平成 14 年度

修士学位論文

秘密分散法を用いた
広域分散ファイルシステム

Widely Distributed File System
Using Secret Sharing Scheme

1055115　　舟橋 釈仁

指導教員　　菊池 豊

2003 年 2 月 24 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

秘密分散法を用いた広域分散ファイルシステム

舟橋 釈仁

近年，ストレージに蓄えられる情報が爆発的に増えており，ストレージ技術における情報管理の重要性はますます増加している．ストレージ技術の中核を占める RAID やバックアップは，もともと磁気ディスク装置の高信頼性と高性能化を行うための技術である．そのため，RAID やバックアップを用いた分散保管方式は，十分な秘匿性と対災害性を同時に持たない．したがって，十分な秘匿性と対災害性を保証した保管方式が必要とされている．

本研究は，秘密分散法を用いた分散データ管理方式を提案している．提案手法と RAID，そしてバックアップによって作成した分散データを比較し，提案手法の分散データが十分な秘匿性と対災害性を持つことを示した．また，提案手法の実装を検討し，分散データの管理とネットワーク分散の容易さからファイルシステムによる実装が最適という結果を得た．そこで，分散ファイルシステムによる実装を行い，実行速度を測定した結果，ブロック長が小さいときに実用的な範囲で動作した．

キーワード　秘密分散法，分散保管，分散ファイルシステム

Abstract

Widely Distributed File System Using Secret Sharing Scheme

FUNAHASHI Sekiji

The data of keeping storages explosively increase and data control method of storage technology is more important. RAID and Backup which is one of most important storage technology is originally to get high reliability and performance for a magnetic disk. Therefore distributed store method using RAID and Backup couldn't have enough safety of cracking and disasters.

We present data distributed management method using Secret Sharing Scheme for high safety of security and disasters. it's method and RAID and backup compared with the result that distribution data of it's method have enough safety of cracking and disasters. Also it is easy to manage distribution data and distribute using network so that implementation with file system is optimum. And so we implement the file system using it and as a result it performed as good speed in small block size.

key words Secret Sharing Scheme, Distrubuted store method, Distributed File System

目次

図目次

表目次

第 1 章

はじめに

1.1 背景と目的

近年，ストレージに蓄えられる情報が爆発的に増えており，ストレージ技術における情報管理の重要性はますます増加している．例えば，地方自治体の管理する住民基本台帳は，住民基本台帳ネットワークのためにデータベース化されている．このような情報の保管には，十分な秘匿性を保ちつつ，広域災害にそなえた分散保管が必要になる．そのため，住民基本台帳や保健・医療・福祉等の個人情報のような重要情報を管理する組織では，たとえコストが若干余計にかかろうとも十分な秘匿性と対災害性を保証して情報を保管したいという要望がある．

広域災害による被害でデータを損失しないために，RAID [*1] と SAN [*2] ，リモートコピーと NAS [*3] を組み合わせた分散保管が実用化されている．しかし，RAID やリモートコピーは，もともと磁気ディスク装置の高信頼性と高性能化を行うための技術である．そのため，RAID やバックアップによって作成した分散情報を分散管理する方式では，十分な秘匿性と対災害性を保証できない．

そこで，本研究は十分な秘匿性と対災害性を保証した分散管理を実現するために，秘密分散法を用いた分散データ管理方式を提案し，ファイルシステムによる実装を行う．

[*1] Redundant Arrays of Inexpensive Disks

[*2] Storage Area Network

[*3] Network Attached Storage

1.2　本論文の概要

本論文では，秘密分散法を用いた分散データ管理方式を提案し，提案方式のファイルシステムによる実装について述べる．

第 2 章では，データを保管する技術であるストレージ技術について述べる．

第 3 章では，提案方式に用いる秘密分散法について述べる．特に本研究で用いる (k, n) しきい値秘密分散法とその演算について述べる．

第 4 章では，秘密分散法を用いた分散データ管理方式を提案する．提案方式と既存の方式の比較を行い有効性を示す．また，提案方式の実装について検討を行う．

第 5 章では，提案方式のファイルシステムについて設計を述べる．v ノードの構造と v ノードの操作，そしてマウントついて述べる．

第 6 章では，作成したファイルシステムの実行速度を測定し考察を行う．

最後に，本論文の成果をまとめ，今後の課題を述べる．

第 2 章

ストレージ技術

データを保管する磁気ディスク装置はコンピュータを構成する部品の中で最も故障率が高い部品である．そのため，ストレージ技術はもともと磁気ディスク装置の大容量化・高性能化を行う技術としてはじまった．1988 年に発表された RAID [4] は，複数の磁気ディスク装置に分割してデータを保管することにより高い信頼性と処理速度の向上を達成する技術である．

コンピュータの普及とともに管理するデータの増加が続いている現在では，大容量のデータを効率よく管理するための研究が行われている．ストレージデバイスをそれぞれのコンピュータの周辺装置として管理するダイレクトアタッチストレージ手法は，大容量のデータを管理ようとすると，それぞれのコンピュータ毎に異なった管理をしなければならず多大なコストを必要とする．そこで，ストレージデバイスを集約してファイバチャンネルのような高速なストレージ専用ネットワークを用いてアクセスする SAN [*1] や，NFS/CIFS のようにストレージデバイス側でファイルシステム機能を持ち IP ネットワークを用いたデータ共有を可能とする NAS [*2] が用いられるようになった．

本章では，データを保管するためのストレージ技術について述べる．最初に，オペレーティングシステムを含むデータ保管の流れを述べる．次に，分散データを作成して冗長化をはかるバックアップと RAID について述べる．そして，ネットワークを用いた分散保管を行う SAN と NAS について述べる．

[*1] Storage Area Network

[*2] Network Attached Storage

2.1　データ保管の階層構造

ストレージに格納されているデータの読み出し，あるいはストレージにデータを格納する書き込み方法はオペレーティングシステムによって提供されている．オペレーティングシステムは，ファイルシステムとデバイスドライバを用いて階層構造を構築し，ユーザの要求するデータの読み書きを行っている．図 2.1 に階層構造におけるデータの流れを示す．図 2.1 は，ユーザがアプリケーションを用いてデータを書き込むと，オペレーティングシステム内のファイルシステムとデバイスドライバを経由して，ストレージにデータが格納される様子を示している．また，ストレージに格納されているデータを読み込む場合，データは書き込みの逆順にオペレーティングシステム内を通過する．

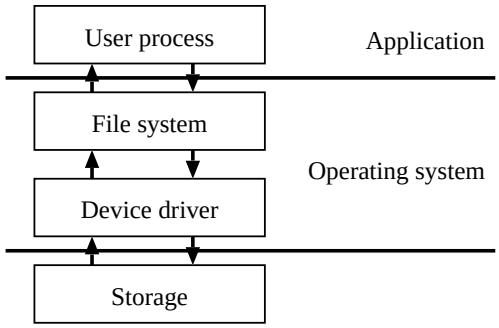

図 2.1　入出力の階層構造におけるデータの流れ

Fig.2.1 Data flow of data storing layer.

ファイルシステムはオペレーティングシステムの機能の一部で，ユーザにファイルという抽象概念とファイルの管理機能を提供する．ユーザはファイルシステムによって提供されるファイルに対し，データの書き込みや読み込みを行い，データをストレージに格納する．また，ユーザはファイルの作成，移動，削除，アクセス制御などを行うことによりデータの管理を行うことができる．ユーザのファイルに関する操作やデータは，ファイルシステムによってストレージ毎に異なる構造に変換され，デバイスドライバを用いてストレージに格納される．デバイスドライバは，ハードウェア毎に異なった制御インタフェースを隠蔽し，ファイルシステムに対して一貫したインタフェースを提供する．

2.1.1　ファイルシステム

ファイルシステムはユーザに対してファイルというデータの論理的な格納場所を提供するためのインタフェースである．ファイルシステムの持つインタフェースは大きく 2 種類に大別できる．1 つは，ファイルに対してデータの読み書きを行う操作で，open，close，read，write に代表される操作である．これらの操作は，ファイルの内容や構造を解釈することはなく，単純にファイルをバイトの集合とみなす．もう 1 つは，データが格納されているファイルに対して名前の変更や場所の移動，ファイルの削除やアクセス制御を行う操作である．これらの操作は，オペレーティングシステムによって一貫したインタフェースとして定義されている．

2.1.2　デバイスドライバ

デバイスドライバは，ファイルシステムに対して一貫したインタフェースを提供する．このインタフェースは，ハードウェアへの入出力に備える open 操作，入出力を終了する close 操作，そしてデータの読み込みと書き込みを行う read，write 操作を提供する．デバイスドライバもファイルシステムと同様に読み書きを行うデータを解釈することはなく，単純にバイトの集合とみなす．

2.2　分散データ作成技術

磁気ディスク装置はコンピュータを構成する部品の中で最も故障率が高い部品である．そこで，磁気ディスク装置が壊れときに備えてデータを復旧できるように，データを冗長化する技術が用いられている．本節では，分散データを作成して冗長化をはかるバックアップと RAID について述べる．

2.2.1　バックアップ

バックアップとは，磁気ディスク装置に保管してあるデータをテープドライブなどにコピーして，データの損失を防ぐことである．すべてのデータのバックアップをとることをフルバックアップ，ある時点からの差分だけのバックアップをインクリメタルバックアップと呼ぶ．常にフルバックアップをとることは効率が悪いので，インクリメンタルバックアップを組み合わせて，効率を上げる．

2.2.2　RAID

RAID は，保管するデータと障害回復のための冗長データを複数の磁気ディスクに分散して格納することで，対障害性と高性能を同時に実現する技術である．冗長データの種類と磁気ディスクへの保管のしかたによって，0〜5 の RAID レベルがある．例えば，RAID5 の場合，冗長データはパリティ計算によって作成され，保管は複数の磁気ディスク装置に順番に分散データを保管することにより，1 つの磁気ディスクに対するアクセスの集中を防ぐので，もっとも効率がよい方法である．

2.3　ストレージネットワーキング

ストレージに蓄えるデータの量が増大するにしたがって，データのバックアップやストレージの増設あるいは障害対応など運用にかかる負担が増加した．そこで，ストレージ装置をそれぞれのサーバの周辺機器としてサーバ単位で管理する手法から，複数のストレージを集約し，ネットワークを介してアクセスすることにより，統一的なデータ管理を行う手法が普及した．このネットワークを用いたデータ管理手法が SAN と NAS である．

2.3.1　SAN

SAN は Fibre Channel とシリアル SCSI-3 プロトコルを用いてストレージにアクセスする仕組みで，主に磁気ディスクやテープドライバを共有する．Fibre Channel を用いて転送

を行うことから，現在の転送速度は 2Gb/s，最長到達距離は 10km である．

図 2.2 に SAN の構成例を示す．図 2.2 では，3 台の機器で 4 つのストレージを共有している．

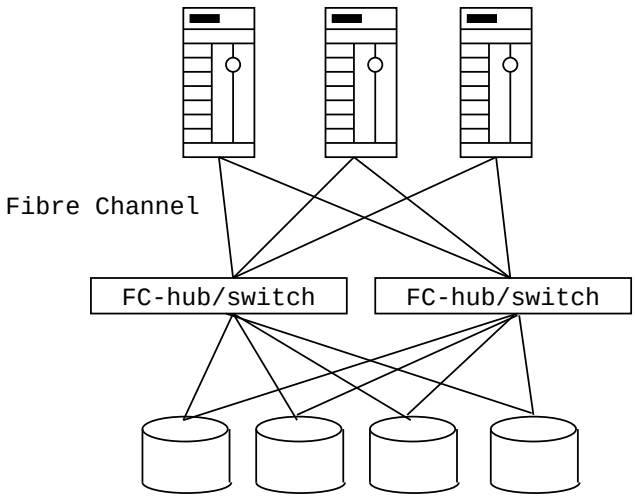

図 2.2　SAN のネットワーク構成

Fig.2.2 Structure of SAN.

