DBSJ Journal, Vol7,No3
December, 2008

平文を生成しない分散ストレージ上での再暗号化手法

A Re-encryption Method on a Distributed Storage without Generating Cleartexts

高山 一樹† 横田 治夫‡
Kazuki TAKAYAMA Haruo YOKOTA

情報セキュリティの重要性が増大する中で，並列ストレージにおける伝送中データを保護する手法として，データを暗号化して格納するencrypt-on-disk方式がある．この方式はデータ転送性能に優れる半面，暗号鍵の更新が必要な場合の処理コストが大きい．encrypt-on-disk方式の高機能ストレージにおいて，再暗号化処理をディスク上で行うことでクライアント側の負担を軽減することができるが，通常の再暗号化方式では一時的に平文が生成される為，情報漏洩の可能性がある．我々は分散ストレージの構造と特定の暗号化方式の性質を利用した，平文を生成しない再暗号化手法を提案する．OFB モードと排他的論理和の性質に着目し，暗号化と復号を逆順に処理することで，平文の生成を防ぐ．本稿で性能面とセキュリティ面で評価し，有用性を示す．

The information security becomes more and more important recently. There is the encrypt-on-disk scheme, in which files are stored in cipher, and it is a scheme for protection of transmitted data in parallel storage systems. This scheme costs re-encryptions with the updates of cryptographic keys, while it has improved performance of data transmission. With a high functional storage adopted encrypt-on-disk scheme, we can reduce the costs in clients by processing re-encryption on storage nodes. However, there are possibilities of leaks because cleartext is generated along normal re-encryption processes. We propose a re-encryption method without generating cleartexts by applying structure of distributed storage and particular encryption mode. We focus on characteristics of OFB mode and XOR. The cleartext generations are prevented by processing encryption and decryption in reverse. In this paper, we evaluate the method with performance and security, and indicate the usefulness of proposed method.

1. はじめに

近年，情報セキュリティの重要性が増大する中，ネットワークストレージにおいても様々なデータ保護要件の考慮が必須となっている[1], [2]．保護要件の一つ，伝送中データの機密性保持の為の手法として，encrypt-on-disk 方式がある．これはデータを暗号化した状態でストレージに格納する方式で，転送時のみ暗号を用いる encrypt-on-wire 方式と比較して転送性能が良く，格納データの機密性も実現できるという利点がある反面，ユーザのアクセス権失効（revocation）に伴い，暗号化データを新しい鍵で再暗号化しなければならない．再暗号化の処理コストは大きく，処理の実行場所及び方法は重要な考慮事項である．

一方で，近年高機能分散ストレージが注目されている（例えば自律ディスク[3]）．高機能分散ストレージシステムではクライアント側の負担を軽減する為に，ストレージノードの演算処理能力を利用し，耐故障化，負荷均衡化，容量分散等の機能を自律的に実行するアプローチがとられる．同様にストレージノードの処理能力を利用して，encrypt-on-disk 方式を高機能ストレージ上で実現し，ストレージノード上で上記の再暗号化処理を実行することで，revocation 発生時のクライアント側の負担軽減が可能となる．

しかし，ノード上の再暗号化に通常の手法を用いた場合，処理の中間生成物として平文が出現してしまう為，ディスクノード上の機密性が提供できない．機密性保持の為にも，処理中に平文が出現しない再暗号化処理手法が必要である．

本稿では，分散ストレージにおける，単一ストレージノード上での機密性を実現できる再暗号化手法 RORE（Reverse Order Re-Encryption）を提案する．ブロック暗号の暗号化モードのうち，OFB（Output Feedback）等一部のモードではその処理の性質上，再暗号化における暗号化処理と復号処理が可逆である為，暗号化を先に処理することで平文を生成しない再暗号化処理を実現でき，ノード陥落によるデータ漏洩を防ぐことができる．提案手法を実装して実験を行い，性能，セキュリティの両面から考察を行う．

2. encrypt-on-disk 方式

2.1 概要及び転送性能

encrypt-on-disk 方式は，ネットワークストレージにおける，伝送路上のデータを悪意あるユーザの傍受から守る暗号利用方式の 1 つである．encrypt-on-disk 方式ではデータを暗号化した状態でストレージノードに格納する．その為，ストレージ側でのデータ送受信時に暗号処理を必要としない．

