情報処理学会論文誌 Vol.55 No.3 1126–1139 (Mar. 2014)

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

大東 俊博[1,a)] 後藤 めぐ美[2,†1] 西村 浩二[1] 相原 玲二[1]

受付日 2013年6月30日, 採録日 2013年12月4日

概要：近年，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，TrueCrypt のようなクライアント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．最近では，暗号文ポリシー属性ベース暗号（CP-ABE）と呼ばれる新しい公開鍵暗号の方式を利用することで，きめ細かなアクセス制御をクライアント主導で行える方法が議論されてきている．しかしながら，既存の方式はファイルの内容（コンテンツ）のみを暗号化の対象にしており，コンテンツの要約などの重要情報が含まれうるファイル名やディレクトリ名の秘匿については議論されていなかった．そこで，本論文では CP-ABE を利用してコンテンツの暗号化だけでなくファイル名やディレクトリ名の暗号化および編集権限の制御が可能な方式を提案する．さらに，提案方式のプロトタイプシステムを実装し，ファイル名/ディレクトリ名の表示およびファイルアップロード時のファイル名の登録に要する時間を評価する．

キーワード：セキュアストレージ，ファイル共有システム，ファイル名暗号化，暗号文ポリシー属性ベース暗号

Implementation and Evaluation of a File Sharing Service with File Name Encryption Using Ciphertext-policy Attribute-based Encryption

Toshihiro Ohigashi[1,a)] Megumi Goto[2,†1] Kouji Nishimura[1] Reiji Aibara[1]

Received: June 30, 2013, Accepted: December 4, 2013

Abstract: Recently, a lot of online storage services, e.g. Dropbox, have been widely used. These services have a weakness, which the storage administrator can obtain contents of user's files. Hence client based encryption systems like TrueCrypt are used in order to protect user's files on online storage server. In particular, several researchers discuss methods based on Ciphertext-Policy Attribute Based Encryption (CP-ABE), which encrypt data with flexible access control by client-side. However, previous methods don't discuss confidentiality of file name and directory name. In this paper, we propose a method to protect not only content of user's file but also file name of it by using CP-ABE. The proposed method can protect directory name too, and can control write permission of file/directory by client-side. In addition, we implement a prototype system of file sharing service based on our method, and evaluate a performance of our method, including processing time for listing file/directory names and registering file/directory names.

Keywords: secure storage, file sharing system, filename encryption, ciphertext-policy attribute-based encryption

1. はじめに

クラウド技術の普及により Dropbox[*1] をはじめとする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．オンラインストレージサービスは自身のファイル

[1] 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University, Higashi-Hiroshima, Hiroshima 739–8511, Japan
[2] 広島大学総合科学研究科
The Graduate of Integrated Arts and Sciences, Hiroshima University, Higashi-Hiroshima, Hiroshima 739–8521, Japan
[†1] 現在，株式会社 OKI ソフトウェア
Presently with OKI Software Co., Ltd.
[a)] ohigashi@hiroshima-u.ac.jp

[*1] http://www.dropbox.com/

のバックアップ以外にもグループ間でのファイル共有の用途にも利用できる．特に自組織にサーバを設置・運用するコストを削減できるため，組織内でのファイル共有目的での利用が期待される．一方，ユーザのデータがつねにオンライン上のサーバに保存されるため，不正アクセスによる情報の漏えいやデータの改ざんなどからデータを保護するセキュリティ対策が課題となる．

Dropbox などのオンラインストレージサービスではサーバ側でアクセス制御や暗号化を行い，アクセス権限のない利用者からデータを保護している．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．特に政治的な理由によりディスクの検閲を実施できる国家に設置されたサーバではその懸念は大きくなる．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加されてしまい，情報漏えいが発生するような事故も防ぐことができない．

このような問題を解決する方法として，TrueCrypt [*2] や Sola [1] のようなクライアント側で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オンラインストレージに保存するという方法がある．この方法ではファイルの機密性に関するセキュリティ管理をユーザ側で行うため，適切に鍵が管理されていれば，オンラインストレージの管理者による覗き見や不正アクセスによるファイルの流出が生じたとしてもファイルを保護することができる．しかし，これらの暗号化システムでは共通鍵暗号や従来の公開鍵暗号を利用してファイルを暗号化するため，ユーザはファイル共有をしたいグループ数の鍵を配布・管理する必要があり，組織内でのファイル共有などの用途では運用コストが大きくなってしまう．さらに既存のシステムの多くはファイルの内容（コンテンツと呼ぶ）のみを暗号化の対象とし，ファイル名やディレクトリ名を保護しない．ファイル名やディレクトリ名はコンテンツの要約など重要情報が含まれる場合も多く，アクセス権限のない利用者に知られることは望ましくない．

クラウド上のサービスに有効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）[2] が提案されている．CP-ABE は ID や属性値（所属・役職など）を利用した論理式で表現されるアクセスポリシー（以下，アクセス権）を暗号文に埋め込むタイプの暗号である．ユーザはアクセス権を公開鍵として利用することで復号できる利用者のグループを任意に設定できる．また，ユーザが管理する秘密鍵の個数は自身の属性数に依存するため，組織内でのファイル共有のような共有するグループ数が多くなる場合でも有効となる．クライアント側で CP-ABE を使ってファイルを暗号化す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制御する研究 [3], [4] は存在するが，暗号化対象はコンテンツのみでありファイル名やディレクトリ名は保護されない．また，既存手法の 1 つである Zhao らの方式 [4] では属性ベース署名 [5] を用いて利用者の属性を確認する機能をストレージサーバに組み込み，ファイルを編集可能な利用者の権限をユーザ側で決定できるように拡張している．組織内でのファイル共有において，Zhao らが議論したファイルの編集権限の制御は必要なものであると考えられる．

本論文では，CP-ABE を用いてファイル名/ディレクトリ名の表示制御およびファイルやディレクトリの編集権限の管理が可能な方式を提案する．ファイル単位/ディレクトリ単位で CP-ABE の復号処理をしてファイル名やディレクトリ名の表示を制御する方法を用いた場合，ファイル数やディレクトリ数の増加によって処理時間が膨大なものとなってしまう．そこで，ファイル名/ディレクトリ名の表示制御を高速に行うために，リストファイルと呼ぶグループ単位でファイル名/ディレクトリ名をまとめて扱うファイルを導入する．また，アップロードマネージャと呼ぶ管理システムを導入し，CP-ABE を用いて属性を認証する処理とリストファイルを利用した編集権限の管理によって，ファイルやディレクトリの編集権限を制御できる．さらに，提案方式のプロトタイプシステムを実装し，ファイル名/ディレクトリ名の表示およびファイルアップロード時のファイル名の登録に要する時間を評価する．

2. 暗号文ポリシー属性ベース暗号

暗号文ポリシー属性ベース暗号（CP-ABE）[2] 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 [6] 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．受信者は鍵発行センタに自分の属性（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，平文を復号可能となる．このような仕組みは関数型暗号とも呼ばれる．**図 1** に CP-ABE の処理の概要を示す．

CP-ABE は以下の 4 つのアルゴリズムからなる．

Setup マスタ公開鍵 PK とマスタ秘密鍵 MK を生成し，出力する．

Encrypt(PK, M, A) マスタ公開鍵 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) マスタ秘密鍵 MK と，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．

[*2] http://www.truecrypt.org/

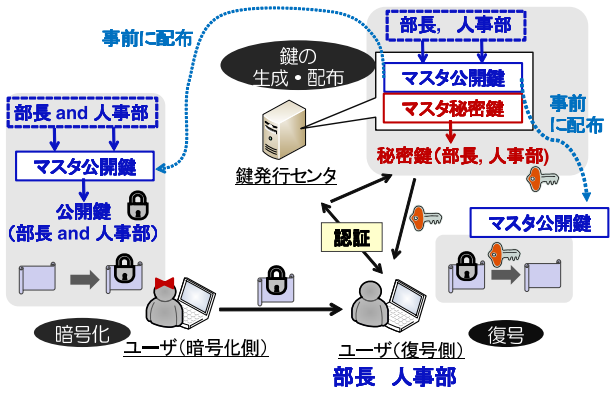

図 1 暗号文ポリシー属性ベース暗号の処理概要
Fig. 1 Description of ciphertext-policy attribute-based encryption.

