安全なクラウドストレージを実現するFADEへの改良の提案

田中 敏之 †　　西出 隆志 ‡　　櫻井 幸一 ‡

† 九州大学大学院システム情報科学府
819-0395 福岡県福岡市西区元岡 744
yuki@itslab.csce.kyushu-u.ac.jp

‡ 九州大学大学院システム情報科学研究院
819-0395 福岡県福岡市西区元岡 744
nishide@inf.kyushu-u.ac.jp, sakurai@csce.kyushu-u.ac.jp

あらまし クラウドサービスの１つであるストレージサービスは，さまざまなファイルをインターネット経由で保存でき，利用したい時に読み出せる．しかし，クラウドは保存したファイルを分散して管理するので，ユーザはコピーの数やどのサーバーに保存されているかを知ることはできない．そのため，保存したファイルの削除をクラウドに要求しても，すべてのコピーが削除されるかどうかは分からない．そこで，本研究では Tang らの File Assured Deletion (FADE) と呼ばれる手法に着目し，FADE で用いられている RSA 暗号以外に ElGamal 暗号も利用可能とする改良と鍵マネージャーの複数化による安全性向上を提案する．

Improvement of FADE to Achieve Secure Cloud Storage

Toshiyuki Tanaka†　　Takashi Nishide‡　　Kouichi Sakurai‡

†Graduate School of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395, JAPAN
yuki@itslab.inf.kyushu-u.ac.jp

‡Faculty of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka 819-0395, JAPAN
nishide@inf.kyushu-u.ac.jp, sakurai@csce.kyushu-u.ac.jp

Abstract A cloud storage service can store various files via Internet accesses and users can retrieve the files whenever users want to use. However, because the cloud distributes and manages stored files, users do not know the number of copies and which server stores data. Therefore, users do not understand whether all the copies are deleted even if users request the deletion of stored files to the cloud. Then, we focus on File Assured Deletion (FADE) and we add ElGamal encryption besides RSA used in FADE and improve the security of FADE by introducing multiple key managers.

1　はじめに

ブロードバンドや無線 LAN の普及により，ユーザはいつでもどこからでもインターネットにアクセスすることが可能になった．そのため，インターネット経由で提供されるクラウドサービスの利便性が向上し，多くのユーザに利用されている．

クラウドサービスの１つにストレージサービスがある．クラウドコンピューティングが提供

するストレージサービスは，ユーザが保有するデータの管理コストを大幅に削減することができると期待されている．しかし，クラウドストレージサービスはバックアップのために多くのデータのコピーを作成し，それらを分散して保存する可能性もあるため，ユーザはバックアップのコピーの数やその格納場所がどこにあるのか知ることはできない．そのため，保存したデータの削除をクラウドストレージサービスに要求しても，全てのデータのコピーが完全に削除されたかどうかはユーザ側からは分からない．

その解決策としてさまざまな手法が提案されているが，本研究では Tang らの，暗号化したデータをクラウドストレージに，暗号化鍵を鍵マネージャーに分離して管理させることでデータのプライバシーと完全性を保証する FADE(File Assured Deletion) と呼ばれる手法 [1] に着目し，その改良を目指している．

本発表では，FADE において，ユーザの秘密鍵解読時にユーザ･鍵マネージャー間で用いられる RSA 暗号の代わりに，高速化のために鍵生成が早い ElGamal 暗号を用いる手法を提案する．また，クラウドストレージと鍵マネージャーが結託していた場合の対策についても考察する．

2 クラウドコンピューティング

クラウドサービスとは，クラウドコンピューティングによって提供されるサービスの総称である．クラウドサービスとして提供されている主なサービスとしては，Amazon Web Services[2], Windows Azure[3]，Google App Engine[4] などを挙げることができる．クラウドサービスの 1 つであるクラウドストレージサービスは，クラウド上にユーザのデータを保存することができるサービスである．クラウドストレージサービスの代表的なものとしては，Amazon S3，Dropbox，SugarSync などがある．このサービスには以下のような利点がある．

- ユーザの PC の容量が少なくても多くのデータをクラウド上に保存することができる．
- 特定の PC 以外からもデータにアクセスできる．
- 自分の PC が壊れても，データは消えない．

