SCIS 2014 The 31st Symposium on Cryptography and Information Security
Kagoshima, Japan, Jan. 21 - 24, 2014
The Institute of Electronics,
Information and Communication Engineers

PRINCESS:プロキシ再暗号化技術を活用したセキュアなストレージシステム
Proxy Re-encryption with INd-Cca security in Encrypted file Storage System

王 立華 * 早稲田 篤志 * 野島 良 * 盛合 志帆 *
Lihua Wang Atsushi Waseda Ryo Nojima Shiho Moriai

あらまし 現在の多くのクラウドストレージシステムでは、平文のままストレージサーバにファイルがアップロードされることが多く、サーバに侵入されると情報漏洩が起きる可能性があった。この問題を解決する方法の一つとして、ファイルを暗号化してからストレージサーバにアップロードすることが挙げられるが、ファイル交換が不便になるなどの課題があった。一つの解決策として、プロキシ再暗号化技術を活用すれば、ファイルを暗号化したまま指定した人とセキュアなファイル交換が可能になる。本稿では、王らにより提案された、委譲したプロキシ復号・再暗号化権限を無効化可能な ID ベースプロキシ暗号方式を用いて、より柔軟に管理できる、機密レベルに応じた暗号化ファイル交換可能なストレージシステムを開発したので紹介する。

キーワード クラウドセキュリティ、代理暗号、再暗号化技術、ペアリング、システム実装

1 まえがき

近年、クラウドコンピューティングに注目が集まり、その利便性の高さから急速に普及・展開した。その中でもとりわけ、Dropbox、Skydrive、Amazon Cloud Drive などのストレージサービスに人気が集まっている。ストレージサービスは利便性が高い一方で、ユーザが保管したファイルがサービス提供者に漏洩する可能性がある。本問題の解決策として暗号化するという選択肢もあるが、グループ間で同じファイルを共有するためには、事前に暗号化用の鍵を共有しておく必要があり、利便性が失われてしまう。

本問題の解決策として、プロキシ再暗号化 (PRE: Proxy Re-Encryption) 技術がある。PRE では、ファイルを暗号化したまま指定したユーザと安全にファイルを交換することが可能である。PRE という概念は、Blaze らによって初めて提案された [2]。PRE では、3 人のユーザ、Alice、Bob、サービス提供者 (Proxy、代理者) を考える。PRE を利用すると、下記のような仕組みを実現できる。

(1) Alice は、事前にサービス提供者に「再暗号化鍵」とよばれる鍵を渡しておく。

(2) ファイルを保管する際に、Alice は自らの鍵でファイルを暗号化し、サービス提供者が管理するサーバに保管する。

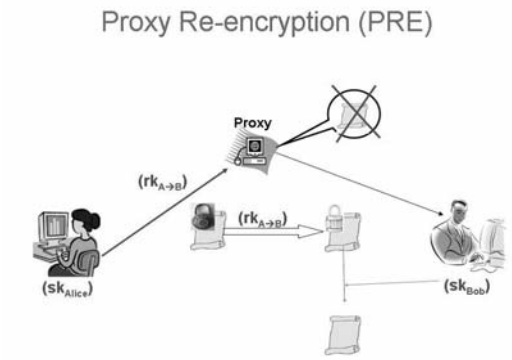

図 1: プロキシ再暗号の概略

(3) サービス提供者は、再暗号化鍵を使い Bob が自らの鍵で復号できるように暗号文を変換する。この際、サービス提供者にファイルが漏れることはない。

(4) Bob はサーバに接続し、暗号化されたファイルを取り出し、自らの鍵で復号する。

PRE とストレージサービスの相性の良さから、PRE を使ったストレージサービスに注目が集まっている。例えば、[1] では PRE のライブラリ、[9] では PRE を使ったストレージシステム「デジタル貸金庫」が開発された。

本稿では、王らにより提案された PRE [7] を使った暗

* (独) 情報通信研究機構, 東京都小金井市貫井北町 4 − 2 − 1, National Institute of Information and Communications Technology (NICT) 4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan (wlh@nict.go.jp)

号化ストレージシステム「PRINCESS」を提案する。[7] を利用することにより、機密レベルに応じたファイル交換が可能となる。

2 関連技術

本章では、PRINCESS に使った暗号技術を紹介し、それから既存のクラウド暗号化ファイル共有システムである Hayashi らの方式 [3] との相違点や優位性を紹介する。

