複数組織対応属性ベース暗号を用いたファイル共有システム

石橋 拓哉[1,a)]　大東 俊博[1]　土田 光[2]　金岡 晃[3]　柿崎 淑郎[4]　相原 玲二[5]

概要：Dropbox に代表されるオンラインストレージサービスが普及してきている．このようなサービスではストレージの管理者によりデータを覗き見られる危険性があることから，ユーザ側で暗号化してデータを保護するシステムが注目されている．大東らは暗号文ポリシー属性ベース暗号を用いてファイル本体およびファイル名・ディレクトリ名を保護できるクライアントベースの暗号化ファイル共有サービスを提案しているが，鍵発行センターの特性上，単一組織での使用に限定せざるを得ない．そこで本稿では，複数組織で利用可能なファイル共有システムについて検討し，実現するために新たに必要となる属性ベース暗号（MA-ABE）の実装・評価を行い，さらに，MA-ABE と学術認証フェデレーション（学認）を利用したファイル共有サービスについての検討を行った．

キーワード：ファイル共有システム，暗号文ポリシー属性ベース暗号，複数組織対応

File Sharing Services using Multi-Authority Attrubute-Based Encryption

Takuya Ishibashi[1,a)]　Toshihiro Ohigashi[1]　Hikaru Tsuchida[2]　Akira Kanaoka[3]
Yoshio Kakizaki[4]　Reiji Aibara[5]

1. はじめに

クラウド技術の普及により Dropbox[*1] に代表されるクラウド上のオンラインストレージサービスが手軽に利用できるようになった．これらのサービスは自身のファイルのバックアップ以外にファイル共有の用途にも利用できる．一方，ユーザのデータは常にオンライン上のサーバに保存されているため，データの機密性や完全性の保護が課題となる．Dropbox などのオンラインストレージサービスでは，サーバでアクセス制御や暗号化を行い，アクセス権限の無いユーザからファイルを保護している．しかしながら，この方法ではストレージサービスの管理者によるデータの覗き見を防ぐことはできない．また，管理者のオペレーションミスでユーザのデータに誤った権限が付与されてしまい，情報漏えいが発生するような事故を防ぐこともできない．このような問題を解決する方法として，ユーザ自身がファイルを暗号化するシステムを利用し，暗号化されたファイルを共有する仕組みが注目されている．

近年，柔軟なアクセス制御が可能な公開鍵暗号方式として暗号文ポリシー属性ベース暗号 (Ciphertext-Policy Attribute-Based Encryption: CP-ABE) [1] が提案されている．CP-ABE は属性値 (ID・所属・役職など) の論理式で表現されたアクセスポリシー（以下，アクセス権）を暗号文に埋め込み，その暗号文をアクセス権を満たす属性を有したユーザの秘密鍵でしか復号できなくすることで，きめ細やかなアクセス制御機能を暗号化処理に付加できる．CP-ABE ではユーザは鍵発行センター (Key Generation Center: KGC) に自身の属性が含まれた秘密鍵を発行してもらい，それを適切な認証を経て取得することで閲覧権限があるファイルを復号できるようになる．KGC は全てのユーザの秘密鍵を作成できる強い権限を持っているため，利用組織内の信頼できる部署が管理することを想定する．

1 東海大学大学院 情報通信学研究科, Graduate School of Information and Telecommunication Engineering，Tokai University
2 日本電気株式会社，NEC
3 東邦大学 理学部，Faculty of Science，Toho University
4 東京電機大学，Tokyo Denki University
5 広島大学 情報メディア教育研究センター，Information Media Center，Hiroshima University
a) t_ishibashi@star.tokai-u.jp
*1 http://www.dropbox.com/

この CP-ABE を用いることでオンラインストレージ上のファイルの閲覧権限を柔軟に制御するシステム [2], [3] が議論されている．大東らはこれを拡張し，ファイルだけではなくファイル名・ディレクトリ名およびディレクトリ構造まで閲覧権限の制御範囲として保護するシステムを提案している [4].

KGC は自身が管理している全ユーザの秘密鍵生成および暗号文の復号が可能という強い権限を有しているため，複数 KGC に対応できない従来研究では，同一組織内での利用に限定せざるを得なかった．組織外への情報漏えいリスクを低減するには各組織が KGC を個別に運用する要求があることから，複数 KGC への拡張が求められる．たとえば，国立情報学研究所が構築している研究データ基盤 GakuNin RDM[*2] は，複数 KGC に対応した属性ベース暗号によるファイル共有サービスを必要とする例である．

そこで，本研究では CP-ABE を用いたファイル共有サービスを複数組織間で相互利用可能にする拡張方法について調査および考察を行い，さらにその方式を利用したファイル共有サービスについての検討および考察を行った．まず複数の KGC が存在可能な属性ベース暗号 (Multi-Authority Attribute-Based Encryption: MA-ABE) について調査した．その結果，我々が想定している利用方法では，中央機関を必要とせず，既存のペアリングライブラリで実装可能な方式として Lewko が提案した方式 [5] が適していることがわかったため，Lewko の方式の実装・評価を行った [6].

しかしながら，Lewko の方式では同一機関内でも，属性ごとの KGC を構築する必要があるため，1 組織あたりの KGC の数が膨大になる．比較的少数の組織間で，限られた属性のみを対象にする場合は問題とはならないが，数千の組織を対象にするサービスではパラメータサイズが膨大になる．そのため，多くの組織が利用するサービスでのファイル共有システムに MA-ABE を使用するためには，管理する属性数に依らず，1 組織ごとの KGC の公開パラメータが定数サイズとなる MA-ABE を使用する必要がある．そこで本稿では，この条件を満たしている方式である Yannis らの方式 [7] を使用することを前提に，この方式の実装・評価を行った．まず初めに，Lewko および Yannis らの方式をペアリングライブラリ PBC を用いて実装・評価した．各 KGC に属性が 1 つとし，Yannis らの方式のメリットが現れない条件のもとアルゴリズム毎に処理時間の計測を行い，このような条件下でも Lewko の方式に比べて Yannis らの方式の方が処理時間が速いことを示した．さらに，MA-ABE と認証フェデレーションを用いた複数組織間での利用を想定したファイル共有システムの仕組みに関する検討を行った．

