DEWS2007 E7-7

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

高山　一樹[†]　　小林　　大[††,†††]　　横田　治夫[††††,††]

† 東京工業大学 工学部 情報工学科
†† 東京工業大学 大学院 情報理工学研究科 計算工学専攻
††† 日本学術振興会特別研究員 DC
†††† 東京工業大学 学術国際情報センター
E-mail: †takayama@de.cs.titech.ac.jp, ††, †††daik@de.cs.titech.ac.jp, ††††, ††yokota@cs.titech.ac.jp

あらまし　近年，情報セキュリティの重要性が大きくなり，ネットワークストレージでは悪意あるユーザによる盗聴から伝送路上のデータを保護することが必須となる．データの保護には主に暗号が用いられる．伝送時のみ暗号を用いるシステムに比べ，データを予め暗号化して格納するシステムは伝送時の効率がよいが，このようなシステムではユーザのアクセス権失効に伴いデータを再暗号化する必要がある．この再暗号化の方法は，直ちに再暗号化を行う active revocation とデータ更新時まで再暗号化を遅延する lazy revocation があるが，active revocation は性能面でのコストが高く，lazy revocation は古い鍵で暗号化された脆弱な状態が残る，という相反する問題がある．本稿では，データの複製を他のディスク装置に置く分散ストレージを前提とし，権限失効に伴うデータ再暗号化のための効率のよい方式を検討する．また実験によりその有用性を示す．

キーワード　プライバシー保護，並列 · 分散 DB，ストレージシステムバックアップ

Revocation for Encrypted Data Stored with Replica

Kazuki TAKAYAMA[†], Dai KOBAYASHI[††,†††], and Haruo YOKOTA[††††,††]

† Department of Computer Science, Faculty of Engineering, Tokyo Institute of Technology
†† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
††† Research Fellow (DC), Japan Society for the Promotion of Science
†††† Global Scientific Information & Computing Center, Tokyo Institute of Technology
E-mail: †takayama@de.cs.titech.ac.jp, ††, †††daik@de.cs.titech.ac.jp, ††††, ††yokota@cs.titech.ac.jp

Abstract　Data transmitted must be prevented from intruder's interception because the information security grows more and more recently. Mostly, cryptography is used for the data protection. Encrypt-on-disk system costs re-encryption work when the user is revoked, while it offers improved performance of transmission over the encrypt-on-wire system. There are two methods for re-encryption after the revocation, active revocation and lazy revocation. When using active revocation, data is re-encrypted immediately after the revocation. While using lazy revocation, delays re-encryption to the next time the data is updated. There is the trade-off that active revocation is expensive because of immediately re-encryption, while lazy revocation is vulnerable because of delayed re-encryption. In this paper, we propose more efficient method for re-encryption after revocation on the basis of the distributed system which makes data replicas on the other disk node. And we evaluate the efficiency of the method by the experiments.

Key words　privacy protection, parallel·distributed DB, storage system backup

1. はじめに

ストレージ装置のネットワーク化が進んでいる．情報セキュリティが重要視されるようになってきている近年において，このようなネットワークストレージでは伝送中のデータの保護は必須である．そのため様々なセキュアストレージシステムで，伝送中のデータの保護機能は重要な機能の一つとして実現されている [1], [2].

一方，我々はこれまで，ストレージ装置の演算処理能力を利用し，耐故障化，負荷均衡化，容量分散などの機能を自律的に

実行する高機能な分散ストレージシステムとして，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って構成される．しかし，これまでは伝送路上におけるデータの保護については考慮していなかった．

本稿では伝送中のデータの保護要件のうち，機密性の保障に着目する．機密性とは，データがアクセス権のないユーザには見ることができない状態であることが保障されるという性質で，ほとんどの場合暗号を用いて実現されている．伝送中のデータに対する機密性の保障として encrypt-on-wire 方式と encrypt-on-disk 方式の 2 方式がある．暗号化されたデータを保存しておく encrypt-on-disk 方式は，伝送時のみ暗号を用いる encrypt-on-wire 方式よりもデータ転送に関して効率がよく，セキュリティ面でも同等に安全である．しかしその反面，複数のユーザによってデータを共有する環境では，encrypt-on-disk 方式ではユーザのアクセス権失効（revocation）に伴ってデータを再暗号化する必要がある．Kallahalla 等の調査 [5] によると，MIT では 7ヶ月で 29,203 回の revocation が発生したとされていて，revocation に伴う処理による影響は決して無視できるものではないと言える．revocation に伴う再暗号化処理の既存方式である active revocation と lazy revocation にはそれぞれ，直ちに実行しなければならない再暗号化による性能低下と，再暗号化を遅延することによるデータの脆弱な状態の持続という相反する欠点が存在するという問題がある [1], [2].