2.3.2　NAS

NAS は IP ネットワークと NFS/CIFS を用いてファイルシステムの共有を行う仕組みである．ストレージに NFS/CIFS によるファイル共有を行うオペレーティングシステムが付属している．

図 2.3 に NAS の構成例を示す．図 2.3 では，3 台の機器は 4 つのストレージそれぞれに付属しているオペレーティングシステムを介してファイルシステムにアクセスすることができる．

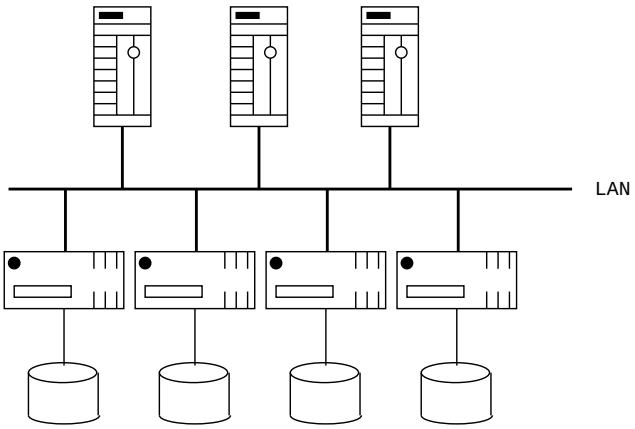

図 2.3　NAS のネットワーク構成

Fig.2.3 Structure of NAS.

第 3 章

秘密分散法

—分散データから元のデータが全く分からず，決められた数の分散データを集めると元のデータに復号できる—

保管したいデータをこのような性質を持つ分散データに符号化する方法が秘密分散法である．

秘密分散法 (Secret Sharing Scheme) に関する研究は 1979 年に Shamir [1] と Blakley [2] が独立に (k, n) しきい値法を提案したことにより始まっている．その後，構成法や特性に関する研究が行われ，さまざまな秘密分散法が提案されている．分散データの作成法に関して幾何学的な手法を用いるものやベクトル空間を利用したもの，マトロイド理論を利用したものがあり、分散データが持つ特性に関してランプ型秘密分散法や検証可秘密分散法などがある [3].

秘密分散法によって作成された分散データの安全性とサイズ (ビット長) には密接な関係があり，秘密分散法を使用する際には適切な構成を選択する必要がある．そこで，本章では，秘密分散法の構成と分散データのサイズ (ビット長) について述べる．

はじめに，一般化秘密分散法と分散データのサイズの範囲について示す．次に，Shamir の (k, n) しきい値秘密分散法の構成法と (k, d, n) ランプ型秘密分散法の構成法を述べる．そして，これらの秘密分散法によって作成される分散データを比較し，Shamir の (k, n) しきい値秘密分散法が理想的であることを述べる．また，Shamir の (k, n) しきい値秘密分散法は有限体上で演算が行われている．しかし，計算機による計算では，2 の拡大体上で演算を行うと効率がよい．そこで 2 の拡大体を用いた演算について述べる．

3.1　秘密分散法

本節では，一般化秘密分散法と (k, n) しきい値秘密分散法，及び (k, d, n) ランプ型秘密分散法の構成について述べる．

3.1.1　一般化秘密分散法

秘密情報 S を分散暗号化して得られる分散情報の集合 W を考える．

$$W = \{W_1, W_2, \ldots, W_n\} \tag{3.1}$$

分散情報 W の部分集合を U としたとき，U から秘密情報 S が完全に復号できるならば，U を S のアクセス集合という．また，アクセス集合から 1 つでも分散情報が減ると S を復号できなくなるとき，U を極小アクセス集合という．

極小アクセス集合 U が与えられたとすると，極小アクセス集合から S が一意に復号できるので，U が与えられたときの S に対する条件付きエントロピーは，

$$H(S|U) = 0 \tag{3.2}$$

となる．

極小アクセス集合から分散情報を 1 つ減らした集合 X が与えられたとすると，X から S を復号することができなくなる．さらに，X から S に関する情報が全く得られない場合，X が与えられたときの S に対する条件付きエントロピーは，

$$H(S|X) = H(S) \tag{3.3}$$

となる．そして，式 (3.3) が成り立つとき，1 つの分散情報 W_i がもつエントロピーは

$$H(W_i) \geq H(S) \tag{3.4}$$

を満たす．

秘密情報のサイズ (ビット長) は秘密情報 S の最大値から最小値を引いた値 $|S|$ で表され，$\log_2 |S|$ となる．よって，分散情報のサイズは式 (3.4) を用いて

$$\log_2 |W_i| \geq H(W_i) \geq H(S) = \log_2 |S| \tag{3.5}$$

となる．特に

$$\log_2 |W| = \log_2 |S| \tag{3.6}$$

となる場合を理想的という．

3.1.2　(k, n) しきい値秘密分散法

Shamir の (k, n) しきい値秘密分散法は n 個の分散情報のうち k 個以上集めると秘密情報を得られるが，k 個未満では秘密情報が全くわからないように構成される．以下にその構成法を示す．

保管するデータを秘密情報 S とし，n 個の分散情報 $W_i (i = 1, \ldots, n)$ を作成する．以後，秘密分散法によって作成された分散情報をシェアと呼ぶ．

秘密情報 S は素数 p を法とする有限体 $GF(p)$ の元とし，同様にシェアも $GF(p)$ の元とする．演算は $GF(p)$ 上で行われ，暗号化の手順と復号化の手順は次のようになる．

分散暗号化

1. $GF(p)$ の元から 0 でない別々のものを n 個選びだし，$x_i (i = 1, 2, \ldots, n)$ とする．ただし，$n < p$ である．
2. $GF(p)$ の元から $k - 1$ 個の乱数 $r_j (j = 1, 2, \ldots, k - 1)$ を選択する．
3. シェア $W_i (i = 1, \ldots, n)$ を

$$W_i = S + \sum_{j=1}^{k-1} r_j {x_i}^j \pmod{p} \tag{3.7}$$

により計算する．

4. x_i をシェア W_i の ID として，x_i と W_i を保管する．

復号化

1. 作成したシェア W_i の中から復号化に使うシェアを W_{i_j}，そのシェアに対応する ID を $x_{i_j} (1 \leq j \leq k, k < n)$ とする．
2. 式 (3.7) に W_{ij} と $x_{ij} (j = 0, 1, \ldots, k)$ を代入し，k 個の線形方程式を求める．

3. k 個の線形方程式から k 個の未知数 $S, r_1, r_2, \ldots, r_{k-1}$ を求めれば，秘密情報 S が得られる．

式 (3.7) は，連立 1 次方程式であることから

$$\begin{aligned} S + r_1 x_1 + r_2 {x_1}^2 + \cdots + r_{k-1} {x_1}^{k-1} &= w_1 \pmod p \\ S + r_1 x_2 + r_2 {x_2}^2 + \cdots + r_{k-1} {x_2}^{k-1} &= w_2 \pmod p \\ &\vdots \\ S + r_1 x_n + r_2 {x_n}^2 + \cdots + r_{k-1} {x_n}^{k-1} &= w_n \pmod p \end{aligned} \tag{3.8}$$

のように表せる．また，これは行列を用いて

$$\begin{bmatrix} 1 & x_1 & {x_1}^2 & \cdots & {x_1}^{k-1} \\ 1 & x_2 & {x_2}^2 & \cdots & {x_2}^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & {x_n}^2 & \cdots & {x_n}^{k-1} \end{bmatrix} \begin{bmatrix} S \\ r_1 \\ \vdots \\ r_{k-1} \end{bmatrix} \equiv \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \pmod p \tag{3.9}$$

と表せる．

$GF(p)$ の原始根 α をを用いて $x_1, x_2, \ldots, x_n$ を表せば，

$$\begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{k-1} \\ 1 & \alpha^2 & \alpha^4 & \cdots & \alpha^{2(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \alpha^{q^n-1} & \alpha^{(q^n-1)2} & \ldots & \alpha^{(q^n-1)(k-1)} \end{bmatrix} \begin{bmatrix} S \\ r_1 \\ \vdots \\ r_{k-1} \end{bmatrix} \equiv \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} \pmod p \tag{3.10}$$

となる．

3.1.3　(k, d, n) ランプ型秘密分散法

ここでは，(k, d, n) ランプ型秘密分散法の構成について述べる．

(k, d, n) ランプ型秘密分散法は n 個のシェアのうち k 個以上集めると秘密情報を得ることができ，$k - d$ 以下集めただけでは秘密情報が全くわからず，$k - d$ より多く k 個未満集めると秘密情報の部分的な情報が分かる性質を持つ．

秘密情報 S を

$$S = S_1 + \cdots + S_d \tag{3.11}$$

とし, $k > d$ が成り立っているとする.

このときシェア $w_i(i = 1, 2, \ldots, n)$ は秘密情報の部分的な値 $S_l(l = 0, 1, \ldots, d)$ と素数 p とその原始根 α ，及び乱数 $r_j(j = 1, 2, \ldots, k - d)$ を用いて

$$w_i = S_0 + S_1\alpha + S_2\alpha^2 + \cdots + S_d\alpha^d + r_1\alpha^{d+1} + \cdots + r_{k-d}\alpha^{k-1} \pmod{p} \tag{3.12}$$

と表せる.

3.2　分散情報のサイズ

秘密分散法が効率的かどうかの指針にシェアのサイズを用いることができる．ランプ型秘密分散法のようにシェアのサイズを秘密情報のサイズより小さくしようとすると，非アクセス集合から部分情報がわかる．逆に，検証化秘密分散法のようにシェアとは別に検証情報を作成すると，検証情報から秘密情報を計算量的に求めることが可能な場合がある.

Shamir の (k, n) しきい値法は多項式補間に基づいて構成されている．この構成法で作成されたシェアのサイズは秘密情報のサイズと等しい．よって Shamir の (k, n) しきい値法は理想的である．また，検証情報を作成しないので，安全性が下がる可能性もない.

3.3　2 の拡大体上の演算

Shamir の (k, n) しきい値法において式 (3.7) が成り立つように x_i を選ぶには，$GF(p)$ の原始根 α のべきを用いればよい．しかし，大きな p の原始根を効率的に求める方法は知られていない．そこで，原始根を求める計算が必要なく，計算機上で効率よく演算できる 2 の n 次拡大体 $GF(2^n)$ 上での演算について述べる.