2.1 ID ベース暗号

ID ベース暗号は 1984 年に Shamir により提案された [5]。ID ベース暗号とは公開鍵暗号の一種であり、その特徴はユーザの公開鍵として一意な識別子 (例えば利用者のメールアドレスなど) を用いる点である。そのため、従来の公開鍵暗号に比べて、利便性が高い。従来の公開鍵暗号システムと違い、ID ベース暗号システムでは、秘密鍵の生成はユーザ自身ではなく、信頼できる鍵生成センター（Private Key Generator: PKG）によって、ユーザの秘密鍵を発行してもらい、ユーザの ID がそのまま公開鍵になる。PKG は生成した秘密鍵を全て知ることができる立場である。そのため、ID ベース暗号は組織内での利用に最適な暗号システムである。

2.2 代理暗号とプロキシ再暗号化

1997 年、Mambo らは事前に依頼した代理人によって復号可能な暗号方式である代理暗号 (Proxy Cryptosystem) を初めて提案した [4]。この方式では依頼人は自分の秘密鍵を渡すのではなく、代理復号鍵を渡す。代理復号鍵を渡された代理人は依頼人の代わりに暗号文を復元することができる。一方、プロキシ再暗号化技術では依頼人 A は他のユーザ B への再暗号化鍵を代理人に渡し、代理人はその鍵を使って A 宛の暗号化されたデータを復号することなく B 宛の暗号化データに変更可能な暗号方式である。その後 B は自分の秘密鍵を使って変換されてきた暗号文を復号する。このように、代理暗号とプロキシ再暗号化は密接に関連している概念である。

密接に関連している二種類の暗号方式は独立であるが、王らが提案した IBPdr システムでは、代理復号鍵で復号できる暗号文は再暗号化が可能である。本方式はこの特徴を活用し、組織内メンバまでのファイル交換と、組織内メンバと外部連携者までを含めたファイル交換を可能にした。

2.3 Hybrid 暗号

暗号システムは大きく共通鍵暗号と公開鍵暗号に分けられる。共通鍵暗号の利点は効率が良いことであるが、問題点は暗号文を複数で共有しにくいこと（鍵共有問題）である。逆に公開鍵暗号の利点は他人と暗号文で情報を共有しやすいことであるが、問題点は効率が悪いことである。そのためよく行われている方法は公開鍵暗号で共通鍵暗号のセッション鍵を共有して、共有した鍵で暗号化するという方式（Hybrid 暗号方式）である。PRINCESS では ID ベースプロキシ再暗号化技術を使って鍵を共有する。サーバに Alice から Bob への再暗号化鍵 $rk_{A\to B}$ を保有すると、Alice と Bob が $file$ の暗号化データを交換したい場合は、Alice のフォルダに Alice の ID を使って計算した暗号文

$$E_K(file), E_{Alice}(K)$$

を保存する。クラウドサーバでは $rk_{A\to B}$ を使って、$E_{Alice}(K)$ を Bob の暗号文 $E_{Bob}(K)$ に変換し、Bob のフォルダに

$$E_K(file), E_{Bob}(K)$$

を保存する。Alice と Bob は各自の秘密鍵で K を復号して、$file$ を復号する。

2.4 Hayashi らの方式との相違点と優位性

最近、東芝ソリューション（株）と東芝による Hayashi ら [3] 提案した再暗号化技術を用いて、デジタル貸金庫という暗号化ファイル共有システムが実装された [9]。この方式は従来の公開鍵暗号インフラの元で構築されたシステムである。再暗号化する作業をクラウドストレージサーバが A から再暗号化鍵を受け取って行う。その際に、暗号化されたデータを復号することなく別のユーザの暗号文を作成する。データは常に暗号化されているため、万が一クラウドストレージサーバが侵入されデータが漏洩してもファイルの内容が流出することはない。

今回、我々は王らの ID ベース再暗号技術 [6, 7, 8] を用いて、PRINCESS システム (Proxy Re-encryption with INd-Cca security in Encrypted file Storage System) を開発した。