本稿では，自律ディスクのような高機能分散ストレージシステムに encrypt-on-disk 方式を採用し，より効率よく revocation に伴う再暗号化処理を実現する方法を考案する．全てのデータについて，バックアップデータを他のディスクに持つ構造を前提とし，バックアップデータを予め異なる暗号鍵で暗号化しておき，revocation 時には実際に再暗号化処理を行う替わりにバックアップデータを置き換えることで，迅速かつ低コストな revocation を実現する．

以下に本稿の構成を述べる．まず 2. で encrypt-on-disk 方式と encrypt-on-wire 方式の特徴を述べる．また 2.4 では encrypt-on-disk 方式における revocation の既存方式とその問題点について述べる．3. で関連研究について述べる次に 4. で前提とするシステム構成を説明し，5. でその前提に基づいた提案方式の提案を行う．6. で 5.3 の処理の流れに基づいた実装を用いた実験の結果と考察を述べ，最後に 7. でまとめと今後の課題を述べる．

2. encrypt-on-wire と encrypt-on-disk

2.1 データ転送時の暗号利用方式

ネットワークストレージにおいて，伝送路上のデータを悪意あるユーザの傍受から守るため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，セッション毎に新たに生成される暗号鍵を用いてデータを暗号化し，伝送する方式である．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に格納しておき，伝送時は暗号化処理を行わずそのまま送信する方式である．

データ転送時におけるパフォーマンスについて両方式を比較すると，encrypt-on-disk 方式ではストレージ側のデータ送信/受信時に暗号化/復号の処理を行う必要がないため，必ず暗号化/復号処理を実行しなければならない encrypt-on-wire 方式よりも効率が良い．さらに encrypt-on-wire 方式ではセッション毎に新しい暗号鍵を生成するコストがかかることからも，encrypt-on-disk 方式のほうが性能面で有利である [1].

2.2 encrypt-on-disk 方式における鍵使用の粒度

encrypt-on-disk 方式では，一つの鍵を使用する粒度によって性能が異なってくる．

アクセス権を所有するユーザが同じである複数のデータは，同じ暗号鍵で暗号化しても問題はない．このようなシステムでは，データ毎に異なる暗号鍵を使用するシステムと比較して，システム全体で生成，配布される暗号鍵の数が少なく，パフォーマンスが向上する反面，一つの暗号鍵の漏洩が複数のデータに及ぶために脆弱性が増すことになる．

2.3 encrypt-on-disk 方式におけるアクセス権失効

複数ユーザによってデータを共有するシステムにおいて，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たな暗号鍵で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）は現在の暗号鍵を知っている可能性があり，たとえアクセス権管理により revoked user のアクセス要求を拒否しても，傍受などで revoked user にデータが渡ると情報が知られてしまうからである．

このため revocation の処理においては encrypt-on-disk 方式は encrypt-on-wire と比べてコストがかかる．しかしこの点を含めた上で，encrypt-on-disk システムは encrypt-on-wire システムよりもパフォーマンス面，セキュリティ面共に優れているという検証結果がある [1].

2.4 再暗号化における既存方式とその問題点

encrypt-on-disk 方式での revocation に伴う再暗号化は，再暗号化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．

2.4.1 active revocation

active revocation は，revocation が発生すると直ちに対象データを新しい暗号鍵で再暗号化を実行する方式である．revoked user は revocation 発生直後から対象データを復号できなくなるため，2.4.2 で後述の lazy revocation と比較して安全であるが，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くなるため，性能を低下させる可能性がある．これは revocation の対象データが複数に及ぶ場合顕著である．

2.4.2 lazy revocation

lazy revocation は，対象データの再暗号化を次の更新時まで遅延する方式である．Cepheus [4] において初めて提案され，Plutus [5] 等で採用されている．データの更新処理は，暗号化処理を伴うため，revocation のための復号，暗号化処理を兼ねることができる．また，頻繁に更新されないデータでは，revocation の度に再暗号化を行わなければならない active revocation と比べて，複数回の revocation の再暗号化処理をまとめることができるので，その性能差は大きなものとなる．この方式では，revocation 発生後まだ更新されていないデータは，revocation 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扱われる．これは，revoked user がすでに知っている可能性のある更新前の情報は，revoked user に漏洩しても問題ない，という考えに基づくものであるが，revoked user が revocation 発生前にデータを取得していないこともあり得るため，active revocation と比較するとセキュリティ面で劣ると言える．

2.4.3 比　　較

パフォーマンス面では lazy revocation の方がはるかに優れているが，セキュリティ面を考慮すると，active revocation を行うべきであると考える．

そこで本稿では，active revocation と同等の安全性を実現し，4. で述べる環境を前提とした，より効率のいい active revocation の手法を提案する．詳細は 5. で説明する．

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 [6]，Plutus [5]，SiRiUS [7] などが挙げられる．Plutus は lazy revocation，SiRiUS は active revocation を採用し，SNAD は両方式のトレードオフの問題から revocation は今後の課題としている．これらは，クライアントはサーバが不正を行わないことを信用できない (trust でない) 状況を想定し，データを復号するのはクライアントマシン上のみとしていて，ストレージ側で処理を行い効率化を目指す本稿の提案とは異なる．