他方式として encrypt-on-wire 方式が挙げられる．この方式ではデータを平文の状態で格納し，送受信時に暗号化及び復号処理を行う．

データ転送性能の面では，encrypt-on-disk 方式の方が優れている．これは encrypt-on-wire 方式ではストレージ側送受信時に必ず暗号処理が発生すること，セッション毎に新しい暗号鍵を生成する為，鍵生成コストがかかることが理由である．また encrypt-on-disk 方式では格納データの機密性も実現可能であるという利点もある．

2.2 encrypt-on-disk におけるアクセス権失効

複数のユーザでデータを共有するシステムの場合，encrypt-on-disk 方式ではユーザのアクセス権失効（revocation）に伴い，対象データを新しい暗号鍵で再暗号化する必要がある．アクセス権を失効されたユーザ（revoked user）は現在使われている暗号鍵を保持している可能性があり，アクセス制御リスト（Access Control List: ACL）等のアクセス管理手法で revoked user のアクセス要求を拒否しても，傍受等不当なアプローチで revoked user にデータが渡ると，情報が漏洩してしまうからである．

† 学生会員 東京工業大学大学院情報理工学研究科計算工学専攻修士課程 takayama@de.cs.titech.ac.jp
‡ 正会員 東京工業大学学術国際情報センター yokota@cs.titech.ac.jp

この為 revocation 処理に関しては encrypt-on-disk 方式は encrypt-on-wire 方式より処理コストが高い．しかし，revocation コストを考慮した上でも，総合して encrypt-on-disk システムは encrypt-on-wire システムより性能面，セキュリティ面共に優れていると検証されている[1].

3. 関連研究

encrypt-on-disk 方式を採用したセキュアストレージシステムとして，SNAD[4]，Plutus[5]，SiRiUS[6]等がある．これらのシステムはサーバが不正を行わない事を信用しない（trust でない）という前提で，データが平文として存在し得るのはクライアントマシン上のみであるとしている．また revocation 処理は所有者主導で実行される．我々の研究ではクライアントの負担を軽減するために，ストレージ側で再暗号化処理を行い，かつ後述の通り処理とデータ配置の考慮によりノード上のセキュリティ実現を目指すため，方針が異なる．

我々は分散ストレージにおける，性能とセキュリティを両立した revocation 時再暗号化手法 BA-Rev（Backup Assist Revocation）[7]を提案した．予めバックアップデータを新しい暗号鍵で暗号化しておき，revocation 発生時にプライマリとすることで，アクセス回復までの時間を再暗号化処理の分短縮することができる．次に元のプライマリをバックグラウンドで再暗号化し，新しいバックアップとして設定する．なお，この構造ではバックアップデータがプライマリと異なる鍵で暗号化されている関係で，通常と比較してディスク故障時の復旧に時間が掛かる為，単一ディスク故障に対しては通常より脆弱であると言える．これに対し，復旧用バックアップと revocation 用バックアップを分けて保存しておくことで，より堅牢なシステムを構築できる．

BA-Rev の再暗号化処理をストレージノード上で処理する点は本稿の方針と同じであり，また本稿の提案手法はこの BA-Rev への適用を考慮するものとする．

4. 想定するシステム要件

以下本稿で前提とするシステムやデータ構造について概略を述べる．

4.1 高機能分散ストレージ

我々はストレージ装置上の演算処理能力を利用してデータの管理を自律的に行うシステムとして自律ディスク[3]を提案してきた．自律ディスクはネットワークに接続された高機能ディスクノードのクラスタにより構成される．この高機能ディスクの演算能力を利用し，ストレージ側で耐故障化，負荷均衡化，容量分散等の機能を自律的に実行し，ユーザによるストレージ管理の負担を軽減する．

また各ストレージノードは並列にネットワーク接続される．これらのストレージノードに対し，同様にネットワークに接続されたクライアントノードからアクセスするものとする．

4.2 暗号の利用

本稿では encrypt-on-disk 方式を採用した，自律ディスクのような高機能ストレージシステムや，ファイルサーバクラスタを対象とする．自律ディスクの方針に従い，revocation に伴う再暗号化処理はストレージシステム側で行い，クライアント側の負担を軽減する．

本節では本研究で利用した暗号の種類と暗号化モード，鍵管理方法について説明する．

4.2.1 暗号の種類と利用

暗号は，暗号化と復号に共通の鍵を用いる共通鍵暗号と，異なる対の鍵を用いる公開鍵暗号に分類できる．共通鍵暗号は，予め暗号化側と復号側で，安全な手段を用いて鍵を共有していなければならない．一方，公開鍵と秘密鍵の対を用いる公開鍵暗号では，対のうちの片方（公開鍵）を全ユーザに公開することができる為，鍵配布の問題がない．しかし，その反面，公開鍵暗号は一般的に共通鍵暗号の数百から数千倍の処理速度であるという問題がある．