この PRINCESS システムもまた Hayashi らの方式と同じく再暗号化技術を利用し暗号化ファイルを暗号化されたまま交換するシステムである。一方で Hayashi らの方式との違いとして、PRINCESS は ID ベース暗号方式を使用している点が挙げられる。そのため、組織管理がしやすいシステムである。これは現行の組織内のシステム管理者が組織内のストレージ内容を把握することができるというシステムをそのままに、外部ストレージに移行が可能であることを意味している。このようなシステムをクラウドサービス上で行うことのニーズは非常に高いと考えられる。例えば、医療データを病院内のサーバに保存すると、震災などで病院や主治医が被災すると医療データが失われ、患者の救助を遅くなる。このような場合、本システムを使用していると外部ストレージに暗号化されて預けられている医療データを、システムの管

表 1: 計算コストの比較

処理主体	処理内容	Hayashi らの方式 [3]		王らの方式 [7]		
		レベル 1	レベル 2	高レベル	中レベル	低レベル
クライアント	暗号化	13E + 1P	6E + 1P	3E	4E	5E
クラウドサーバ	有効性チェック	1E + 14P	1E + 8P	6P	6P	6P
クラウドサーバ	再暗号化	11E	-	-	3P	3P
クライアント	復号	2E + 4P	1E + 1P	1E + 2P	1E + 2P	1E + 4P

理者が外部の医療関係者に復号鍵を発行して復号できるようにすることで、医療の早期再開が可能となる。

PRINCESS もう一つ有用な特徴は機密レベルに応じて暗号化ファイルを共有できることである。ファイルを共有する範囲により高中低の 3 つのレベルに分けられる。高レベルの場合は指定した人とだけファイルを交換する。中レベルの場合はプロジェクト毎で組織内グループメンバまでファイル共有する。低レベルの場合は組織外の連携者とファイル交換することが可能である。この秘密レベルは暗号化するときに選択する。

さらに、PRINCESS にはユーザは誰でもリーダとしてプロジェクトを立ち上げ、メンバの追加・削除することができ、さらに、プロジェクトを削除して終了させることもできるという柔軟性もある。ここで述べているプロジェクト削除とメンバの削除は単に削除するメンバをメンバリストから削除するという意味ではない。それに加えて、既にサーバに保存した削除メンバに関係する再暗号化鍵、代理復号鍵を無効化しなければならない。王らにより提案された方式は、委譲したプロキシ復号・再暗号化権限を無効化可能という性質を満たす [7]。したがって、プロジェクトとメンバだけではなく、クラウドストレージサービスも必要がなくなったときにやめる事ができる。Hayashi らの方式も鍵生成するときに有効期間を付けておくという方法で、期間が過ぎたら自動的に無効化することが可能であるが、これはむしろ受動的な無効化方法で、削除されるメンバとストレージサービス管理者が結託すれば、有効期間までの間、データを続けて読むことができる。PRINCESS では鍵の発行をするときに、時間パラメータを分離し、後日コントロールための common data に設置する、この方法により、能動的な無効化作業がリーダ（当然 PKG も）の管理によって可能になる。

Hayashi らの方式のデータの機密レベルは２つのレベルに分けられ、レベル１は再暗号化が可能なレベルであり、レベル２は再暗号化が不可能なレベルである。よって、レベル２の暗号文は PRINCESS における高レベルに設定された暗号文に相当する。同様にレベル１の暗号文は中レベルに設定した暗号文に相当する。

特に意味があるのは組織内メンバのみならず、組織外のメンバにも暗号化データを共有することができることである。具体的な分類方法は第 3 節を参照。

両者計算量コストの比較は論文のデータによって整理しておく（表 1）。E は指数演算、P はペアリング演算を示す。

3 PRINCESS システムの紹介

PRINCESS は王らが提案した IBPdr システムに基づいて共有した鍵を使って、AES で暗号化した Hybrid システムである。このシステムは、図 2 で示したように、6 つのエンティティと１２個のアルゴリズムにより構成される。