一方 encrypt-on-wire 方式のシステムとしては iSCSI などに用いられる IPsec 等がある．また，iSCSI を用いた IP-SAN において，IPsec を利用するよりも効率よく暗号化処理を行うミドルウェアの研究 [8] がある．

また，Seagate はハードディスクに内蔵するための暗号化技術として DriveTrust [9] を開発した．これはハードディスク内の全データを暗号化する技術であるが，その目的はハードディスク紛失時における内部のデータの保護であり，本稿の目的である伝送路上のデータの保護とは異なる．

4. 想定システム

4.1 前提となるストレージシステム

4.1.1 高機能ストレージシステム

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の研究，開発として，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．

本稿では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に encrypt-on-disk 方式を適用することを考える．revocation にともなう再暗号化処理や，新しい暗号鍵の配布などの処理を極力ストレージシステム側で行う．また，このような処理を行うために，各ストレージは trust であることを前提とする．

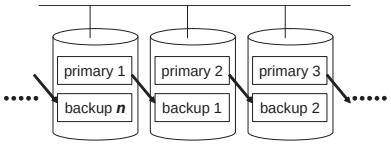

図 1　Chained Declustering による複製配置

4.1.2 プライマリ・バックアップ構造

耐故障性機能実現のために，並列ストレージシステムの各ディスクノードは，アクセスされるプライマリデータと，他のディスク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．本稿の提案においては，このように他のディスクにデータの複製がある構造を前提とする．

また，このバックアップ配置には制約がある場合が存在する．例えば自律ディスクのプライマリ・バックアップは標準では図 1 のように Chained Declustering [10] に従って配置されている．このように配置に制約がある場合と，配置に制約がない場合を分けて考える．

4.1.3 ネットワーク構成

各データ格納ノードは図 1 のように並列にネットワークへ接続されている．これらのストレージ装置に対し，同様にネットワークに接続されたクライアントノードによってアクセスされるものとする．

4.2 データ暗号化構造

暗号を用いるシステムにおいて，鍵管理の概念は重要である．分散ストレージシステムのディスク上で暗号鍵を管理する場合，暗号鍵は分散して管理されるべきである．一部で集中して管理を行うとそこが SPOF となり，また悪意あるユーザの攻撃対象となり得るからである．

4.1.1 で述べたようにディスク上で再暗号化処理を行うことを考えると，暗号鍵はそれが用いられるデータと一緒に管理するのが効率の面からも良いと考える．本稿で利用した鍵管理構造を以下で説明する．

4.2.1 暗号の種類と利用法

暗号は，暗号化と復号に共通の鍵を用いる共通鍵暗号と，異なる鍵を用いる公開鍵暗号に分類することができる．共通鍵暗号では，予め暗号化側と復号側で安全な方法によっ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，公開鍵を公開することができるので，鍵配布の問題はないが，その反面一般的には共通鍵暗号の数百〜数千倍の処理速度であるという問題がある．これらの特性より，一般的には共通鍵の配布のために公開鍵暗号を用い，その共通鍵を用いてデータのやりとりをする場合が多い．本稿においてもこの方式をとり，ファイルは共通鍵で暗号化し，その共通鍵は送信先のクライアントまたはディスク固有の公開鍵で暗号化して送信するものとする．

また，共通鍵アルゴリズムは攻撃に強いとされるものを選ぶ必要がある．revoked user はアクセス権を失う直前のファイルを持っている可能性があり，revocation 発生後，更新発生前のファイルを傍受すると，その平文と暗号文の組から既知平文攻

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

K_A^+ :ユーザAの公開鍵
K :ファイルfの共通鍵

図 2　SNAD の *key object*

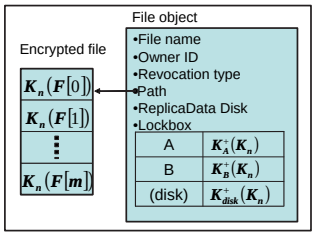

図 3　データ構造

撃により鍵を特定されてしまう可能性があるからである．

4.2.2　ロックボックス

ロックボックスとは，暗号鍵が格納されていて，その使用権を持つユーザのみ，それを取り出すことができる鍵管理構造である．SNAD [6] におけるロックボックスである *key object* の例を図 2 に示す．*key object* はファイルを暗号化するための 1 つの共通鍵 K を格納している．*key object* には 1 個または複数個のファイルが関連付けられていて，関連付けられたファイルは全てその暗号鍵で暗号化されているものとする．

key object のタプルはそのファイルへのアクセス権を与えられているユーザの ID，ユーザの公開鍵 K_x^+ で暗号化されたファイルの共通鍵，ユーザが認可されている読み込みや書き込みなどのアクセス権 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ある秘密鍵を用いて復号しなければならないので，正しく認可されたユーザのみが共通鍵を得ることができる．