多項式を元とする集合を考え，$GF(2)$ 上の $m - 1$ 次以下の多項式の集合を X_m とする．$GF(2)$ 上の m 次の多項式で既約であるものを $P(x)$ とする．このとき，X_m の元 $F_1(x)$ と $F_2(x)$ の積を，多項式の $F_1(x)F_2(x)$ を $P(x)$ で割った剰余多項式として定義する.

$$F_1(x)F_2(x) \pmod{P(x)} \tag{3.13}$$

集合 X_m は非零の任意の元に対し乗法逆元が存在する．よって，X_m において加減乗除が定義できるので，体 $GF(2^m)$ が定義できる．有限体 $GF(2^m)$ を $GF(2)$ の m 次の拡大体と呼ぶ．

既約多項式 $P(x)$ で生成される $GF(2^m)$ の元は，$P(x)$ の 1 つの根を α とすると，α の $m-1$ 次以下の多項式として表される．言い換えれば，α^{m-1}，α^{m-2}，...，α，1 により展開した形で表現できる．この $\alpha^{m-1},\ldots,\alpha,1$ を $GF(p^m)$ の多項式基底と呼ぶ．$GF(p^m)$ の元 $\xi_{m-1}\alpha^{m-1}+\cdots+\xi_1\alpha+\xi_0$ の係数を並べて $GF(p)$ 上の m 次元ベクトル $(\xi_{m-1},\ldots,\xi_1,\xi_0)$ とし，(多項式基底による) ベクトル表現と呼ぶ．$GF(p^m)$ の元の加算，減算は係数どうしの加算，減算として演算できる．また，$GF(p^m)$ の元の乗算は，ベクトル表現が多項式基底による展開の係数をであることから，多項式として計算できる．

このことから，計算機における各ビットをベクトル表現の係数と対応させれば，加算は各ビットの排他的論理和になる．

$a=(a_{m-1},\ldots,a_0)$ と $b=(b_{m-1},\ldots,b_0)$ との加算は

$$c_i = a_i \oplus b_i (i=0,1,\ldots,m-1) \tag{3.14}$$

となる．ここで，$c=(c_{m-1},\ldots,c_0)$ である．

乗算は

$$d_j = \sum_{k=0}^{i} a_k \cdot b_{m-k-1} (j=0,1,\ldots,2m-1)$$
$$c = d \pmod{P(x)} \tag{3.15}$$

となる．ここで，$c=(c_{m-1},\ldots,c_0)$ で $d=(d_{2m-1},\ldots,d_0)$ である．

第 4 章

秘密分散法を用いた分散データ管理方式

広域災害によるデータの損失を防ぐことを目的として，RAID と SAN を組み合わせた方法やリモートコピーと NAS を組み合わせた方法が実用化されている．しかし，RAID やリモートコピーを用いて作成した分散データでは，十分な秘匿性と対災害性を同時に保証することができない．

そこで，十分な秘匿性と対災害性を保証したデータ保管を行うために，秘密分散法を用いた分散データ管理方式を提案する．本章では，はじめに秘密分散法を用いた分散データ管理方式について述べる．次に提案方式と既存の方式により作成した分散データについて比較を行い，提案方式が分散保管に適していることを述べる．そして，提案方式を実現するためにネットワーク転送を考慮した実装の検討を行う．

4.1 概要

広域災害によるデータ損失を防ぐために，RAID やリモートコピーを用いて分散データを作成し，リモートホストに分散保管する方法が行われている．本質的に広域災害によるデータの損失を防ぐには，冗長性をもった分散データを作成し，距離が離れた複数のリモートホストに保管するしかない．このため，データの重要性が高くなるほど，より多くの分散データを作成して保管したいという要望がある．

しかし，分散データを保管するリモートホストが増えるほど不正アクセスの機会が増大

し，情報漏洩の可能性も増加する．したがって，情報漏洩の可能性を減らすために，分散データ数を減らすことが必要になり，対災害性と矛盾する．

そこで，秘密分散法を用いて分散データを作成する分散データ管理方式を提案する．提案する分散データ管理方式は，ローカルホストで (k,n) しきい値秘密分散法により n 個 のシェアを作成し，作成したシェアをネットワークを介して n 個の場所で分散保管する．データを読み出すときは，シェアを k 個 集めて (k,n) しきい値秘密分散法の復号化を行いデータを得る．

提案する分散データ管理方式のモデルを図 4.1 に示す．図の (a) はデータを分散暗号化して分散保管するときのデータの流れを，(b) はデータを復号化して読み出すときのデータの流れをそれぞれ示している．

提案方式では，(k,n) しきい値秘密分散法を用いて作成したシェアをそれぞれ物理的に離れた場所に分散保管することにより，広域災害などでいくつかのシェアが損失しても残っているシェアから元のデータを復号することができる．よって多くのシェアを作成することにより，データ損失による被害を減らすことができる．また，シェアの数が増えるとシェアの管理が難しくなると同時にシェアへの不正アクセスの可能性が増加する．しかし，提案方式では秘密分散法を用いているため，1 つ 1 つのシェアから元のデータは全くわからないため，安全である．

4.2　提案方式の分散保管手順

提案方式における分散保管の手順は次のようになる．

1. 保管したいデータに対してブロック化を行う．
2. ブロック毎に (k,n) しきい値秘密分散法を適用し，シェアを作成する．
3. 作成された n 個のシェアをそれぞれの場所に分散転送する．
4. 保管場所でシェアを保管する．

データを取り出す時の手順は次のようになる．

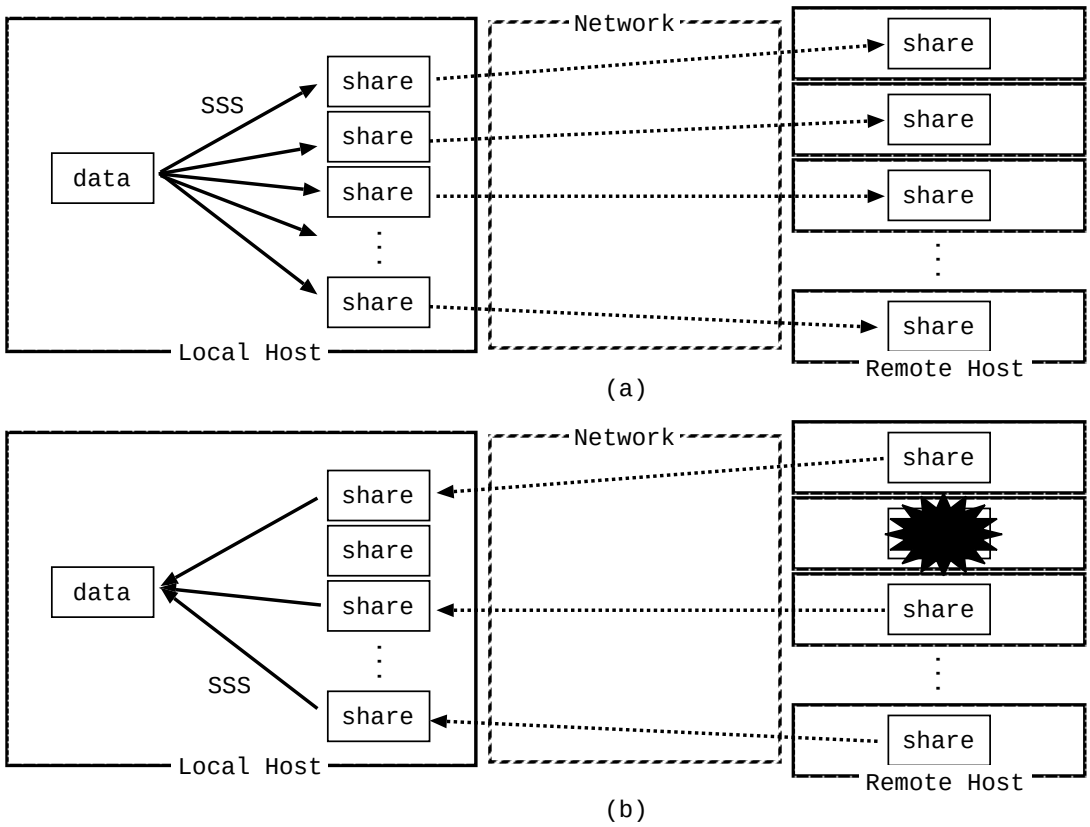

図 4.1　提案する分散データ管理方式
Fig.4.1 Distribution data management system.

1. 復号化に使用するシェアを選択する.
2. 保管場所からシェアを転送する.
3. シェアの損失などにより転送できない場合もある．そこで，k 個集まるまで 1，2 を繰り返す.
4. (k, n) しきい値秘密分散法を用いてデータを復号化する.

シェアが損失したときの復旧手順は次のようになる.

1. 残っているシェアを選択する.
2. シェアを k 個転送する.
3. (k, n) しきい値秘密分散法を用いてデータを復号化する.
4. (k, n) しきい値秘密分散法を用いて新たにシェアを作りなおす.

5. 作成された n 個のシェアをそれぞれの保管場所に分散する．

4.3　対災害性と秘匿性

既存のストレージ技術における分散データの作成は，RAID5 に代表されるデータ分割する方式と，バックアップのようにデータ複製を行う方式が主流である．そこで本節では，秘密分散法で作成されるシェアと RAID5，バックアップで作成される分散データの比較を行う．

4.3.1　対災害性

ここでは，分散データ数と冗長分散データ数から対災害性能の比較を行う．

データ分割方式やデータ複製方式では，遠隔地の保管場所で分散データを盗聴から守るために暗号化を行う必要がある．したがって，復号化を行う鍵も分散データの 1 つとして考える．提案方式と RAID5 とバックアップにおける分散データの作成方法と分散データ数を表 4.1 に示す．

提案方式は (k, n) しきい値秘密分散法を用いているので，分散データの数は n である．RAID5 は，データ分割により任意の個数 x の分散データと 1 つのパリティ，そして，暗号化で 1 個分散データの個数が増えるので合計で $x + 1 + 1$ になる．バックアップでは，データ複製により任意の個数 y の分散データを作成でき，暗号化で 1 つ分散データの個数が増えるので合計で $y + 1$ になる．

表 4.1　分散データ作成方法と分散データ数
Table 4.1 Distribution data makeing scheme and Distribution data number.

	提案方式	RAID5	バックアップ
分散データ作成	秘密分散法	データ分割+パリティ+暗号化	データ複製+暗号化
分散データ数	n	$x + 1 + 1$	y+1

(n, x, y は任意の整数)

次に冗長分散データ数を比較する．分散データの冗長性が増えるとデータのサイズが増加するので，同時にサイズも比較する．ただし，RAID5 とバックアップで暗号の鍵が損失すると元に戻らなくなるので，その場合は除いて考える．

提案方式の冗長分散データ数は，(k, n) しきい値秘密分散法を用いるときのパラメータである n と k を用いて $n-k$ $(k \geq 2)$ と表せる．RAID5 は分散データの 1 つにパリティを持つため，冗長分散データ数は 1 である．バックアップでは分散データのうちの 1 つでも残っていればいいので $y-1$ となる．また，提案方式のシェアの合計サイズはほぼ n 倍である．RAID5 の分散データの合計サイズはパリティのデータサイズだけ増加する．バックアップの分散データの合計サイズはほぼ y 倍である．

表 4.2 に冗長分散データ数と分散データの合計サイズを示す．表 4.2 で s は元のデータのサイズを示している．また，α はデータの復元に必要な情報，β と γ は復号化を行う鍵のサイズであり，元のデータに対して $\alpha \ll s, \beta \ll s, \gamma \ll s$ が成り立つのでサイズの比較では無視できるとする．

表 4.2　冗長分散データの数
Table 4.2 Number of redundant distribution data.