2. 準備

本研究は CP-ABE を用いたファイル共有サービスを複数組織に対応させることを目的としている．本章では各方式の説明で用いる対称ペアリング群について述べた後に，既存のシステムとして大東らの方式 [4] を説明する．まず初めに CP-ABE について説明し，次にそれを用いてファイル本体・ファイル名・ディレクトリ名を暗号化して開示制御が可能なシステムを概説する．

[*2] https://rcos.nii.ac.jp/service/rdm/

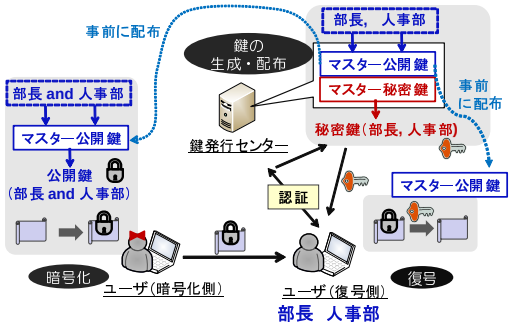

図 1 暗号文ポリシー属性ベース暗号の概要

2.1 対称ペアリング群

対称ペアリング群 $\mathsf{param} = (p, \mathbb{G}, \mathbb{G}_T, g, e)$ はそれぞれ，ビット長 λ の素数 p，位数 p の乗法的巡回群 $\mathbb{G}, \mathbb{G}_T$，生成元 $g \in \mathbb{G}$，多項式時間で計算可能な非退化性を有する双線形写像 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ から成る．セキュリティパラメータ λ を入力に取り，対称ペアリング群 param を出力するアルゴリズムを $\mathcal{G}_{\mathsf{SPG}}(1^\lambda)$ とする．

2.2 大東らのファイル共有サービス [4]

2.2.1 暗号文ポリシー属性ベース暗号

CP-ABE[1] は所属や役職などの属性を公開鍵として利用する属性ベース暗号 [8] の一種である．属性の論理式で表現されたアクセス権（例：人事部 OR（総務部 AND 部長））を暗号文に埋め込むことで復号可能な人のグループを決定できる．受信者は鍵発行機関に自分の属性（例: 人事, 部長, ○○担当）が埋め込まれた秘密鍵を発行してもらい，秘密鍵に埋め込まれた属性集合が暗号文のアクセス権を満たすとき，暗号文を復号可能となる．CP-ABE の処理の概要を図 1 に示す．

CP-ABE は以下の 4 つのアルゴリズムから成る．

Setup(1^λ) セキュリティパラメータ λ を入力しマスター公開鍵 PK とマスター秘密鍵 MK を生成し，出力する．

Encrypt(PK, M, A) マスター公開鍵 PK と平文 M とアクセス権 A を入力すると，暗号文 CT を出力する．

KeyGen(MK, S) マスター秘密鍵 MK と，秘密鍵を識別するための属性集合 S を入力すると，秘密鍵 SK を出力する．

Decrypt(PK, CT, SK) マスター公開鍵 PK，秘密鍵 SK，暗号文 CT を入力すると，CT に埋め込まれたアクセス権 A にマッチする SK のみ平文 M を復号で

きる．

KGC は信頼できる機関であり，**Setup** で生成したマスター公開鍵とマスター秘密鍵を管理し，全ユーザにマスター公開鍵を配布する．ユーザ（暗号化側）は **Encrypt** でマスター公開鍵とアクセス権を利用して平文を暗号化する．属性に対応する秘密鍵は鍵発行機関が **KeyGen** でマスター秘密鍵と属性値を用いて発行する．ユーザ（復号側）は **Decrypt** でマスター公開鍵と秘密鍵を利用して暗号文を復号する．

ユーザが自身の秘密鍵を取得するとき，鍵発行機関はユーザの ID と属性の対応表を参照し，ユーザを認証した上でユーザの属性と紐付いた秘密鍵を SSL/TLS を利用するなどして安全に配布する．ここで，ユーザの秘密鍵は属性が更新されない限り，ユーザの秘密鍵を変更する必要がない．そのため，秘密鍵の配布の頻度は低いと考えられるため，鍵配布の処理時間は比較的大きい場合でも運用上問題にならないと思われる．

CP-ABE は公開鍵暗号であるため柔軟な暗号化が可能であるが，AES などの共通鍵暗号と比べると処理に時間がかかる．これを解決するため，サイズが比較的大きいデータ本体は共通鍵暗号で暗号化し，それに用いる共通鍵（セッションキー）を CP-ABE で暗号化して保護するハイブリッド型の暗号化処理が用いられることが多い．Bethencourt ら [1] が開発した CP-ABE のライブラリである cpabe toolkit[*3] もこのハイブリッド型の処理が実装されている．

2.2.2 大東らのシステムの概要

大東らは，CP-ABE を用いた従来のファイル共有サービスと異なり，コンテンツだけでなくファイル名/ディレクトリ名を含むディレクトリ構造全体を暗号化し，ファイル名/ディレクトリ名の秘匿および編集権限の制御を行うシステムの提案をしている (図 2)．このシステムでは，ファイル名やディレクトリ名はランダムな文字列に置き換えられ，その文字列と本来のファイル名・ディレクトリ名の対応をディレクトリごとのファイル (リストファイルと呼ぶ) で管理している．リストファイル内のファイル名・ディレクトリ名を閲覧が許可されているアクセス権ごとにまとめて CP-ABE による暗号化をすることで，高速に処理することを実現している．なお，ディレクトリへのファイル・ディレクトリの追加処理を安全にするために，アップロードマネージャという登録専用のサーバを導入し，そこでの処理も CP-ABE を用いた認証を利用することで権限が無いユーザのファイルの登録も防いでいる．

2.3 複数組織対応属性ベース暗号

クラウドサービスなどの利用をする際には，複数の組織のユーザが共同で利用する場合が考えられる．こういった場合，従来の属性ベース暗号では KGC を複数機関で共同で管理することは非常に難しい．このような場面でも属性