また，*key object* が改竄されても検出できるよう，編集を行ったユーザは *key object* のハッシュと自身の秘密鍵を用いて Signature を生成しなければならない．

ファイルの所有者は，これを用いることで共有者と直接通信を行うことなく暗号鍵の受け渡しを行うことができる．また，安全な暗号鍵保存機能だけでなく，そのまま各ユーザに転送しても問題ない形式で保存してあることから，鍵配送の効率がよいという機能もある．

ロックボックスの概念は Plutus [5] でも用いられているが，Plutus のものは SNAD とは構造が大きく異なり，複数の鍵を使用したより堅固な構造となっている．しかし，こちらはロックボックス解除のための鍵の一つを所有者が共有者へ配布しなければならない．また非常に複雑であることから，SNAD の方式を採用した．

4.2.3　file object

file object は SNAD のデータ構造の一つで，ファイル本体へのポインタ，*key object* へのポインタ，ファイルのメタデータで構成される．データにアクセスする時は *file object* から本体情報と暗号鍵情報を取得し，アクセスする．

4.3　本稿でのデータ構造

本稿は 4.2 で述べた基本構造を基にした，図 3 の構造を用いる．各ユーザ及びディスクは公開鍵，秘密鍵の対 (K_x^+, K_x^-) を持つこと，ファイルは共通鍵 K_n で暗号化されていることを前提とする．ここでファイルはそのまま，あるいは複数個の固定長ブロックに分割して暗号化する．これは，クライアントがファイルを更新する際の差分データの単位をこのブロックにすることで，転送および暗号化のコストを削減するためである．

ファイルオブジェクトにはファイル名，所有者の ID，revocation 時の処理方式，暗号化ファイルの位置，複製データのあるディスク，ロックボックスの情報を持つ．ロックボックスはファイルへのアクセス権を持つユーザ用の暗号鍵の他に，再暗号化等の処理を行うために，格納されたディスク専用の鍵も保存している．さらに提案手法におけるバックアップデータのファイルオブジェクトでは，プライマリデータに使用されている暗号鍵を保存している．これはバックアップデータに使用されている暗号鍵はユーザに知られてはならないため，更新データなどはプライマリの暗号鍵で暗号化し，やり取りされるからである．

SNAD においては 1 つの暗号鍵を，同じユーザグループによって共有される複数のファイルに使用することが認められているが，複数のディスクに同じ鍵で暗号化されているファイルが分散している場合などで処理が複雑になるため，本稿では簡単のためにファイル毎にユニークな暗号鍵を使用するものとする．そのため，SNAD では 1 つの *key object* が複数の *file object* から参照される可能性があったが，本稿では 1 対 1 対応とするため，ロックボックスをファイルオブジェクトの 1 要素として扱っている．暗号鍵が複数ファイルに使用される状況への対応は今後の課題とする．

また，ロックボックスの完全性の保障や，各ユーザのアクセス権の区別（read only など）は今回考慮しないものとし，そのため SNAD の *key object* と比較してこのロックボックスが持つ情報は最小限のものとしている．

5.　提案手法の概要

本稿では，プライマリ・バックアップ構造を持つ，暗号を利用した高機能分散ストレージシステムにおける効率のよい revocation 処理実現のために，プライマリデータとバックアップデータに異なる暗号鍵を用い，revocation 発生時にバックアップデータをプライマリに昇格させることで，低コストかつ迅速に処理を終了させる方式を提案する．

これにより，実際に再暗号化処理を実行する前に対象ファイルへアクセスできるようになるため，active revocation よりも待機時間が短くなる．active revocation のように複数のディスクで再暗号化処理を実行している状態がなくなり，性能低下を抑えることができる．

5.1　複製データへの異なる暗号鍵の適用

バックアップに複製データを作成する時，プライマリデータのコピーをそのまま置くのではなく，異なる暗号鍵を生成し，その暗号鍵で暗号化して格納するようにする．このバックアップデータはユーザからアクセスされず，よってバックアップデータの暗号鍵も知られていないものとする．revocation が発生した際には，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あるディスクのプライマリに移動して新たなプライマリデータとし，元のプライマリデータは新たなバックアップとして別の暗号鍵を生成し，再暗号化する．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データへ

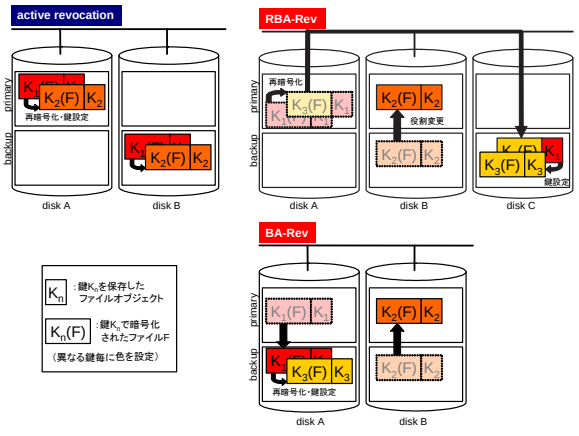

図 4　既存方式と提案方式の比較

のアクセスができなくなるが，提案方式では再暗号化処理はアクセスされないバックアップデータのみであり，すぐにアクセスを受け付けることができる．

5.2　制約条件による提案方式

提案方式は，4.1.2 で述べたプライマリ・バックアップの配置の制約の有無を想定し，以下のように区別して処理を実行する．またその比較を図 4 に示す．

これ以降，revocation 対象ファイルのプライマリデータがあるディスクをディスク A，そのバックアップデータがあるディスクをディスク B，配置に制約がある場合における提案方式で，新たにバックアップデータを置くべきディスクをディスク C と表記する．