	提案方式	RAID5	バックアップ
冗長分散データ数	$n-k$	1	$y-1$
分散データ合計サイズ	$ns+\alpha$	$s+\frac{s}{y}+\beta$	$ys+\gamma$

表 4.1 と表 4.2 の比較の結果より，バックアップが最も広域災害に強いことが分かる．そして，提案方式の冗長データ数はバックアップのデータ数と同じぐらいに設定できるので，バックアップと同じぐらいの性能であるといえる．

4.3.2　秘匿性

分散データには解読の危険がある．そこで，解読に必要な分散データの最小数と最大数を用いて秘匿性の比較を行う．また，暗号化を行う速度を含めた全体の処理速度の比較を行

う．ただし，RAID5 とバックアップに適用する暗号は理想的な暗号で，鍵を用いないと復号化できないとする．重要なデータは一部分でも解読されてはいけないので，一部分でも解読されたら情報が洩れたらと判定する．

表 4.3 にそれぞれの方式で解読に必要な分散データの最小数と最大数処理速度を示す．

表 4.3　安全性と処理速度の比較
Table 4.3 Compare of security and throughput.

	提案方式	RAID5	バックアップ
解読最小数	k	2	2
解読最大数	k	$x+1+1$	$y+1$
処理速度	遅い	速い	中

提案方式では，情報理論的に安全でそれぞれのシェアから元のデータは全くわからない．最小数，最大数は $2 \leq k \leq n$ の範囲で任意に決定できる k になる．

RAID5 において解読に必要な最小数の分散データのパターンは，復号化を行う鍵と分割データのどれか 1 つなので最小数は 2 になる．また，最大数は分割データとパリティデータをすべて集めて，最後に鍵が集まる場合なので $x+1+1$ になる．

バックアップでは RAID5 の場合と同様のパターンになるので，最小数は 2，最大数は $y+1$ となる．ただし，最小数のときも部分データだけでなく全てのデータが解読される．また，処理速度について比較すると速い順に RAID5，バックアップ，本方式となる．RAID5 がストレージに対して分割したデータの同時書き込みを行うため一番速く，提案方式は秘密分散法の演算処理に時間がかかるため一番遅くなる．

第 3 者が不正な分散データの取得を試みたとき，小数のリモートホストから分散データを取得できたとしても，多数のリモートホストから分散データを集めるのは難しい．よって，RAID5 やバックアップでは解読に必要な最小の分散データの数が 2 なので，解読される危険性は高い．一方，(k, n) しきい値秘密分散法を用いた提案方式は，任意に k を設定できるので，k を適切に設定すれば解読される危険性を 0 に近付けることができる．

4.4　ネットワークを用いた分散

対災害生を実現するには，地理的に離れた場所で分散保管することが必要になる．そこで，ネットワークを用いて分散を行うための検討を行う．

ネットワークを用いたストレージ技術に SAN と NAS がある．SAN はデバイスドライバ層の技術で，NAS はファイルシステム層の技術である．ストレージシステムの階層構造において，上位層は下位層の機能を間接的に使用できる．例えば，アプリケーション層は IP Network と Fibre Channel を使用できる．同様に，ファイルシステム層は IP Network と Fibre Channel を使用できる．ただし，ファイルシステム層は Fibre Channel に対応したプロトコルがないので，実装は困難である．表 4.4 にストレージシステムの階層構造において使用できるネットワークの種類を示す．

表 4.4　使用できるネットワーク
Table 4.4 Network of using.

layer	Network
Application	IP Network, Fibre Channel
File System	IP Network
Device Driver	Fibre Channel

そこで，提案方式を実現するレイヤとして次の 3 つを比較検討する．

- アプリケーション
- ファイルシステム
- デバイスドライバ

アプリケーションレイヤで実現する場合，Fibre Channel と IP Network におけるさまざまなファイル転送プロトコルを使うことができる．しかし，秘密分散法の適用や遠隔地への分散などのシェアの管理をユーザがコマンドで行う必要があり，保管するデータが増えるとシェアの管理は困難になる．したがって，アプリケーションレイヤではネットワークを介し

た分散は容易に行うことができる反面，シェアの管理コストは大きい．

次にファイルシステムで実現する場合，IP Network を用いることができるため，NFS などで容易に転送できる．また，もともとファイルを管理するためのレイヤであることから，秘密分散法の適用を自動化することやシェアの管理も容易である．

最後に，デバイスドライバで実現する場合，データはバイト・ストリームとして扱われるので，シェアの管理コストも小さくなる．しかし，IP Netwrok を用いることができず，Fibre Channel を用いた転送では，伝送距離が 10km までという限界があるため実現は困難である．

これらのことをまとめると表 4.5 のようになる．表 4.5 よりシェアの管理コストが少なくネットワークを介したシェアの分散が容易に行えるファイルシステムが秘密分散法を用いた本方式に最適といえる．

表 4.5　実装レイヤの比較
Table 4.5 Comparison of implementation layer.

	ネットワーク分散	管理コスト
アプリケーション	◎	×
ファイルシステム	○	○
デバイスドライバ	×	◎

高いレベルで要求事項を満たしていれば ◎

要求事項を満たしていれば ○

要求事項を満たしていないならば ×

第 5 章

分散ファイルシステムの設計

初期の UNIX オペレーティングシステムは，単一のファイルシステムにしか対応していなかった．しかし，ディスクの配置とディスク上の構造などを改良した FFS[*1]をはじめ，ネットワークを用いたファイル共有を可能とする RFS[*2]や NFS[*3]といった分散ファイルシステムが開発され，複数のファイルシステムに対応する新たなフレームワークが必要になった．そこで，ファイルシステム毎の違いを吸収する VFS(Virtual File System) を用いて，統一的なインタフェースを提供する v ノード/vfs アーキテクチャ [5] が開発されデファクトスタンダードとなった．VFS に対応したファイルシステムは，ファイルを表す v ノードと v ノードに対して読み込みなどを行う操作から構成される．これにより，ユーザからはあたかも単一のファイルシステムであるかのように複数のファイルシステムを扱うことができるようになった．

さらに，ファイルに対応する v ノードを積み重ねることにより，上位層のファイルシステムから下位層のファイルシステムの機能を利用できるスタッカブルファイルシステム [6] が開発された．スタッカブルファイルシステムを用いると，既存のファイルシステムに機能を付加したファイルシステムが容易に作成できる．そこで，提案方式のファイルシステムはスタッカブルファイルシステムを用いて UFS [*4]や NFS などの既存のファイルシステムを利用する．

本章では FreeBSD 上の分散ファイルシステムの設計と実装について述べる．はじめに実

[*1] Fast File System

[*2] Remote File Sharing

[*3] Network File System

[*4] Unix File System．この場合は VFS に対応した FFS を指す．

装するファイルシステムの構成について述べる．次に実装するファイルシステムの v ノードと v ノードの操作について述べる．そして，ファイルシステムを使用できるようにするための操作であるマウントについて述べる．

5.1　ファイルシステムの構成

提案方式に必要な機能は，秘密分散法を用いた分散暗号化/復号化機能とシェアの分散保管機能，そしてシェアの管理機能である．スタッカブルファイルシステムを用いれば，UFS や NFS を利用したシェアの分散保管が可能である．さらに，ファイルシステムが扱うデータは住民基本台帳のように基本的に 1 つの組織で管理されるとする．そのような場合，分散暗号化/復号化機能とシェアの管理機能は，同じホスト上にあればよい．そこで，分散暗号化/復号化機能とシェアの管理機能を持つファイルシステムを作成する．

ローカルホストのファイルシステム構造は，分散暗号化/復号化とシェアの管理を行うレイヤ SML (Share Management Layer) とシェアの分散保管を行うレイヤ SDL (Share Distribution Layer) となる．SML は新たに作成する SSFS (Secrect Sharing File System) で，SDL (Share Distribution Layer) は既存のファイルシステムである．

図 5.1 にローカルホストのファイルシステム構成を示す．図 5.1 は，ユーザのファイル操作はシステムコールから SSFS で分散暗号化が行われ，SDL の各ファイルシステムによって分散保管される様子を示している．

SDL のファイルシステムはシェアを分散保管するファイルシステムでそれぞれのファイルシステムがシェアと 1 対 1 に対応する．ローカルホストのファイルシステムを VFS として，SDL のファイルシステムに NFS を用いたときのリモートホストとのファイルシステム構成を図 5.2 に示す．図 5.2 ではローカルホストで分散暗号化したシェアのうち 1 個がローカルなストレージに保管され，残りの 3 つのシェアがリモートホストのストレージに保管される様子を示している．

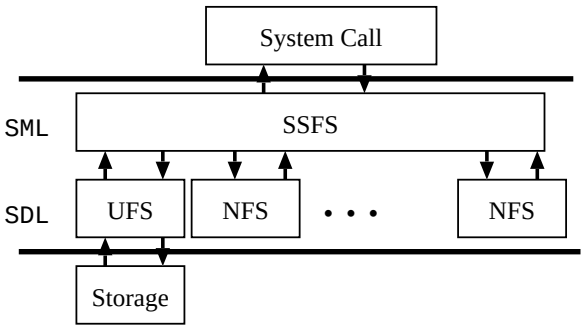

図 5.1 ローカルホストのファイルシステム

Fig. 5.1 Filesystem of local host.

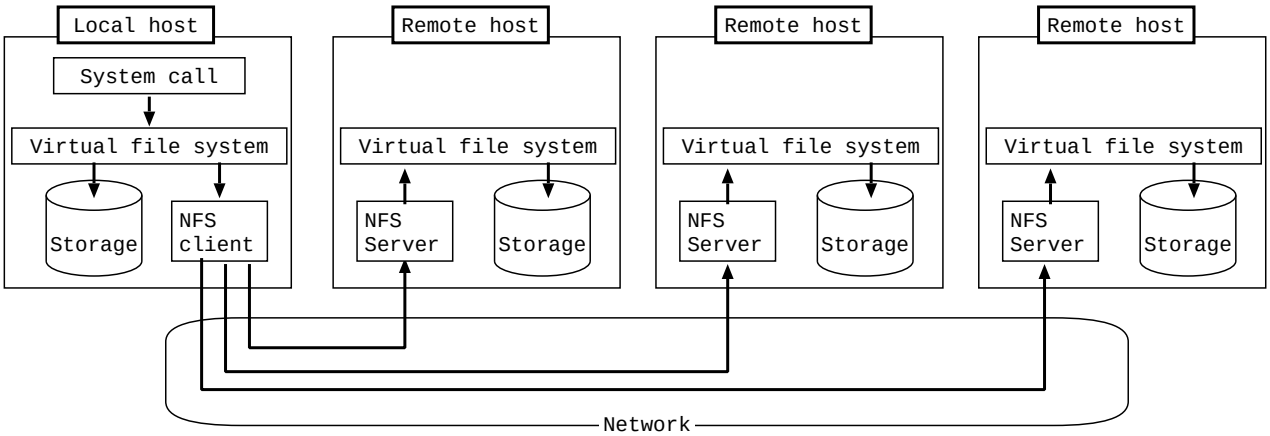

図 5.2 リモートホストを含んだファイルシステム

Fig. 5.2 File system with remote host .