[*3] http://acsc.cs.utexas.edu/cpabe/

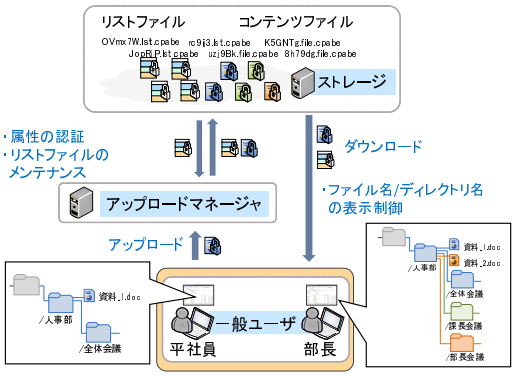

図 2 大東らの方式 [4] の概要

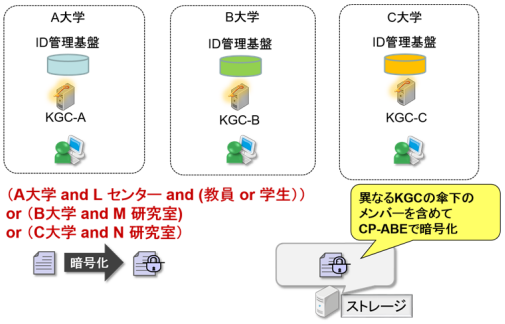

図 3 複数組織での KGC 管理の概要

ベース暗号を複数機関で利用できるように，KGC を組織ごとに用意して複数機関で連携して使用できる方式である MA-ABE が提案されている（図 3）．MA-ABE には大きく分けて，各機関の KGC を中央機関を使用して管理する方式 [9][10] と中央機関を必要とせず KGC が複数存在可能な方式 [5][7][11][12][13] が存在する．これらの方式は，複数機関で利用する場合などの現実的な属性管理ができるため，従来の鍵発行機関が 1 つのみの属性ベース暗号と比べ優れている．

中央機関を必要とせず複数の KGC が存在可能な属性ベース暗号では，ユーザ同士の結託攻撃に対する耐性を実現する必要がある．そこでこれらの方式では，ユーザごとに固有の識別子 GID を決め，KGC に依らず GID を秘密鍵生成の演算に含めることで結託耐性を与えている．たとえば，「A 大学教員かつ B 大学客員研究員」のユーザ向けの暗号文を解読するために「A 大学教員」と「B 大学客員研究員」がそれぞれ秘密鍵を提供し合ったとしても，それぞれの秘密鍵の GID が異なることから結合ができないため解読を防ぐことができる．

2.3.1 Lewko の方式のアルゴリズム

Lewko の方式では複数の KGC が独立にユーザを管理しており，DPVS(Dual Pairing Vector Space) という技術を用いて複数の組織（KGC）が連携した暗号化を実現する．Lewko の方式のシンタックスを以下に示す．

定義 2.1 (Lewko の方式 [5])

Lewko のアルゴリズムは以下の 5 つのアルゴリズムで構成

される．

Global Setup: Global Setup はセキュリティパラメータ λ を入力とし，グローバルパラメータ GP を出力する．

Authority Setup: Authority Setup は GP を入力とし，公開パラメータ PK と秘密鍵 SK を出力する．

KeyGen: KeyGen は GP，KGC の識別子 i，ユーザ固有の識別子である GID，SK を入力とし，復号するユーザの秘密鍵 $K_{i,\mathrm{GID}}$ を出力する．

Encrypt: Encrypt は GP，データ M，アクセス構造 (A,ρ)，公開パラメータの集合 {PK} を入力とし，暗号文 CT を出力する．ここで A は属性の論理式に合致したときに復号されるように線形秘密分散法 (Linear Secret Sharing Scheme: LSSS) を用いて作成された $\ell \times n$ のアクセス行列であり，ρ は A の各行と対応する KGC および属性を紐づけるための関数である．つまり，A の各行を $A_j (j = 1,\ldots,\ell)$，アクセス権を満たす属性の集合に対応したラベルの集合を $I \subseteq [\ell]$ としたとき，$\sum_{j\in I} \omega_i A_j = (1,\ldots,0)$ となるような $\{\omega_j\}_{j\in I} (\omega_j \in \mathbb{Z}_p)$ が存在する．

Decrypt: Decrypt は GP，CT，$\{K_{i,\mathrm{GID}}\}$ を入力とする．ここで，$\{K_{i,\mathrm{GID}}\}$ に関連するユーザ GID の属性の集合 $\Gamma_{\mathrm{GID}} := \{(\mathrm{GID}, i)\}$ が CT に付与されたアクセス権を満たすなら，データ M を出力する．そうでないならば，$\perp$ を出力する．

正当性として，GP $\leftarrow$ Global Setup(1^λ), (PK, SK) $\leftarrow$ Authority Setup(GP)，$K_{i,\mathrm{GID}} \leftarrow$ KeyGen(GP, i, GID, SK)，CT $\leftarrow$ Encrypt(GP, M, (A,ρ), {PK}) に対し，ユーザ GID の属性集合が暗号文 CT に付与されたアクセス権を満たすなら，$M =$ Decrypt(GP, CT, $\{K_{i,\mathrm{GID}}\}$) が成り立つことを要求する．

このとき，Lewko の方式のアルゴリズムの詳細は以下のように与えられる．

Global Setup (1^λ)**:** $\mathsf{param} \leftarrow \mathcal{G}_{\mathsf{SPB}}(1^\lambda)$ を実行する．さらに，H: $\{0,1\}^* \to \mathbb{G}\times\mathbb{G}$ をハッシュ関数と定義する．GP $= \{\mathsf{param}, H\}$ を出力する．

Authority Setup (GP): KGC_i は 12 次正方行列 $B \in \mathbb{Z}_p^{12\times 12}$ をランダムに生成し，正規直交基底の組 $(\mathbb{B} = (\vec{b}_1,\ldots,\vec{b}_{12}), \mathbb{B}^* = (\vec{b}^*_1,\ldots,\vec{b}^*_{12}))$ for $\mathbb{Z}_p^{12}$ を生成する．さらに，鍵発行機関 i は 2 つの一様乱数 ${\alpha_i}^1, {\alpha_i}^2 \in \mathbb{Z}_p$ を選択する．鍵発行機関 i は公開パラメータ PK $= \{e(g,g)^{a_i^1}, e(g,g)^{a_i^2}, g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, g^{\vec{b}_{3,i}}, g^{\vec{b}_{4,i}}\}$ と秘密鍵 SK $= \{g^{{\alpha_i}^1\vec{b}^*_{1,i}}, \vec{b}^*_{1,i}, \vec{b}^*_{2,i}, \vec{b}^*_{3,i}, g^{{\alpha_i}^2\vec{b}^*_{3,i}}, \vec{b}^*_{4,i}\}$ を出力する．