鍵発行センタは信頼できる機関であり，**Setup** で生成したマスタ公開鍵とマスタ秘密鍵を管理し，全ユーザにマスタ公開鍵を配布する．ユーザ（暗号化側）は **Encrypt** でマスタ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センタが **Keygen** でマスタ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，鍵発行センタはユーザの ID と属性の対応表を参照し，ユーザを認証したうえ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は同じものを使い続けることができ，秘密鍵の配布の頻度は低いと考えられるため，鍵配布コストは必ずしも高くならないことに注意する．ユーザの秘密鍵はユーザの属性と紐付いているため，ユーザの属性が変更された場合（例：人事部から総務部への異動）に配布した鍵の効力を失効させる必要がある．この鍵の失効問題については，アクセス権の中に日時など有効期限を埋め込むことや，失効リストを管理し復号時にそれを参照させる方法 [3]，アクセス権の中に論理否定（NOT）を含んだ論理式を実現することで特定のユーザに対して権限を奪う方法（内積述語暗号）[7]，プロキシ再暗号化に基づく方法 [8] などが知られている．

CP-ABE は柔軟な暗号化が可能であるが公開鍵暗号であるため，AES などの共通鍵暗号と比べると低速である．これを解決するために，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処理が用いられることが多い．文献 [2] の著者らが開発した 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．文献 [2] で CP-ABE が提案されて以降，前述した鍵失効の仕組みも含めて数々の改良方式 [9], [10], [11] が提案されているが，著者らが知る限りでは CP-ABE 単体では cpabe toolkit 以外で公開されているライブラリは存在しない．したがって，一般に実装が入手しやすいこと，提案する方式は一般の CP-ABE に適用可能であること，上記の改良方式との性能の違いは各方式単体の暗号化/復号処理時間の違いから見積もれると期待できることから，本論文では cpabe toolkit を利用することを前提に実装・評価を行うこととする．なお，このライブラリに実装されている CP-ABE のアルゴリズム [2] の詳細な計算式については紙面の都合上割愛する．

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

3. 提案方式

3.1 概要

一般的なファイル暗号化システムではコンテンツの暗号化のみを行うが，提案方式では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．ファイル名やディレクトリ名は利便性を高めるために，その内容を要約する情報（例：○○課長昇進人事_20130630.docx）が含まれている場合があり，高度なセキュリティを提供するためにはそれらの情報も秘匿する必要がある．なお，本論文において，アクセス権のうちコンテンツまたはファイル名/ディレクトリ名を復号できる権限を read 権，それらを作成または編集できる権限を write 権と定義して議論をしていく．

提案方式の read 権による制御は CP-ABE によって暗号化することで実現する．ファイルのコンテンツ部分は単純に CP-ABE で暗号化し，ファイル名は保存用ファイル名（ユニークな擬似乱数列）に置き換えて保存する．さらに，元のファイル名と保存用ファイル名の対応関係を CP-ABE による暗号化で保護することで read 権がない一般ユーザからファイル名を秘匿している．提案方式では，同じ read 権を持つ複数のファイル名の対応関係およびディレクトリ名，ディレクトリ間のリンク構造を含めた情報をリストファイルという単位で管理することで高速に処理できるように工夫をしている．一般ユーザはこのリストファイルを復号することで自身の read 権に応じたディレクトリ構造を表示できる．リストファイルに関する詳細は 3.3 節で説明する．

提案方式の write 権による制御は上述したリストファイルとそれを管理するアップロードマネージャによって実現する．アップロードマネージャはリストファイル内に記述している write 権に対応する属性を一般ユーザが持っているかについて認証をする．認証に成功した場合，アップロードマネージャはリストファイルの更新およびストレージへのコンテンツファイルのアップロードを行う．なお，一般ユーザがアップロードマネージャを介してしかストレージにファイルをアップロードできないようにすること

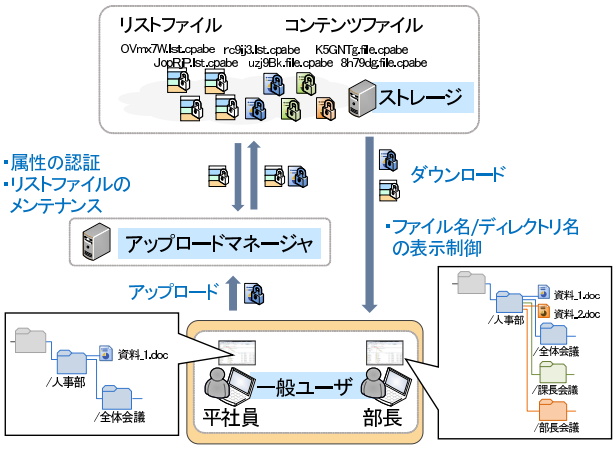

図 2 提案方式の概要
Fig. 2 Description of the proposed method.

で，権限がない一般ユーザによる不正なファイルの削除を防いでいる．

提案方式の概要を図 2 に示す．なお，この図では鍵発行センタ，各一般ユーザとアップロードマネージャが鍵発行センタから事前に秘密鍵を取得する処理，ストレージやアップロードマネージャ自体の利用者認証（パスワード認証など）の処理は省略している．ファイルのアップロード（ファイル名/ディレクトリ名の登録）の際には必ずアップロードマネージャを介するため負荷が集中することが想定されるが，ダウンロード時には一般ユーザは直接ストレージからファイルを取得しクライアント側で復号処理させることで負荷分散を考慮している．

3.2 各管理者と一般ユーザの役割と権限

提案方式における各管理者と一般ユーザの役割と権限について以下に整理する．

ストレージ管理者

役割 オンラインストレージサービスの管理者．本論文では，ユーザ認証した者に対してのみファイルのアップロードを許可し，ダウンロードは認証なしで許可する機能が提供されていることを想定している．たとえば，Dropbox では利用者 ID を持たないユーザともファイル共有を行えるようにリンク共有機能によって生成する読み出しのみ可能な URL を利用できる[*4]．

権限 ストレージ管理者はストレージにアップロード可能なユーザを制御でき，本システムではアップロードマネージャのみにその権限を与え，鍵発行センタ，一般ユーザおよびそれ以外の第 3 者にはアップロード権限を与えない．また，CP-ABE のマスタ公開鍵を取得して暗号化を行うことは可能である．しかしながら，ストレージ管理者は対応する CP-ABE の秘密鍵を持たないため，ファイル名/ディレクトリ名および暗号化されたコンテンツを復号することはできない．

[*4] WebDAV で実現する場合には，たとえばアップロードには認証を要求し，ダウンロードの際には認証が不要な URL から対象のディレクトリをシンボリックリンクや同期機能などで参照する方法などで実現できる．

アップロードマネージャ管理者

役割 アップロードマネージャ管理者は，一般ユーザからコンテンツが暗号化されたファイルを受け取り，write 権を有することを確認したうえで，リストファイルを用いたファイル名の秘匿化を施してからストレージにアップロードする．また，ディレクトリの作成の際にも write 権を有することを確認したうえでリストファイルを編集する．なお，アップロードの要求ができるユーザを制限するために，利用者 ID とパスワードを用いた利用者認証を行う．

権限 アップロードマネージャ管理者はストレージの利用者 ID とパスワードを管理しており，ストレージ上へ暗号化ファイルを作成・削除できる．また，アップロードマネージャにアップロードの要求ができるユーザを制御でき，本システムでは一般ユーザのみにその権限を与え，ストレージ管理者，鍵発行センタおよびそれ以外の第 3 者には権限を与えない．さらに，リストファイルの編集のための属性のみ埋め込まれた秘密鍵およびマスタ公開鍵を持つため，ファイル名・ディレクトリ名の復号およびディレクトリ構造の取得ができ，内容の新規作成または変更をして暗号化することでリストファイルを作成・編集することもできる．しかしながら，コンテンツの暗号化に用いている属性の秘密鍵を持たないためにコンテンツ（ファイルの本文）の復号はできない．

鍵発行センタ管理者

役割 一般ユーザやアップロードマネージャ管理者の CP-ABE に関する ID やそれに紐づく属性を参照し，対応する秘密鍵の生成と配布を行う．信頼できる機関としての役割を持つ．

権限 秘密鍵の生成に用いるマスタ秘密鍵を管理するため，すべての属性に関するコンテンツやファイル名/ディレクトリ名を復号，および任意の属性の認証を成功させることができる．しかしながら，ストレージおよびアップロードマネージャのユーザ認証用の利用者 ID とパスワードを有していないため，ストレージへのファイルのアップロードを行うことはできない．

一般ユーザ

役割 アップロードしたいファイルのコンテンツを暗

表 1 管理者および一般ユーザの権限
Table 1 Authorities of administrators and users on the proposed method.