鍵生成センタはエンティティ「PKG」として活動し、Setup、Ext、PKD(PRKGen) の 3 つのアルゴリズムを実行する。組織に所属するユーザは以下の二つのエンティティとして活動する。「Decryptor」はプロジェクトのリーダとしてのエンティティであり、PRKGen、PDKGen、Common Data、Rev、Dec の 5 個のアルゴリズムを実行する。「Re-Decryptor」はプロジェクトのメンバとしてのエンティティであり、アルゴリズム RDec を実行する。組織に所属していないユーザはエンティティ「Proxy Decryptor」として活動し、アルゴリズム PDec を実行する。暗号化行う所属する組織を他わないエンティティ「Encryptor」として活動し、アルゴリズム Enc を実行する。クラウドストレージはエンティティ「Proxy Server」として活動し、アルゴリズム REnc を実行する。各アルゴリズムの詳細は以下のようになる。

Setup p は素数とする。PKG はランダムに $\alpha \in \mathbb{Z}_p$ を選び、IBPdr システムの公開パラメータとマスターキーを生成する。

$$params = (G_1, G_2, p, \hat{e}, g, g_1 = g^{\alpha}, g_2, \{w_i\}_0^{2l}), msk = g_2^{\alpha}.$$

ここで、G_1 と G_2 は位数 p を持つ乗法群とし、$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ となる双線形写像とする。g は G_1 の任意の生成元とし、g_2 と $\{w_i\}$ は G_1 の任意の元である。公開パラメータを組織とストレージサーバに送信する。マスターキーを PKG 内に保存する。

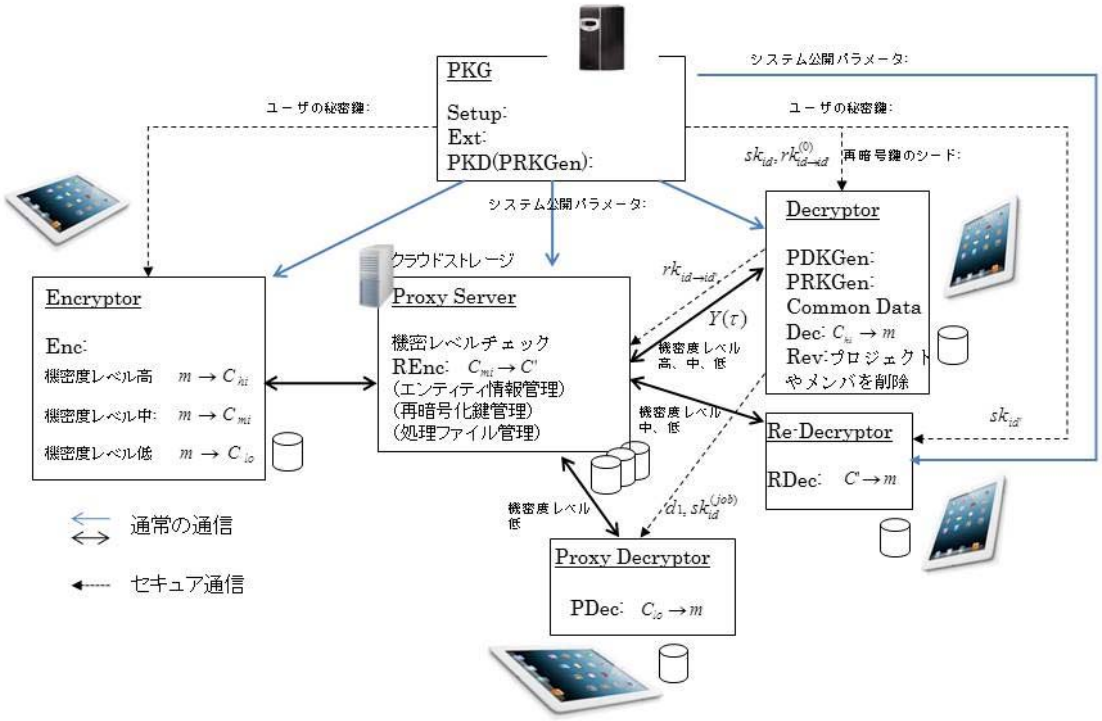

図 2: システムアルゴリズムの概略

Ext　PKG は乱数 $u \in \mathbb{Z}_p$ を生成し、PKG に保存する。PKG は u とユーザの識別子 (id) とマスターキーを使ってユーザの鍵 $sk_{id} = (d_0, d_1)$ を生成して、ユーザに安全な通信路を使って配布する。

$$(d_0,\ d_1) = (g_2^{\alpha} H(id_{PKG}||id)^u, g^u),$$

ここで、H は Waters のハッシュ関数とする。d_0 は秘密鍵として、ユーザで秘密に保存する。d_1 はユーザの秘密鍵、及び代理復号鍵の一部として利用する。