KeyGen (GP, i, GID, SK): 鍵発行機関 i は $H(\mathrm{GID}) = (H^1_{\mathrm{GID}}, H^2_{\mathrm{GID}}) \in \mathbb{G}$ を計算する．復号するユーザの秘密鍵 $K_{i,\mathrm{GID}} = g^{{\alpha_i}^1\vec{b}^*_{1,i}} g^{{\alpha_i}^2\vec{b}^*_{3,i}} (H^1_{\mathrm{GID}})^{\vec{b}^*_{1,i}-\vec{b}^*_{2,i}} (H^2_{\mathrm{GID}})^{\vec{b}^*_{3,i}-\vec{b}^*_{4,i}}$ を出力する．

Encrypt (GP, M, (A,ρ), PK): 暗号化するユーザは一様乱数 $s \in \mathbb{Z}_p$ を選択する．アクセス権が定義されている $\ell \times n$ 行のアクセス行列 (A,ρ) に関して，要素数 n のベクトル $v, w^1, w^2 \in Z_p^n$ をランダムに選択する．ただし，v の 1 つ目の要素を s とし，w^1, w^2 の 1 つ目の要素を 0 とする．さらに，$j \in [1,\ell]$ に対し r_j^1, r_j^2 をランダムに選択する．$C = Me(g,g)^s, D_j = e(g,g)^{A_j\cdot v} e(g,g)^{\alpha^1_{\rho(j)} r^1_j} e(g,g)^{\alpha^2_{\rho(j)} r^2_j}$，$C_j = g^{r^1_j \vec{b}_{1,\rho(j)}} + (r^1_j + A_j\cdot w^1)\vec{b}_{2,\rho(j)} + r^2_j \vec{b}_{3,\rho(j)} + (r^2_j + A_j\cdot w^2)\vec{b}_{4,\rho(j)}$ を出力する．暗号文 CT は (A,ρ) と $C, \{D_j\}, \{C_j\}$ によって構成される．

Decrypt (GP, CT, $\{K_{i,\mathrm{GID}}\}$): アクセス構造 (A,ρ) 下で復号 j するユーザは $\sum_{j=1}^{l} \omega_j A_j = (1,0,\ldots,0)$ になるような $\omega_1,\ldots,\omega_j \in \mathbb{Z}_p$ を選ぶ．ただし，$\rho(j)$ が秘密鍵を持っている復号するユーザの属性であった場合のみ，$\omega_j \neq 0$ とする．ユーザは $\omega_j \neq 0$ が成り立つ j に対してのみ，$F_j = D_j / e_{12}(K_{\rho(j),\mathrm{GID}}, C_j)$ を計算し，$M = C/\prod_{j=1}^{\ell} F_j^{\omega_j}$ を出力する．

ここで $e_{12}(\cdot,\cdot)$ は DPVS の計算である．ペアリング演算 $e(\cdot,\cdot)$ は 2 つの曲線の変数が入力される演算であったが，DPVS はそれをベクトルに拡張したものである．$e_{12}(\cdot,\cdot)$ は 12 個の要素を持つベクトルが 2 つ入力され，各ベクトルの要素間でペアリング演算を行い，計算結果を全て乗算する演算となる．たとえば，$\vec{v} = (v_1,\ldots,v_{12}), \vec{w} = (w_1,\ldots,w_{12}) \in \mathbb{Z}_p^{12}, g \in \mathbb{G}$ のときは $e_{12}(g^{\vec{v}}, g^{\vec{w}}) = \prod_{i=1}^{12} e(g^{v_i}, g^{w_i}) = e(g,g)^{\vec{v}\cdot\vec{w}}$ のように演算する．

2.3.2 Yannis の方式のアルゴリズム

Yannis らの方式では DPVS は用いておらず，楕円曲線上のペアリング演算などのみを使用して実現されている．Yannis らの方式のシンタックスを以下に示す．

定義 2.2 (Yannis らの方式 [7])

Yannis らのアルゴリズムは以下の 5 つのアルゴリズムで構成される．

Global Setup: Global Setup はセキュリティパラメータ λ を入力とし，グローバルパラメータ GP を出力する．

Authority Setup: Authority Setup は GP と KGC 番号 θ を入力とし，公開パラメータ PK_θ と秘密鍵 SK_θ を出力する．

KeyGen: KeyGen は GP，KGC_θ 内のユーザの属性情報 u，ユーザ固有の識別子である GID，SK_θ を入力とし，復号するユーザの秘密鍵 $\mathrm{SK}_{\mathrm{GID},u}$ を出力する．

Encrypt: Encrypt は GP，データ M，アクセス構造 $\mathbb{A} = (A,\delta)$，$\ell \times n$ のアクセス行列が A，δ が A の各行と属性を結びつける関数となる．なお，各属性に対応する公開鍵の集合 $\{\mathrm{PK}_\theta\}$ を入力とし，暗号文 CT を出力する．ここで A は LSSS を用いて作成された行列である．つまり，A の各行を $A_i (i = 1,\ldots,\ell)$，アクセス権を満たす属性の集合に対応したラベルの集合を $I \subseteq [\ell]$ としたとき，$\sum_{i\in I} c_i A_i = (1,\ldots,0)$ となるような $\{c_i\}_{i\in I} (c_i \in \mathbb{Z}_p)$ が

存在する．

Decrypt: Decrypt は GP，CT，$\{\mathrm{SK}_{\mathrm{GID},u}\}$ を入力とする．ここで，$\{\mathrm{SK}_{\mathrm{GID},u}\}$ に関連するユーザ GID の属性の集合 $\Gamma_{\mathrm{GID}} := \{(\mathrm{GID}, u)\}$ が CT に付与されたアクセス権を満たすなら，データ M を出力する．そうでないならば，$\perp$ を出力する．