しかし，クラウドストレージに保存したデータはバックアップのためにコピーされ，分散して保存されるため，それらのコピーの削除を要求したときに，すべて確実に削除されるかどうかは分からず，そこからデータが漏えいするなどのリスクが発生する可能性がある．そのため，データが削除されなかった場合のことも考え，データがユーザ以外の誰からも見られないようにする方法が必要である．

3 関連研究

クラウドストレージに保存したファイルを保護する既存研究として，Wang らはユーザのアクセス権と保存したファイルの変化をサポートする安全なアクセスメカニズムを提案した [5]．また，Sahai らは属性ベースの暗号化（ABE）を提案した [6]．その後 ABE の実装や拡張が行われた [7, 8]．さらに Li らによって，個人健康記録（PHR）に対するきめ細かいアクセス制御を可能にするために ABE が利用された [9]．

また，Perlman により，有効期限が過ぎた後はファイルに永久にアクセスすることができなくなる時間ベースのファイル削除が提案された [10]．これはまずファイルをデータ鍵で暗号化し，そのデータ鍵をさらに制御鍵で暗号化する．制御鍵は鍵マネージャーによって管理され，有効期限が過ぎれば鍵マネージャーによって制御鍵は取り除かれる．制御鍵がないとデータ鍵を復元できず，制御鍵がないとファイルを復元できないので，ファイルは暗号化されたままでアクセスすることができないと考えられる．そのプロトタイプに Vanish[11] がある．

さらに，時間ベースのファイル削除の拡張としてポリシーベースのファイル削除が提案された．各ファイルはファイルアクセスポリシー（以下ポリシー）と結びついており，各ポリシーは鍵マネージャーによって管理された制御鍵と結びついている．ファイルはデータ鍵で暗号化され，データ鍵はポリシーに対応する制御鍵でさらに暗号化される．ポリシーが無効になったとき，対応する制御鍵は鍵マネージャーから取り除かれる．このように，ポリシーが無効になっ

たとき，制御鍵が取り除かれるので，制御鍵で暗号化したデータ鍵を復元できず，データ鍵で暗号化したファイルも復元できない．

4 FADE (File Assured Deletion)

FADE はポリシーベースのクラウドストレージサービスであり，次の 3 者で成り立っている．

- データ所有者
  クラウドに保存するファイルを生成する．
- クラウドストレージ
  第 3 者のクラウドプロバイダによって管理される．
- 鍵マネージャー
  データ鍵を暗号化するために使うポリシーベースの制御鍵を管理する．
  制御鍵の暗号化，復号化，更新，取り消しを行うことでユーザのリクエストに応じる．

4.1 データ所有者・クラウドストレージのみのファイルのアップロード/ダウンロード

鍵マネージャーが存在しない場合について考える．この場合，データ所有者とクラウドストレージの 2 者だけでファイルのアップロード/ダウンロード行う．

- アップロード
  データ所有者はポリシー P_i に合うランダムな秘密鍵 S_i を生成する．データ所有者は S_i で暗号化されたファイル $\{F\}_{S_i}$ をクラウドストレージに送る．
- ダウンロード
  データ所有者はクラウドストレージから $\{F\}_{S_i}$ をダウンロードする．データ所有者は S_i を使って解読し，ファイル F を得る．

この場合，データ所有者は自分で秘密鍵を管理しなければならないので，利用環境に注意を払わなければならい．また，複数の鍵の利用には適していない．

4.2 データ所有者・クラウドストレージ・鍵マネージャーのファイルのアップロード/ダウンロード

鍵マネージャーが存在する場合について考える．この場合，データ所有者とクラウドストレージと鍵マネージャーの 3 者でファイルのアップロード/ダウンロード行う．図 1 にファイルのアップロード，図 2 にファイルのダウンロードを示す．

- アップロード
  (1) ポリシー P_i の公開鍵 (n_i, e_i) を要求する．ここで，(n_i, e_i) は RSA 暗号の公開鍵を意味する．
  (2) ポリシー P_i の公開鍵 (n_i, e_i) を返す
  (3) ランダムなデータ鍵 K とポリシー P_i に合うランダムな秘密鍵 S_i を生成する
  (4) P_i, $\{K\}_{S_i}$, $S_i^{e_i}$, $\{F\}_K$ をクラウドに送る
  (5) データ所有者は K と S_i を削除する