5.2 SSFS の v ノード

vfs/v ノードアーキテクチャでは，すべてのファイルシステムのファイルは，v ノードによって表現される．v ノードはプライベートデータ領域にファイルシステムに依存したデータを持つことにより，ファイルシステム毎に異なったファイルを表現する．UFS の場合のプライベートデータは i ノードであり，NFS の場合はファイルハンドルである．そして，スタッカブルファイルシステムを用いている場合のファイルは，スタックに積まれた各ファイルシステムで 1 つずつの v ノードから構成される．

ローカルホストのファイルシステムは SML に SSFS，SDL に複数のファイルシステムか

らなるので，図 5.3 のようになる．図 5.3 は SDL が UFS と NFS のときの v ノードのスタック構成を示している．

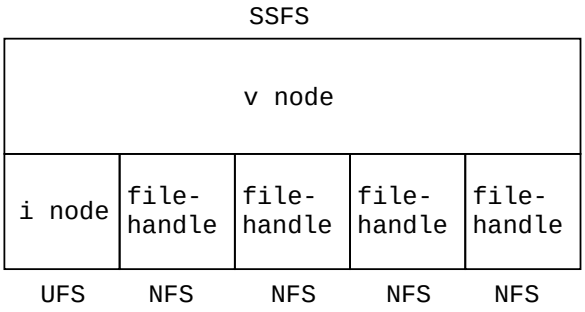

図 5.3　SML と SDL の v ノード

Fig. 5.3 v node of SML and SDL.

5.3　SSFS の v ノード操作

v ノード/vfs アーキテクチャでは v ノードに対する操作はあらかじめ決められている．例えば，FreeBSD による実装では read や write，rename などをはじめてとして，54 個の操作がある．そして，スタッカブルファイルシステムを用いる場合は，v ノードのスタック毎に，次の 2 つのどちらか 1 つを実行する．

- v ノードは操作を実行し，結果を呼び出し側に返す．
- 何らかの処理を行い，スタックの下層の v ノードへ処理を渡す．

これにより，ファイルによる処理は上から下へ各層で行われる．

本節では SSFS に必要な v ノードの操作を述べる．

5.3.1　v ノードの操作

表 5.1 は SSFS に必要な操作とその簡単な説明を行っている．

SSFS の機能は，分散暗号化/復号化機能とシェアの管理機能である．分散暗号化/復号化

表 5.1 SSFS の v ノードの主な操作
Table 5.1 v node operation.

操作	説明
bypass	SSFS では何もしないで SDL に操作をスルーする
access	SDL の v ノードにアクセス許可があるかどうか調べる
close	SDL のファイルを閉じる
create	SDL にファイルを作る
getattr	SDL のファイルの属性を取得する
lock	v ノードをロックする
lookup	パス名解決を行う
mkdir	SDL にディレクトリを作る
open	SDL のファイルを開く
read	SDL のファイルを読み込む
readir	SDL のディレクトリを読み込む
remove	SDL のファイルを消す
rename	SDL のファイル名を変える
rmdir	SDL のディレクトリを消す
setattr	SDL のファイルの属性を設定する
unlock	v ノードのロックを解除する
write	分散暗号化を行って，シェアをファイルに書き込む

はファイルのデータに行うので，read と write 操作を用いてできる．SSFS のシェアの管理を行う操作は，基本的に SDL の v ノードに対する操作を用いてできる．そのため，SSFS のシェア管理では，SDL の複数の v ノードに処理を渡すだけである．図 5.4 に SML の v ノードから SDL の v ノードに処理を渡す様子を示す．図 5.4 では，SSFS の v ノードに setattr 操作が行われたとき，UFS や NFS に setattr 操作を渡す様子を示している．

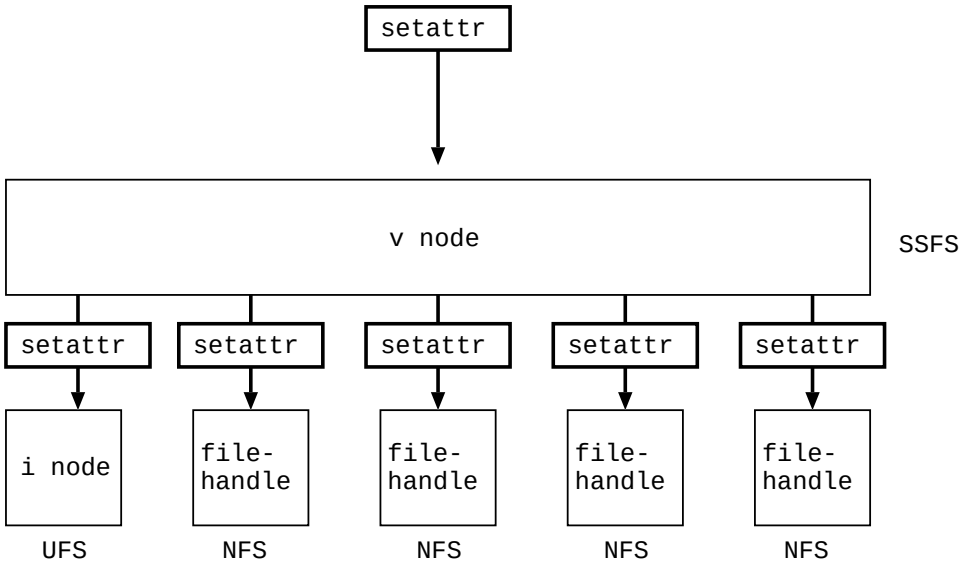

図 5.4 SSFS の v ノード操作 1

Fig. 5.4 v node operation of SSFS 1.

しかし，複数回行う必要のない操作もある．例えば，ファイルのサイズを調べるための getattr 操作は，シェアのファイルはどれも同じ大きさなので，わざわざ SDL の v ノードを全て調べなくてもよい．そこで，このような種類の操作の場合，後述するマウントのときに決定するメインシェアファイルシステムの v ノードに対してだけ行う．

図 5.5 に SML の v ノードから SDL の v ノードのうち 1 つに処理を渡す様子を示す．図 5.5 では，SSFS の v ノードに getattr 操作が行われたとき，UFS にだけ getattr 操作を渡す様子を示している．

5.3.2 読み込みと書き込み

SSFS は read 操作及び write 操作のときに秘密分散法によって分散暗号化及び復号化する．秘密分散法の処理は一定サイズのデータに対して行うので，これを秘密分散法のブロック処理と呼ぶ．図 5.6 に秘密分散法のブロック処理の様子を示す．図 5.6 はファイルに対して start から end までの長さの要求があったときに，SSS のブロック化した断片を加えて，ファイルを処理する様子を示している．

5.3　SSFS の v ノード操作

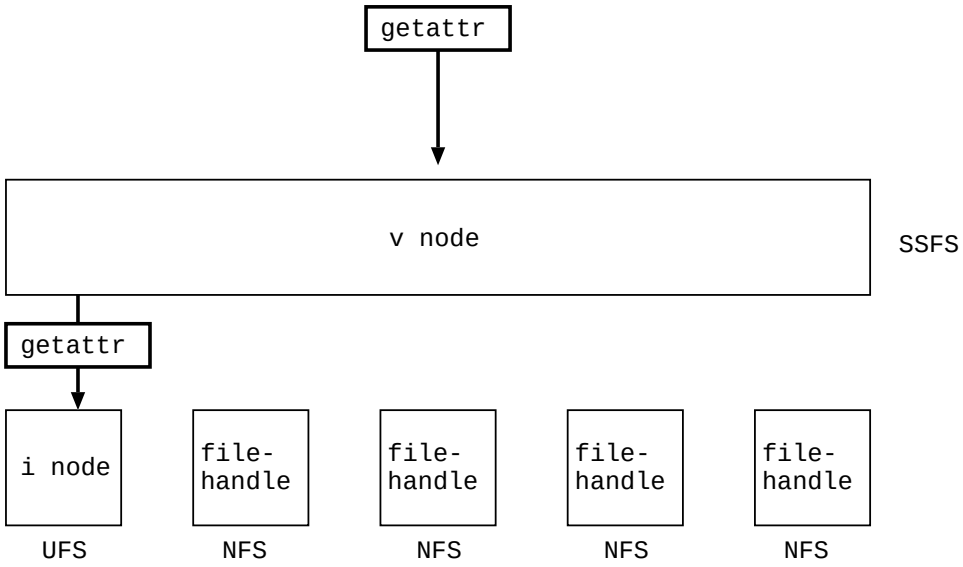

図 5.5　SSFS の v ノード操作 2

Fig. 5.5 v node operation of SSFS 2.

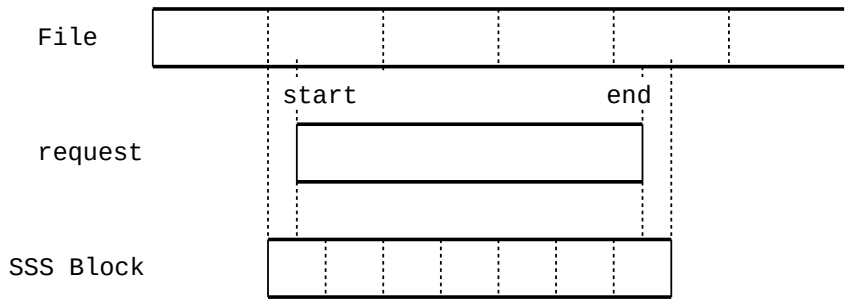

図 5.6　秘密分散法のブロック化

Fig. 5.6 SSS Blcok.