正当性として，GP $\leftarrow$ Global Setup(1^λ), $(\mathrm{PK}_\theta, \mathrm{SK}_\theta) \leftarrow$ Authority Setup(GP, θ), $\mathrm{SK}_{\mathrm{GID},u} \leftarrow$KeyGen(GID, θ, u, SK_θ, GP)，CT$\leftarrow$ Encrypt $(M, (A, \delta), \{\mathrm{PK}_\theta\}, \mathrm{GP})$ に対し，$\{\mathrm{SK}_{\mathrm{GID},u}\}$ に関連するユーザ GID の属性の集合 Γ_{GID} が CT に付与されたアクセス権を満たすなら，M = Decrypt $(\mathrm{CT}, \{\mathrm{SK}_{\mathrm{GID},u}\}, \mathrm{GP})$ が成り立つことを要求する．

Yannis らの方式で用いている対称ペアリングを次のように定義する．このとき，Yannis らの方式のアルゴリズムの詳細は以下のように与えられる．

Global Setup (1^λ)**:** $\mathsf{param} \leftarrow \mathcal{G}_{\mathsf{SPG}}(1^\lambda)$ を実行する．また，U, U_Θ はそれぞれ属性，KGC 番号である θ の母集団，T は属性からその属性が属している KGC へとマッピングする関数と定義し，さらに，H は GID を，F は文字列をそれぞれ $\mathbb{G}$ の要素へとマッピングするハッシュ関数とする．GP $= \{\mathsf{param}, H, F, U, U_\Theta, \mathsf{T}\}$ を出力する．

Authority Setup (GP, θ): 鍵発行機関 θ は，2 つの一様乱数 $\alpha_\theta, y_\theta \xleftarrow{R} \mathbb{Z}_p$ を選択する．鍵発行機関 θ は公開パラメータ $\mathrm{PK} = \{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$ と秘密鍵 $\mathrm{SK} = \{\alpha_\theta, y_\theta\}$ を出力する．

KeyGen (GID, θ, u, SK_θ, GP): 鍵発行機関 θ は $t \xleftarrow{R} \mathbb{Z}_p$ を選択する．そして，ユーザの秘密鍵 $\mathrm{SK}_{\mathrm{GID},u} = \{\mathrm{K}_{\mathrm{GID},u} = g^{\alpha_\theta} H(\mathrm{GID})^{y_\theta} F(u)^t, \mathrm{K}'_{\mathrm{GID},u} = g^t\}$ を出力する．

Encrypt $(M, (A, \delta), \{\mathrm{PK}_\theta\}, \mathrm{GP})$**:** 暗号化するユーザは一様乱数 $z, v_2, \ldots, v_n, w_2, \ldots, w_n \xleftarrow{R} \mathbb{Z}_p$ を選択し，ベクトル $v = (z, v_2, \ldots, v_n)^\top$, $w = (0, w_2, \ldots, w_n)^\top$ を生成する．そして A_x と v の内積を計算し，$\lambda_x = \langle A_x, v\rangle$ とする．同様に $\omega_x = \langle A_x, w\rangle$ も計算する．$t_x \xleftarrow{R} \mathbb{Z}_p$ を生成して，$C_0 = Me(g,g)^z, \{C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} t_x}, C_{2,x} = g^{-t_x}, C_{3,x} = g^{y_{\rho(x)} t_x} g^{\omega_x}, C_{4,x} = F(\delta(x))^{t_x}\}_{x\in[\ell]}$ を出力する．なお，$\rho : [\ell] \to U_\Theta$，つまり $\rho(\cdot) = T(\delta(\cdot))$ とする．

Decrypt $(\mathrm{CT}, \{\mathrm{SK}_{\mathrm{GID},u}\}, \mathrm{GP})$**:** 復号するユーザは自身の復号用の鍵 $\{\mathrm{SK}_{\mathrm{GID},u}\}$ を使用し，暗号文 CT から平文 M を計算し出力する．ここで，A の各行を A_x，アクセス権を満たす属性の集合に対応したラベルの集合を $I \subseteq [\ell]$ としたとき，$\sum_{x\in I} c_x A_x = (1, \ldots, 0)$ とする．

$$C_{1,x} \cdot e(\mathrm{K}_{GID,\delta(x)}, C_{2,x}) \cdot e(\mathrm{H}(\mathrm{GID}), C_{3,x})$$
$$\cdot e(K'_{\mathrm{GID},\delta(x)}, C_{4,x}) = e(g,g)^{\lambda_x} e(H(\mathrm{GID}), g)^{\omega_x},$$
$$\prod_{x\in I} \left(e(g,g)^{\lambda_x} e(H(\mathrm{GID}), g)^{\omega_x} \right)^{c_x} = e(g,g)^z,$$
$$M = C_0 / e(g,g)^z.$$

2.3.3 Lewko の方式と Yannis らの方式の違い

Yannis らの方式ではユーザの復号用の鍵を生成する KeyGen においてユーザの属性情報である u と KGC 番号である θ を入力として使用しているのに対し，Lewko の方式では KeyGen において KGC 番号である i のみを入力として使用しておりユーザの属性情報を使用していない．これは Lewko の方式では一つの KGC では扱うことができる属性が一種類であることを示しており，扱う属性数を増やしたい場合は同じサーバ上で扱う属性の個数だけ Authority Setup を実行して複数の KGC を管理する必要があることを意味する．

Lewko の方式は Yannis らの方式より強い安全性を実現している．Lewko の方式では Adaptive 安全性を達成していることが証明されているのに対し，Yannis らの方式では Selective 安全性しか達成されていない．ここで，Selective 安全性とは，Authority Setup よりも前の段階で攻撃者が攻撃対象となるアクセス権をチャレンジャに発行することが強いられているモデルでの安全性である．一方，Adaptive 安全性とは，鍵クエリを行った後に攻撃者が攻撃対象となるアクセス権をチャレンジャに発行するモデルでの安全性である．Adaptive 安全性の方が Selective 安全性よりも自然なモデルであるが，それらの違いが実用上どのような影響を及ぼすか，著者らが知る限り明らかにされてはいない．本研究では KGC のコストを重視し，管理する属性数に依らず KGC の公開パラメータが一定のサイズとなる Yannis らの方式を採用する．

