# 論文 / 著書情報
# Article / Book Information

| | |
|---|---|
| 論題(和文) | 暗号化データの複製を利用した権限失効処理手法の評価 |
| Title(English) | Evaluation of a Revocation Method Utilizing Encrypted Backup Data |
| 著者(和文) | 高山一樹, 横田治夫 |
| Authors(English) | Kazuki TAKAYAMA, Haruo YOKOTA |
| 掲載誌(和文) | DEIM2009論文集 |
| Citation(English) | Proceedings of DEIM2009 |
| Vol, no, pages | ， ， |
| 発行日 / Pub. date | 2009, 3 |

# 暗号化データの複製を利用した権限失効処理手法の評価

高山　一樹† 　横田　治夫††,†

† 東京工業大学大学院情報理工学研究科計算工学専攻　〒 152–8552 東京都目黒区大岡山 2–12–1
†† 東京工業大学学術国際情報センター　〒 152–8550 東京都目黒区大岡山 2–12–1
E-mail: †takayama@de.cs.titech.ac.jp, ††,†yokota@cs.titech.ac.jp

**あらまし**　暗号化データ格納ストレージでは，一部利用者のアクセス権失効時に再暗号化が必要となり，アクセス不能の期間が発生する．我々は複製を持つ暗号化データ格納ストレージにおける再暗号化手法として BA-Rev を提案している．これは複製データを予め異なる暗号鍵で暗号化し，失効処理時に主データと切り替えることでアクセス不能期間を短縮し，処理コストも低減する．既存方式の active revocation と同等のセキュリティを維持しながら読み出し性能が向上できる反面，更新性能が悪化するという問題点がある．本稿では，BA-Rev の更新性能を改善するため，更新内容の複製側への適用を遅延させる DW/DRW 戦略を BA-Rev に適用した場合に，書き込み遅延と失効処理の関係を調べる．実験により，提案方式と active revocation を比較し，更新がある環境でのアクセス性能と DW/DRW 戦略が失効処理に与える影響を評価する．

**キーワード**　revocation，encrypt-on-disk 方式，高信頼ストレージ，セキュアストレージ

# Evaluation of a Revocation Method Utilizing Encrypted Backup Data

Kazuki TAKAYAMA† and Haruo YOKOTA††,†

† Department of Computer Science, Graduate School of Information Science and Engineering,
Tokyo Institute of Technology　2–12–1 Ookayama, Meguro-ku, Tokyo 152–8552, Japan
†† Global Scientific Information and Computing Center, Tokyo Institute of Technology
2–12–1 Ookayama, Meguro-ku, Tokyo 152–8550, Japan
E-mail: †takayama@de.cs.titech.ac.jp, ††,†yokota@cs.titech.ac.jp

**Abstract**　In storage systems storing encrypted data, there are terms in which users cannot access data because of revocation processes involved in data re-encryption. We proposed BA-Rev, the re-encryption method with storage system having backup data. It makes backup data encrypted with the key different from that for primary. When a revocation occurs, BA-Rev shortens the inaccessible term by setting the backup data as the primary and cuts total processing cost. While it improves readout performance and is as secure as the existing method, active revocation, it degrades update performance. In this paper, we examine the impact of the DW/DRWstrategy on BA-Rev, which delay the applying differential data in backup, and compare the proposed method with active revocation on environment including update.

**Key words**　revocation, encrypt-on-disk, high dependable storage, secure storage

## 1. はじめに

近年，情報セキュリティの重要性が増大し，分散ストレージでもデータ保護要件の考慮が必須となってきている [1], [2]．保護要件の一つである，分散ストレージにおけるノード間伝送中データの機密性保護手法として，encrypt-on-disk 方式がある．これはデータを暗号化された状態でストレージに格納する方式で，転送時のみ暗号を利用する encrypt-on-wire 方式と比較して転送性能が高く，ノード内のデータの機密性も実現可能である．しかしその反面，利用者のアクセス権失効（revocation）に伴い，新しい暗号鍵で対象データを再暗号化する必要がある．この revocation 時再暗号化処理の既存方式である active revocation と lazy revocation には，再暗号化を直ちに実行することによる性能低下と，再暗号化を遅延することによる脆弱性という，相反する欠点が存在する．

我々は，active revocation と同等の安全性を保ちつつ，効率の良い再暗号化処理を実現する BA-Rev（Backup Assist Revocation）を提案し，更新の無い環境での性能評価を行った [3].

BA-Rev はプライマリ・バックアップ構造を前提とし，バックアップデータを予め新しい暗号鍵で暗号化しておき，revocation 時にプライマリデータと置き換えることで，迅速かつ低コストな revocation を実現する．

本稿では，BA-Rev の環境に DW（Delayed Writing）戦略 [4] 及び DRW（Delayed Re-encrypting and Writing）戦略を適用し，更新性能と revocation 処理時の性能に与える影響を評価する．DW 戦略は更新処理時にバックアップ側の差分データの書き込みを，DRW 戦略は再暗号化及び書き込みを遅延することで，負荷分散と複数回の更新に対する適用処理を集約し，性能向上を図る．これらの提案方式と，既存方式である active revocation の性能を比較し，既存方式に対する優位性を示す．

以下に本稿の構成を述べる．2. で暗号化データを格納する encrypt-on-disk 方式の特徴を説明する．3. で関連研究について述べる．4. で本研究が前提とするシステムの概要を述べる．5. で BA-Rev の概要と DW/DRW 戦略を説明する．6. で実験とその結果に対する考察を行い，7. で本稿のまとめと今後の課題について述べる．