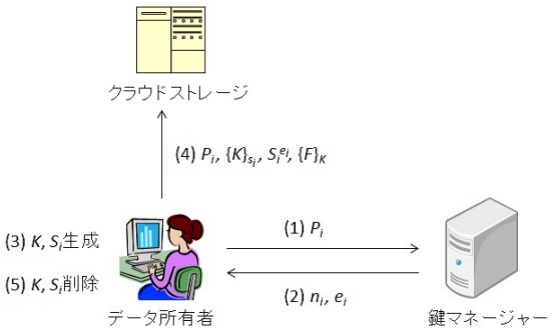

図 1: ファイルアップロード

- ダウンロード
  (1) P_i, $\{K\}_{S_i}$, $S_i^{e_i}$, $\{F\}_K$ をダウンロードする
  (2) 秘密の乱数 R を生成する
  (3) R^{e_i} を計算する
  (4) P_i, $S_i^{e_i} R^{e_i} = (S_i R)^{e_i}$ を送る
  (5) $((S_i R)^{e_i})^{d_i} = S_i R$ を計算し返す
  (6) $S_i R$ から R を取り除き，S_i を手に入れる

(7) $\{K\}_{S_i}$ を復号する

(8) $\{F\}_K$ を復号する

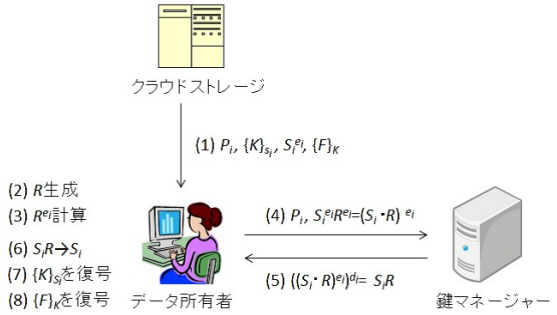

図 2: ファイルダウンロード

この場合，データ所有者は自分で鍵を持つ必要がない．これはデータ所有者の利用環境が安全でない場合に有効である．また，鍵を管理する必要がないので，複数の鍵を持つ場合に有効である．復号時，$S_i^{e_i}$ に R^{e_i} をかけて鍵マネージャーに送った理由は，$S_i^{e_i}$ のまま送ると，鍵マネージャーが復号したとき，データ所有者の秘密鍵 S_i がそのまま鍵マネージャーに見られてしまうため，R^{e_i} をかけることで，鍵マネージャーにデータ所有者の秘密鍵がわからないようにするためである．

4.3 ポリシー更新

FADE では，ファイルはポリシーと結びついている．例えば，ファイルの有効期限を延長したい場合には，ポリシーを更新する必要がある．図 3 にポリシー更新を示す．

(1) P_i, $S_i^{e_i}$ をダウンロードする

(2) P_i, $S_i^{e_i}R^{e_i} = (S_iR)^{e_i}$ を送る

(3) $((S_iR)^{e_i})^{d_i} = S_iR$ を計算し返す

(4) 新しいポリシー $P_{i'}$ の公開鍵 $(n_{i'}, e_{i'})$ を要求する

(5) ポリシー $P_{i'}$ の公開鍵 $(n_{i'}, e_{i'})$ を返す

(6) 公開鍵 $(n_{i'}, e_{i'})$ で S_i を $S_i^{e_{i'}}$ に再暗号化し，新しいポリシー $P_{i'}$ と共にクラウドストレージに送る

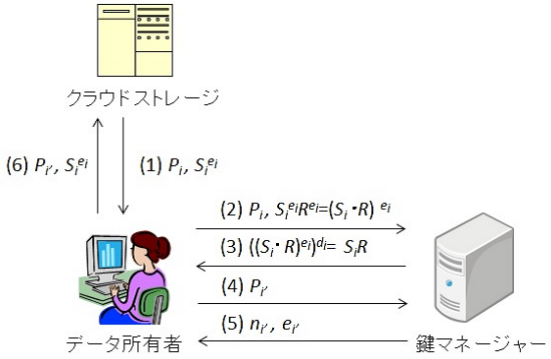

図 3: ポリシー更新

5 提案手法

5.1 ElGamal 暗号による FADE の改良

FADE では，データ所有者の秘密鍵の暗号化・復号化に RSA 暗号を用いていた．本研究では，RSA 暗号の代わりに，ElGamal 暗号を用いる方法を提案する．RSA 暗号は素数 p,q を生成するのに時間がかかるが，ElGamal 暗号は一度生成した素数 p は何度も使えるため，鍵生成が速い．鍵はポリシーの更新時に生成されるので，ポリシー更新が頻繁に起こる場合に ElGamal 暗号を用いると，高速化することができると考えられる．p を大きな素数とし，g を p を法とする原始根，x を秘密鍵，$y = g^x \bmod p$ とする．このとき (y, p, g) が公開鍵となる．平文を M，暗号文を (C_1, C_2)，r を乱数とする．M を暗号化すると，