	コンテンツの復号	ファイル名・ディレクトリ名の復号/ディレクトリ構造の取得	ストレージ上への暗号化ファイルの作成
ストレージ管理者	×	×	×
アップロードマネージャ管理者	×	○	○
鍵発行センタ管理者	○	○	×
一般ユーザ（属性合致の場合）	○	○	×

号化し，ディレクトリの中の write 権を満たす場所へのファイルの登録をアップロードマネージャに依頼する．ストレージからリストファイルや暗号化されたコンテンツをダウンロードすることで，ファイル名/ディレクトリ名の表示やコンテンツの復号を実行する．

権限 自分の属性が埋め込まれた秘密鍵とマスタ公開鍵を用いて（read 権に属性が合致する）コンテンツの復号やファイル名・ディレクトリ名の取得を行える．また，マスタ公開鍵を用いてコンテンツ本体を暗号化することができる．さらに，アップロードマネージャの利用者 ID とパスワードを有しており，自身の持つ属性に対応したディレクトリへのファイルのアップロードおよびファイル名の登録・削除をアップロードマネージャを介して行える．しかしながら，ストレージの利用者 ID とパスワードを有していないため，ストレージ上へ直接リストファイルやコンテンツをアップロードすることはできない．

表 1 に上記のプレイヤが持つ権限をまとめた．表 1 より，コンテンツ本体を復号できるのは一般ユーザおよび鍵発行センタのみであり，アップロードマネージャやストレージ管理者およびそれ以外の第 3 者からはコンテンツの内容を秘匿できる．また，ファイル名/ディレクトリ名の復号およびディレクトリ構造の取得ができるのは，そのディレクトリの read 権に合致した一般ユーザとリストファイル専用の復号権限を持ったアップロードマネージャおよび鍵発行センタのみであり，ストレージ管理者やそれ以外の第 3 者からは秘匿できる．さらに，ストレージ上のリストファイルを編集して更新できるのはストレージの利用者 ID とパスワードを持ったアップロードマネージャのみであり，一般ユーザや鍵発行センタやそれ以外の第 3 者からの不正な書き換えを防ぐことができる．なお，ストレージ管理者によるファイルの修正・削除は CP-ABE など暗号化技術のみでは防ぐことは困難である．しかし，Web 改ざんをリモートで監視するシステム[*5]などを導入し，アップロードマネージャがアップロードしたファイルが不正に変更されることを監視すれば単純な置き換えによる改ざんを防ぐことは可能である．この理由により，表 1 のストレージ管理者の該当する項目は，ストレージ上への暗号化ファイルの作成を自由にはできないという意味で（×）としている．

[*5] http://www.kaizankenchi.jp/

3.3 リストファイルの役割と利用方法

3.3.1 アクセス権

リストファイルに関する詳細な説明の前に，read 権や write 権として書かれるアクセス権（属性を用いた論理式）について提案システムでどのような言語で書けるかについて定義をする．これは本システムで利用している cpabe toolkit で用いられる形式を概説したものである．

CP-ABE では個人の ID やその人が持っている役割を含む属性値を論理演算子で連結して論理式を構成する．この属性値は文字列または数値で表現されており，たとえば “人事部”，“部長”，“student”，“staff” などの文字列や “hour = 10” などの数値の条件のように記述できる．論理演算子は AND 演算子，OR 演算子，閾値型演算子（列挙した n 個の属性のうち k 個の属性を満たす場合に真を返す関数，$n \geq k$）および比較演算子（<，>，<=，>=，=）をサポートしている．本システムでは後者の 2 つの演算子については利用シーンが多くないため，AND と OR の演算子のみ利用することとする．

cpabe toolkit では，これらの属性と演算子を組み合わせてアクセス権を文字列として表現したものを入力として，それに対応した適切な暗号化を実行する．たとえば，連結する 2 つの属性値を X_1，X_2 としたとき，それらを AND で結合する場合は “X_1 and X_2”，OR で結合する場合は “X_1 or X_2” のように半角スペースを区切り文字として演算子で結合する形でポリシーを記述できる．また，各演算子の順序を指定する際に括弧 () を利用もでき，たとえば 3 つの属性値 X_1，X_2，X_3 を “(X_1 and X_2) or X_3” と記述すれば，X_1 and X_2 が先に評価され，その後，その結果と X_3 の OR が評価される．リストファイルでアクセス権を記述する際には，cpabe toolkit で扱いやすいように上記のフォーマットに沿った文字列で記述している．

3.3.2 ファイル名/ディレクトリ名の表示制御

通常，ファイル名やディレクトリ名は一度に複数表示するため，ファイル名やディレクトリ名の暗号化・復号

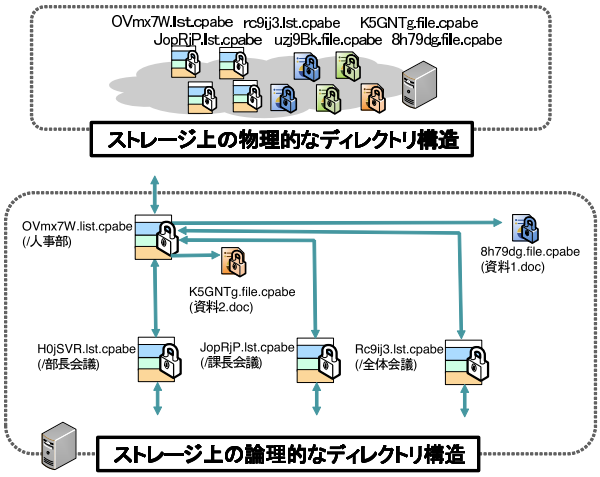

図 3 ストレージ上の論理的なディレクトリ構造
Fig. 3 Logical directory structure on online storages.

をファイル/ディレクトリのそれぞれで個別に行うことはCP-ABE の小さな処理を多数発生させることになる．2 章で述べたように CP-ABE のライブラリは共通鍵暗号を利用したハイブリッド型の処理が用いられるため，小さいサイズの多数のファイルを暗号化/復号するより，大きなサイズの少数のファイルを処理したほうが高速に動作する．そこで，ある 1 つのディレクトリにおいて，同じ read 権を持つファイル名や子ディレクトリ名を 1 つのファイル（リストファイル）にまとめて CP-ABE で暗号化する方法を考える．この場合，属性が合致している複数のファイル名/ディレクトリ名を同時に復号できるため，リストファイルのサイズは大きくなるが，それぞれを個別に復号するより高速に処理できる．

リストファイルは read 権およびディレクトリごとに作成し，一般ユーザはディレクトリ内のファイル名/ディレクトリ名を表示する際に自身の属性に合致する read 権の個数の CP-ABE の復号を実行する．また，親・子ディレクトリ名と対応するリストファイルの URL の対を現在のリストファイル内に記載することでディレクトリ間の相互リンクを実現する．**図 3** で示すように，この URL を利用した参照の仕組みによって，リストファイル/コンテンツファイルの物理的な保存場所とディレクトリの論理的な構造を結び付けている．

一般ユーザはディレクトリを移動するたびにリストファイルを復号し，ファイル名/ディレクトリ名の表示を行う（Linux において cd コマンドの後に ls コマンドを実行する場合に相当する）．このようにして一般ユーザは自身の属性に合致するディレクトリを自由にたどることができるが，実際に利用をする際には起点となるリストファイルが必要となる．この起点となるリストファイルをルートリストファイルと呼ぶ．ルートリストファイルは唯一削除できないルートディレクトリとしてのリストファイルであり，固定の URL がつけられている．この固定の URL を閲覧用のプログラム（ビューア）に内蔵（preset）しておくことにより，ユーザはルートリストファイルからディレクトリをたどっていくことで必要なディレクトリに到達できる．ビューアが保持しておく必要があるのはルートリストファイルの URL であり，その内容に関しては必要に応じてルートリストファイルをダウンロードして最新の情報を得ることで，同じストレージを共用するビューア間でルートリストファイルの同期をとることになる．なお，ルートリストファイルのフォーマットは 3.4.4 項で説明する通常のリストファイルから親ディレクトリの記述をなくしたものとなる．

3.3.3 ファイル名/ディレクトリ名の登録

リストファイルは同じ read 権を持つ複数の一般ユーザで共有されるため，不正な書き換えや削除を防ぐために一般ユーザはアップロードマネージャを介してしかリストファイルの編集を行えないようにする．具体的には，リストファイルの中にファイルやディレクトリの write 権を表す文字列を格納し，アップロードマネージャはこれを利用して一般ユーザが write 権に合致する属性を持っているかを以下のように確認する．

(1) アップロードマネージャは，セッションごとにユニークなセッション ID を生成し，write 権に合致した属性を持つ一般ユーザの秘密鍵でなければ復号できないように CP-ABE で暗号化する．
(2) アップロードマネージャは生成した暗号文をチャレンジとして一般ユーザに送信する．
(3) 一般ユーザは自身の秘密鍵でチャレンジを復号し，得られた平文をアップロードマネージャに返信する．
(4) 一般ユーザから送られてきた平文がセッション ID と一致するとき，アップロードマネージャは一般ユーザが write 権に合致する属性を持っていると判断する．