## 2. 暗号化データ格納ストレージ

分散ストレージにおける暗号利用方式の一つである encrypt-on-disk 方式の特徴，利点及び問題点について説明する．

### 2.1 encrypt-on-disk 方式と encrypt-on-wire 方式

分散ストレージでは各ノード間で転送されるデータの保護が必須である．伝送路上のデータの機密性保護の為の暗号利用方式として，encrypt-on-wire 方式と encrypt-on-disk 方式がある．encrypt-on-wire 方式は，セッション毎に新しく暗号鍵を生成し，それを用いてデータを暗号化し，転送する方式である．一方，encrypt-on-disk 方式は，予めデータを暗号化した状態でストレージノードに格納しておき，転送時はその暗号化データをそのまま転送する方式である．

二方式をデータ転送時におけるパフォーマンスに関して比較すると，encrypt-on-disk 方式ではストレージ側でのデータ送受信時に暗号化及び復号処理を行う必要がない為，encrypt-on-wire 方式よりも効率が良い．また encrypt-on-wire 方式ではセッション毎に暗号鍵生成コストが掛かり，性能面で問題がある．

またセキュリティ面で比較すると，伝送路上の機密性は二方式で同等だが，encrypt-on-disk 方式ではストレージノード上の機密性も実現が可能である為有用である [1].

### 2.2 アクセス権失効処理（revocation）

複数ユーザでデータを共有する，encrypt-on-disk 方式を採るシステムでは，あるユーザのあるデータに対するアクセス権の失効（revocation）に伴い，対象データを再暗号化する必要がある．これは，アクセス権を失ったユーザ（revoked user) が対象データに用いられている暗号鍵を保持している可能性があり，アクセス制御により revoked user のアクセス要求を拒否していても，傍受等の不正アクセスによりデータが revoked user に渡ると，情報が漏洩する危険性がある為である．

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 処理時の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ると検証されている [1].

### 2.3 revocation 時再暗号化手法

encrypt-on-disk 方式での revocation 処理における再暗号化は，処理を行うタイミングによって active revocation と lazy revocation に分類できる．

#### 2.3.1 active revocation

active revocation は，revocation 発生後直ちに，新しい暗号鍵を生成し，対象データの再暗号化処理を実行する方式である．revoked user は新しい暗号鍵を持たず，revocation 直後から対象データを復号できなくなる為，後述の lazy revocation と比較してセキュリティ面で優れる．しかし一方で，直ちに再暗号化処理を実行しなければならない，再暗号化処理が終了するまで対象データにアクセスできない等の原因より，性能を低下させる可能性がある．これは revocation の対象が複数データに及ぶ場合顕著である．

#### 2.3.2 lazy revocation

lazy revocation は，対象データの再暗号化処理を次回の更新時まで遅延する方式である．Cepheus [5] で提案され，Plutus [6] 等で採用されている．暗号化データの更新処理は暗号化処理を伴う為，revocation の為の暗号化処理を兼ねる事でコストを削減できる．また，更新頻度が低いデータでは，revocation の度に再暗号化を行う active revocation と比べ，複数回の revocation に対する再暗号化処理をまとめることができるので，性能差は大きくなる．

この方式では，revocation 発生後の未更新データは，発生前と同じ，revoked user が保持している恐れのある暗号鍵で暗号化された状態である．これは未更新データの情報は revoked user が知っている可能性がある為漏洩しても問題ない，という考えに基づく．しかし revoked user が revocation 発生前に対象データを取得していない可能性もある為，active revocation と比較するとセキュリティ面で劣ると言える．

#### 2.3.3 両手法の比較

パフォーマンス面では lazy revocation が優れているが，セキュリティ面を考慮すると active revocation を採用すべきであると考える．その為我々は，active revocation の安全性を維持しつつ，性能を向上させることを目指す．

## 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 [7]，Plutus [6]，SiRiUS [8] がある．Plutus は lazy revocation，SiRiUS は active revocation を採用し，SNAD は両方式における性能とセキュリティのトレードオフの問題から revocation の処理については今後の課題としている．これらのシステムは，サーバが不正を行わないということを信用できない（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ユーザのクライアントマシン上のみであるとしている．また，revocation の処理はデータの所有者主導で，所有者のクライアントで実行される．我々の研究ではクライアント側の負

担を軽減する為に，revocation 時の再暗号化処理をストレージ側で実行する為，これらのシステムと方針が異なる．

暗号処理専用回路を HDD（Hard Disk Drive）に組み込むことで全格納データを暗号化する技術として，Seagate の DriveTrust [9] や富士通の MTZ2 CJ がある．これらは暗号化データを格納する点で encrypt-on-disk システムと同等であるが，主に HDD 紛失時におけるデータの保護が目的である為，本稿で対象とする伝送路上のデータの機密性実現とは異なる．

## 4. 想定するシステム要件

本研究で前提とするシステムやデータ構造の概略を述べる．

### 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 [10] 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処理能力を利用し，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．

本研究では自律ディスクの様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に encrypt-on-disk 方式を採用することを考える．クライアント側の負担の軽減の為，revocation に伴う再暗号化処理や，暗号鍵の配布等を極力ストレージ側で行う．

### 4.2 プライマリ・バックアップ構造

耐故障化実現の為，並列ストレージシステムの各ストレージノードは，主にユーザアクセスを受けるプライマリデータと，他のノードのプライマリデータの複製であるバックアップデータを持つ．