RBA-Rev（Restricted Backup Assist Revocation）：　バックアップ配置に制約がある場合

配置を崩さないように，元のプライマリデータをディスク C へ転送する．この時，ファイルの再暗号化は転送前，ロックボックスの新しい暗号鍵での更新は転送後に行う．これは，revocation 発生前の鍵で暗号化された脆弱なデータはネットワークに流すべきではなく，またその時点でアクセス権のあるユーザは，次回の revocation 時にプライマリ暗号鍵となる新しい鍵が登録されたロックボックスからまだ知られてはいけない，新しい鍵を取り出すことが可能だからである．

BA-Rev（Backup Assist Revocation）：　バックアップ配置に制約がない場合

元のプライマリデータは再暗号化され，そのままディスク A のバックアップに置く．

5.3　各 revocation 処理方式による処理の流れ

5.3.1　revocation 発生時の処理

active revocation，RBA-Rev，BA-Rev において revocation が発生した時の，共通の処理（ファイル名，ユーザ名受信）以降の処理の流れを図 5 に示す．それぞれ縦の矢印が一つのディスクでの処理を表し，状況に応じて複数スレッドによる並行処理を実行している様子を表している．

実際には図で示す以外にファイルオブジェクトの I/O や転送も行っているが，本稿における環境ではファイルオブジェクトのサイズは 1KB 未満で I/O 及び転送時間は充分小さいため，図では省略した．

なお，両提案方式は revocation に伴いファイルの位置が変化するため，本来はインデックスを更新する必要があるが，今回はこの処理については考慮しないものとする．

（1）　active revocation

プライマリデータとバックアップデータを直ちに再暗号化する．ディスク A で新しい共通鍵を生成後，ファイルの読み出し（read），復号（dec），暗号化（enc），ファイルの書き込み（write）を実行しつつ，別スレッドで鍵，ファイル名，ユーザ名をディスク B に送信し，バックアップ側も同様の処理を行う．プライマリの処理が終了した時点で，要求元クライアントへ終了通知を送信する．

また再暗号化処理が終了するまで対象のプライマリデータへのアクセスはブロックする．

（2）　RBA-Rev

ディスク A が要求を受け取ると，その情報をディスク B へ送信し待機する．ディスク B はファイルオブジェクトから revoked user 用の鍵を削除し，プライマリデータとして設定しディスク A へ終了通知を送信する．この時点で新しいプライマリデータへのアクセスが可能な状態なので，要求元クライアントへ終了通知を送信する．またここまでの処理が終わる前にディスク A に対象ファイルへのアクセス要求が来た場合は，処理終了後に新しい位置情報とともに再アクセス要求を返し，クライアントはその情報に従って再アクセスする．

その後，ディスク A はディスク B から，新しいプライマリの共通鍵を受信し，保存する（プライマリからの更新データ復号のため．4.3 参照）．次に，別に生成した新バックアップ用の共通鍵でファイルを再暗号化後，プライマリ・バックアップ配置制約に従ってディスク C へファイルを送信する．

（3）　BA-Rev

新しいバックアップのデータを再暗号化するまでの処理は RBA-Rev と同じである．ここではプライマリ・バックアップ配置に制約がないため，そのままそのディスク A のバックアップへ置く．

（4）　lazy revocation

本稿では，lazy revocation はセキュリティ面で他方式より劣ると言う考えから，性能面での比較対象としない．よって以下省略する．

5.3.2　更新発生時の処理

以下に，データ更新が発生した時の，各 revocation 方式毎の処理をまとめる．クライアントは，現在のプライマリデータの共通鍵で差分データを暗号化し，送信するものとする．この更新単位は 4.3 で述べたブロック単位とする．