まず，write のアルゴリズムを述べる．

1. 秘密分散法のブロック処理

   書き込むデータが秘密分散法のブロック長の倍数になっているか調べる．

   (a) 倍数になっていれば 2 へ．

   (b) 倍数になっていなければ，ブロックにあうように SDL の k 個の v ノードから秘密分散法の断片部分を読み出して，復号化し，データに付け足す．

2. 分散暗号化を行いシェアを作成する．
3. シェアを n 個の下層の v ノードに対して書き込み操作を行う．

読み込みでも書き込みと同様に秘密分散法のブロック処理を行う．また，読み込みの際，秘密分散法の復号化に用いるシェアによって式 (3.9) に示す行列が変化する．このシェアの組み合わせをシェアのパターンと呼ぶ．秘密分散法の復号化で計算量的に最も効率的なのは，事前に逆行列を求めておく方法である．しかし，逆行列はあるシェアのパターンに依存した値なので，シェアのパターンが変化すると逆行列を再度求める必要がある．そのため，読み込みのアルゴリズムでは，最初にあるシェアのパターンについて計算しておき，シェアのパターンが変わればその都度，求め直すようにする．読み込みのアルゴリズムは次のようになる．

1. 秘密分散法のブロック処理判定

   読み込むデータが秘密分散法のブロック長の倍数になっているか調べる．倍数になっていなければ，ブロックに合うようにデータを大きくする．
2. SDL の v ノードで k 個読めるまで読み込を行う．
3. 読み込みに使用した v ノードのパターンを判定する．
   (a) シェアのパターンが事前に求めておいた逆行列のパターンならば，逆行列を用いて復号化する．
   (b) シェアのパターンが違う場合，データのサイズを判定
      i. データのサイズが秘密分散法のブロック長より小さいなら，消去法を用いて復号化する．
      ii. データのサイズが秘密分散法のブロック長より大きいなら，逆行列を作成し，その逆行列を用いて復号化する．

ここで，秘密分散法のブロック処理によって発生する磁気ディスクのオーバーヘッドについて検討する．

SDL のファイルシステムは，一般的に固定長セクタの磁気ディスクにデータを書き込む．

磁気ディスクはセクタの倍数でしか読み書きできないので，秘密分散法によるブロック処理によっては，余分なオーバーヘッドが発生する可能性がある．

磁気ディスクはユーザが単一のバイトを読み書きしたいとしても，セクタの倍数でしか読み書きできない．そこで，ファイルシステムは，更新すべきバイトを含むセクタを読み込み，そのバイトを置き換え，ディスクに書き戻すように動作する．図 5.7 にユーザによるファイルアクセスに必要なブロック処理を示す．図 5.7 では，ファイルを読み込むとき，システムバッファによりブロック化されて，最終的に論理ブロックの大きさで磁気ディスクにアクセスされる様子を示している．

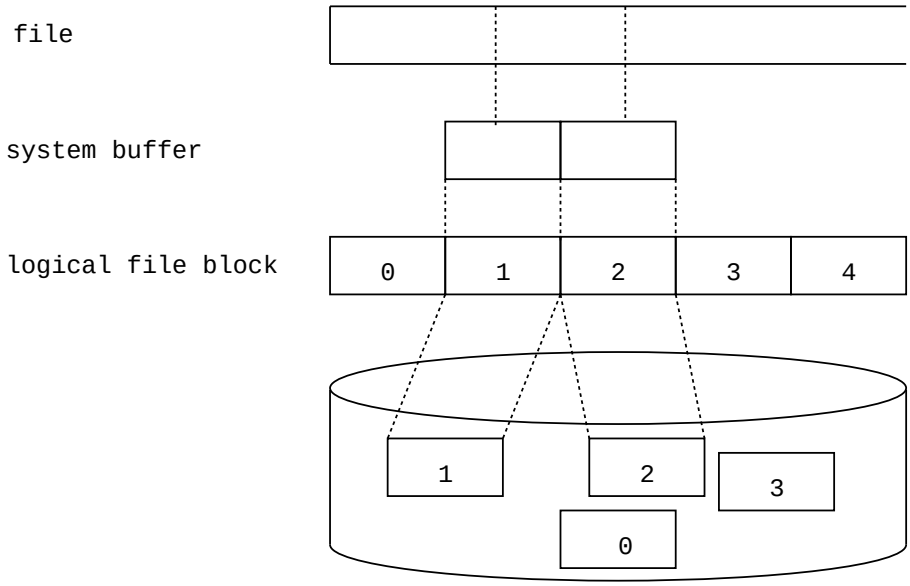

図 5.7 ファイルシステムブロック

Fig. 5.7 Filesystem block.

秘密分散法に用いるデータはブロック化しなければならない．そのため，分散暗号化/復号化するときにデータのブロック処理を行う．図 5.8 に秘密分散法のブロック処理を示す．図 5.8 は秘密分散法のブロックより大きいブロックのシステムバッファがあり，そのバッファにしたがって論理ブロックの処理が行われて，最終的にストレージをブロック毎に処理していることを示している．

よって秘密分散法によるブロック処理が論理ブロックを越えなければ，ストレージデバイ

スに対する余分な読み書きのオーバヘッドと保管領域のロスは発生しない.

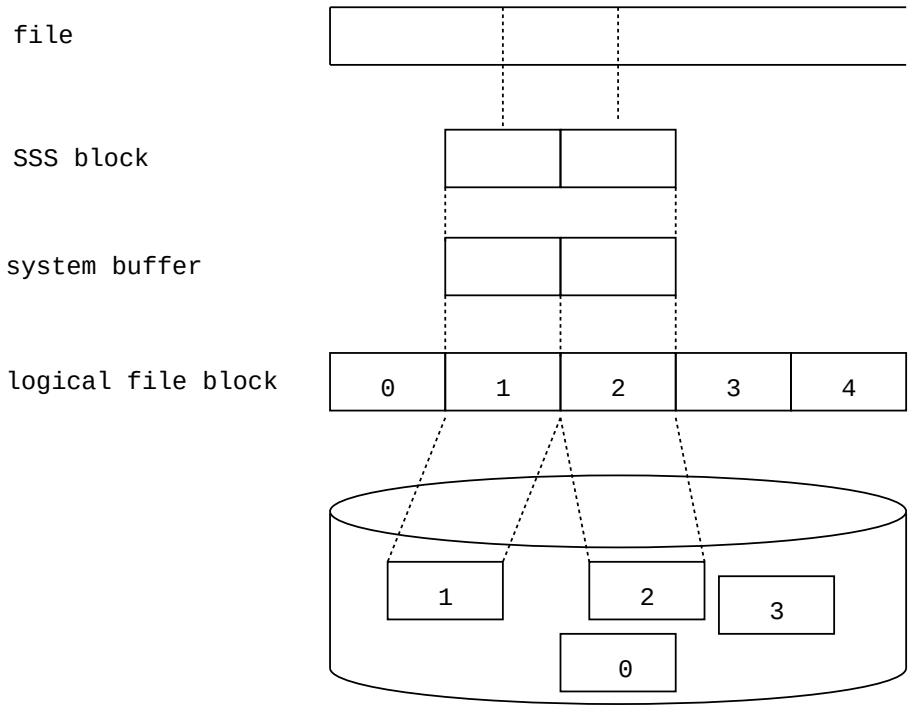

図 5.8 秘密分散法のブロック処理

Fig. 5.8 SSS block.

5.3.3 パス名解決のキャッシュ

ファイル名を v ノードに変換する操作は一般的にパス名解決と呼ばれ，SSFS では lookup 操作で行われる．lookup 操作では，他の SSFS のシェア管理操作と同じように SDL の v ノードに lookup 操作を行う．ただし，ファイルシステムにおける lookup 操作では，ファイルシステム内の v ノードが同じファイルに対して 2 つ以上同時に割りあたらないようにように，ファイルシステムに用意されたキャッシュを用いる．もし，キャッシュを調べてヒットしたとすれば，すでにそのファイルを指す v ノードがあるということなので，v ノードの参照カウントと呼ばれる値をインクリメントする.

SSFS では，時間短縮のために，SDL の v ノードのうち 1 つに lookup 操作を行い，その後にキャッシュを参照することで，時間短縮を計っている.

SSFS の lookup 操作のアルゴリズムを次に示す.

1. メインシェアファイルシステムの v ノードに lookup を試みる.
   (a) できれば 2 へ.
   (b) できなければエラーを返して終了.
2. メインシェアファイルシステムの v ノードを key にしてキャッシュを探索.
   (a) ヒットすれば，参照カウントをインクリメントし正常に終了.
   (b) ヒットしなければ，SSFS の新しい v ノードを取得して 3 へ.
3. メインシェアファイルシステム以外の SDL のファイルシステムに対して名前解決を行う.
   (a) 名前解決ができたファイルシステムが k 個以下の場合エラーを返す.
   (b) それ以外の場合は正常終了.

5.4　シェアの再作成

SML では，SDL のファイルシステムに障害が起こると，ファイルシステムのエラーとして検知する．そして，ファイルシステムの障害がなくなれば，新たにシェアを再作成しなければならない．シェアを再作成するには，残っているシェアからデータを読みこんで復号化したあと，分散暗号化して書き込む.

書き込みのアルゴリズムを示す.

1　シェアを保管している SDL のファイルシステムを選択し，シェアの個数が k 個になるまで SDL から読み込みを行う.

2　シェアのパターンを判定する.

2.1　保存しておいたシェアのパターンと同じなら逆行列を用いて復号する.

2.2　シェアのパターンが違う場合，データのサイズを判定

2.2.1　データのサイズがブロック長より小さいなら，消去法を用いて復号する.

2.2.2　データのサイズがブロック長より大きいなら，逆行列を作成し，その逆行

列を用いて復号する．

3 シェアを書き込む

5.5 マウント

ファイルシステムを使用するにはマウントを行わなければならない．スタッカブルファイルシステム用いたファイルシステムのマウントは，SDL のファイルシステムをマウントしたディレクトリの v ノードとマウントポイントを指定する．mount システムコールが実行されるとカーネルはファイルシステムを表現する vfs オブジェクトを作成し，カーネル内の管理リストに追加する．図 5.9 に FreeBSD の vfs オブジェクトの管理リストを示す．この図 5.9 では，マウントを行うと右にリストが増えて，アンマウントを行うと右からリストが減っていく．

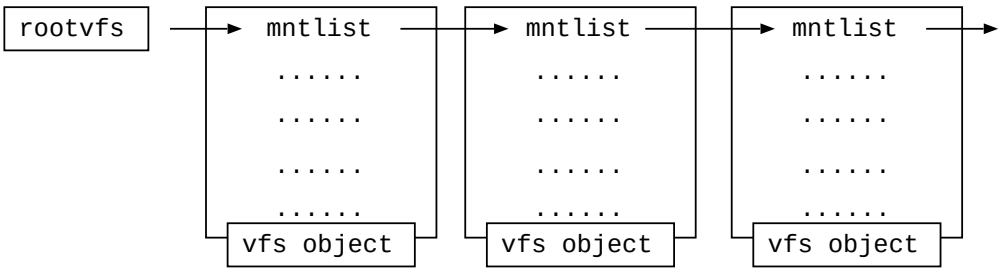

図 5.9 vfs オブジェクトとマウント

Fig. 5.9 vfs object and mount.