3. 提案システム

Yannis らの方式を使ったファイル共有システムの応用先として，共同研究グループでの情報共有システムを検討する．情報共有システムは，現在，大学単位などの組織ごとに管理がなされている．しかしながら，現実の利用シーンでは研究室単位や共同研究における研究グループごとなど，小規模な単位の組織間でデータにアクセスできるグループを作成したい場合が考えられる．たとえば，複数の研究室間で共同研究を行っていた場合に，従来の大学単位で管理されているファイル共有システムを使った場合，大学側の管理者に研究データを見られてしまう可能性があり，研究グループ内にのみ公開したいデータの場合には使用することが好ましくない．特に学外の研究機関と共同研究している場合，研究室単位で NDA（秘密保持契約）を締結しているとすると，大学単位での KGC を持つシステムでは利用に適さない．そこで，MA-ABE を利用して各研究グルー

プ単位で KGC を管理するようにし，暗号方式で必要となる GID を大学管理の ID 管理基盤の ID を利用する方法を提案する（図 4）．

提案する方式では研究グループ単位での KGC の管理に拡張したことで，システム内に存在する KGC の数は多くなってしまう．さらに，Lewko の方式では研究グループの属性ごとに KGC を構築する必要があるため，KGC の数が膨大になってしまう．そこで Yannis らの方式を利用することにより，グループごとに一つの KGC を用意するだけで使用可能にし，より実用的なものにする．

このシステムを複数の大学間のグループで利用できるシステムへ拡張することは，学術認証フェデレーション[*4]を利用して KGC を連携させることにより実現できる．まず学認に参加している自組織の認証プロバイダ（IdP）で認証を行い認可用トークンを発行してもらい，そのトークンを利用してユーザはサービスを利用したい組織の鍵発行センター（KGC=SP）で属性に対応する秘密鍵を発行してもらう．このとき，ユーザ固有の値である GID には ePPN（eduPersonPrincipalName）[*5] を利用できる．

KGC から発行される秘密鍵に対応する属性の管理については，個々の KGC に任せる方法を取ることにする．KGC を運営している組織について，GID に紐づける属性を事前にデータベース等に登録しておき，鍵生成の要求があったときに対応する秘密鍵を払い出すようにする．この登録については，当該組織で属性を持つ場合に何らかの事務手続きが生じると思われるため，その際に本人確認および対応付ける属性を精査した上で登録する．なお，ここで扱う属性の種類については，学認が提供している統一した属性を用いても良いし，個々の組織で定義している属性を用いても良いことにする．これらの属性がどの範囲のメンバーを包括するかについては，共同研究等をする関係であれば事前に共有や確認ができることを期待している．しかしながら，属性の範囲の誤解などは起こりうることであり，さらに KGC を設置する組織の大きさ次第では同じ属性を持つメンバーを把握できないことも考えられるため，属性の利用範囲に関しては何らかの工夫が必要になると思われる．この点の詳細な議論は今後の課題としたい．

4. 実装・評価

本章では，まず初めに Lewko の方式と比較することで Yannis らの方式を用いることの妥当性を確認し，その後，Yannis らの方式を提案システムに利用する場合の処理速度について評価する．

今回の実装において，開発言語は C 言語を用いている．Lewko の方式および Yannis らの方式は対称ペアリングから構成されるため，C 言語用のペアリング暗号ライブラリ

[*4] https://www.gakunin.jp/
[*5] https://meatwiki.nii.ac.jp/confluence/display/GakuNinShibInstall/eduPersonPrincipalName

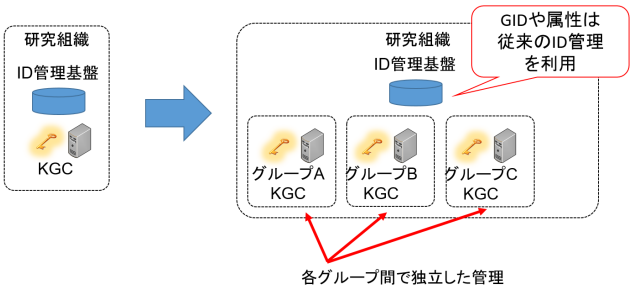

図 4　情報共有システムでの利用

である PBC Library (version 0.5.14) の対称ペアリング用の曲線である Type A curve を用いて実装した．ただし，Type A curve として PBC library で提供されている楕円曲線の位数が 160 ビットで 80 ビット安全性しか有していないため，PairingParametersGenerator API を用いて 256 ビット位数の曲線を生成して 128 ビット安全性を確保している．なお，同様にペアリング計算の出力となる拡大体 $\mathbb{G}_T$ のサイズは 3072 ビットとした．この変更をするためには，PairingParametersGenerator API により Type A Curve で $rBits = 256$, $qBits = 1536$ を指定して生成をすれば良い．

4.1 Lewko の方式と Yannis らの方式の比較

本節では初めに，Lewko の方式と Yannis らの方式の公開パラメータの比較を行い，Yannis の方式が公開パラメータの管理の点で優位であることを確認する．次に，方式ごとの処理速度の違いを比較し，Yannis らの方式が Lewko の方式と比べて処理速度が劣っていないことを確認する．

4.1.1 公開パラメータの比較

3 章で提案したシステムでは複数の組織間での使用を想定している．そこで，KGC が管理する属性ごとに公開パラメータのサイズが増大する Lewko の方式と，KGC が管理する属性数に依らず公開パラメータが定数サイズで済む Yannis らの方式では，公開パラメータの大きさがどの程度変わるのかを比較する．

まず提案したシステムに登録している全組織の数を $\mathrm{N_{KGC}}$ とし，各機関それぞれの属性数が $\mathrm{N_{Attr}}$，であるとする．1 つの KGC 当たりの公開パラメータの大きさを $\mathrm{D_{Param}}$ とすると，Lewko の方式では公開パラメータのサイズは $\mathrm{N_{KGC}}\times\mathrm{N_{Attr}}\times\mathrm{D_{Param}}$ となり，Yannis らの方式では $\mathrm{N_{KGC}}\times\mathrm{D_{Param}}$ となる．たとえば，全組織数が $\mathrm{N_{KGC}} = 2^{10}$，属性数がそれぞれ $\mathrm{N_{Attr}} = 2^{10}$，1 つの公開パラメータのサイズが $\mathrm{D_{Param}} = 1\mathrm{KB}$ と仮定すると，Lewko の方式では公開パラメータのサイズは $2^{10}\times 2^{10}\times 1\mathrm{KB} \fallingdotseq 1\mathrm{GB}$ となる．一方，Yannis らの方式では $2^{10}\times 1\mathrm{KB} \fallingdotseq 1\mathrm{MB}$ となる．この公開パラメータは暗号化処理で使われる．3 章で説明したような共同研究時の情報共有のような相互が対等に情報を提供し合う場面では，大多数のユーザが暗号化を行うこ