このようにアップロードマネージャは write 権を確認するために一般ユーザに対して CP-ABE を用いたチャレンジアンドレスポンス型の認証を実行する[*6]．

アップロードマネージャは write 権を満たすリストファイルを復号したうえで編集し，再度該当する read 権により CP-ABE で暗号化することでリストファイルを更新する．この操作のためにアップロードマネージャはすべてのリストファイルを復号できる必要がある．これは，すべてのリストファイルの read 権にアップロードマネージャの属性を OR で追加することで実現できる．

[*6] Zhao らの方式 [4] のように属性ベース署名 [5]（CP-ABE の電子署名版）を用いて一般ユーザの属性を確認することもできる．しかしながら，属性ベース署名用のライブラリは著者らの知る限りでは公開されておらず性能評価のためのハードルが高いこと，提案方式のモデルでは電子署名の機能（否認防止など）までは必要なく属性を認証できればよいこと，から CP-ABE だけで構成できるチャレンジアンドレスポンス型の認証方式を採用した．

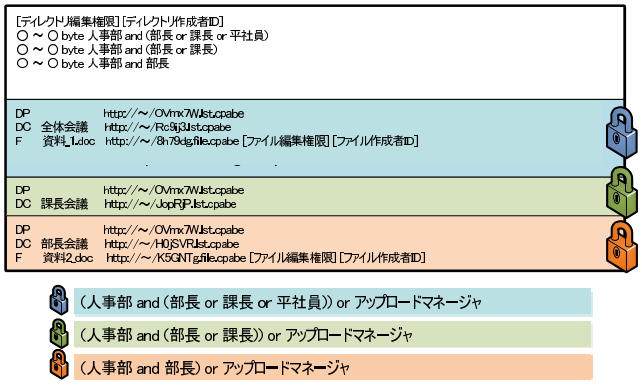

図 4 リストファイルのフォーマット
Fig. 4 File format for the list files.

3.3.4 リストファイルのフォーマット

1 つのディレクトリの中には異なる read 権を持つリストファイルが複数存在し，かつそのリストファイルの数は利用をしていくうえで増減することが想定される．read 権ごとにリストファイルを管理する場合，同一ディレクトリに属するリストファイルでも保存用ファイル名が異なるランダムな文字列となり区別できないため，ディレクトリ単位でリストファイルのバックアップをとる際に必要なファイルを探す操作が煩雑となるという問題がある．特に同一ディレクトリに属する read 権の種類が動的に変更されることが想定されるためバックアップ対象のリストファイルを事前にリスティングしておくような対策も必ずしも有効とはならない．そこで，図 4 のようにディレクトリ単位で一括して扱えるように，リストファイルを read 権ごとに暗号化されたブロックと平文のヘッダ情報から構成して管理する．このようにすることで，ディレクトリに対応するリストファイルのバックアップの際の操作が容易となる．ヘッダの 1 行目にはカレントディレクトリの write 権を表す文字列およびディレクトリ作成者の ID が保存されている．ヘッダの 2 行目以降には，すべての暗号化ブロックについて復号のための read 権および暗号化ブロックのファイル上の位置を表す情報が保存されている．

暗号化ブロック内の平文データは，行頭の 3 種類の識別子（DP，DC，F）で内容を区別し，以下のようにタブ区切りで情報を整理して保存する．

- DP[tab] 親ディレクトリのリストファイルの URL
- DC[tab] 子ディレクトリ名の平文 [tab] 子ディレクトリのリストファイルの URL
- F[tab] カレントディレクトリに存在するコンテンツ名の平文 [tab] コンテンツの URL[tab] ファイルの write 権 [tab] ファイル作成者の ID

3.4 関連研究

ファイル名やディレクトリ名などメタ情報を暗号化によって保護する方式は複数の製品などですでに実現されている．EncFS [*7]，CryptSync [*8]，CarotDAV [*9] などは，パスワードや共通鍵を利用してファイル本体およびそのファイル名やディレクトリ名を暗号化する仕組みを提供している．この方法ではファイルを共有したいグループ数のパスワードや鍵を配布・管理する必要があり，本論文で対象としている組織内でのファイル共有などの用途では運用コストが大きくなってしまい適していない．Boxcryptor [*10] は共通鍵暗号と公開鍵暗号を組み合わせたハイブリッド型の方式でファイル本体およびそのファイル名を暗号化する仕組みを提供している．この方式ではファイル本体とファイル名を共通鍵暗号で暗号化し，その共通鍵を公開鍵暗号（RSA 暗号）で暗号化してファイルと連結して保存する．複数人でのファイルの共有は，共有したいメンバの各公開鍵で共通鍵を暗号化し，暗号化された共通鍵群を暗号文と連結することで実現している．ただし，ファイル名を表示させる際には，ファイル名ごと個別に公開鍵暗号の復号処理を実行する必要があるため，ディレクトリ内のファイル数が多くなった場合に処理時間が問題となる．この方式の共有の仕組みを CP-ABE を用いたものに置き換えた場合，3.3.2 項の冒頭に書いている CP-ABE を用いた素朴な方式に相当し，Boxcryptor などの製品はそれに包括されていると考えることができる．本論文は，CP-ABE，ハイブリッド型暗号化の性質，およびリストファイルを使ったアクセス権ごとの一括したファイル名の閲覧制御というアイディアによって，ファイル名やディレクトリ名の表示時間（復号時間）の短縮という問題を解決することを目的としている．

ポリシーをベースとしたアクセス制御の方式として RBAC（Role-based access control）[12], [13] が知られている．この方式はサーバ側でロール（役割）に基づくアクセス制御によってファイルやディレクトリの閲覧範囲を柔軟に制御することができる．CP-ABE は役割に代表される個人の属性を利用してグループを定義してアクセス制御を行うため RBAC と似た技術ではあるが，それを暗号化によって実現しているという点で異なる．本論文で想定しているクラウド上での覗き見耐性を持つためには，クライアント主導の暗号化によるアクセス制御をする必要があるため，RBAC の技術をそのまま用いることは困難であると考えられる．

4. 実装

本章では提案方式の評価のために実装したプロトタイプシステムについて説明する．

提案方式での一般ユーザの処理はクライアント端末に実

*7 http://www.arg0.net/encfs
*8 http://stefanstools.sourceforge.net/CryptSync.html
*9 http://rei.to/carotdav.html
*10 https://www.boxcryptor.com/en/features

装する方法と自分が管理している個人用のサーバに代理で処理をさせる方法が考えられる．クライアント端末に実装する方法では，各種 OS に対応させるためには CP-ABE のライブラリの移植が必要であり，実装負荷が大きい．さらに処理性能の低い携帯端末などでは処理の性能が低くなってしまう．そこで，プロトタイプシステムでは Web インタフェースを有した個人用のエージェントサーバを介する実装により，マルチプラットフォームでも利用可能で端末によって処理速度が低下しないようにした．

エージェントサーバは，従来クライアント端末で行うコンテンツの暗号化・復号，リストファイルの復号，およびアップロードマネージャを介したアップロード処理を代理で実行する．また，一般ユーザの認証情報を利用して鍵発行センタから秘密鍵を取得し，アップロードマネージャへのアクセスの権限も持つことから，エージェントサーバは一般ユーザと同じ権限を持つ．そのため，エージェントサーバは個人専用の仮想マシンなど管理権限を有したサーバを利用して安全性を確保する必要がある．なお，エージェントサーバと一般ユーザのエンド間の通信は HTTPS などの従来の技術で安全性を確保する．

エージェントサーバおよびアップロードマネージャのユーザインタフェースと通信部分は Perl（v5.10.1）で実装した．両者間の通信はポート制限を受けにくい HTTPS で行い，Perl のモジュールである LWP::UserAgent クラスを利用した．CP-ABE の処理については cpabe toolkit（cpabe-0.11, libbswabe-0.9）を用いて C 言語で実装している．

ストレージは世界中で広く使われている Dropbox を対象にした．アップロードマネージャと Dropbox 間の通信は Perl 用の Dropbox API である WebService::Dropbox（v.1.09）[*11] を利用し，通信路は HTTPS で保護する．Dropbox API は Dropbox と通信をするために OAuth で認可を与える必要がある．OAuth によって得られたアクセストークンを安全な場所に保存し，運用時にはそれを使って Dropbox に接続する．共有ファイルの置き場はアップロードマネージャのユーザ ID で取得した Dropbox の領域を利用する．Dropbox からファイルをダウンロードするときには，リンク共有機能によって生成された読み出しのみ可能な公開用 URL を用いる．今回の実装では単一のストレージで動作させている．なお，クラウドストレージへのアクセスは既存ライブラリの機能に任せた操作のみを行うことを想定しており，一般的なファイルシステムにおける問題に関する対応についてはストレージの機能として抽象化することとし，本論文では設定・実装の対象としていない．たとえば，リストファイルやコンテンツなどの転送（アップロードやダウンロード）の失敗の検知，再送やファイルの世代管理をしたうえでの書き戻しなどファイルの復旧（回復）に関する処理などはライブラリや既存の解決法に任せることとする．