（1）　active revocation

差分データを受信後，それを該当箇所へ上書きして反映させる．これと並行して差分データをそのままバックアップ

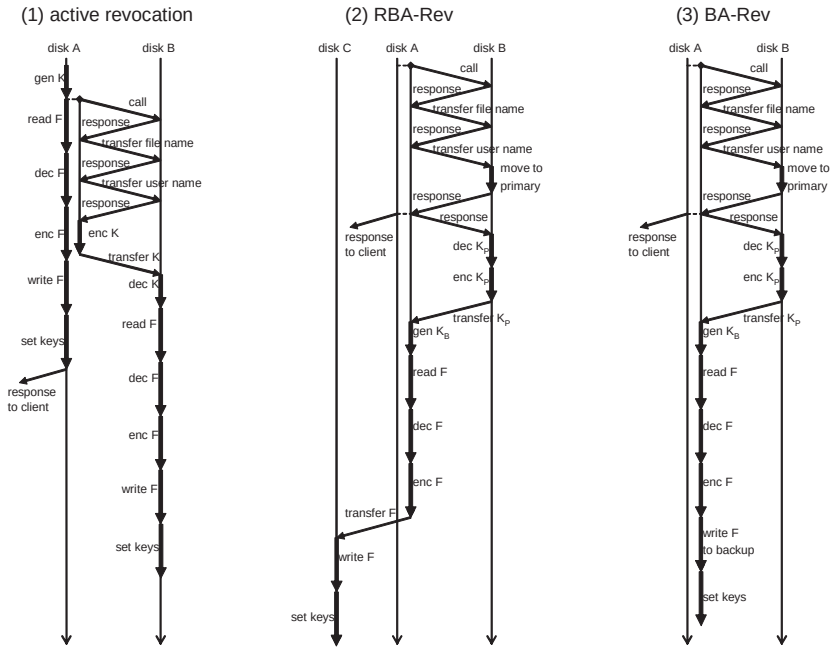

図 5　revocation 発生時の各方式の処理の流れ

にも送信し，同様に上書きして反映させる．

（2） RBA-Rev，BA-Rev

プライマリの更新は active revocation と同じである．バックアップは，プライマリデータの共通鍵で暗号化された差分データを受け取り，保存してあったプライマリの共通鍵で復号し，バックアップデータの共通鍵で再暗号化して，該当箇所へ上書きして反映させる．

6. 実　　験

6.1 実行プログラム

提案方式の active revocation に対する優位性を確かめるために，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た．データ構造は 4.3 のものに従う．クライアントプログラムでは，put，get，update，アクセス権付与，アクセス権失効の 5 つを実行できる．

6.2 実 験 環 境

実験は表 1 に示す構成の PC クラスタ上で行った．また固定のパラメタとして表 2 のものを使用した．

各ディスクに保存されるデータのサイズは一定で，ファイル数はその総データサイズに従う．また受信ファイルは，revocation 対象ファイルへのアクセスが重なりやすいよう，アクセスされるファイルの選択に偏りを持たせるために，パラメタ θ によって決まる Zipf 分布に従って選ばれるものとした．

共通鍵アルゴリズムとしては，既知平文攻撃などの攻撃に強いとされる AES を用いた．また，4.3 で述べた，ファイルを分割して暗号化した場合における更新処理を実行する状況を作るために，暗号化モードとして共通鍵固定のブロックサイズ毎に独立して暗号化を行う ECB を用いた．

6.3 予 備 実 験

ファイル受信時における，encrypt-on-disk 方式の encrypt-on-wire 方式に対する優位性を確認するため，暗号を用いない方式と encrypt-on-wire 方式のクライアント・サーバプログラムを作成し，比較を行った．encrypt-on-wire 方式プログラムは，予めサーバとクライアントが共通の鍵を所持している場合はそれを使用し，所持していない場合はサーバが新たな鍵を生成し，ファイルと共にクライアントへ送信する．

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	PKCS5
総プライマリデータサイズ	500MB/disk
Zipf 母数 θ	0.9

これらのプログラムで，get の要求を送信してからファイルを受信し，暗号化されている場合は復号し，ディスクに書き込むまでの時間を応答時間として測定し，比較した．その結果を図 6 に示す．

図 6 より，ファイルサイズが大きくなるほど encrypt-on-wire 方式に対して encrypt-on-disk 方式が有利になることがわかる．これは，ファイルサイズが大きいほど encrypt-on-disk 方式において予め暗号化しておくことによるコスト削減の効果が大きいことを示している．