PKD(PRKGen)　PKG はユーザ A からの要望を受けて、ユーザ A と B の ID とマスターキーを使ってユーザ A の暗号文をユーザ B の暗号文に変換する再暗号化鍵作成のためのシードを作成して、ユーザ A に安全な通信路を使って配布する。

$$rk^{(0)}_{id \to id'} = \left(\frac{H(id_{PKG}||id_A)}{H(id_{PKG}||id_B)} \right)^{u_B}$$

PRKGen　Decryptor は設定するプロジェクト（プロジェクト名 job とする）の Revocation keystone $\gamma = \Gamma(id, 0) \in G_1$ と乱数 $d_\gamma \in \mathbb{Z}_p$ を生成し、受け取った再暗号化鍵のシードとこの γ に合わせて再暗号化鍵を生成する。クラウドストレージのプロキシサーバに安全な通信路を使って送信する。

$$\begin{aligned} rk_{id_A \to id_B} &= (rk^{(1)}_{id_A \to id_B}, rk^{(2)}_{id_A \to id_B}) \\ &= \left(\gamma \cdot H(id_{PKG}||id_A)^{d_\gamma} \cdot rk^{(0)}_{id_A \to id_B},\ g^{d_\gamma} \right) \end{aligned}$$

PDKGen　Decryptor は乱数 $d_{job} \in \mathbb{Z}_p$ を生成し、自分の秘密鍵を使って、γ に合わせて代理復号鍵を作成し、プロジェクトに所属していないユーザである代理復号人（Proxy Decryptor）に安全な通信路を使って送る。

$$sk_1^{(job)} = d_0 \cdot H(job \parallel id)^{d_{job}\gamma}, \quad sk_2^{(job)} = g^{d_{job}}, d_1.$$

Common Data　Decryptor は γ を使って時間 τ の関数である common data $Y(\cdot)$ を作成する。$Y(\tau) = (y_1(\tau),\ y_2(\tau))$

$$y_1(\tau) = \gamma^{-1} H(\tau \parallel id)^{d_\tau}, \quad y_2(\tau) = g^{d_\tau}.$$

ここで、d_τ は $\mathbb{Z}_p$ からとる乱数とする。

Enc　Encryptor は機密レベルを設定し、Decryptor の ID、現在時刻 τ とプロジェクト名 job を使ってファイル m を暗号化する。暗号文は $C = (C_1, C_2, C_3, C_4, C_5)$ の五つ組からなり、$C_1 = m \cdot \hat{e}(g_1, g_2)^r$、$C_2 = g^r$、$C_3 = H(id_{PKG}||id)^r$、$C_4 = H(\tau \parallel id)^r$、$C_5 = H(job \parallel id)^r$ とし、暗号化レベルに応じて以下のように生成される。

高: $C = (C_1, C_2, C_3, *_4, *_5)$

中: $C = (C_1, C_2, C_3, C_4, *_5)$

低: $C = (C_1, C_2, C_3, C_4, C_5)$

ここで r は $\mathbb{Z}_p$ 上の乱数、$*_4, *_5$ は G_1 上の任意の要素とする。

REnc　Proxy Server は $\langle id, C = (C_1, C_2, C_3, C_4, C_5), job, \tau \rangle$ を受け取ったら、受けた Decryptor 宛の暗号文の機密レベルをチェックし、Decryptor のフォルダに保存する。さらに中と低レベルの暗号文がある場合は、Decryptor から受け取った再暗号化鍵と common data を使って、以下を計算する：

$$C_1' = C_1 \cdot \frac{\hat{e}(C_4, y_2(\tau)) \cdot \hat{e}(C_3, rk^{(2)}_{id_A \to id_B})}{\hat{e}(C_2, y_1(\tau) \cdot rk^{(1)}_{id_A \to id_B})}$$