プロトタイプシステムでは鍵発行センタの実装は行わず，事前に cpabe toolkit の cpabe-keygen コマンドによって属性に対応する秘密鍵を生成し，アップロードマネージャやエージェントサーバに置くようにしている．鍵発行センタについては，類似の暗号方式である ID ベース暗号において鍵発行センタの実装の試みが行われており [14]，同様の枠組みで実装できる．

リストファイルに記述する AND や OR の論理式は加法標準形で表記する．加法標準形は，全体を論理最小項の論理和で構成し（例：A and (B or C) or D → (A and B) or (A and C) or D），論理式を一意に表現できるため，同一リストファイル内の複数ブロックに同じ権限が付与されることを防ぐことができる．なお，本システムの実装では，属性の並び順を文字列順でソートしたうえで加法標準形で書いた論理式をユーザにシステムへ入力させるようにしている．

4.1 ファイル名/ディレクトリ名表示の処理手順

図 5 にファイル名/ディレクトリ名表示の処理手順を示す．一般ユーザは (D-1)〜(D-4) によってストレージから指定したリストファイルを取得し，復号することによりディレクトリの移動およびファイル名/ディレクトリ名を表示する．リストファイルはルートリストファイルを起点としてディレクトリ構造を相対パスで保持しており，(D-1)〜(D-4) によるディレクトリの移動を繰り返すことで目的のファイルがあるディレクトリに到達する．ダウンロード対

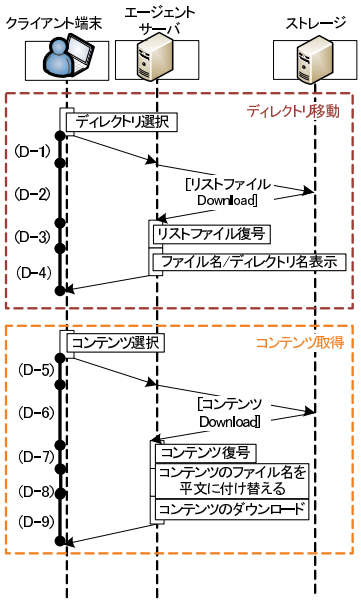

図 5 ファイル名/ディレクトリ名表示の処理手順
Fig. 5 Processing for listing file/directory names.

[*11] http://search.cpan.org/dist/WebService-Dropbox/

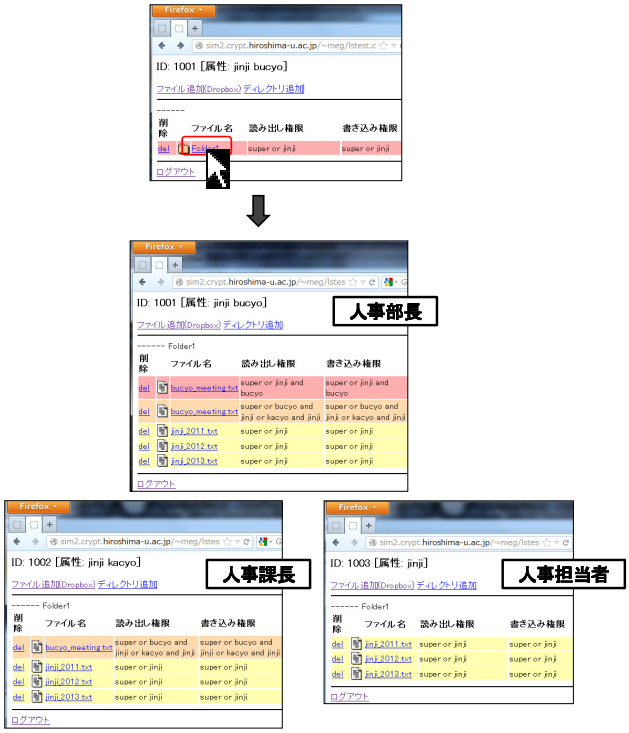

図 6 ディレクトリの移動（ファイル名/ディレクトリ名表示）のスクリーンショット

Fig. 6 Screen shot of a processing for listing file/directory names.

象のファイルを発見した場合，(D-5)～(D-9) の処理によってコンテンツのダウンロードおよび復号，ファイル名の復元をしてから一般ユーザはファイルを取得できる．図 6 は (D-1)～(D-4) によるディレクトリの移動の際の画面遷移であり，3 種類の属性により表示が異なることを確認できる．

4.2 ファイルのアップロードの処理手順

図 7 にファイルのアップロードの処理手順を示す．コンテンツの read 権，write 権はアップロードするディレクトリの write 権をデフォルトとし，Web インタフェースにより変更することも可能とする．

ファイルアップロードボタンを押したとき，(1-1)～(1-12) の CP-ABE によるチャレンジアンドレスポンス認証の処理によって一般ユーザが write 権を満たす属性を有しているかを確認する．

属性の確認後，(2-1)～(2-16) の処理によってアップロードファイルのファイル名をリストファイルに登録し，(3-1)～(3-8) の処理によってコンテンツが暗号化されたファイルをファイル名を秘匿した状態でストレージにアップロードする．(2-3)～(2-15) はロックファイルをストレージに置いて管理することでリストファイルの変更について簡易の排他制御を行う．同様に，(2-11)～(3-6) の処理ではアップロードするファイルの保存用ファイル名（擬似乱数によって生成）が衝突して上書きしない/されないように，保存用ファイル名が存在していないことを確認 (2-9) したうえで，空

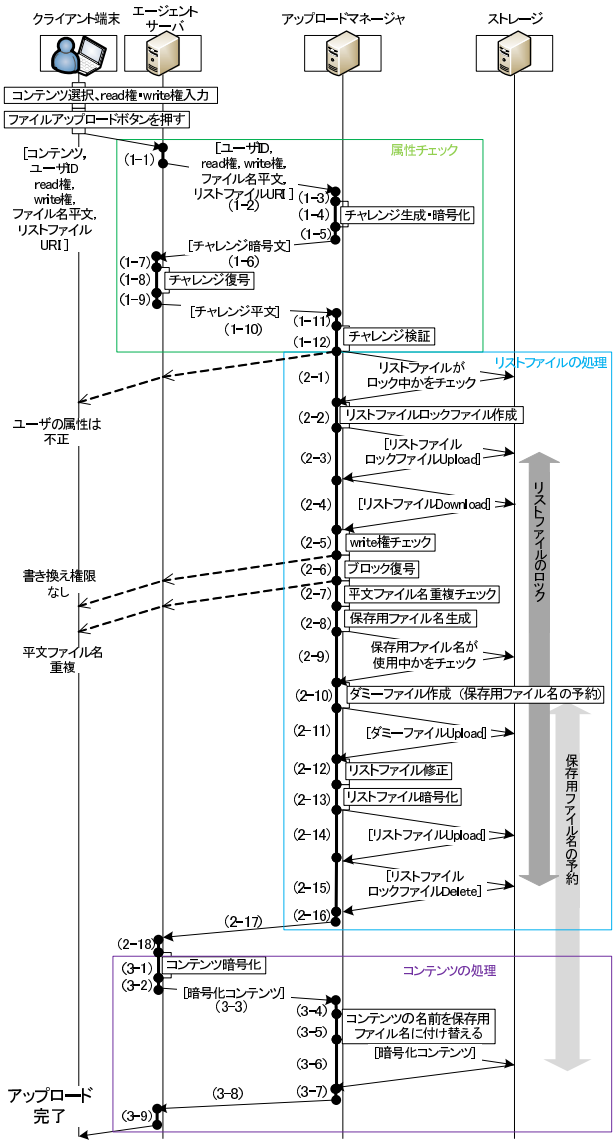

図 7 ファイルアップロードの処理手順

Fig. 7 Processing for uploading a user's file.

のファイル（ダミーファイル）をストレージに置くことで予約状態にする．そのほか，同じファイル名が同じ read 権のブロックの中に存在することは制限しているため，もしすでにそのファイル名が登録されていたなら (2-7) でアップロード処理を中断する．

5. 評価

本章の評価実験で使用した機器の仕様を表 2 に，実験ネットワークの構成を図 8 に示す．

5.1 CP-ABE の暗号化と復号時間の測定

cpabe toolkit を用いて，CP-ABE の暗号化/復号に要する時間を属性数やその結合の方法およびファイルサイズを変更して計測した．計測には表 2 のエージェントサーバを用い，暗号化/復号を 100 回試行した平均値を算出した．CP-ABE の暗号化/復号に要する時間は，コンテンツ

表 2 性能測定に使用した機器の仕様
Table 2 Specifications used for performance evaluation.