表 1　計測に使用した機器の仕様

CPU	intel(R) core(TM) i7-920 @ 2.67GHz
Memory	6GB
OS	Debian GNU/Linux 9.5

とが想定されるため，公開パラメータのサイズは現実的でなければならない．Lewko の方式では 1 GB と非常に大きくなってしまうのに対して，Yannis らの方式では 1MB と Lewko の方式と比べて小さいものとなっており，ユーザの負担が Lewko の方式に比べ軽減される．したがって，今回の提案したシステムには Yannis らの方式を使用するのが現実的かと思われる．

実際の $\mathrm{D_{Param}}$ のサイズについて検討する．今回の実装では，使用している楕円曲線とペアリングから，$\mathbb{G}_T$ の要素のサイズは 3072 ビット（384 バイトで格納），$\mathbb{G}$ の要素のサイズは 1537 ビット（193 バイトで格納）となる．Lewko の方式および Yannis らの方式における公開パラメータ $\mathrm{D_{Param}}$ をそれぞれ $\mathrm{D_{Lewko}}, \mathrm{D_{Yannis}}$ と表記する．Lewko の方式の公開パラメータは $\mathrm{PK} = \{e(g,g)^{a_i^1}, e(g,g)^{a_i^2}, g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, g^{\vec{b}_{3,i}}, g^{\vec{b}_{4,i}}\}$ であり，$\mathbb{G}_T$ の要素が 2 個，$\mathbb{G}$ の要素が 4×12 個であるため，$\mathrm{D_{Lewko}} = 2 \times 384 + 48 \times 193 = 10032$ Bytes となる．Yannis の方式の公開パラメータは $\mathrm{PK} = \{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$ であり，$\mathbb{G}_T$ の要素が 1 個，$\mathbb{G}$ の要素が 1 個であるため，$\mathrm{D_{Yannis}} = 1 \times 384 + 1 \times 193 = 577$ Bytes となる．これは，システム全体の公開パラメータの大きさの差が上記で比較したものよりさらに大きいことを意味している．

4.1.2　処理速度の比較

公開パラメータの大きさは，Lewko の方式より Yannis らの方式の方が提案したシステムで使用するには現実的である．しかしながら，Yannis らの方式の処理時間が Lewko の方式より大幅に遅い場合にはシステムに使用することは望ましくないため，Lewko の方式と Yannis らの方式の処理時間の計測を行った．

今回の実験では，暗号化/復号での平文 M は拡大体 $\mathbb{G}_T$ の一つの要素にエンコードする実装にしている．$\mathbb{G}_T$ のサイズは 3072 ビットあるため，128 ビットなどの共通鍵を暗号化するには十分であるため，ハイブリッド型暗号化の公開鍵部分の処理の評価としては十分であると考えられる。実験での KGC の数は 4 とし，それぞれの KGC は 1 つの属性を有しているとして暗号化を行う条件で実験をしている．各 KGC が有している属性を A, B, C, D という文字としたとき暗号化の際のアクセス権は A AND ((B AND C) OR D) と固定し，A AND D のユーザの鍵で復号し実験を行った．表 1 の実験環境において各アルゴリズムを 100 回実行した平均処理時間を比較したものを表 2 に示す．この表より，各 KGC が一つの属性を持つという Yannis の方式の利点を生かしていない条件であっても，Lewko の方式に比べ Yannis の方式は処理時間が全てのアルゴリズムにおいて同程度の処理時間または Lewko の方式よりも早く処理できていることがわかる．

表 2　各アルゴリズムの処理時間の比較 [sec]

	Lewko の方式	Yannis らの方式
Global Setup	0.0846	0.0840
Authority Setup	1.4901	0.0816
KeyGen	0.6520	0.5379
Encrypt	0.6632	0.4888
Decrypt	1.0172	0.2543

表 3　AND 連結の属性の場合の処理時間 (Yannis らの方式) [sec]

属性の個数	1	2	3	4	5
Encrypt	0.1105	0.2457	0.3675	0.4892	0.6071
Decrypt	0.0636	0.1272	0.1907	0.2542	0.3179

表 4　OR 連結の属性の場合の処理時間 (Yannis らの方式) [sec]

属性の個数	1	2	3	4	5
Encrypt	0.1105	0.2199	0.3283	0.4370	0.5419
Decrypt	0.0636	0.0637	0.0635	0.0635	0.0635

4.2　Yannis らの方式の計測・評価

4.1 節で Lewko の方式と Yannis らの方式を比較し，Yannis らの方式が Lewko の方式よりも現実的に使用可能であることを確認した．そこで本節では，Yannis らの方式を使用してファイル共有システムを構築することを目標とし，実際に Yannis らの方式を利用した際の条件式ごとの処理時間を評価する．

まず，Global Setup，Authority Setup，KeyGen については暗号化する際のアクセス権とは関係なく一定の処理速度となることから，表 2 の値を採用する．Global Setup は本ファイル共有サービスを開始するときに初めに 1 回だけ実行する処理，Authority Setup は新たな組織がサービスに参加するときに KGC を構築するときに 1 回だけ実行する処理である．それぞれの処理は 0.085 秒未満で実行できていることから，十分現実的な処理速度であると考えている．KeyGen はユーザがシステムを利用するときに秘密鍵を取得する際に各 KGC で実行される処理である．ユーザの鍵の取得する頻度によるが，例えば始めに 1 回だけ実行する場合や 1 日に 1 回実行される場合もある．実験結果はユーザが持っている一つの属性に対応する鍵の生成に必要な時間である．例えば 10 個の属性を持ったユーザの場合，一つの KGC から全ての属性に関する鍵を取得するような最悪のケースを考えても KeyGen の結果の 10 倍の 6 秒程度の時間で鍵が生成されることになる．