### 4.3 ネットワーク構成

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を行うものとする．

### 4.4 暗号の利用

本研究で利用した暗号法と鍵管理方法について説明する．

#### 4.4.1 暗号法の種類と利用法

暗号は，暗号化と復号に共通の暗号鍵を用いる共通鍵暗号法と，異なる対の暗号鍵を用いる公開鍵暗号法に分類できる．共通鍵暗号は，予め暗号化側と復号側で，安全な方法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）を公開することができる為，共通鍵暗号のような鍵配布の問題はない．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，一般的には公開鍵暗号を用いて共通鍵の配布を行い，その共通鍵を用いてデータのやりとりを行う場合が多い．本研究でもその方式に従う．

#### 4.4.2 暗号鍵管理構造

暗号鍵を管理する構造としてロックボックスがある．ロックボックスは，暗号鍵を格納し，その鍵が関連するデータに対して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．図 1 に SNAD [7] におけるロックボックスである *key object* の例を示す．ここでユーザ $x$ は公開鍵 $K_x^+$ と秘密鍵 $K_x^-$ を持ち，あるデータが共通鍵 $K_i$ 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 $K_x^+(K_i)$，ユーザに認可されているアクセス権の種別 $P_x$ からなる．格納する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 $K_x^-$ 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共通鍵を獲得できる．

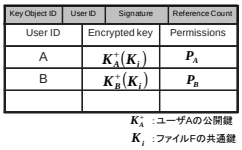


図 1 *key object*

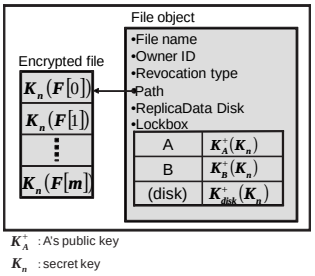


図 2 データ構造

本研究では，この *key object* の構造を利用して共通鍵を管理する．1 ファイルに対し 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都合上，各ストレージノードもユーザと同様に公開鍵と秘密鍵を持ち，処理対象のファイルの *key object* に鍵を保持する．

### 4.5 データ構造

本稿における 1 ファイル相当のデータ構造を図 2 に示す．この構造は SNAD [7] におけるデータ構造である *file object* を参考にした．ここで 4.4 で述べた通り各ユーザ及びストレージノードは公開鍵と秘密鍵の対を持ち，各ファイルは共通鍵で暗号化されている．この時ファイルの暗号化粒度は全体或いは固定長ブロック単位とする．後者の場合ファイル更新時の差分データの単位をこのブロックとすることで転送及び暗号化コストを削減できる．

ファイルへのアクセスはファイルオブジェクトを通して実行するものとする．ファイルオブジェクトにはファイル名や位置，処理方式等の情報，及び鍵を格納したロックボックスを持つ．

## 5. 複製データを利用した再暗号化

2. 3. 1 で述べた active revocation 適用時に性能が低下する問題に対し，我々は 4. で述べたシステムを前提として構成した分散ストレージにおける，バックアップデータを利用した，効率の良い再暗号化手法 BA-Rev（Backup Assist Revocation）[3] を提案した．本節で BA-Rev について説明する．

### 5.1 BA-Rev の概要

encrypt-on-disk 方式を採用した，プライマリ・バックアップ構造を持つ分散ストレージシステムを前提とする．なお，以降プライマリとバックアップの配置制約は無いものと仮定するが，制約が存在する場合も以降の処理は可能である（RBA-Rev [3]）．BA-Rev 及び既存方式の active revocation における再暗号化処理

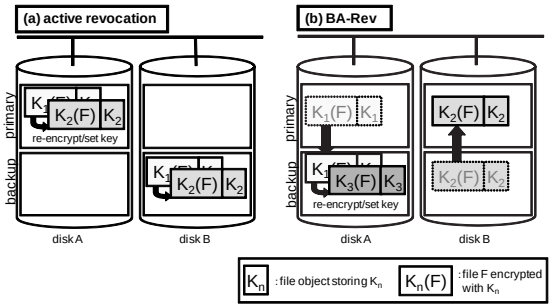


図 3　再暗号化方式の比較

の流れを図 3 に示す.

BA-Rev を採用するシステムでは，バックアップデータを作成する時，プライマリデータと同じ共通鍵で暗号化された状態ではなく，新たに生成した共通鍵で予め再暗号化し，格納する．ここでバックアップデータ及びバックアップデータを暗号化した共通鍵はユーザにアクセスされず，未知であるものとする．

revocation 発生時には，対象データのバックアップデータを，バックアップデータが格納されたストレージノードのプライマリに移動して新しいプライマリデータとして設定し，元のプライマリデータに代わってアクセスを受けるものとする．続いて，元のプライマリデータを新しくバックアップデータとして設定する為，新たに暗号鍵を生成し，再暗号化してバックアップとして格納する．

既存方式である active revocation ではプライマリデータの再暗号化中はそのファイルへのアクセスが不可能になるのに対し，BA-Rev では直ちにアクセス受付可能となる．また再暗号化処理を行うのは元のプライマリデータが格納されていた 1 ノードのみである為，全体での総処理コストも抑えられる．