これらの特性を考慮し，一般的には共通鍵の配布を，公開鍵暗号を用いて行い，その共通鍵を用いてデータのやり取りをする場合が多い．本研究においてもこの方式を採る．

4.2.2 鍵管理構造

データの暗号鍵を管理する構造としてロックボックスがある．ロックボックスとは，暗号鍵を格納し，アクセス権を持つユーザのみ鍵を取り出すことが出来る鍵管理構造である．SNAD[4]におけるロックボックスである *key object* の例を図 1 に示す．ここでユーザ x は公開鍵 K_x^+と秘密鍵 K_x^-を持ち，あるデータは共通鍵 K_i で暗号化されているものとする．*key object* の各行はユーザ ID，ユーザの公開鍵で暗号化された共通鍵 $K_x^+(K)$, ユーザに認可されているアクセス権の種類 P_x からなる．格納された共通鍵を獲得するには，公開鍵と対を成す，ユーザ自身のクライアントノード上に保存された秘密鍵を用いてのみ復号できる為，正しく認可されたユーザのみが共通鍵を獲得できる．

本研究では，この *key object* の構造を利用して鍵を管理する．1 ファイルに対して 1 つの *key object* が対応し，ファイルを暗号化した共通鍵を格納し，ストレージノードに格納される．また本研究ではストレージ側で再暗号化等の暗号処理を行う為，各ストレージノードもユーザと同様に公開鍵を持ち，処理対象のファイルの *key object* に獲得可能な状態の鍵を保持するものとする．

Key Object ID	User ID	Signature	Reference Count
User ID	Encrypted key	Permissions	
A	$K_A^+(K_i)$	P_A	
B	$K_B^+(K_i)$	P_B	

K_A^+ ：ユーザAの公開鍵

K_i ：ファイルfの共通鍵

図 1 ロックボックスの例：*key object*
Fig 1 Example of Lockbox：*key object*

4.2.3 暗号化と復号が可逆となる暗号化モード

本研究の前提である，暗号化処理と復号処理が可逆となるブロック暗号モードとして，OFB（Output FeedBack）モードがある．本研究ではこの暗号化モードを使用した．OFB での暗号化処理の流れを図 2 に示す．

OFB モードの特色として，暗号処理部分を平文とは独立に実行でき，実際の平文に対する処理は暗号処理によって生成された疑似乱数ビット列との排他的論理和のみである点，ストリーム暗号的に利用する点が挙げられる．また，暗号化と復号の処理が同じであるのも特徴である．

OFB モードの欠点として，平文と暗号文の排他的論理和

より，暗号化に用いた疑似乱数ビット列が得られてしまう点が挙げられる．これにより，ある平文と暗号文の組を奪取された場合，同じ暗号鍵と初期ベクトル（Initial Vector: IV）によって暗号化したデータを復号できてしまう．この攻撃を防ぐ為，対象データ毎に暗号鍵あるいは初期ベクトルを変更する必要がある．

また OFB モードは初期ベクトルの暗号化に始まり，先頭ブロックから順次処理しなければならない為，ランダムアクセスに弱いという欠点がある．これに対し，CTR (counter) モードというランダムアクセスに強いモードがある．CTR では暗号処理の入力として，前ブロックの疑似乱数ビット列ではなく，ブロックの位置に対応した値を持つカウンタの値を用いる為，途中のブロックからの処理が可能となる．但し，OFB と比較すると疑似乱数ビット列の生成元が周期性を持つ為，若干安全性が損なわれる．本稿の実験環境では OFB を用いたが，代わりに CTR を用いることも可能である．