	クライアント端末	エージェントサーバ	アップロードマネージャ	ローカルストレージ
CPU	Intel®Atom™N270 1.60 GHz	Intel®Core™i7 965 @ 3.20 GHz	Intel®Core™i7 920 @ 2.67 GHz	Intel®Core™i7 920 @ 2.67 GHz
Memory	1 GB	12 GB	6 GB	6 GB
OS	Windows XP SP3	Debian 6.0.5	Debian 6.0.5	Debian 6.0.5

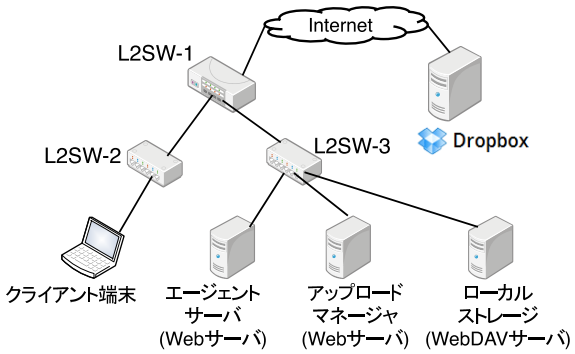

図 8 性能測定のネットワーク構成
Fig. 8 Network topology for performance evaluation.

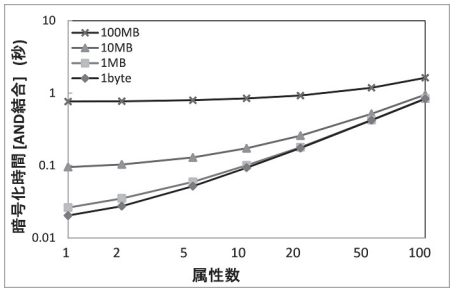

図 9 CP-ABE の暗号化時間（AND 結合）
Fig. 9 Evaluation of the encryption using CP-ABE with logical formula combined by only AND operation.

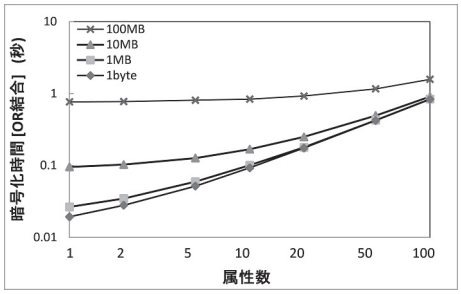

図 10 CP-ABE の暗号化時間（OR 結合）
Fig. 10 Evaluation of the encryption using CP-ABE with logical formula combined by only OR operation.

サイズやアクセス権の複雑さによって変化する．2 章で述べたように，アクセス権は属性を AND や OR で結合して表現される（例：人事部 AND（部長 OR 課長））．つまり，AND や OR の利用回数と，それにより結合される属性の個数によりアクセス権の複雑さが決まる．そこで，コンテンツサイズおよび属性数やその結合を変更して計測した．具体的には，処理するファイルは 1 byte，1 MB，10 MB，100 MB の 4 種類を用意し，それぞれに属性を AND のみで結合した場合（A001 AND A002 AND ···）と OR のみで結合した場合（A001 OR A002 OR ···）について，属性数を変化させて計測した．

cpabe tookit に実装されている CP-ABE のアルゴリズム [2] では，秘密分散法の技術を利用して AND や OR の論理演算を実現している．n 個の属性を AND で連結する場合，秘密情報を n 個の分散情報に分割し，n 個すべての分散情報が揃わなければ元の情報が得られないように秘密分散法のパラメータを指定する．さらに，それぞれの分散情報に対して各属性の鍵を対応付けて暗号化を施すことで，n 個の属性に対応する秘密鍵をすべて持つ場合のみ復元できるようになり AND が実現される．この場合，暗号化も復号も暗号化/復号処理を n 回実行することになるため，属性数に比例して CP-ABE の処理時間が増加すると予想される．また，n 個の属性を OR で連結する場合は n 個の分散情報を作成するときに，そのうちの 1 つが得られた場合に元の情報が得られるように秘密分散法のパラメータを指定する．そうすれば，任意の 1 つの属性の鍵を有している場合に復元できるため OR が実現される．この場合，暗号化は AND と同じく n 回の暗号化処理が必要となり属性数に比例するが，復号の際には合致する属性を用いた 1 回の復号処理だけでよいため属性数が増えても一定の時間になると考えられる．また，cpabe tookit はハイブリッド型の処理をしていることから，ファイルサイズが小さい場合には共通鍵暗号の処理は無視できるため CP-ABE の処理時間が支配的になり属性数の影響を受けやすくなり，ファイルサイズが十分に大きくなった場合には共通鍵暗号の処理時間が支配的になり属性数によらずに一定の処理時間となると予想される．

実験によって得られた属性数と暗号化時間の関係を**図 9** と**図 10**，属性数と復号時間の関係を**図 11** と**図 12** に示す．ファイルサイズが 1 byte のように小さい場合では CP-ABE の処理速度が支配的になっており，AND 結合では暗号化/復号の処理時間が属性数に比例し，OR 結合では暗号化は属性数に比例した処理時間，復号の処理時間は一定値（AND で属性数が 1 のときと同程度）となった．これは前段落で述べた CP-ABE の処理の性質と合致している．また，ファ

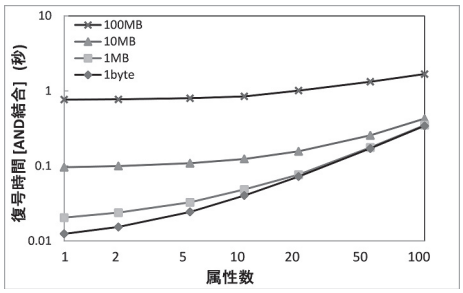

図 11 CP-ABE の復号時間（AND 結合）

Fig. 11 Evaluation of the decryption using CP-ABE with logical formula combined by only AND operation.

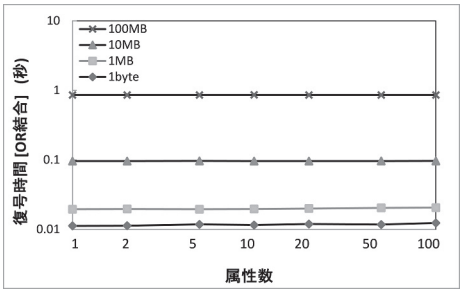

図 12 CP-ABE の復号時間（OR 結合）

Fig. 12 Evaluation of the decryption using CP-ABE with logical formula combined by only OR operation.

イルサイズが 100 MB のように大きい場合では共通鍵暗号の処理時間が支配的となり，連結数が 20 以下など小さい範囲では処理時間が 1 秒程度の一定の値をとり，CP-ABE の処理時間は無視できる程度であることが分かった．

5.2 プロトタイプシステムによる提案方式の評価

プロトタイプシステムを用いて提案方式によるファイル名/ディレクトリ名表示とアップロード時の処理の時間について計測する．

実験の条件について述べる．リストファイルの一行（ファイル名 1 つ分の情報）を約 500 byte と仮定し，リストファイルが保持するファイル名の数を 200 個（100 KB），2,000 個（1 MB），20,000 個（10 MB）として 3 種類のリストファイルを作成した．リストファイルの中身は 10 個のブロックに分割し，各ブロックを暗号化するアクセス権は AND や OR を利用しないシンプルな属性値を用いる．ファイル名表示時間およびファイルのアップロード時間におけるリストファイルサイズによる影響を調べるため，上記の 3 種類のサイズのリストファイルを利用して，図 5 の (D1)〜(D4) および図 7 の各ステップを測定した．ファイル名/ディレクトリ名表示の計測では 10 個のブロックをすべて復号する．また，ファイルのアップロード時間の測定ではリストファイルの影響が出やすいように，小さなサイズ（1 KB）のファイルを 1 つアップロードする処理で評価した．

測定はエージェントおよびアップロードマネージャの実行プログラム内に Perl の Time::HiRes モジュールの time 関数を埋め込み，図 5 と図 7 で示した各ステップにおけるタイムスタンプの差分を算出することによって行った．各測定は 10 回行い，その平均値を測定値とした．

表 3 リストファイルサイズとファイル名/ディレクトリ名表示時間（単位：秒）

Table 3 Processing time for listing file/directory names.