しかし一方で，BA-Rev 環境では，active revocation 環境よりデータの更新コストが高く，性能劣化の可能性があった．これは，プライマリとバックアップの共通鍵が異なる環境では，更新がある場合，バックアップ側で差分データの再暗号化処理が必要であることが理由である．ユーザはバックアップ側で用いられる共通鍵を知らない為，差分データはプライマリデータに用いられている共通鍵で暗号化され，ストレージ側に転送される．その為，このままバックアップデータに適用はできない．

### 5.2　バックアップデータへの差分データ適用の遅延

上記の問題に対するアプローチとして，バックアップデータへの差分書き込みを遅延する DW（Delayed Writing）戦略 [4], 及び差分再暗号化と書き込みを遅延する DRW（Delayed Re-encrypting and Writing）戦略を適用する．DW 戦略ではバックアップデータ用差分データを再暗号化後，DRW 戦略では転送されてきた状態のまま，メモリ上に保持し適用を遅延することで，処理を分散して性能劣化を抑える．同時に，更新頻度が高い場合，バックアップ側では複数回の更新に関する処理をまとめ適用回数を削減できる．

本稿では DW 戦略でメモリ上に保持された差分データの書き込みタイミングの評価，及び DW/DRW 戦略が BA-Rev の処理に与える影響を評価する．適用が遅延された差分データに関して，何れかのタイミングでメモリ上の差分データをファイルへ適用しなければならないが，過去の研究 [4] ではこの点を考慮しておらず，また BA-Rev の処理への影響も未評価であった．

#### 5.2.1　差分データ適用時期の設定

メモリ上に保持した，バックアップデータへの差分データを適用する時期として，以下の 3 パターンを考える．ここで遅延された差分適用とは，DW 戦略では書き込みを，DRW 戦略では再暗号化と書き込みを指す.

1. revocation 発生時
   BA-Rev では，revocation 処理時バックアップデータをプライマリデータに昇格させる．従って，対象のバックアップデータへの差分が未適用である場合，プライマリとしてアクセスを受け付ける為に revocation 発生に伴って適用する必要がある．

2. 差分データ保存数が一定値を超えた時
   ストレージノードのメモリは有限である為，適当な閾値を設け，未適用の差分データの総数或いは総サイズが閾値を超えた場合に差分データを適用する．

3. 特定の条件を満たした時
   上記以外で，ある条件を満たした場合に差分データを適用する．本稿では以下の二項目のうちどちらかを選択し，その条件を満たした時に適用を行う．
   - 3-a　更新後一定時間経過後 ― 差分データ再暗号化しをメモリに保存後，一定時間待機し，その後書き込みを行うことで負荷を分散する．
   - 3-b　負荷が一定以下 ― 定期的に各ストレージノードの負荷を測定し，負荷の値が一定以下の時のみ差分データの書き込みを行うことで，負荷の集中を防ぐ．

図 4 に，(1) プライマリとバックアップで共通鍵が同じである環境（ここでは active revocation 環境），(2) 共通鍵が異なる環境（通常の BA-Rev 環境），(3) 共通鍵が異なり，かつ DW 戦略を適用した環境，(4) 同様に DRW 戦略を適用した環境における，更新発生時のクライアントマシン上，プライマリ及びバックアップでの処理の流れを示す．ここで (3) 及び (4) の環境では，上記何れかの条件を満たした場合にメモリに保持した差分データの適用を行うことを示している．また，いずれの方式においても，ユーザからデータを受信してから，バックアップ側からの応答を受信するまでの間，排他制御によりこのプライマリデータへの他のアクセスはブロックされるものとする．

## 6. 実　　験

更新がある環境における BA-Rev の性能，及び DW/DRW 戦略が BA-Rev に与える影響を評価する為，実験を行った．

### 6.1　実験で比較する環境

以下の環境について実装，実験し，比較を行う．

i. active revocation 環境
   revocation 発生時は既存方式である active revocation を実

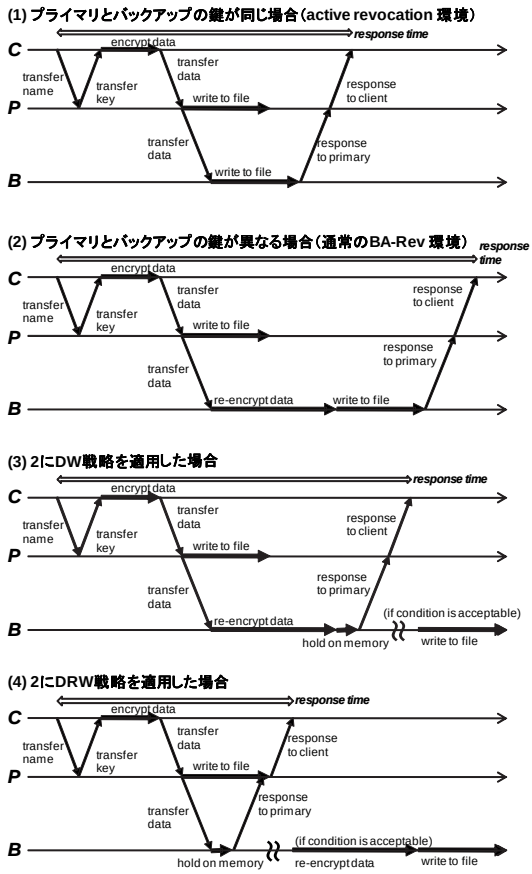


図 4　クライアント（*C*），プライマリ（*P*）とバックアップ（*B*）における更新処理の流れ

行する．あるファイルのプライマリデータとバックアップデータは同じ共通鍵で暗号化される．

ii. BA-Rev 環境
revocation 発生時は BA-Rev 方式で処理を実行する．バックアップデータはプライマリデータと異なる共通鍵で暗号化される．更新（update）発生時は図 4 の (2) の通り，バックアップ側でも即時書き込みを行うものとする．