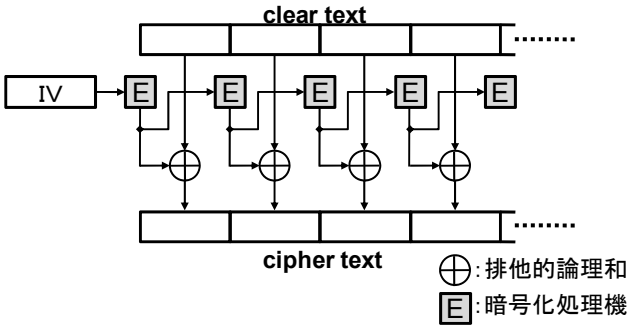

図 2 OFB モード
Fig 2 OFB mode

4.3 想定する攻撃モデル

攻撃者は，対象となる分散ストレージシステムと同じネットワーク上に存在し，以下の攻撃を実行できるものとする．

- 伝送路へ攻撃し，転送中のデータを傍受する．
- あるストレージノードに攻撃し，陥落させて内部のデータを奪取する．

なお，攻撃者は格納されたファイルへのアクセス権を持たず，暗号鍵等いかなるデータも所持していないものとする．

5. RORE の提案

通常の再暗号化手法では，暗号化データを元の暗号鍵で復号し，次に新しい暗号鍵で暗号化する為，中間生成物として平文が出現する．その為 encrypt-on-disk 方式でも，ノード上で再暗号化を実行すると，ノード上の機密性が保持できないという問題があった

この問題に対し，処理中に平文が生成されない再暗号化手法 RORE（Reverse Order Re-Encryption）を提案する．

4.2.3 で説明した OFB モードでは，暗号化は平文と疑似乱数ビット列の排他的論理和を求めることで実行される．また復号は暗号文に対して同様の処理を行うことで実行される．ここで，排他的論理和は交換則が成り立つ為，再暗号化処理のように復号と暗号化を連続して実行する場合，暗号化と復号は可逆である．この性質を利用し，通常“復号”,“暗号化”の順に実行しなければならない再暗号化処理を，“暗号化”,“復号”の順に処理することで，平文を生成しない再暗号化を実現でき，再暗号化処理中の攻撃者による当該ノード陥落によるデータ漏洩を防ぐことが可能となる．

なお，定常状態を含めて，1 ノード陥落による漏洩を防ぐ為には，鍵管理の考慮が必要である．再暗号化をストレージ側で行う為に，共通鍵をストレージノードが利用可能な形で保存する必要があるが，この無防備な状態の鍵を暗号化ファイルと同じノードに置くと，そのノードが陥落した時点でデータが漏洩してしまう危険性がある．対策としては，後述の実験での方法のように鍵をデータとは異なるノードのロックボックスで管理する方法がある．また，ネットワーク上に安全性を信用できる暗号鍵サーバを設置してそこに格納する方法や，暗号処理専用のハードウェアを各ノードに搭載してその内部で鍵を管理する方法がある．後者の例としては，HDD に暗号化回路を組み込んだ Seagate の DriveTrust[8] や富士通の MTZ2 CJ 等がある．但しこれらは revocation 時の再暗号化処理に対応していない為，そのままの適用はできない．

6. 実験

提案手法の適用による，性能面への影響の評価の為，PC クラスタ上で動作する encrypt-on-disk 方式のファイルサーバ・クライアントプログラムを作成し，実験を行った．

6.1 実験プログラム

実験プログラムは以下の 2 環境について，ファイル獲得（get）の応答時間と，ストレージ上での再暗号化処理の実行時間を測定できる．

i. **normal**：通常の再暗号化を行う環境（図 3）
復号，暗号化の順に再暗号化処理を行う．ロックボックスは対象の暗号化ファイルと同じストレージノード上に格納する．

ii. **RORE**：再暗号化に RORE を適用した環境（図 4）
5.で述べたとおり，ロックボックスは図 4 のようにファイルを格納したノードの隣接ノードに置くものとする．

暗号の利用方法や鍵管理方法は 4.の記述に従う．ここでは，提案の環境では，暗号鍵を管理するロックボックスを対応する暗号化ファイルとは異なるノードに置くことで，1 ノード陥落によるデータの漏洩を防ぐものとする．

図 3，4 に各環境での再暗号化処理の概略を示す．提案手法である RORE における処理の流れは以下の通りである．初期状態として，ファイル F はユニークな共通鍵 K_1 により，OFB モードで暗号化されてディスクノード A に格納され，その K_1 を格納するロックボックスは隣接ノード B に格納されている．ロックボックスには再暗号化処理用に，ロックボックス格納ノード N の公開鍵 K_N^+で暗号化された K_1 を保存している．なお，このプログラムにより，この手順で処理を行うことで実際に平文を生成せず再暗号化処理を完了できることを確認した．

step 1. 暗号化ファイルを格納したノードで新しい共通鍵 K_2 を生成し，暗号化ファイルを暗号化する．この段階でファイルは二重に暗号化された状態 $K_1K_2(F)$となる．

step 2. ノード B のロックボックスに K_2 を転送し，代わりに K_1 を受信する．この時鍵交換は送信先ノードの公開鍵で暗号化した状態で行う為，送信前後及びロックボックスの再構成時に鍵の復号，暗号化の必要がある．

step 3. 古い鍵 K_1 でファイルの復号を行い，その後 K_1 を破棄する．

プログラムで実行できる処理のうち，get はロックボックスに格納された対象ユーザ用の共通鍵の転送も含む．その為，ロックボックスの格納場所が異なる 2 環境では，再暗号化処理だけでなく get の処理も異なる為，以下評価の対象とする．