マウントは，1 つのルートノードとマウントポイントを 1 対 1 に対応させる．しかし，SSFS は SDL の複数のファイルシステムが対応するため，複数のルートノードをマウントポイントに対応させなければならない．このような構造を実現するための自然な拡張として，ネットワーク RAID [7] のように同一のマウントポイントに多重マウントを行う方法がある．同一のマウントポイントへの多重マウントは，カーネルが vfs オブジェクトのリストをマウント時の順番で管理していることで可能となっている．しかし，多重マウントを行った場合，必ずマウントの逆順にアンマウントを行なわなければならないため，障害が発生し

たときに問題がある．

説明のために同一のマウントポイントに次の順番でファイルシステムのルートノードを多重マウントする場合を考える．

1.UFS　2.NFS　3.UFS　4.NFS　5.NFS

このときマウントを行う順番に vfs オブジェクトが割り当てられ，vfs オブジェクトはマウント時の順番でリスト構造を作っている．例えば，4 のファイルシステムをアンマウントする場合は，事前に 5 のファイルシステムをアンマウントしなければならない．同様に 1 のファイルシステムをアンマウントするには，2〜5 のファイルシステムをアンマウントしなければならない．ここで，1 の UFS でマウントしているストレージに障害が発生したとする．UFS でマウントしているストレージを交換するためには，ファイルシステムをアンマウントしなければならない．そのため，2 から 5 のファイルシステムをアンマウントしなければならず，運用を継続できない．

そこで，SSFS のマウントに 1 つの vfs オブジェクトだけを割り当てるように設計する．SDL のファイルシステムの 1 つをメインシェアファイルシステムとし，通常のマウントを行って vfs オブジェクトを作成する．それ以外のファイルシステムは，メインシェアファイルシステムのマウントによって作成される vfs オブジェクトの中に SDL の v ノードだけを持つようにする．SSFS のアンマウントを行うには，メインシェアファイルをアンマウントする．これにより，前述のような障害が発生しても対応できるようになる．

第 6 章

評価

本章では，作成したファイルシステムの処理量について述べる．そして，実際の処理時間を計測し考察を行う．

6.1 SSFS の処理量

SSFS のパス名解決操作，書き込み操作，読み込み操作の理想的な処理量を示す．ここで，n と k は (k, n) しきい値秘密分散法のパラメータである．

L を SDL のファイルシステムによるパス名解決の実行時間，そして，キャッシュにヒットしなかったときのキャッシュの検索時間を H とすると，SSFS のパス名解決の実行時間は

$$H + \sum_{i=0}^{n} L_i \tag{6.1}$$

になる．

分散暗号化の実行時間を E，復号化する実行時間を D，逆行列を計算する復号化する処理時間を I とする．SDL のファイルシステムに書き込みを行う実行時間を W とする．SDL のファイルシステムに読み込みを行う実行時間を R とする．ファイルのサイズが秘密分散法のブロック処理より大きいときは，分散暗号化/と復号化を複数回行うので，$\sum$ で表す．

書き込みの操作は，SSFS の分散暗号化処理と SDL への n 回の書き込みで，秘密分散法のブロック処理による読み込みが発生する場合があるので，

$$\max\left(\sum_{i=0}^{k} R_i + 2D + \sum E, \sum E\right) + \sum_{i=0}^{n} W_i \tag{6.2}$$

である．

SSFS の読み込み操作は，SDL のファイルシステムから読み込みを行い，復号化する．シェアの損失が $n-k$ 個であった場合，SDL すべてのファイルシステムでシェアを読み込もうと試み，逆行列を再作成する．

復号化処理の実行時間は，

$$\max\left(\sum_{i=0}^{n} R_i + I, \sum_{i=0}^{k} R_i\right) + \sum D \tag{6.3}$$

になる．

6.2 実行速度

実装したファイルシステムを用いて実行速度を測定した．

6.2.1 測定環境

測定に用いた環境を次に示す．

OS FreeBSD 5-CURRENT (2002/10/24)

Compiler gcc version 3.2.1

CPU PentiumII 400MHz

HDD IDE ATA-33

測定に用いたデータは，1024Byte，2048Byte，4096Byte，8192Byte，16384Byte，32768Byte，65536Byte，262144Byte，524288Byte，1048576Byte である．

測定は読み込み書き込みとも 11 回の処理速度を計測し，1 回目の処理時間は，v ノードを作成する時間などが入っているため読み捨てる．そして，2 回目から 11 回目の処理速度の平均を取る．

6.2.2　結果

秘密分散法のブロック長をそれぞれ 32Byte，16Byte，8Byte，4Byte として分散暗号化を行ったときのファイルシステムの実行速度と秘密分散法による分散暗号化処理をしないときのファイルシステムの実行速度の値を表 6.1 に示し，値のグラフを図 6.1 に示す．表の値は秒単位である．

表 6.1　ブロック長ごとの分散暗号化の処理時間
Table 6.1 Transaction time of encrypt.

File Size	no encrpt	4 Byte	8 Byte	16 Byte	32 Byte
1024	0.000471	0.021820	0.027888	0.050207	0.079163
2048	0.000582	0.043141	0.055243	0.099707	0.157658
4096	0.000832	0.085605	0.110139	0.198693	0.314633
8192	0.001251	0.168871	0.217547	0.395168	0.626568
16384	0.002413	0.334652	0.432710	0.786705	1.270568
32768	0.004562	0.666645	0.861753	1.574336	2.495246
65536	0.009848	1.333856	1.724356	3.137772	4.986578
131072	0.017878	2.669944	3.452634	6.269262	9.971790
262144	0.118104	5.420093	6.976395	12.618760	20.005109
524288	0.357706	10.945579	14.101171	25.321250	40.127101
1048576	0.780067	21.961246	28.291778	50.754147	80.331774

読み込み時にシェアのパターンが変わる場合，逆行列を再計算してから復号化する．固定パターンのときは逆行列を用いて復号化する．それぞれの場合のファイルシステムの実行速度の値を表 6.2 に示し，値のグラフを図 6.2 に示す．表の値は秒単位である．

秘密分散法のブロック長をそれぞれ 32Byte，16Byte，8Byte，4Byte として逆行列を計算して復号化したときの実行速度と秘密分散法による復号化処理をしないときの実行速度の

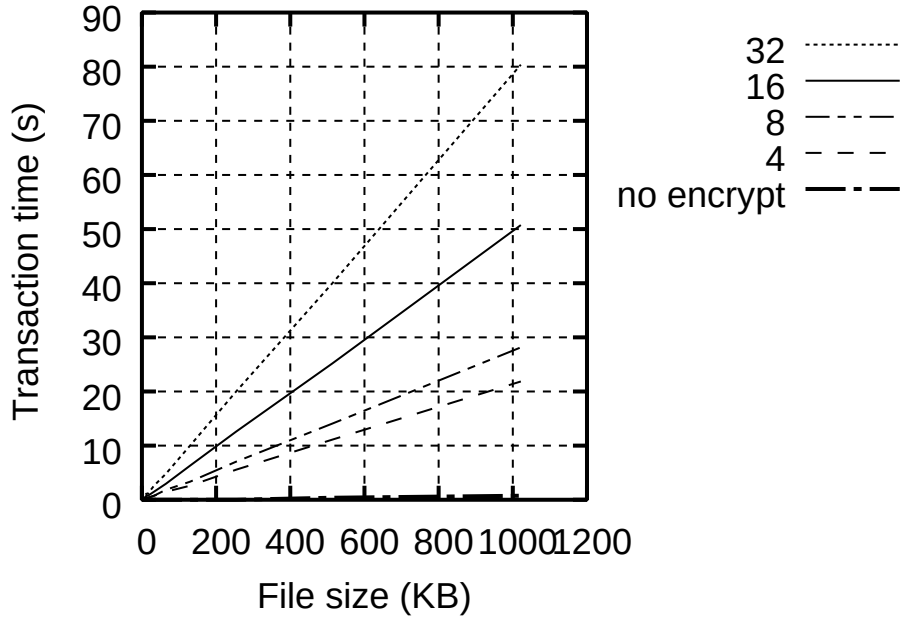

図 6.1 ブロック長ごとの分散暗号化の処理時間

Fig. 6.1 Transaction time of encrypt.

比較を表 6.3 に示し，値のグラフを図 6.3 に示す．表の値は秒単位である．

また，各ブロック長でのスループットを図 6.4 に示す．

6.3 考察

読み込みの速度について考察する．式 (6.3) で理論的な処理速度が遅いのは，逆行列を求める場合である．

しかし，図 6.2 より復号化に用いるシェアのパターンの違いによる処理速度の変化はほんどない．よって，シェアのパターンが変わって行列の演算の計算量が多くなっても，実際には問題にならない結果となった．

式 (6.2)，(6.3) の最小値で，SDL のファイルシステムに対する書き込みと読み込みの違いはあるものの，操作を行う SDL のファイルシステムの数は同じである．図 6.1 と図 6.3 を比較すると書き込み時の方が大幅に時間がかかっている．したがって，秘密分散法の処理時間がボトルネックになっていると予測できる．

表 6.2 シェアのパターンが動的な場合と静的な場合の復号化の処理時間
Table 6.2 Transaction time of dynamic and static inverse matrix.

File Size	Dynamic	Static
1024	0.063867	0.063937
2048	0.125029	0.125028
4096	0.249370	0.246001
8192	0.450452	0.445090
16384	0.831375	0.829050
32768	1.599722	1.596746
65536	3.132511	3.133318
131072	6.198174	6.195464
262144	12.348280	12.339915
524288	24.607520	24.608510
1048576	49.153362	49.137892

また，図 6.1 と図 6.3 より，ファイルサイズが 1MB のときの各ブロック長の処理時間をシェアの数 5 で割ると，一つのシェアにかかる処理時間が分かり，表 6.4 になる．

よって，1MB のときの処理時間をブロック長に依存した定数 a と，シェアの数 n を用いて，本ファイルシステムの処理時間 T は次のように近似できる．

$$T = a \times n \quad (s). \tag{6.4}$$

例えば，ブロックサイズが 32Byte のときの復号化では，$a = 16$，$n = 3$ より処理時間は約 48 秒である．

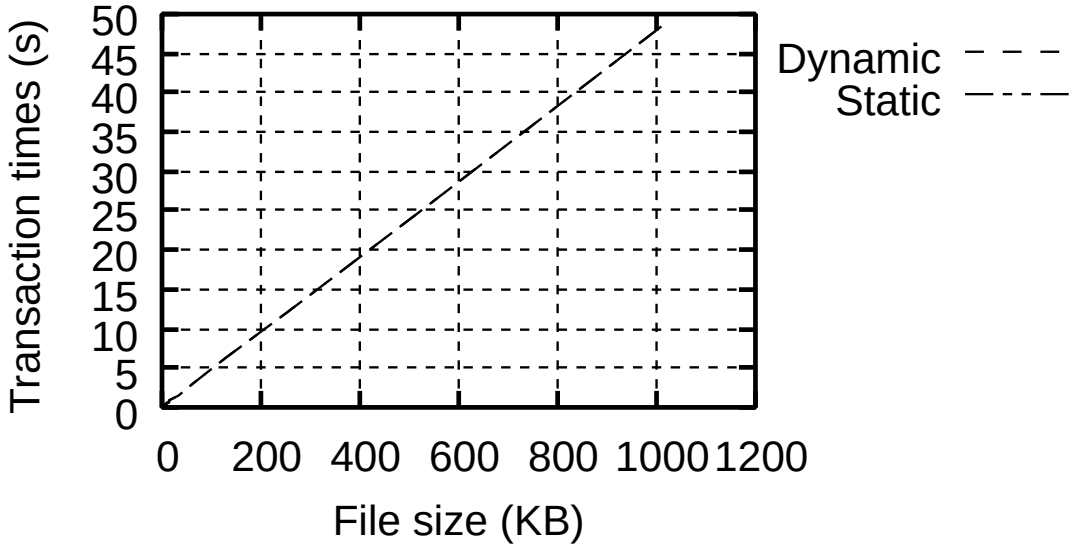

図 6.2 シェアのパターンが動的な場合と静的な場合の復号化の処理時間
Fig. 6.2 Transaction time of dynamic and static inverse matrix.