iii. BA-Rev + $DW_{raw}$/$DRW_{raw}$
ii. に加え，更新処理は DW/DRW 戦略に従う．差分データの書き込み条件は 5.2.1 の (1)，(2) のみとする．一定間隔で未適用差分データ数を確認し，閾値を超えていたらその分を適用する．

iv. BA-Rev + $DW_{const:n}$/$DRW_{const:n}$
iii. に加え，差分データ適用条件として 5.2.1 の（3-a）を用いる．制御パラメタを $n$ とし，バックアップ側で差分データを再暗号化後，$n$ 秒待機して適用を行う．なお，待機中に同ファイルの更新が発生した場合，待機中の差分データを後続の差分データに置き換えるものとする．

v. BA-Rev + $DW_{load:n}$/$DRW_{load:n}$
iii. に加え，差分データ適用条件として 5.2.1 の（3-b）を用いる．制御パラメタを $n$ とし，一定間隔でストレージノード上のアクティブなスレッドの数を確認し，$n$ 未満の時未適用の差分データを適用する．この処理をスレッド数が $n$ 以上になるまで繰り返す．

### 6.2　実験環境とプログラム

表 1 のノードからなる PC クラスタ上で動作する，encrypt-on-disk 方式のファイルサーバ及びクライアントプログラムを作成した．実験に用いた各パラメタを表 2 に示す．

サーバプログラムは 6.1 の環境を構築する．データの初期配置は chained declustering [11] に従い，バックアップデータは，プライマリデータを格納するノードの論理的に“ 右隣 ”となるノードに配置するものとする．ただし処理中の配置制約は無く，BA-Rev の処理で配置が変わる可能性はある．

クライアントプログラムはサーバプログラムに対し，ファイル獲得（get），ファイル更新（update），revocation，及びファイルの格納とユーザの認可を要求できる．また get と update は応答時間を計測できる．ここで応答時間はクライアント側で以下の処理に掛かる時間と定義する．なお update に関しては図 4 も参照されたい．get は update と異なり各環境で処理は同じである為，比較図は省略する．

（1） **update**

1. サーバに対象ファイル名を送信する．
2. サーバから対象ファイルのプライマリデータに使われている，クライアントの公開鍵で暗号化された共通鍵を受信する．
3. 暗号化された共通鍵を秘密鍵で復号する．
4. 差分データを作成し，共通鍵で暗号化してサーバに送信する．
5. サーバ側の終了通知を受信する（終了通知のタイミングは図 4 参照）．

（2） **get**

1. サーバに対象ファイル名を送信する．
2. サーバから暗号化ファイル（プライマリデータ）及びクライアントの公開鍵で暗号化された共通鍵を受信する．
3. 暗号化された共通鍵を秘密鍵で復号する．
4. 共通鍵で暗号化ファイルを復号する．
5. ファイルを記憶装置に書き込む．

get，update の実行間隔は，平均到着間隔 $1/\lambda$ の指数分布 $f(t) = \lambda e^{-\lambda t}$ に従い決定する．ここで固定の get:update 比に従いアクセスを実行する．また対象ファイルは，偏りを持たせる為に，ノードに格納されたファイルからパラメタ $\theta$ に従う Zipf 分布 [12] に基づき選択される．

共通鍵暗号アルゴリズムは既知平文攻撃等の攻撃に強いとされる AES を選択した．また，4.5 で挙げたように，更新処理を固定長ブロック単位で行う為に，暗号化モードとして共通鍵アルゴリズム固定のブロックサイズ毎に独立して暗号化を行う ECB を用い，このブロックを更新処理の単位として用いた．

表 1　ストレージノード諸元

| | |
|---|---|
| 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　固定パラメタ

| | |
|---|---|
| 公開鍵 | RSA 1024bit |
| 共通鍵 | AES 128bit |
| 暗号化モード | ECB |
| パディング | PKCS5Padding |
| ストレージノード数 | 3 |
| ノード当たりデータサイズ | 1MB×500 |
| 差分データサイズ | 100KB |
| Zipf 母数 $\theta$ | 0.7 |
| 未適用差分データ数/スレッド数確認間隔 | 10sec |

### 6.3　実験 1：通常時のアクセスの応答時間

各環境における通常時の応答性能を評価する為の実験を行った．

#### 6.3.1　実 験 方 法

3 台のストレージノードに，各ノードのプライマリデータ数が 500 になるよう 1MB のファイルを格納する．この状態で，1 ストレージノードに対し 1 つのクライアントノードから，一定の get:update 比（50:50）に従い，アクセスを行う．アクセスの平均到着間隔 $1/\lambda$ を 500 ミリ秒から 150 ミリ秒まで変化させて，平均応答時間の変化を観測した．

#### 6.3.2　考　　察

DW 戦略又は DRW 戦略を適用した環境での update の平均応答時間を図 5，6 に示す．比較の為，active revocation 及び BA-Rev 環境での平均応答時間は両図に記載した．また get の平均応答時間を同様に図 7，8 に示す．

active revocation 環境と BA-Rev 環境における update の平均応答時間を比較すると，BA-Rev 環境の応答時間が大きい．プライマリとバックアップに用いる共通鍵が同じである active revocation 環境に対し，共通鍵が異なる BA-Rev 環境ではバックアップ側で差分データの再暗号化処理が必要であり，この処理に時間が掛かる為である．

