Theory of Computing, STOC 2009, pp.169-178 (2009).
2) Brakerski, Z. and Vaikuntanathan, V. : Efficient Fully Homomorphic Encryption from (standard) LWE, In IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, pp.97-106 (2011).
3) Lyubashevsky, V., Peikert, C. and Regev, O. : On Ideal Lattices and Learning with Errors Over Rings, In Advances in Cryptology - EUROCRYPT 2010, pp.1-23 (2010).
4) Regev, O. : On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. J. ACM 56, 6, 34:1-34:40 (2009).
5) Gentry, C., Halevi, S. and Smart, N. P. : Homomorphic Evaluation of the AES Circuit, In Advances in Cryptology - CRYPTO 2012, pp.850-867 (2012).
6) Yasuda, M., Shimoyama, T., Kogure, J., Yokoyama, K. and Koshiba, T. : Secure Pattern Matching Using Somewhat Homomorphic Encryption, In 2013 ACM Workshop on Cloud Computing Security Workshop, ACM, pp.65-76 (2013).
7) Lu, W.-J., Yamada, Y. and Sakuma, J. : Privacy-preserving Genome-wide Association Studies on Cloud Environment Using Fully Homomorphic Encryption, In BMC Medical Informatics and Decision Making, vol.15, BioMed Central, p.S1 (2015).
8) Lu,W., Kawasaki, S. and Sakuma, J. : Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data, In Network and Distributed System Security Symposium(NDSS), pp.1-16 (2017).
9) Halevi, S. and Shoup, V. : HELib, http://shaih.github.io/HElib
10) Chen, H., Laine, K. and Player, R. : Simple Encrypted Arithmetic Library - SEAL v2.1. IACR Cryptology ePrint Archive 2017, 224 (2017).
11) Chillotti, I., Gama, N., Georgieva, M. and Izabachène, M. : Faster Fully Homomorphic Encryption : Bootstrapping in less than 0.1 seconds. In Advances in Cryptology - ASIACRYPT 2016, Part I, pp.3-33 (2016).
12) Fan, J. and Vercauteren, F. : Somewhat Practical Fully Homomorphic Encryption. IACR Cryptology ePrint Archive 2012, 144 (2012).

（2018 年 7 月 11 日受付）

佐久間 淳　jun@cs.tsukuba.ac.jp

2003 年東京工業大学大学院総合理工学研究科博士後期課程修了．博士（工学）．同年日本アイ・ビー・エム（株）入社，東京基礎研究所に配属．2004 年東京工業大学総合理工学研究科助手，2007 年同助教，2009 年筑波大学大学院システム情報工学研究科准教授，2016 年同教授．その間，2009 ～ 2012 年科学技術振興事業団さきがけ研究員兼任，2012 ～ 2014 年国立情報学研究所客員准教授，2016 年理化学研究所革新統合知能研究センターチームリーダ，現在に至る．

陸 文傑　riku@mdl.cs.tsukuba.ac.jp

2016 年筑波大学大学院システム情報工学研究科博士前期課程修了，同年同博士後期課程入学，2017 年学術振興会特別研究員 (DC2)，現在に至る．