6.4 実 験 方 法

3 台のディスクに対し，予め 1 台のオーナークライアント PC

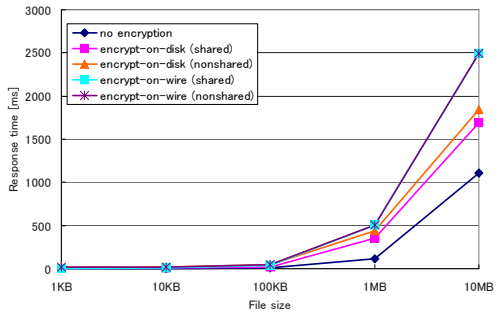

図 6　予備実験：各方式における応答時間

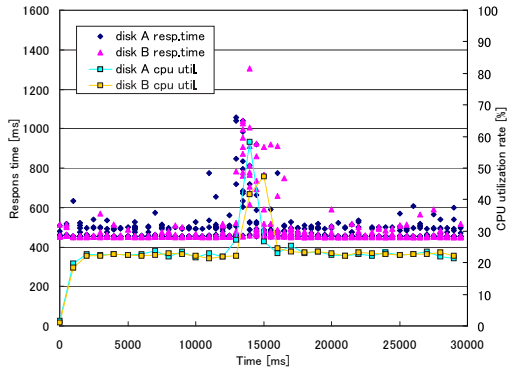

図 7　active revocation 方式で 1 回 revocation が発生した場合の変化

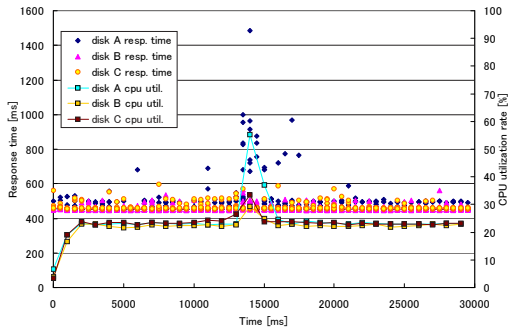

図 8　RBA-Rev 方式で 1 回 revocation が発生した場合の変化

から 1MB のファイルを 1 ディスク当たり 500 回 put しておく．この 3 台のディスクそれぞれに対し，1 台のクライアント PC によって 500ms の間隔で get リクエストを出し，その応答時間の変化を記録する．この間隔 500ms は，一定間隔で get リクエストを出してその応答時間を測定する実験を，様々な間隔について行い，平均応答時間が大きく変化しない範囲で最小のものを選んだ．この状態で，オーナークライアント PC によって最もアクセスされる確率が高いファイルについて revocation を発生させた時に応答時間がどのように変化したかを観測した．また，同時に vmstat コマンドにより CPU 使用率を測定した．

6.5　1 個のファイルに対して revocation が発生した場合

active revocation と RBA-Rev，BA-Rev で，1 ファイルについて revocation が発生した時の応答時間と CPU 使用率の変化を図 7〜9 に示す．実験は各方式毎に 10 回ずつ行い，応答時間は

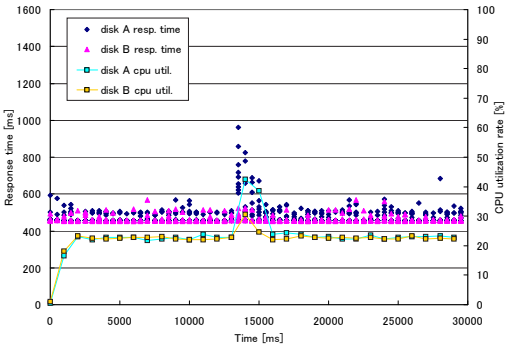

図 9　BA-Rev 方式で 1 回 revocation が発生した場合の変化

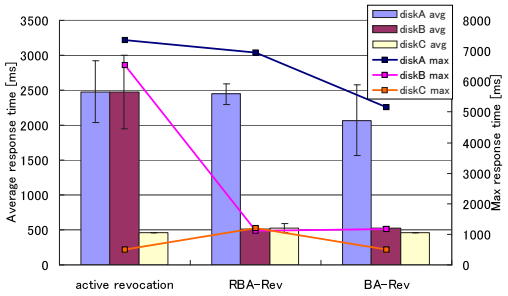

図 10　1 ディスクで複数回 revocation が発生した場合の平均および最大応答時間

全てについてプロットし，CPU 利用率は平均をプロットした．

6.5.1　考　　察

応答時間の分布を比較すると，図 7 の active revocation では revocation に伴い，対象ファイルのプライマリデータがあり，それにアクセスできなくなるディスク A だけでなく，対象ファイルのバックアップデータがあるものの，それはアクセスの対象ではないディスク B でも応答時間が遅くなる傾向があることがわかる．これはアクセスされないファイルの再暗号化処理も応答時間に大きく影響してくることを示している．

これに対し，図 8，9 の提案方式ではどちらもディスク B の応答時間はほとんど変化していない．ディスク B は再暗号化処理を行わなくてよいということにより，性能低下が大きく抑えられているといえる．

さらに，ディスク A から C への転送を行わなくてよい BA-Rev は，active revocation よりもディスク A での応答時間が増大しない傾向がある．これは対象ファイルへのアクセスが来た場合，ディスク B におけるプライマリ昇格の処理が終了次第そちらへの再アクセス要求を返していることによる効果だと考える．

6.6　1 ディスクの複数のファイルに対して revocation が発生した場合

1 つのディスクにある複数のファイルに対して，同時に revocation が発生した場合を想定し，実験を行った．ディスクのプライマリデータのうち，アクセスされる確率が上位である 24 個のファイルに対して revocation を発生させた時の，発生後 20 秒間の平均応答時間及び 95 %信頼区間，最大平均応答時間を測定した．結果を図 10 に示す．