通常の BA-Rev と DW 戦略を適用した各環境で update の平均応答時間を比較すると，低負荷時ではわずかに DW 戦略を適用した環境の方が性能が良かったものの，大きな差は出なかった．これは，本実験の実装では，あるファイルで複数回更新が発生した場合，キャッシュの利用により記憶装置との同期が取られず，高速に処理されている為と考える．

一方 DRW 戦略を適用した環境における update の平均応答時間は，BA-Rev と比較して大きく改善した．これは時間が掛かっていた再暗号化処理を遅延したことによる効果である．また適用条件による差異は，$\mathrm{DRW}_{\mathrm{load:10}}$ では高負荷時にまとめて差分データを適用してしまう場合がある為僅かに性能が落ちるものの，他の環境では適用条件毎に一長一短があり，大きな差が現れない．

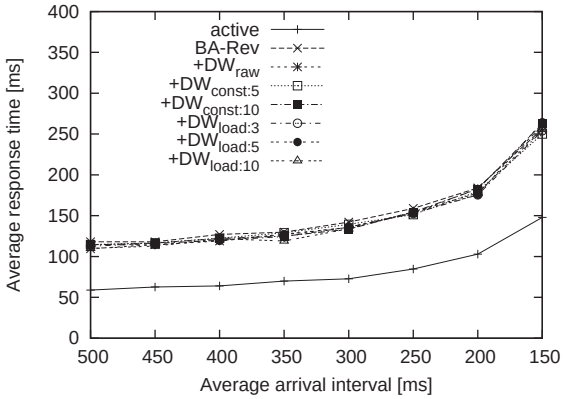


図 5　update の平均応答時間（DW 戦略適用）

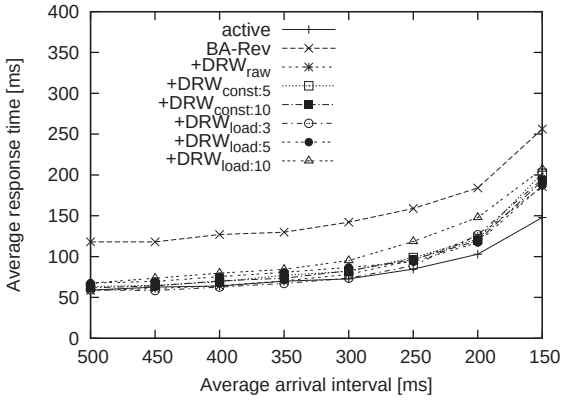


図 6　update の平均応答時間（DRW 戦略適用）

get の処理に関しては，何れの環境でも処理が同じである為，高負荷時にはバックアップでの update に関する処理の影響で僅かに差が現れるものの，全体として大きな差は見られない．

この実験により，BA-Rev 環境での更新の応答時間に，暗号処理の性能が大きく影響を与えていることがわかる．例えば，低負荷時の update 応答時間の内，80 パーセント強がクライアント上及びバックアップ側における暗号処理に掛かっていた．暗号処理時間は CPU 性能に依存する．その為，ストレージノードの性能を考慮することでも更新性能を改善できる．表 1 のストレージノードの環境と，高性能サーバ機（CPU: AMD Opteron 248（2.2GHz）× 2）で，100KB のデータの暗号化及び再暗号化処理の実行時間を比較したところ，後者は前者と比較して処理時間が約 40 パーセント改善した．

### 6.4　実験 2：revocation を集中して発生させた場合の比較

1 つのストレージノードで revocation が発生した場合における，各環境での性能比較の為実験を行った．

#### 6.4.1　実 験 方 法

6.3.1 と同様に 3 ストレージノードにファイルを格納し，3 クライアントからアクセスを行う．ここでは平均到着間隔は 400 ミリ秒，DW 戦略における制御パラメタ $n$ は 5 に固定する．また初期状態ではノード A，B，C のプライマリに対するバックアップはそれぞれ B，C，A にある．