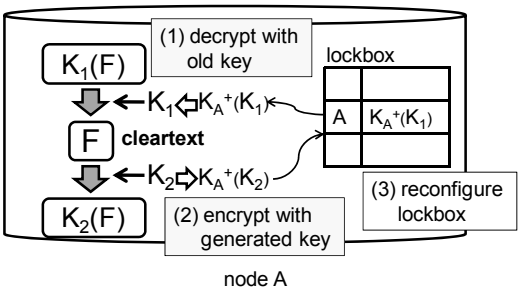

図 3 normal: 通常の再暗号化を行う環境
Fig 3 Environment with Normal Re-encryption Method

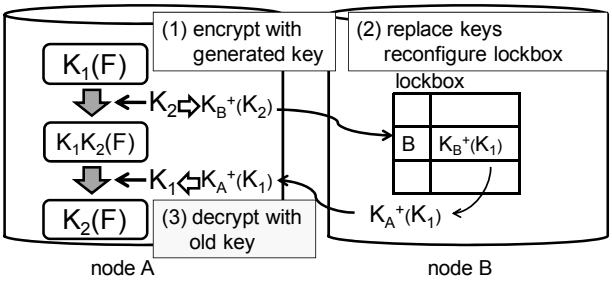

図 4 RORE: 再暗号化に RORE を適用した環境
Fig 4 Environment with RORE

表 1 ストレージノード諸元
Table 1 Spec of Storage nodes

CPU	AMD Athlon XP-M1800+ (1.53GHz)
Memory	PC2100 DDR SDRAM 1GB
HDD	TOSHIBA MK3019GAX (30GB, 5400rpm, 2.5inch)
Network	TCP/IP + 1000BASE-T
OS	Linux 2.4.20
Java VM	Sun J2SE SDK 1.5.0_03 Server VM

表 2 実験に用いた固定パラメタ
Table 2 Fixed Parameter used for Experiments

公開鍵	RSA 1024bit
共通鍵	AES 128bit
暗号化モード	OFB
パディング	NoPadding
ストレージノード数	3
ノード当たりデータサイズ	1MB × 500
Zipf 母数 θ	0.7

6.2 実験環境

実験は表 1 に示すノードで構成した，PC クラスタ上で行った．また，実験におけるパラメタとして表 2 のものを用いた．

表の記述の通り，実験では 3 台のストレージノードにそれぞれ 1MB，500 個のファイルを暗号化し，格納する．ただし総データサイズにロックボックスのサイズは含まない．

また，get の対象となるファイルは，選択したストレージノードの全ファイルの中から，パラメタ θ によって決まる Zipf 分布[9]に従い偏って選ばれるものとした．

6.3 実験 1:get の応答時間の比較

ファイルを格納した 3 ノードに対し，それぞれ 1 つのクライアントノードから get リクエストを 1000 回実行し，その応答時間を測定した．図 5 は各環境における，3 ノードの平均応答時間，及び 95%信頼区間を表している．

実験の結果，RORE を適用した環境では get の応答時間が平均で約 4.5 パーセント増加した．これは，提案の環境では，ファイル格納ノードが隣接した鍵格納ノードから，アクセスしたユーザに対応する共通鍵を取得しなければならない為であると考える．ただし，共通鍵は短い固定長である為，その転送の影響は非常に小さいことがわかる．

6.4 実験 2:再暗号化の処理時間の比較

ファイルを格納した 3 ノード上で，それぞれが再暗号化処理を 1000 回実行し，その実行開始から終了までに掛かる時間を測定した．ここで再暗号化対象のファイルはランダムに選択するものとした．図 6 はそれぞれの環境における処理時間の平均と，95%信頼区間を表している．

図より，RORE の再暗号化処理時間は，通常に比べて約 100 ミリ秒，約 47 パーセント増加した．これはノード間通信が必要であることに加え，共通鍵転送時に公開鍵暗号による暗号処理が必要である為だと考える．ここでは，ノード A からノード B への新しい共通鍵の送受信時，新しい共通鍵のロックボックスへの設定時，B から A への古い共通鍵の送受信時に公開鍵暗号での暗号処理が必要となる．6.3 の結果より，共通鍵の転送処理自体の影響は非常に小さいことより，公開鍵暗号による処理が性能に与える影響が非常に大きいことがわかる．