$$\begin{aligned} C_1 &= My^r \bmod p \\ C_2 &= g^r \bmod p \end{aligned}$$

となる．復号化する場合は，

$$\begin{aligned} C_2^x &= (g^r)^x \bmod p \\ &= (g^x)^r \bmod p \\ &= y^r \bmod p \\ M &= \frac{C_1}{C_2^x} \bmod p \end{aligned}$$

より，平文 M を復号できる．FADE に対応させると，暗号化の時の平文 M は S_i，復号化の時の平文は S_iR となる．

5.2 鍵マネージャーの複数化

FADE では鍵マネージャーは信頼できるものであると仮定しているが，実際に鍵マネージャーが信頼できるかどうかはわからない．そこで，鍵マネージャーを複数用意することで，安全性を高めることができるのではないかと考えられる．

- 単純な鍵マネージャーの複数化
  例えば，鍵マネージャーが 2 つの場合，ユーザはデータ鍵 K_1,K_2 とポリシー P_i,$P_{i'}$ に合うランダムな秘密鍵 S_i,$S_{i'}$ を生成する．K_1 は S_i で暗号化し，S_i を鍵マネージャー 1 の公開鍵で暗号化する．K_2 は $S_{i'}$ で暗号化し，$S_{i'}$ を鍵マネージャー 2 の公開鍵で暗号化する．ユーザのファイルは K_1 と K_2 の XOR などで暗号化し，これらすべてをクラウドストレージに保存する．仮に鍵マネージャー 1 とクラウドストレージが結託していたとしたも，復号されるのは S_i と K_1 である．しかし $S_{i'}$ と K_2 がわからなければファイルを復号できないので，安全性が向上したといえる．図 4 に鍵マネージャーが 2 つの場合のファイルアップロードを示す．

  (1) ポリシー P_i の公開鍵 (n_i, e_i) を要求する

  (2) ポリシー P_i の公開鍵 (n_i, e_i) を返す

  (3) ポリシー $P_{i'}$ の公開鍵 $(n_{i'}, e_{i'})$ を要求する

  (4) ポリシー $P_{i'}$ の公開鍵 $(n_{i'}, e_{i'})$ を返す

  (5) ランダムなデータ鍵 K_1 とポリシー P_i に合うランダムな秘密鍵 S_i を生成する

  (6) ランダムなデータ鍵 K_2 とポリシー $P_{i'}$ に合うランダムな秘密鍵 $S_{i'}$ を生成する

  (7) P_i, $P_{i'}$, $\{K_1\}_{S_i}$, $S_i^{e_i}$, $\{K_2\}_{S_{i'}}$, $S_{i'}^{e_{i'}}$, $\{F\}_{K_1 \oplus K_2}$ をクラウドに送る

  (8) データ所有者は K_1 と S_i を削除する

  (9) データ所有者は K_2 と $S_{i'}$ を削除する

- 秘密分散法による鍵マネージャーの複数化
  秘密分散法である (k, n) しきい値法を利用する．データ鍵 K を n 個の分散情報に分散し，その分散情報を n 個の鍵マネージャーがそれぞれ一つずつ管理する．この分散情報から任意の k 個を集めると元のデータ鍵 K を復元できる．しかし，任意の $k-1$ 個の分散情報を集めても，元のデータ鍵 K を復元することはできない．図 5 に $(3, 5)$ しきい値法の例を示す．