アクセスを実行している状況で，異なるクライアントで 1 つのストレージノード（ここではノード B）中の，ランダムに

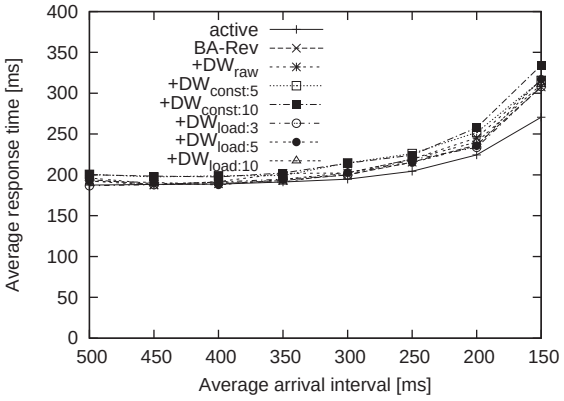


図 7　get の平均応答時間（DW 戦略適用）

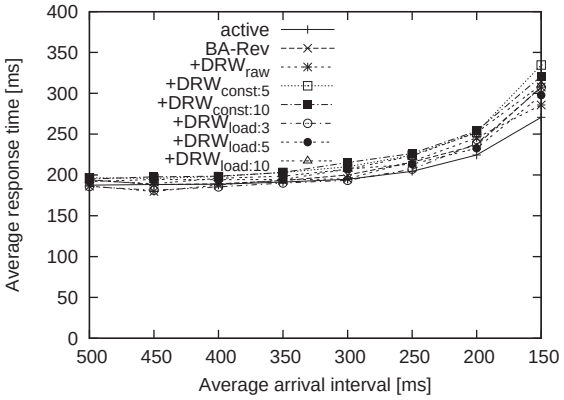


図 8　get の平均応答時間（DRW 戦略適用）

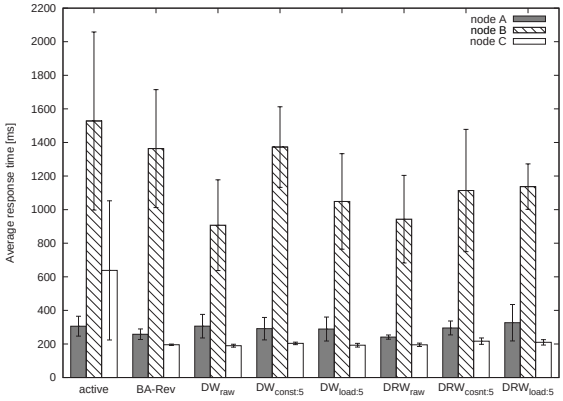


図 9　revocation が集中した場合の平均応答時間（get）

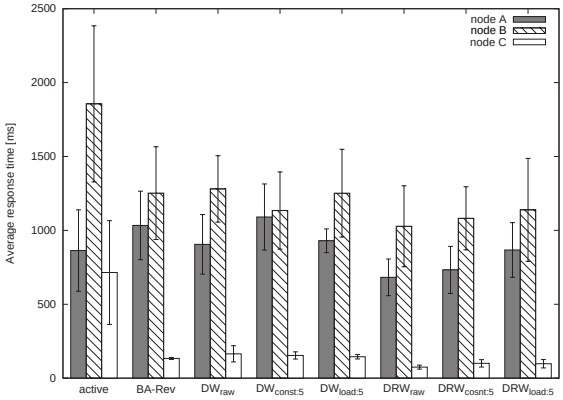


図 10　revocation が集中した場合の平均応答時間（update）

選択した 50 個のファイルに関して revocation を実行し，get と update の応答時間の変化を測定した．ただし権限を失効されるユーザはこの 4 クライアントと関わりのない仮想的なユーザであるとする．

#### 6.4.2　考　　察

上記の実験を 6 回行った．get 及び update それぞれについて，各ノード毎の，revocation 発生から収束までの期間を含んだ 100 アクセス分の応答時間の平均，及び各実験の平均の 95 パーセント信頼区間を図 9，10 に示す．

revocation 対象ファイルを持つノード B 及びそのバックアップデータがあるノード C に関して，active revocation と比較して提案方式の各環境で，get，update 共に性能が改善された．これは BA-Rev ではノード C で再暗号化処理が不必要である点，ノード B では再暗号化処理が必要であるものの，対象がアクセスされないバックアップデータである点が理由として挙げられる．

ノード A に関しては，active revocation と比較して提案の各環境の性能は同等，或いは若干劣化した．これは 6.4 で示した，通常時の性能差に起因するものと考える．ただしノード B，C と比較すると，手法間の差は小さい．

BA-Rev と DW/DRW 戦略を適用した環境を比較すると，get では特にノード B で，DW/DRW 戦略がより良い結果を示した．これはノード B では revocation 発生に伴い多量の再暗号化及びディスク I/O が発生するのに対し，両戦略では適用遅延により差分適用回数を削減でき，更なる性能劣化を引き起こしにくい為だと考える．一方 update に関しては，差分データ再暗号化まで遅延する DRW 戦略の環境が，BA-Rev 及び DW 戦略の環境より性能が良い．これは update の処理が再暗号化処理による影響を受けやすく，差分再暗号化遅延の影響が大きい為だと考える．

なお本稿では，DW/DRW 戦略における差分適用条件の差異による性能評価は割愛する．本実験及び次節の実験における平均及び信頼区間を考慮すると，条件間に明確な優劣を見出せなかった．その為，実験方法や回数，条件を改めて評価を行うことを今後の課題とする．

### 6.5　revocation が分散して発生させた場合の比較

複数のストレージノードに渡って同時に revocation が発生する場合を想定し，実験を行った．

#### 6.5.1　実 験 方 法

実験 2 と同様にデータを格納し，アクセスを行っている環境で，3 ストレージノード全てで同時に，それぞれランダムに選択した 15 ファイルについて revocation を発生させ，応答時間の変化を観測した．

#### 6.5.2　考　　察

この実験を 6 回実行した．get 及び update それぞれについて，revocation 発生から収束までの期間を含んだ 100 アクセス分の，全ノード応答時間の平均，及び各実験の平均応答時間の 95 パーセント信頼区間を図 11，12 に示す．