6.5 実験 3:再暗号化処理が他アクセスに与える影響の比較

再暗号化処理が他アクセスに対してどの程度の影響を与えるかを実験により測定した．ノード A，B にファイルを格納した状態で，ノード A で再暗号化処理を連続で実行する．この時ノード B は，提案手法の環境においてノード A のファイルに対応するロックボックスが格納されるノードである．すなわち，normal ではノード A のみで再暗号化処理が実行され，RORE では図 4 の通りノード A で再暗号化処理，ノード B で対応するロックボックスの処理が実行される状態である．この状態でノード A，B に対して get を 1000 回実行し，応答時間を測定した．図 7 は各環境，各ノードに対する get の平均応答時間，95%応答時間を表す．

結果は，再暗号化の無い平常状態（図の normal：ノード B に相当）と比較して，get の応答時間は通常の環境で約 29 パーセント，RORE でノード A，B 共に約 14 パーセント増加した．通常では再暗号化処理が 1 ノード内で行われる分，そのノードへの他アクセスに大きく影響を与えるのに対し，処理が 2 ノードに分散される RORE では，他アクセスへの影響も分散されることがわかる．

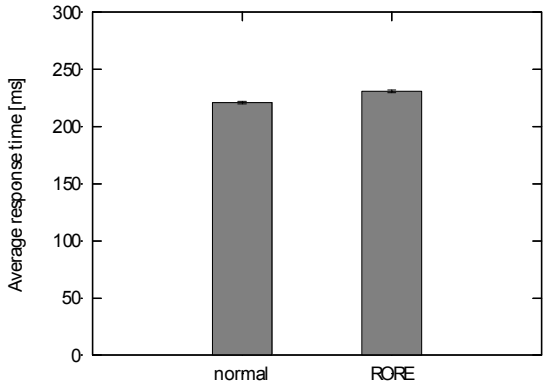

図 5 get コマンドにおける平均応答時間の比較
Fig 5 Comparison of Average Response Times about GET Command

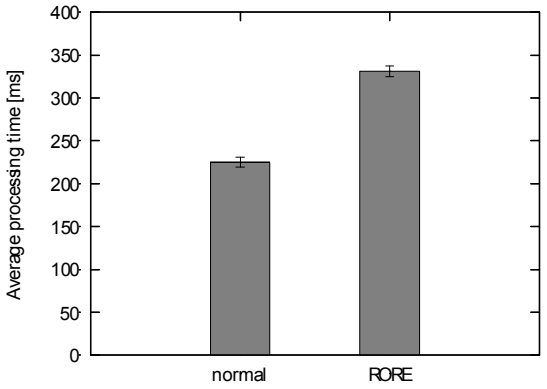

図 6 再暗号化の平均処理時間の比較
Fig 6 Comparison of Average Processing Times about Re-encryption

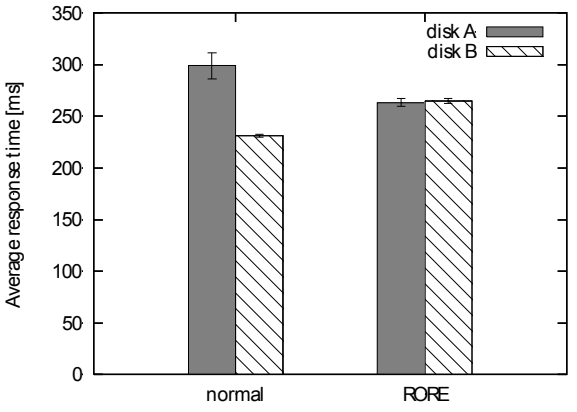

図 7 バックグラウンドの再暗号化処理が get に与える影響の評価
Fig 7 Evaluation of the Impact of Re-encryption Process on GET Command

6.6 性能に関する考察

6.3 の結果より，ファイル本体とロックボックスの格納ノード分離による get の性能低下は非常に小さいことがわかる．この性能低下は無視できる程度であると考える．

6.4 より，RORE では再暗号化の処理時間は増加する．その為，再暗号化の発生箇所とタイミングによっては，システムの性能低下を引き起こす可能性がある．例えば，アクセス対象のファイルが再暗号化中の場合，処理が終わるまで他のアクセスがブロックされ，応答時間が増大してしまう可能性がある．