表 6.3 逆行列を計算した場合のブロック長ごとの復号化の処理時間
Table 6.3 Transaction time of decrypt with dynamic inverse.

File Size	no encrpt	4 Byte	8 Byte	16 Byte	32 Byte
1024	0.000405	0.015836	0.021137	0.035539	0.063867
2048	0.000487	0.030277	0.040767	0.069592	0.125029
4096	0.000724	0.058903	0.079509	0.137799	0.249370
8192	0.001105	0.113953	0.149238	0.259138	0.450452
16384	0.006486	0.222639	0.285642	0.497430	0.831375
32768	0.003881	0.439951	0.558500	0.979805	1.599722
65536	0.008251	0.875669	1.105234	1.925645	3.132511
131072	0.012511	1.746037	2.192171	3.827149	6.198174
262144	0.025522	3.478191	4.368258	7.626717	12.348280
524288	0.049790	6.965234	8.721424	15.246322	24.607520
1048576	0.098752	13.892446	17.434554	30.453958	49.153362

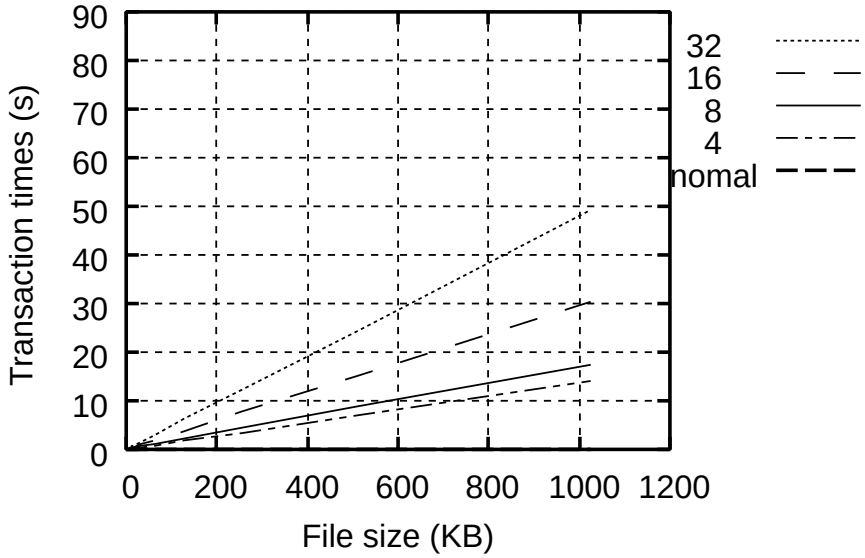

図 6.3 逆行列を計算した場合のブロック長ごとの復号化の処理時間

Fig. 6.3 Transaction time of decrypt with dynamic inverse.

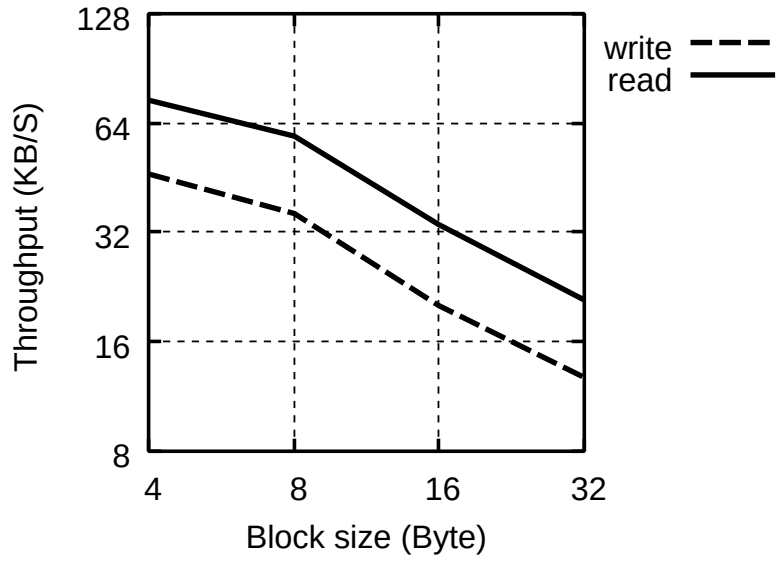

図 6.4 ブロック長毎のスループット

Fig. 6.4 Throughput by blocksize.

表 6.4 1 つのシェアに対する処理時間
Table 6.4 The slope for a share.

	4	8	16	32
slope	4.4	5.6	10	16

第 7 章

まとめ

本論文では秘密分散法を用いた分散データ管理方式を提案し，ファイルシステムによる実装を行った．その結果，秘密分散法のブロック長が小さいときに実用的な範囲で動作するファイルシステムを実現した．

秘密分散法を用いた分散データ管理方式を提案し，既存の方式と比較を行い提案方式の有効性を明らかにした．また，提案方式の最適な実装を検討し，ファイルシステムによる実装が最適という結果を得た．提案方式を実現するためにスタッカブルファイルシステムによるファイルシステムの設計と実装を行った．v ノードの構成と v ノードの操作，マウントの設計を行った．また，秘密分散法のブロック処理における検討を行った．そして，実装したファイルシステムの実行速度を測定した．秘密分散法のブロック長が小さいときに，実用的な範囲で使用できるという結果を得た．

以下では今後の課題について述べる．

秘密分散法の実装では，演算を $GF(2^n)$ 上で実行している．実装では $GF(2^n)$ の元を多項式表現しているため，加算と減算は高速に実行される．しかし，乗算と除算は加算と減算に比べ実行速度が格段に遅い．秘密分散法は，復号化の過程で方程式をとく必要があるため，除算が必要になる．そのため，別の多項式表現を用いるなどの方法により除算の高速化をはかる必要がある．

提案法式のファイルシステムは，リモートホストに保管してあるシェアのデータが改竄された場合の検知手段がない．また，ファイルシステムにも，データが正しいかどうか判定する情報を付加する領域がない．そこで，ファイルシステムに対応できる検知法を検討する必要がある．

本論文でデバイスドライバによる実装は，現在最も普及している IP Network が使用できないことからネットワーク分散が困難であると述べた．しかし，SCSI over IP [8] プロトコルが提案されているので，今後は，デバイスドライバによる実装も検討する必要がある．

謝辞

本研究を遂行するにあたり，終始御指導並びに御鞭撻を賜わりました高知工科大学情報システム工学科の菊池豊助教授に謹んで感謝致します．菊池先生には，勉学に励む環境を用意していただきました．

本研究を遂行するにあたり，終始御指導並びに御鞭撻を賜わりました高知工科大学情報システム工学科の福本昌弘助教授に謹んで感謝致します．特に福本先生には，勉強の仕方を 0 から教えていただきました．

また，本研究を遂行するにあたり，御助言を賜わりました大森洋一助手，妻鳥貴彦助手に深く感謝致します．大森先生には，C 言語に関する質問にお答えいただきありがとうございます．妻鳥先生には，学部のころから長い間面倒を見ていただいて，いろいろな社会勉強もさせていただきました．

有益な講義をしていただいた David Greene 教授に深く感謝致します．David Greene 先生の講義で勉強した論理的な文章の構成は，本論文のために大変役に立ちました．

本論文の文書スタイルファイルを作成してくださった橋本学氏，故井上富幸氏，中平拓司氏に心より感謝致します．橋本学氏と中平拓司氏には，まだ研究室に入って何も分からないときからたくさんの御迷惑をおかけしました．しかし，先輩がいない一期生にとって，大変大きな存在で，両氏なくして今の自分は考えられません．

また，故井上富幸氏には，さまざまな場面で有益な見識を御教授いただき，大変勉強になりました．ここで故井上富幸氏に深く哀悼の意を捧げ，御冥福をお祈り申し上げます．

そして，一緒に輪講をしていただいた福本研究室の秋山 由佳女史，武田秀丞氏，坂本研究室の登伸一氏，島村研究室の山岡徹也氏，清水研究室の上岡隆氏，福富英次氏に感謝します．

最後に，御協力をいただいた菊池研究室と福本研究室の皆様方に深く感謝いたします．

参考文献

[1] A. Shamir, “How to Share a Secret,” Communication of the ACM, Vol.22, No.11, pp.612–613, Nov. 1979.

[2] G. Blakley, “Safeguarding Cryptographic Keys,” Proc. of AFIPS 1979 Nat. Computer Conf., Vol.48, pp.313–317, Sept. 1979.

[3] 岡本龍明，山本博資，現代暗号，14. 秘密分散法産業図書，1997.

[4] D.A. Patterson, G.A. Gibson, and R.H. Katz, “A Case for Redundant Arrays of Inexpensive Disks (RAID),” In Proc.of Int. Conf. on Management of Data, pp.109–116, May 1988.

[5] M. Karels, and M. Mckusick, “Toward a Compatible Filesystem Interface,” Proceedings of the Autumn 1986 European UNIX Users’ Group Conference, pp.481–496, Oct. 1986.

[6] J.S. Heidemann, and G.J. Popek, “File-system Development with Stackable Layers,” ACM Transactions on Computer Systems, Vol.12, No.1, pp.58–89, 1994.

[7] 松本尚，“NIC を活用したネットワーク RAID 方式の提案,” 情報処理学会研究会報告，Vol.2000, No.74，pp.79–84，Aug. 2000.

[8] RFC3347, “Small computer systems interface protocol over the internet (iscsi) requirements and design considerations,” , July 2002.

[9] 尾形わかは，黒沢馨，“秘密分散共有法,” 現代暗号とマジックプロトコル，今井秀樹（編），第 8 章，サイエンス社，2000.

[10] U. Vahalia，最前線 UNIX のカーネル，ピアソン・エデュケーション，2000，徳田英幸，中村 明，戸辺 義人，津田 悦幸 訳.

[11] 大木敦雄，FreeBSD カーネル入門 改訂版，アスキー出版局，1998.

[12] S.J.Leffler, M.K.Mckusick, M.J.Karels, and J.S.Quarterman, The Design and Im-

plementation of the 4.3BSD UNIX Operating System, 丸善株式会社, 平成 3 年, 中村 明，相田 仁， 計 宇生，小池 汎平 共訳.

[13] B. Callaghan，NFS バイブル，株式会社アスキー，2001，株式会社クイープ 訳.

[14] 今井秀樹，符号理論，3. ガロア体電子情報通信学会，平成 2 年.

[15] 宇野俊夫，デイスクアレイテクノロジ RAID，エーアイ出版，2000.

[16] W.C. Preston，SAN & NAS ストレージネットワーク管理，オライリー・ジャパン，2002，金崎 裕己 監訳， 豊沢聡 訳.

[17] 喜連川優（編），ストレージネットワーキング，オーム社，平成 14 年.

[18] 舟橋釈仁，福本昌弘，菊池豊，“秘密分散法を用いたファイルシステムの実装,” 情報理論とその応用シンポジウム，Vol.1, pp.235–238，Dec. 2002.