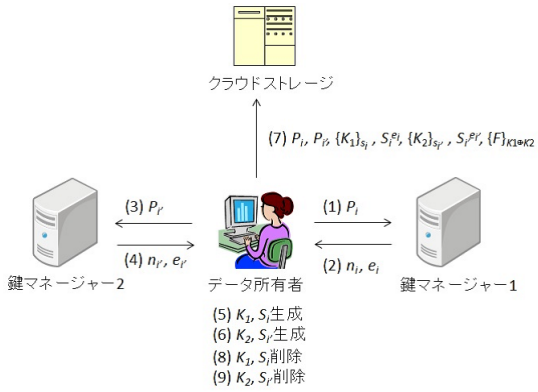

図 4: 複数鍵マネージャーのファイルアップロード

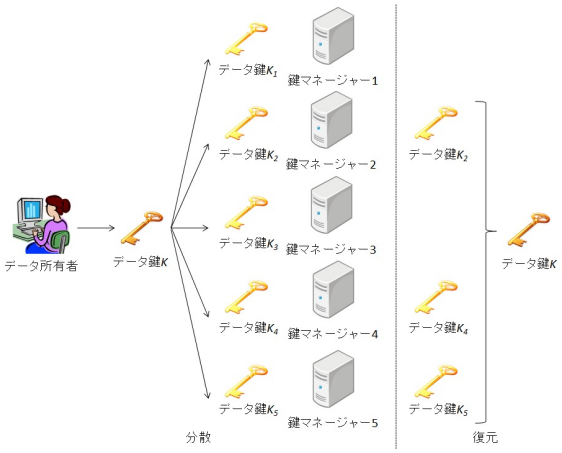

図 5: (3, 5) しきい値法の例

単純な鍵マネージャーの複数化の場合，全ての鍵マネージャーからデータ鍵を集める必要があったが，一つでも鍵マネージャーが動作していない場合，ファイルを復元することはできない．しかし，秘密分散法を用いた場合，任意の k 個の分散情報を集めるだけでよいので，数個の鍵マネージャーが動作していなかったとしてもファイルを復元することができる．

6 まとめと今後の課題

本論文ではクラウドストレージを利用する場合のデータ紛失・漏えいの問題に対して，Yang らのファイルを確実に削除する FADE と呼ばれるポリシーベースのクラウドストレージシステムを紹介した．また，FADE ではユーザの秘密鍵の暗号化・復号化に RSA 暗号を用いていたが，代わりに ElGamal 暗号を用いる手法を提案した．ElGamal 暗号は鍵生成が速いため，鍵生成が頻繁に起こる場合，つまりポリシー更新が頻繁に起こる場合に有利である．さらに，鍵マネージャーとクラウドストレージが結託していた場合に，安全性を向上させるために複数の鍵マネージャーを用いる場合の考察を行った．

今後の課題として，ElGamal 暗号を用いた FADE の実装を行う．また，鍵マネージャーを複数にした場合の性能評価を行う．さらに，FADE ではクラウドストレージに Amazon S3 を用いていたが，他のクラウドストレージが利用できないか検討を行いたい．

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 (23700021) による補助のもとで行われた．

参考文献

[1] Yang Tang, Patrick P. C. Lee, john C. S. Lui, and Radia Perlman, ”FADE: Secure Overlay Cloud Storage with File Assured Deletion”, SecureComm 2010, Sep 2010

[2] Amazon Web Services. `http://aws.amazon.com/jp/`

[3] Windows Azure. `http://www.microsoft.com/windowsazure/`

[4] Amazon Google App Engine. `http://code.google.com/appengine/`

[5] W. Wang, Z. Li, R. Owens, and B. Bhargava, ”Secure and Efficient Access to Outsourced Data”, Cloud Computing Security Workshop, Nov 2009

[6] A. Sahai, and B. Waters, ”Fuzzy Identity-Based Encryption”, EUROCRYPT, May 2005

[7] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters, ”Secure Attribute-Based Systems”, Computer and Communications Security, Oct-Nov 2006

[8] R. Perlman, C. Kaufman, and R. Perlner, ”Privacy-Preserving DRM”, Identity and Trust Ion the Internet, Apr 2010

[9] Ming Li, Shucheng Yu, Kui Ren, and Wenjing Lou, ”Securing Personal Health Records in Cloud Computing: Patient-centric and Fine-grained Data Access Control in Multi-owner Settings”, SecureComm 2010, Sep 2010

[10] R.Perlman, ”File System Design with Assured Delete”, Network and Distributed System Security Symposium, Feb-Mar 2007

[11] R. Geambasu, T. Kohno, A. Levy, and H. M. Levy, ”Vanish: Increasing Data Privacy with Self-Destructing Data, USENIX Security Symposium, Aug 2009