しかし一方で，本研究では 3.で述べた revocation 時再暗号化手法である BA-Rev[7]への適用を考えており，再暗号化処理時間増大の影響は小さいと考える．BA-Rev では revocation 発生後に使われるデータを予めバックアップに保持しており，そちらにアクセスを回送し，実際の再暗号化処理はバックグラウンドで実行される．その為再暗号化処理中のファイルが直接アクセスされることは無いからである．

バックグラウンドでの再暗号化処理による性能への影響は，6.5 の結果より，RORE では複数ノードに影響が出るが，その反面 1 ノード当たりの性能劣化度合は小さい，QoS（Quality of Service）の観点から見ると，突出して性能劣化するノードがない RORE の環境は優れていると言える．なお，実際に BA-Rev に適用した場合の性能評価は今後の課題とする．

また，本実験ではロックボックスによって鍵を管理したが，これにより発生する公開鍵での鍵転送が性能へ大きく影響を与えている．その為，ノード内に鍵管理用ハードウェアを搭載できる場合，公開鍵暗号の使用が減るため，性能は改善できると考える．

7. セキュリティに関する考察

7.1 実験の環境に関する考察

6. の環境では，RORE を適用することにより，5. で述べた通り，再暗号化処理中に平文が生成されず，また 1 ノード中に平文が生成可能となるデータ（暗号化ファイルとその共通鍵）が同時に存在する期間が無い．その為，1 ノード陥落時のデータ漏洩を防ぐことができる．また，あるファイルに関する，ファイル格納ノードと対応する鍵格納ノードのうち，いずれかが陥落しない限り，データ漏洩を防ぐことができる．一方で，あるファイルの格納ノードと対応する鍵格納ノードが陥落した場合，或いは鍵格納ノードが陥落し，かつ通信中の対象ファイルが傍受されている場合，そのデータは漏洩する．しかし，再暗号化処理で平文が生成される通常の方式のように，1 ノード陥落のみでデータが漏洩する危険性のあるシステムと比較すると，提案方式では分散ストレージシステムの構造上，攻撃者が複数ステップを踏まなければデータが漏洩することがない為，提案方式の環境の方がセキュリティ面で優れていると言える．

本稿の実験では鍵管理をファイルと異なるノードのロックボックスで管理する方式を採ったが，ストレージノード外に trust である鍵管理サーバを設置できる場合，ノード内に鍵管理に特化したハードウェアを導入できる場合は，実験環境と比較して堅牢性が増し，より RORE の有効性も大きいと考える．

7.2 既存のセキュアストレージとの比較

3. で挙げた既存のセキュアストレージシステムのように，システムの安全性を信用できないと仮定した環境と比較すると，本稿の提案はストレージ側で再暗号化処理等を実行する為に生の暗号鍵を扱っている分，セキュアでないと言える．しかしその反面，再暗号化処理をユーザの手を煩わせずスト

レージ側で行うことで，クライアント側の負担を大きく軽減することが可能となっている．その為，RORE を適用した高機能ストレージシステムは，多人数でデータを共有する大容量分散ファイルサーバ等への適用が適している．このようなシステムではデータ管理コストが大きく，ストレージ主導で管理を行い，ユーザ側の負担を軽減することが望ましい．RORE 及び BA-Rev を適用することで, revocation 時のクライアント側の負担及びシステムの性能劣化を防ぎ，かつ通常の手法で再暗号化処理を行うよりも高い機密性を維持することができる．

7.3 脆弱性

本実験の環境では処理中の各時点における，1 ノード中のデータから平文が得られることはない．しかし，攻撃者によりノード中の処理がトレースされる等の攻撃で，再暗号化前のデータ $K_1(F)$，再暗号化後のデータ $K_2(F)$，処理中のデータ $K_1K_2(F)$が奪取された場合，それらの排他的論理和をとることで平文 F が得られてしまう．その為，攻撃を受けた時点でそれを検出する方法等，上記 3 データが同時に得られない様な方法や，単純な排他的論理和演算で F が求められないようにする工夫が必要である．

8. まとめと今後の課題

分散ストレージにおける，単一ストレージノード上での機密性を実現できる再暗号化手法 RORE を提案した．再暗号化処理中の暗号化処理と復号処理を可逆とできる OFB 等暗号化と復号が可逆な暗号化モードを利用することで，中間生成物として平文を生成しない再暗号化処理を実現でき，再暗号化処理時のノード陥落によるデータの漏洩を防ぐことができる．それに加えて，暗号化ファイルの保存と暗号鍵管理を異なるノードで管理する，或いは堅牢な鍵管理サーバや鍵管理に特化したハードウェアの導入で鍵を奪取できないようにする等，1 ノード内のデータから平文を生成可能な期間が存在しないようにｓｙ理を考慮することで，上記の 1 ノード陥落に対する機密性が実現可能となる．