再暗号文として $\langle id, C = (C_1', C_2, C_3, C_4, C_5), job, \tau \rangle$ と設定し、中レベルの暗号文である場合は Re-Decryptor のフォルダに保存し、低レベルの暗号文である場合は Proxy Decryptor のフォルダに転送する。

Dec　Decryptor は $\langle id, C = (C_1, C_2, C_3, *, *), *, * \rangle$ を受け取ったら、自分の秘密鍵を使って、自分の ID で暗号化された高中低レベルの暗号文暗号文を復号する

$$m = C_1 \cdot \frac{\hat{e}(C_3, d_1)}{\hat{e}(d_0, C_2)}.$$

RDec　Re-Decryptor は再暗号化された $\langle id', C' = (C_1', C_2, C_3, *, *), *, * \rangle$ を受け取ったら、自分の秘密鍵を使って暗号文を復号する

$$m = C_1' \cdot \frac{\hat{e}(C_3, d_{B1})}{\hat{e}(d_{B0}, C_2)},$$

PDec　Proxy Decryptor は低レベルの暗号文 $\langle id, C = (C_1, C_2, C_3, C_4, C_5), job, \tau \rangle$ を受け取ったら、Decryptor から受け取った代理復号鍵を使って復号する。

$$m = C_1 \cdot \frac{\hat{e}(C_4, y_2(\tau)) \cdot \hat{e}(C_3, d_1)}{\hat{e}(C_2, y_1(\tau) \cdot sk_1^{(job)})} \cdot \hat{e}(C_5, sk_2^{(job)}).$$

Rev　Decryptor はプロキシサーバに送ったある再暗号化鍵を無効化するとき、或いは代理復号人の代理復号鍵を無効化するとき、Revocation keystone を更新して、上記のアルゴリズムを実行することで、無効化しない宛先の再暗号化鍵だけを更新する。

4　システムの試作と評価

本システムの実用性や性能などを評価するために試作および実装評価を行った。

Proxy-Server、Decryptor、Re-Decryptor、Encryptor、Proxy-Decryptor の各エンティティの一連の処理の所要時間の計測結果を表 2, 3 に示す。

表 2, 3 において、それぞれ Proxy-Server は機密度レベルのチェック、再暗号化、Decryptor は復号、Revocation keystone の計算、Proxy common data の計算、再暗号化用鍵の生成、プロキシ復号鍵の作成、プロキシ復号鍵配列作成の一連の処理、Re-Decryptor は再暗号化された暗号文の復号、Encryptor は暗号化、Proxy-Decryptor のプロキシ復号鍵の正当性チェック、プロキシ復号の一連の処理の所要時間 (単位：ミリ秒) を示している。

表 2: PC 上でのエンティティごとの計測時間（ミリ秒）

ハッシュ関数	Waters	IEEE P1363/D6
ライブラリ	IND-CPA/CCA	IND-CPA/CCA
Proxy-Server	26.86 / 26.76	38.32 / 38.61
Decryptor	21.12 / 21.41	30.60 / 31.30
Re-Decryptor	3.06 / 3.39	3.07 / 3.43
Encryptor	8.42 / 9.71	10.23 / 11.74
Proxy-Decryptor	6.13 / 6.42	6.12/6.51

表 3: Android 上でのエンティティごとの計測時間（ミリ秒）

ハッシュ関数	Waters	IEEE P1363/D6
ライブラリ	IND-CPA/CCA	IND-CPA/CCA
Proxy-Server	279.98/ 278.57	380.25 / 381.23
Decryptor	174.96 / 180.75	259.80 / 266.43
Re-Decryptor	34.66 / 40.26	34.31 / 39.86
Encryptor	70.41 / 84.55	87.18 / 102.09
Proxy-Decryptor	69.08 / 74.17	68.71/74.49

計測方法は、128 バイトの平文データに対して、一連の動作を 100 回実行し、各関数の 1 回あたりの計測時間の和を取っている。ここで平文データは毎回変化させ、鍵は固定している。

本システムではハッシュ関数として Waters [7] および IEEE P1363/D6 の 2 通りを実装しており、方式としても IND-CPA および IND-CCA の安全性を達成する方式 2 通りを実装している。

PC 環境としては、Intel Core i7 3770 (動作周波数 3.4GHz, 64bit Memory 32GB)、Android 端末環境としては、Nexus 7 上で計測した。