Step	100 KB	1 MB	10 MB
(D-1)	0.00	0.00	0.00
(D-2)	1.94	4.74	15.42
(D-3)，(D-4)	0.13	0.25	1.11
(D-*) 合計	2.07	4.99	16.53

5.2.1 ファイル名/ディレクトリ名表示時間

ファイル名/ディレクトリ名表示時間の測定結果を**表 3** に示す．ファイル名/ディレクトリ名表示時間のうち，各リストファイルサイズにおいて約 94 パーセントが Dropbox からリストファイルを取得する時間 (D-2) であり，通信時間が支配的となっていることが分かる．(D-3)，(D-4) の復号処理はリストファイルサイズが 100 KB で 0.13 秒，1 MB で 0.25 秒，10 MB で 1.11 秒であった．各リストファイルのブロック数および属性数は同じであるため，CP-ABE によるセッション鍵の復号処理は定数の処理時間，共通鍵暗号の復号処理はファイルサイズに比例する処理時間となる．CP-ABE の復号時間を a，共通鍵暗号の復号時間を $b \cdot x$ とすると，ハイブリッド型処理の合計時間は $a + b \cdot x$ のように表すことできる．ここで，b は定数，x はファイルサイズに依存した変数とする．100 KB を $x = 1$ とおくと，1 MB は $x = 10$，10 MB は $x = 100$ として計算することができる．ここで 100 KB と 1 MB の実験結果から連立方程式を解いて a と b の値を求めると $a = 0.13 - 0.12/9 \sim 0.117$，$b = 0.12/9 \sim 0.013$ となる．これを 10 MB に代入すると $a + 100\,b = 1.417$ となり，実験値である 1.11 と近い値であることが確認できる．これによって CP-ABE の処理時間は $a = 0.117$ 秒程度，共通鍵暗号の処理時間は X KB のファイルを扱う場合に $X/100 \cdot b = X \cdot 0.00013$ 秒程度であったと見積もることができる．

Dropbox を用いたプロトタイプシステムでも，たとえば一度に 200 個のファイル名（100 KB のサイズのリストファイル）を表示する程度までの使い方であれば許容できると考えている．なお，各ディレクトリでリストファイルは異なるため，この制限でも運用可能な場面は十分にあると考えられる．

5.2.2 ファイルのアップロード時間

ファイルのアップロード時間に関する測定結果を**表 4** に示す．表 4 には Dropbox との通信部分および CP-ABE の暗号化/復号処理に関するステップのみ抜粋した．CP-ABE を用いたチャレンジアンドレスポンス方式による一般ユー

表 4 リストファイルサイズとファイルのアップロード時間（単位：秒）
Table 4 Processing time for uploading a user's file.

Step	100 KB	1 MB	10 MB
(1-4)	0.02	0.02	0.02
(1-8)	0.02	0.02	0.02
(1-*) その他	0.12	0.12	0.12
(1-*) 合計	0.26	0.26	0.26
(2-1)	0.66	0.65	0.69
(2-3)	1.25	1.26	1.26
(2-4)	2.42	4.59	15.68
(2-6)	0.02	0.02	0.01
(2-9)	0.66	0.68	0.76
(2-11)	1.16	1.52	1.24
(2-13)	0.04	0.04	0.06
(2-14)	1.85	4.22	18.89
(2-15)	0.83	0.90	0.91
(2-*) その他	0.00	0.00	0.02
(2-*) 合計	8.89	13.88	37.41
(3-1)	0.03	0.03	0.03
(3-6)	1.43	1.58	1.45
(3-*) その他	0.08	0.09	0.07
(3-*) 合計	1.54	1.69	1.55
合計	10.68	15.82	41.36

ザの属性の確認処理は約 0.26 秒であり，高速に処理されている．各リストファイルサイズにおいてアップロード時間のうち約 90 パーセント以上が Dropbox とのファイル転送時間であった．Dropbox との通信は (2-1)，(2-3)，(2-4)，(2-9)，(2-11)，(2-14)，(2-15)，(3-6) のように計 8 回発生するため，ストレージの処理速度や通信速度が大きく影響する．特に，ロックファイルやダミーファイルの処理はリストファイルサイズに関係なく発生するにもかかわらず，(2-1)，(2-3)，(2-9)，(2-11)，(2-15) の 5 回の通信で合計約 5 秒も要するためリストファイルサイズが小さいときの影響が大きい．リストファイルの編集の際には 1 つのブロックにのみ追記処理として復号および暗号化を行うため，リストファイルサイズが 10 MB のときでも対象となるブロックサイズは 1 MB のように小さくなる．この編集に関する処理に対応するのは (2-6) と (2-13) であり，100 KB，1 MB，10 MB のすべてにおいて合計で 0.07 秒以下と全体に占める割合は小さい．

5.3 ローカルストレージによる評価

プロトタイプシステムでは Dropbox を利用していたが，本章の実験によりファイル名/ディレクトリ名表示処理およびアップロード処理の両方でストレージとの通信時間が支配的となり，全体の処理時間を大きくしているという結果を得た．そこで，ストレージの性能が向上することによる影響を調べるため，ネットワークの遅延の影響が極力小さくなるように，図 8 で示したようにアップロードマネージャやエージェントと同一 LAN 内に WebDAV で構築したオンラインストレージ（ローカルストレージと呼ぶ）を設置して Dropbox に該当する処理を置き換えて実験を行った．なお，このローカルストレージは専用で使うためサーバへの負荷が小さい状態で計測される．

表 5 ローカルストレージを利用した場合の処理時間（単位：秒）
Table 5 Processing time of the proposed method using a local storage.

リストファイルサイズ	100 KB	1 MB	10 MB
ファイル名/ディレクトリ名表示時間	0.20	0.30	0.89
ファイルのアップロード時間	2.95	2.99	3.39

ローカルストレージを利用した場合のファイル名/ディレクトリ名表示とアップロード時の処理の時間の測定結果を**表 5** に示す．ローカルストレージを利用した場合，ファイル名/ディレクトリ名表示は 1 秒以内，ファイルのアップロードは約 3 秒で処理できており，Dropbox の結果から大きく改善できている．この結果では一度に 20,000 個のファイル名（10 MB のサイズのリストファイル）を表示するような使い方でも許容できる程度の時間で処理が完了する．

なお，ファイルのアップロード時間について改善の効果が比較的小さい理由は，リストファイルのロック処理や保存ファイル名の予約処理の影響でアップロードマネージャ・ストレージ間の通信回数が増加し，応答時間に優れたローカルストレージを使って 1 回の応答時間が改善されたとしても一定の処理時間が残ってしまっているからと考えられる．

6. 考察

6.1 リストファイルの分割による処理時間の削減

5 章の実験結果より，Dropbox をストレージとして用いる場合，ファイル名/ディレクトリ名表示とアップロード時の処理の時間のうち Dropbox とのファイル転送時間が支配的であることが分かった．そこで，リストファイルのフォーマットを変更することで転送時間を削減することを考える．3 章で述べたリストファイルのフォーマットでは，あるディレクトリ内に存在するファイルの read 権の数だけブロックを作成し，それらを結合したものを 1 つのリストファイルと呼んでいた．これらのブロックを個別のファイルとして分割し，そのファイル名をブロック番号で管理して（例：***.1.lst），論理的にグループ化することでブロック数の増加に対する通信時間の影響を抑えることが可能である（**図 13**）．ブロックの番号と read 権の対応関係はリストファイルのヘッダに記述し，ヘッダも個別のファイルとして扱う．ファイル名表示のときは，最初にヘッダファイルをダウンロードして，復号権限のあるブロック番号を調べることで必要なものだけを取得する．

上記の方法に変更した場合，ディレクトリ内の read 権の

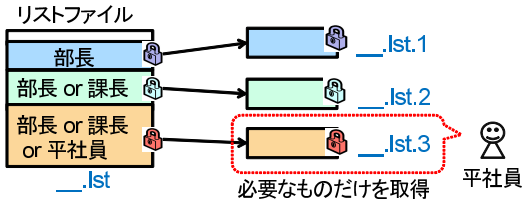

図 13 リストファイルの転送時間を削減する方法
Fig. 13 The method to reduce data transfer time for list file.

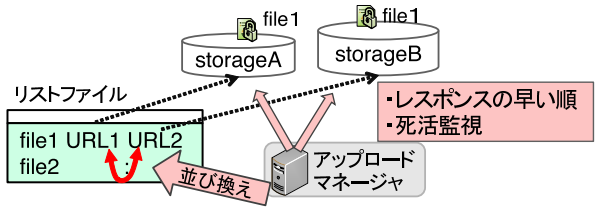

図 14 複数のストレージを利用したファイルの冗長化方法
Fig. 14 Redundancy method by using multiple online storages.