次にアクセス権として設定する論理式の AND と OR の影響について確認するために，属性を AND のみで連結する論理式と OR のみで連結する論理式の 2 つの場合で実験を追加で行った．属性を AND で連結した場合の結果を表 3，OR で連結した場合の結果を表 4 に示す．AND 連結で属性数を増やした場合，暗号化（Encrypt）および復号（Decrypt）の処理時間は連結した属性の個数に比例して増加している．OR 連結で属性数を増やした場合は，暗号化は属性の個数に比例するが，復号処理は属性の個数に依

らず一定の時間となることが確認できる．暗号化と AND 連結の復号に関しては，属性の個数が増える毎にそれぞれ 0.13 秒未満の処理時間と 0.07 秒程度の処理時間が増加しており，OR 連結の復号に関しては 0.07 秒未満で処理できている．

属性を AND で連結した場合の暗号化・復号処理時間について検討する．属性を AND で連結するのは復号できるグループを制限する場合であり，絞り込む範囲によって連結する個数が決まる．例えば，「ある組織」の「ある役職」の「何年に入社した人」かと言った条件の場合は 3 つの属性が AND で連結させるわけである．通常の使い方では，絞り込む条件数は比較的大きくならないと予想されるため，例えば高々 10 個の属性の AND 連結が上限と想定すると，暗号化は 1.3 秒未満，復号は 0.7 秒未満で処理ができると見積もることができる．

次に属性を OR で連結した場合の暗号化・復号処理時間について検討する．属性を OR で連結するのは復号できるグループを拡張するときに生じる．共同研究などで 3 つのグループのメンバーがそのファイルにアクセスすることを認めようとすると，3 つの属性を OR で連結する必要が出てくる．本提案システムでは，組織間の比較的大規模な共同研究なども想定しているため，共同でファイルを扱う組織数とその内部でもグループ分けによって，OR で連結される属性の個数は比較的多くなると予想している．例えば，100 グループの属性 OR で連結されることを想定すると，暗号化は 13 秒未満かかることになるが，各グループでの復号処理は 0.07 秒ですむことから閲覧する側は効率的となる．実際は AND 連結して表現されたグループの属性が OR 連結されるため，復号処理は各グループを表現する AND 連結の個数に依存すると思われる．上記の AND 連結の例のように高々 10 個であれば，それを OR でどれだけ連結したとしても復号処理は 0.7 秒までに抑えられるということになる．

5. まとめ

本稿では複数の KGC が存在可能な属性ベース暗号を用いたファイル共有システムについて検討し，研究グループ単位で KGC を運営するモデルについて具体的に考察した．次に，そのような KGC の数が比較的多い利用形態では Yannis らが提案した属性数に公開パラメータのサイズが依存しない方式が適していることを評価実験および理論的な見積もりにより示した．

今後の課題として，KGC およびクライアントプログラムを実装し，実際のネットワーク上で通信時間を含めた評価を行うことが挙げられる．さらに，学認の具体的な仕組みを考慮した処理手順についても検討する予定である．

謝辞 本研究の一部は JSPS 科研費 JP16H02808 の助成，MIC/SCOPE (課題番号 162108102) の委託を受けたものである．

参考文献

[1] Bethencourt, J., Sahai, A. and Waters, B.: Ciphertext-Policy Attribute-Based Encryption, *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pp. 321–334 (2007).

[2] Zhao, F., Nishide, T. and Sakurai, K.: Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems, *Information Security Practice and Experience - 7th International Conference, ISPEC 2011, Guangzhou, China, May 30 - June 1, 2011. Proceedings*, pp. 83–97 (2011).

[3] 松本悦宜，苦木大輔，内田 恵，近藤伸明，満永拓邦，五十嵐寛，力宗幸男：属性ベース暗号を用いたオンラインストレージサービス用クライアントの実装評価，信学技報，Vol. 111, No. 382, pp. 73–78 (2012).

[4] 大東俊博，後藤めぐ美，西村浩二，相原玲二：暗号文ポリシー属性ベース暗号を利用したファイル名暗号化ファイル共有サービスの実装と性能評価，情報処理学会論文誌，Vol. 55, No. 3, pp. 1126–1139 (2014).

[5] Lewko, A. B.: Functional encryption: new proof techniques and advancing capabilities, PhD Thesis (2012).

[6] 石橋拓哉，鈴木達也，伊藤勝彦，大東俊博，相原玲二：属性ベース暗号を用いたファイル共有サービスの複数組織対応に関する考察，電子情報通信学会技術研究報告= IEICE technical report: 信学技報， Vol. 117, No. 472, pp. 79–84 (2018).

[7] Rouselakis, Y. and Waters, B.: Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption, *Financial Cryptography and Data Security - 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pp. 315–332 (online), DOI: 10.1007/978-3-662-47854-7_19 (2015).

[8] Sahai, A. and Waters, B.: Fuzzy Identity-Based Encryption, *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pp. 457–473 (2005).

[9] Chase, M.: Multi-authority Attribute Based Encryption, *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pp. 515–534 (2007).

[10] Müller, S., Katzenbeisser, S. and Eckert, C.: Distributed Attribute-Based Encryption, *Information Security and Cryptology - ICISC 2008, 11th International Conference, Seoul, Korea, December 3-5, 2008, Revised Selected Papers*, pp. 20–36 (online), DOI: 10.1007/978-3-642-00730-9_2 (2008).

[11] Lewko, A. and Waters, B.: Decentralizing attribute-based encryption, *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, Springer, pp. 568–588 (2011).

[12] 土田 光，金山直樹，西出隆志，岡本栄司：Non-Programmable ランダムオラクルモデルで安全性証明可能かつ複数の鍵発行機関が存在可能な属性ベース暗号，信学技報，Vol. 115, No. 502, pp. 197–204 (2016).

[13] Okamoto, T. and Takashima, K.: Decentralized Attribute-Based Signatures, *Public-Key Cryptography - PKC 2013 - 16th International Conference on Practice and Theory in Public-Key Cryptography, Nara, Japan, February 26 - March 1, 2013. Proceedings*, pp. 125–142 (2013).