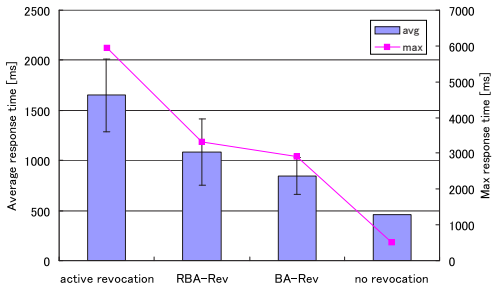

図 11　複数ディスクで revocation が発生した場合の平均及び最大応答時間

6.6.1　考　　察

図 10 より，対象ファイルが 1 個の場合と同様，active revocation と比較すると両提案方式ではディスク B の応答時間の変化は非常に小さく，ディスク B で再暗号化処理を実行しないことによる効果が確認できる．

また，BA-Rev ではディスク A でも応答時間の増加が抑えられていることがわかる．これは，再暗号化処理を待たずにアクセス可能となることによる効果だと考える．RBA-Rev でも同様の効果があるが，同時にディスク C へのファイル転送のコストが発生するため，既存方式と同等の結果となったと考える．

一方最大応答時間を見ると，平均ではほぼ同じであるのにも関わらず，active revocation よりも RBA-Rev の方が小さくなっている．これは，RBA-Rev のディスク A での再暗号化処理はディスク B の処理を待って実行されるために開始タイミングが分散し，瞬間的に負荷がかかるのを防いでいるからと考える．

6.7　連続する複数ディスクで revocation が発生した場合

連続する複数のディスクで同時に revocation が発生した場合を想定し，実験を行った．各ディスクの 8 個のプライマリデータで同時に revocation を発生させ，全ディスクにおける発生後 20 秒間の平均及び最大応答時間を測定した．結果を図 11 に示す．

6.7.1　考　　察

RBA-Rev 及び BA-Rev の平常状態からの応答時間増加量は，それぞれ active revocation での増加量より 47 %，68 %削減されている．この実験では 1 ディスクでこれまでのディスク A，B，C に相当する全ての処理コストがかかるため，単純に各方式毎の総処理量が影響する．そのため，元のバックアップデータの再暗号化を必要としない両提案方式，特にバックアップの転送を行わない BA-Rev の結果が良くなったと考える．

7.　まとめと今後の課題

本研究では，データの複製を他ディスクに持つ encrypt-on-disk 方式分散ストレージシステムにおける，active revocation よりも高速かつ効率のよい revocation 時再暗号化処理方式 RBA-Rev, BA-Rev を提案した．アクセスされない複製データを予め異なる暗号鍵で暗号化して準備しておくことによって，revocation 対象ファイルへアクセスできない状態の継続を抑え，同時に再暗号化処理を行うディスク数が減少して，複数のディスクで大きく性能が低下することを抑えることができた．その効果は配置に制約がない場合において特に大きいことを確認した．

今後の課題としてはまず，データの更新がある状況において，lazy revocation も比較対象に加えての実験，評価が必要である．

また，今回はファイル単位でアクセスするものとしたが，より現実に近い環境での評価のため，ブロック単位のアクセスで実験を行うことは有用だと考える．

一方，この提案方式は自律ディスクの機能の 1 つであるデータマイグレーションに対応していない．データマイグレーションはその効率化のために，バックアップデータをそのディスクのプライマリに昇格させることで実現している．ここでプライマリデータとバックアップデータで使用されている暗号鍵が異なる状況での処理を考えなければならない．さらに今回はデータの位置が移動することによるインデックスの更新処理を考慮していないため，本来はインデックス更新処理コストを考慮すべきである．このことを含めて，提案方式を自律ディスクに実装し，実験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，情報ストレージ研究推進機構 (SRCs)，NHK，文部科学省科学研究費補助金特定領域研究 (18049026) および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行なわれた．

文　　献

[1] Erik Riedel, Mahesh Kallahalla, and Ram Swaminathan. A framework for evalating storage system security. In *FAST '02: Proc. of the 1st USENIX Conf.* , pp. 15–30. USENIX Association, 2002.

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

[3] Haruo Yokota. Autonomous Disks for Advanced Database Applications. In *Proc. of International Symposium on Database Applications in Non-Traditional Environments (DANTE'99)*, pp. 441–448, Nov. 1999.

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

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

[6] Ethan Miller, Darrell Long, William Freeman, and Benjamin Reed. Strong Security for Network-Attached Strorage. In *FAST '02: Proc. of the 1st USENIX Conf.*, p. 1, Berkeley, CA, USA, 2002. USENIX Association.

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

[8] 神坂紀久子, 山口実靖, 小口正人. 高遅延環境下における IP-SAN を用いた暗号処理最適化手法の実装と評価. 電子情報通信学会論文誌 D, Vol. J90-D, No. 1, pp. 16–29, Jan. 2007.

[9] Seagate. Drivetrust[tm] technology: A technical overview. http://-www.seagate.com/docs/pdf/whitepaper/TP564_DriveTrust_Oct06.pdf.

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