本システムは 2013 年 11 月に行われた NICT オープンハウスでデモ展示を行った。

5　今後の展望と課題

一つの応用例として医療情報・電子カルテ管理への適用を検討する。

5.1　非常時も対応可能な医療データ管理への応用

医療データを安全な形でクラウドストレージ上に管理することができれば、震災など非常事態が起こった際も継続して医療データにアクセスすることができるため、

事業の早期復旧・継続という BCP の観点から高い要望がある。

本システムは、下記の手順で、非常時も対応可能な医療データ管理へ応用することができる。

医療機関の情報管理室は鍵生成センターとして各医師の秘密鍵を生成し、安全な通信路を介して各医師の端末へ送付する。

ある患者が初めてその病院に来院した際、担当した医師が主治医としてその患者に関する「プロジェクト」を立ち上げる。診療後、主治医は患者の医療データをデータの共有先によって機密レベル「高」「中」「低」のいずれを適用するか検討し、それぞれの暗号化処理を行う。特に、緊急時の救急医療の際に有用と考えられるデータは外部医療機関の医師が解読可能なように「低」に該当する暗号化処理を行う

患者が後日、他の診療科に受診する場は、受付の際に、サーバにて該当医療データを他の診療科の医師が復号できるよう、再暗号化を行い、その医師が復号してデータを参照する。新しい医療データは、機密レベルに応じて主治医や外部医療機関の医師が解読可能なように暗号化する。

震災時等の緊急時は、主治医は自分の秘密鍵を渡すことなく、自分の秘密鍵を用いて外部医療機関の医師のための代理復号鍵を生成する。これにより患者の医療データを活用して適切な治療を行うことができる。

医療機関内にとどめておくべき医療データについては、本システムの無効化 (Revocation) の機能が医療機関からの医師の転出等の対応に有効である。主治医は他の医師への再暗号化鍵のシードを鍵生成センターからもらい、プロジェクトに対する再暗号化鍵を更新してクラウドストレージサーバに保持することで転出した医師が復号できないよう無効化処理ができる。

また、本システムは、医療データ管理への応用以外にも、汎用的なセキュアファイル共有サーバとして活用できる。

謝辞

本研究は JSPS 科研費 23500031 の助成を受けたものです。またシステム仕様の検討するにあたり多大な貢献を頂いた、情報通信研究機構セキュリティ基盤研究室黒川貴司技術員に感謝いたします。

参考文献

[1] G. Ateniese, K. Fu, M. Green, and S. Hohenberger: “Improved proxy re-encryption schemes with applications to secure distributed storage”, In Internet Society (ISOC): NDSS 2005, pages 29-43, (2005).

[2] M. Blaze, G. Bleumer, and M. Strauss: “Divertible protocols and atomic proxy cryptography”, In: EUROCRYPT 1998, LNCS, vol. 1403, pp. 127-144, Springer (1998)

[3] R. Hayashi, T. Matsushita, T. Yoshida, Y. Fujii, and K. Okada: “Unforgeability of Re-Encryption Keys against Collusion Attack in Proxy Re-Encryption”, IWSEC 2011, LNCS 7038, pp. 210-229 (2011)

[4] M. Mambo, and E. Okamoto: “Proxy cryptosystem: Delegation of the power to decrypt ciphertexts”, In: IEICE Trans. Fundamentals, Vol.E80-A, No.1, pp. 54-63, Springer (1997)

[5] A. Shamir: “Identity-based cryptosystems and signature schemes”, In: CRYPTO 1984, LNCS, vol. 196, pp. 47-53, Springer (1985)

[6] L. Wang: “An ID-based 2-functional proxy cryptosystem”, IEICE Technical Report ISEC2008-188, Vol.108, pp.563-568, (2009).

[7] L. Wang, L. Wang, M. Mambo, and E. Okamoto: “Identity-Based Proxy Cryptosystems with Revocability and Hierarchical Confidentialities”, IEICE, E95-A(1), pp. 70–88, (2012).

[8] 王立華: “二機能付き ID ベース暗号化方法及び暗号システム”, 特許第 5298394 号.

[9] 吉田琢也、松下達之: “クラウドサービス上でより安全なデータ共有を実現する再暗号化技術”, 東芝レビュー Vol.66 No.11(2011)