種類が変化した場合でも保存ファイル名のブロック番号が異なるファイルが増えるだけであり，保存ファイル名の共通部分によって同一ディレクトリのリストファイルか否かを判断すれば，3.3.4 項で議論したディレクトリ単位でのリストファイルのバックアップ処理への悪影響は生じない．

なお，小さなデータでもサーバの応答時間がかかるので，最適なパラメータを調べる必要がある．最適なパラメータに関する検討は今後の課題とする．

6.2 複数のストレージを利用したファイルの冗長化

システム障害などを考慮するとファイルの冗長化は重要である．提案システムにおいて，リストファイルとアップロードマネージャを利用することでファイルの複製による多重化を実現できる．ファイルの保存場所はリストファイル中に URL で保持されているので，ファイルを複製する個数だけ URL を列挙する（**図 14**）．

複数のストレージにファイルが配置されているとき，すべてのストレージに同時にアクセスするとトラフィックが増大する．そこで，リストファイル中の URL の並びに優先順位をつけ，ファイルをダウンロードするときは処理性能の高いストレージから順にアクセスする．優先順位はアップロードマネージャがストレージを定期的に観測し，レスポンス速度やストレージ容量を考慮しながら決定する．この方法は，ストレージの障害発生を検出することも可能であるため死活監視にも役立つ．

ファイルの冗長化に関して，単純にファイルを複製して多重化するだけでなく秘密分散法 [15] を使って分散管理をサポートすることも可能である [16]．特に個人情報の取扱い [17] によっては秘密分散法の技術を要求される場合がある[*12]．

7. おわりに

本論文では暗号文ポリシー属性ベース暗号を用いて，ファイル名/ディレクトリ名を効率的に秘匿する方式を提案した．この方式では同じ復号権限を持つファイル名/ディレクトリ名をリストファイルという単位で管理して一括で暗号化/復号することで高速化を図った．さらにリストファイルの編集権限を制御するためにアップロードマネージャというサーバを定義し，リストファイルを編集するための適切なプロトコルを与えた．提案方式を実装したプロトタイプシステムを用いてファイル名/ディレクトリ名表示およびファイルのアップロード時間について評価した結果，処理時間の中でストレージとのファイル転送時間が支配的であることが分かった．また，ローカルストレージを使用した実験より，応答時間にすぐれたストレージを使うことで，ファイル名/ディレクトリ名の表示やファイルのアップロード処理が大幅に高速化できることが分かった．

今後の課題として，リストファイルのフォーマットや処理手順の見直しにより，逐次的にファイル名/ディレクトリ名を表示させるなどして，通信速度が速くないストレージに対してもストレスなく利用できるように改良することがあげられる．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金若手研究（B）（課題番号 25730085）および基盤研究（B）（課題番号 23300026，24300025）の助成を受けたものである．

[*12] 文献 [17] のガイドラインでは個人情報の判断について「暗号化等によって秘匿化されているかどうかを問わない」との記載があり，セキュリティポリシーによっては，暗号化の有無にかかわらず外部のサーバに置けないことがある．一方，秘密分散法で情報を分割させた場合，復元に必要なすべてのファイルを集めた場合に限り個人情報となるため，分割されたファイルであれば外部サーバに置ける可能性がある．

参考文献

[1] 永見健一，伊波源太，笹川 浩，脇谷康宏：セキュアなオンラインストレージシステムの提案，情報処理学会研究報告 IOT，Vol.2011, No.7, pp.1–5 (2011).
[2] Bethencourt, J., Sahai, A. and Waters, B.: Ciphertext-Policy Attribute-Based Encryption, *IEEE Symposium on Security and Privacy*, pp.321–334, IEEE Computer Society (2007).
[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，電子情報通信学会技術研究報告 LOIS，ライフインテリジェンスとオフィス情報システム，Vol.111, No.383, pp.73–78 (2012).
[4] Zhao, F., Nishide, T. and Sakurai, K.: Realizing Fine-Grained and Flexible Access Control to Outsourced Data with Attribute-Based Cryptosystems, *ISPEC*, Bao, F. and Weng, J. (Eds.), Lecture Notes in Computer Science, Vol.6672, pp.83–97, Springer (2011).
[5] Maji, H.K., Prabhakaran, M. and Rosulek, M.: Attribute-Based Signatures, *CT-RSA*, Kiayias, A. (Ed.), Lecture Notes in Computer Science, Vol.6558, pp.376–

392, Springer (2011).
[6] Sahai, A. and Waters, B.: Fuzzy Identity-Based Encryption, *EUROCRYPT*, Cramer, R. (Ed.), Lecture Notes in Computer Science, Vol.3494, pp.457–473, Springer (2005).
[7] Okamoto, T. and Takashima, K.: Fully Secure Functional Encryption with General Relations from the Decisional Linear Assumption, *CRYPTO*, Rabin, T. (Ed.), Lecture Notes in Computer Science, Vol.6223, pp.191–208, Springer (2010).
[8] Jahid, S., Mittal, P. and Borisov, N.: EASiER: Encryption-based Access Control in Social Networks with Efficient Revocation, *ASIACCS*, Hong Kong (2011).
[9] Nishide, T., Yoneyama, K. and Ohta, K.: Attribute-Based Encryption with Partially Hidden Ciphertext Policies, *IEICE Trans.*, Vol.92-A, No.1, pp.22–32 (2009).
[10] Emura, K., Miyaji, A., Omote, K., Nomura, A. and Soshi, M.: A ciphertext-policy attribute-based encryption scheme with constant ciphertext length, *IJACT*, Vol.2, No.1, pp.46–59 (2010).
[11] 市川幸宏，松田　規，坂上　勉：関数型暗号アプリケーションにおける適切な述語付与方式の検討，情報処理学会研究報告 CSEC，Vol.2013, No.5, pp.1–6 (2013).
[12] 毛　亮堅，櫻井幸一：ロール・ポリシーに基づくドキュメントアクセス制御モデル，電子情報通信学会技術研究報告 ICSS，情報通信システムセキュリティ，Vol.110, No.266, pp.41–46 (2010).
[13] 山崎　航，平石広典，溝口文雄：動的なセキュリティポリシのための RBAC システムの設計，情報処理学会論文誌，Vol.47, No.6, pp.1932–1940 (2006).
[14] 金岡　晃，祝　拓也，岡本栄司：ID ベース暗号における鍵生成センタの実装，電子情報通信学会技術研究報告 ICSS，情報通信システムセキュリティ，Vol.110, No.115, pp.133–139 (2010).
[15] Shamir, A.: How to share a secret, *Comm. ACM*, Vol.22, No.11, pp.612–613 (online), DOI: 10.1145/359168.359176 (1979).
[16] 熊谷悠平，西村浩二，大東俊博，近堂　徹，相原玲二：認証フェデレーションに基づく分散ファイル管理システムの提案，情報処理学会研究報告 IOT，Vol.2012, No.8, pp.1–6 (2012).
[17] 経済産業省：個人情報の保護に関する法律についての経済産業分野を対象とするガイドライン，経済産業省（オンライン），入手先 ⟨http://www.meti.go.jp/policy/it_policy/privacy/kojin_gadelane.htm⟩（参照 2009-10-09）.

大東 俊博　（正会員）

2002 年徳島大学工学部知能情報工学科卒業．2004 年同大学大学院工学研究科博士前期課程修了．2008 年神戸大学大学院自然科学研究科博士課程後期課程修了．現在，広島大学情報メディア教育研究センター助教．博士（工学）．暗号理論，ネットワークセキュリティ，認証プロトコルに関する研究に従事．電子情報通信学会 SCIS20 周年記念賞，SCIS2013 イノベーション論文賞を受賞．電子情報通信学会正員．

後藤 めぐ美

2011 年広島大学工学部第二類（情報系）卒業．2013 年同大学大学院総合科学研究科博士前期課程修了．現在，株式会社 OKI ソフトウェア．修士（学術）．情報システムの構築・開発に関する業務に従事．

西村 浩二　（正会員）

1989 年広島大学工学部第二類（電気系）卒業．1991 年広島大学大学院工学研究科博士課程前期修了．広島大学総合情報処理センター助手，同大学情報メディア教育研究センター准教授等を経て，2011 年より同教授．博士（工学）．コンピュータネットワークの運用管理，移動透過通信，情報セキュリティに関する研究に従事．電子情報通信学会会員．

相原 玲二　（正会員）

1981 年広島大学工学部第二類（電気系）卒業．1986 年同大学大学院工学研究科博士課程後期修了．同大学助手，同大学集積化システム研究センター助教授を経て，現在，同大学情報メディア教育研究センター教授．工学博士．コンピュータネットワークに関する研究に従事．電子情報通信学会，IEEE Computer Society，IEEE Communications Society 各会員．