実験により，RORE の性能面の評価を行った．ファイルとその暗号鍵を管理するロックボックスを異なるノードに格納した環境に，提案手法の RORE を導入した場合のファイル獲得（get）の応答時間を測定したところ，通常の再暗号化手法の環境との差は小さかった．一方，再暗号化の処理時間は通常より 47 パーセント増加したものの，バックグラウンドでの再暗号化処理が，そのノードに対する他のアクセス（本稿における実験では get）に与える影響として，get の応答時間の変化を見たところ，通常の方式では 1 ノードで 29 パーセント増加するのに対し，提案方式では 2 ノードにそれぞれで 14 パーセント増加となって分散された．その為，我々が提案した BA-Rev[7]のように，アクセスされないバックアップデータに対してのみ再暗号化を実行する手法には，性能を低下させることなく適用できると考える．また，通常の再暗号化処理のように，突出して才能が劣化するノードの出現を防ぐことができ，QoS の観点から考えると，提案方式は優れていると言える．

今後の課題としては，本稿で提案した RORE を，revocation 等，ユーザによって実際に要求されるコマンドの処理に組み込み，その性能面，及びセキュリティ面双方について評価することが挙げられる．その中で，我々が提案した revocation 手法である BA-Rev にも組み込み，その評価を行うことが必要である．

[謝辞]

本研究の一部は，独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST，および文部科学省科学研究費補助金特定領域研究（19024028）の助成により行われた．

[文献]

[1] E. Riedel, M. Kallahalla, and R. Swaminathan, “A framework for evaluating storage system security”, FAST ’02: Proceedings of the 1st USENIX Conference on File and Storage Technologies, pp.15-30, USENIX Association, 2002.

[2] P. Stanton, “Securing Data in Storage: A Review of Current Research”, ArXiv Computer Science e-prints, 2004.

[3] H. Yokota, “Autonomous Disks for Advanced Database Applications”, Proc. Of International Symposium on Database Applications in Non-Traditional Environments (DANTE’99), pp.435-442, Nov. 1999.

[4] E. Miller, D. Long, W. Freeman, and B. Reed, “Strong Security for Network-Attached Storage”, FAST ’02: Proceedings of the 1st USENIX Conference on File and Storage Technologies, p.1-13, Berkeley, CA, USA, 2002, USENIX Association.

[5] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, “Plutus: Scalable Secure File Sharing on Untrusted Storage”, FAST ’03: Proceedings of the 1st USENIX Conference on File and Storage Technologies, pp.29-42, USENIX Association, 2003.

[6] E.J. Goh, H. Shacham, N. Modadugu, and D. Boneh, “SiRiUS: Securing Remote Untrusted Storage”, the proceedings of the Internet Society (ISOC) Network and Distributed Systems Security (NDSS) Symposium 2003.

[7] 高山一樹，小林大，横田治夫，“複製を利用したストレージ中での暗号化データの権限失効処理”，第 18 回データ工学ワークショップ（DEWS2007）予稿集，電子情報通信学会データ工学専門委員会，2 月/3 月　2007.

[8] Seagate, “Drivetrust™ technology: A technical overview”, http://www.seagate.com/docs/pdf/whitepaper/TP564_DriveTrust_Oct06.pdf.

[9] D.E. Knuth, Sorting and Searching, Addison-Wesley Publishing Company, 1973

高山 一樹 Kazuki TAKAYAMA

東京工業大学大学院情報理工学研究科計算工学専攻修士課程在学中．2007 同大工学部情報工学科卒業．日本データベース学会学生会員．

横田 治夫 Haruo YOKOTA

東京工業大学学術国際情報センター教授．1980 東京工業大学工学部電子物理工学科卒業．1982 同大大学院理工学研究科情報工学専攻修士課程修了．同年富士通（株）．同年 6 月（財）新世代コンピュータ開発機構研究所．1986 （株）富士通研究所．1992 北陸先端科学技術大学院大学情報科学研究科助教授．1998 東京工業大学大学院情報理工学研究科計算工学専攻助教授．2001 より現職．工学博士．日本データベース学会理事．電子情報通信学会，情報処理学会，人工知能学会，IEEE，ACM 各会員．