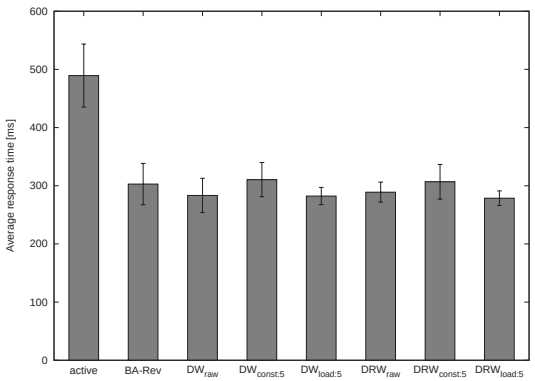


図 11　revocation が分散した場合の平均応答時間（get）

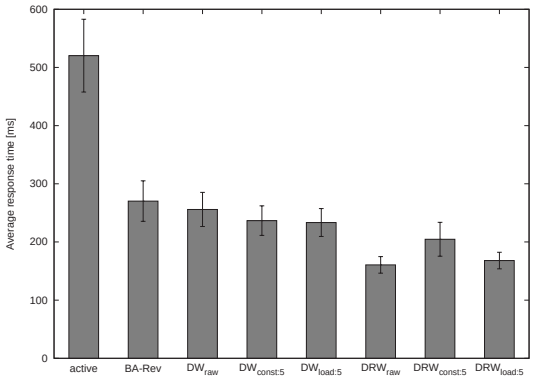


図 12　revocation が分散した場合の平均応答時間（update）

6.4 と同様，active revocation と比較して，BA-Rev を用いた各環境では get，update 共に性能が良い．これはバックアップ側で再暗号化が不必要である点，プライマリ側の再暗号化がバックグラウンドで処理される点の両方が理由であると考える．BA-Rev と DW/DRW 戦略適用環境の比較では，update に関して DRW 戦略が他環境より良い結果となった部分を除き，全体に大きな差が出ない．これは，この実験では revocation 要求を全ノードに分散させた分，1 ノード当たりの負荷が小さい為であると考える．また，この結果より，1 ノード当たりの revocation 対象ファイル数が少なければ，BA-Rev 及び何れの DW/DRW 戦略適用環境でも，条件によらず安定して active revocation より良い性能を実現できることを表していると言える．

## 7. まとめと今後の課題

本研究では，encrypt-on-disk 方式を採用する分散ストレージシステムにおける，高速かつ効率の良い revocation 時再暗号化処理方式 BA-Rev に，更新時にバックアップ側の差分データの書き込みを遅延する DW 戦略，及び差分データの再暗号化と書き込みを遅延する DRW 戦略を適用し，更新がある環境での評価を行った．BA-Rev ではバックアップデータを予め新しい共通鍵で暗号化しておくことで，revocation 対象ファイルへアクセスできない状況を即時回復でき，同時に再暗号化処理を行うノード数が減少する為，複数ノードでの性能低下を抑える．実験により，更新がある環境でもその効果を確認できた．また，特に DRW 戦略を適用し，再暗号化処理も含めてバックアップ側の差分データ適用を遅延し，複数の更新に対する処理を集約することで，active revocation の環境と同等の更新性能を維持しつつ，さらに BA-Rev の性能を向上できることを確認した．

なお，DW/DRW 戦略において，差分データを未適用のままメモリ上に保持することによる，信頼性の低下はほぼ無い事を確認している [13]．

今後の課題として，DW/DRW 戦略における差分データ適用条件による評価がある．今回の実験の実装及び方法では，適用条件による明確な優劣は現れなかった．その為，実験方法や適用条件を再考する必要がある．また自律ディスク等のストレージシステムに実装し，異なるサイズのファイルが存在する等，より実環境を想定した状況での評価が必要である．

## 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，及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] Erik Riedel, Mahesh Kallahalla, and Ram Swaminathan. A framework for evaluating storage system security. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, pp. 15–30. USENIX Association, 2002.

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

[3] 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.

[4] Kazuki Takayama, Dai Kobayashi, and Haruo Yokota. Consideration of Experimental Evaluation about Encrypted Replica Update Process. In *International Workshop on Advanced Storage System 2007 in conjunction of Proc. of the Second IEEE International Conf. of Digital Information Management*, pp. 545–550. IEEE, Oct 2007.

[5] Kevin Fu. Group sharing and random access in cryptographic storage file system. Master's thesis, MIT, 1999.

[6] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *FAST '03: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, pp. 29–42. USENIX Association, 2003.

[7] Ethan Miller, Darrell Long, William Freeman, and Benjamin Reed. Strong Security for Network-Attached Storage. In *FAST '02: Proceedings of the 1st USENIX Conference on File and Storage Technologies*, pp. 1–13, Berkeley, CA, USA, 2002. USENIX Association.

[8] Eu-Jin Goh, Hovav Shacham, Nagendra Modadugu, and Dan Boneh. SiRiUS: Securing Remote Untrusted Storage. In *the proceedings of the Internet Society (ISOC) Network and Distributed Systems Security (NDSS) Symposium 2003*, 2003.

[9] Seagate. Drivetrust$^{tm}$ technology: `http://www.seagate.com/docs/pdf/whitepaper/TP564_DriveTrust_Oct06.pdf`.

[10] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of Int'l Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 435–442, Nov. 1999.

[11] Hui-I Hsiao and David J. DeWitt. Chained declustering: A new availability strategy for multiprocessor database machines. In *Proceedings of the Sixth International Conference on Data Engineering*, pp. 456–465, Washington, DC, USA, 1990. IEEE Computer Society.

[12] Donald E. Knuth. *Sorting and Searching*. Addison-Wesley Publishing Company, 1973.

[13] 高山一樹. 暗号化データ格納分散ストレージにおける性能とセキュリティの両立に関する研究. 修士論文, 東京工業大学大学院, 1 月 2009.