システムソフトウェアと
オペレーティング・システム 79－6
（1998. 8. 6）

分散ファイルシステムにおける細粒度セキュリティドメイン

石井秀浩* 高野陽介 黒岩実
NEC C&Cメディア研究所

計算機のスーパーユーザの特権濫用等による盗み見からファイルを守るための暗号ファイルシステムが、これまでいくつか考案されて来た。しかしこれら既存の方式を用いると、ユーザ間でのファイル共有のためには、各ユーザがたくさんの鍵を憶えなくてはならないなどの問題が生じる。本論文では、アクセス権のあるユーザごとにファイルの鍵を暗号化することによる解決をまず示す。ただしこの方式では、メンバーの追加・削除をファイルごとに処理しなくてはならないなどの問題が生じる。そこで、この問題もさらに解決できるユーザ単位およびグループ単位のアクセス制御方法として、ユーザの代理としてアクセス制御を行なう「ファイル鍵サーバ」を用いる方法を提案する。

Fine-grained Security Domains of Distributed File Systems

Hidehiro Ishii* Yosuke Takano Minoru Kuroiwa
C&C Media Labs, NEC Corporation

Cryptographic file systems had been invented in order to prevent super users or other unauthorized users from eavesdropping. However, some problems arise when users try to share encrypted files. One of those problems is that users have to memorize as many keys as the number of groups they belong to. In this paper, we first present a solution that encrypts file keys for individual readers. But with this solution, admitting or expelling a user into or from a group requires to re-make encrypted keys of every file which the group is allowed to read. Then we will propose a per-user and per-group access control method introducing "file key server" which controls file access on behalf of its owner.

1 はじめに

1.1 一般的なファイルシステムにおけるセキュリティ

通常のマルチユーザのファイルシステムにおいては、ファイルやディレクトリの所有者等が、どのユーザあるいはどのグループのメンバがそれらにアクセスできるかを設定できる。

ところが、ファイルを保存している計算機のスーパーユーザ（システム管理者）はその計算機における全権を持っているので、旧来のファイルシステムでは、設定されているアクセス権に関わらず、このスーパーユーザはすべてのファイルに対して読むことも書くことも自由にできてしまう。特に分散ファイルシステムにおいては、ファイルサーバが多くの計算機のユーザの

*E-mail: *ishii@ccm.cl.nec.co.jp*

ファイルを保存することになり、ファイルサーバのスーパーユーザの権限は極めて大きい。

一般的には、セキュリティの必要なファイルは部門内などの比較的信用できる範囲の計算機にしか置かないので、上記のような事柄は必ずしもさほど大きな問題として扱う必要はない場合が多い。しかし、それで済まない場合もある。例えば、人事情報（従業員の異動、人事考課、給与等の情報、大学等における学生の成績に関する情報、等）等は組織内でも秘密にしなくてはならないものである。そして、そのような情報を扱うユーザ（管理職者など）の多くは複雑なシステム管理のスキルを持たず、システム管理は部下や学生などにさせる場合が多いため、スーパーユーザによる盗み見は大きな問題となる。

ファイルのセキュリティを守るに当たっては、ユーザ自身がシステム管理を行なってその計算機でファイルを扱い保存するか、あるいは他者が管理するファイルサーバに保存してそのスーパーユーザを信用するしかなかった。前者の場合には、システム管理スキルの獲得やデータのバックアップ作業を含む余計な管理コストを個々のユーザが背負う事になる。

結果として、このようなセンシティブな情報は、危険に晒してしまうか、余計な管理コストをかけるか、計算機では扱えないか、ということになっていた。

1.2 既存の暗号ファイルシステム

このような問題を解決できる手段として、暗号ファイルシステムがいくつか提案されている（Cryptographic File System[Blaze93]、FreeBSD 用の暗号ファイルシステム [斉藤 97]、他）。これらは、ファイルの書き込み時に暗号化し、読み出し時に復号することで、スーパーユーザ等による盗み見を防ぐ事ができる。これらのシステムは、特定の個人のみがアクセスするファイルのセキュリティに関しては極めて有用と考えられる。しかし暗号化したファイルをユーザ間で共有しようとすると、鍵を複数のユーザすなわちグループで共有しなくてはならない。これには次のような不都合がある。

- 複数のユーザで共有している鍵は、誰と誰と誰がそれを知らされているのかを常に確実に把握しておく事が困難である。また共有している鍵は、秘密の維持の責任が分散されるため、本来知るべきでないユーザにも洩れやすい。
- グループ外への秘密を保ちながらグループのメンバに鍵を配布する作業には、ユーザの負担が大きい。
- グループごとに異なる鍵を設けなくてはならないので、複数のグループに属すユーザはそのグループの数だけの鍵を憶えなくてはならない。多くの鍵を秘密にしたまま記憶する事は、困難である。[1]
- 鍵は時々変更しないとセキュリティを強く守る事は難しいが、複数のユーザで共有している鍵は変更が困難である。

このような問題を解決し、スーパーユーザへの信用を前提とせずにファイルを保存・共有できるようにするのが、本論文で提案する方式の目的である。

以下第 2 章で、電子メールの配送等に使われている原理を応用して上記の問題を解決する方法を簡単に説明し、その方式で残る問題点を検証して、次に第 3 章で、それを解決するためにグループ単位でのアクセス制御を実現する方法を述べる。そして第 4 章で、第 3 章のシステムの前提条件の妥当性と耐障害性について検証する。

2 読み出し者ごとの暗号化によるアクセス制御（既存原理の応用）

電子メールでは、PEM[Linn93], S/MIME [Dusse98] 等、暗号を用いて一人以上の受信者に安全にメールを配送する方法が確立されている。この原理を利用して、下記の情報をファイ

[1] 複数の鍵を忘れないためには、紙や計算機のファイル等に鍵を記録しておく事が考えられる。しかし、鍵やパスワードを紙やファイル等に記録する事は危険であるとされている。

ルサーバに保存する事で、ファイルシステムにおいてもスーパーユーザの盗み見を防いで、一人以上の特定のユーザにのみファイルを読めるようにする事ができる。

- 共通鍵暗号系で暗号化したファイル（以後、この際の暗号鍵を「ファイル鍵」と呼ぶ）
- (<読み出し権のあるユーザの識別子>, <そのユーザの公開鍵で公開鍵暗号系によりファイル鍵を暗号化したもの>) の組みのリスト（以後これを ACL と呼ぶ）

ファイルの読み出しの際は、読み出し者は自分の公開鍵で暗号化されたファイル鍵を ACL から取り出し、これを自分の秘密鍵で復号してファイル鍵を得、ファイル鍵でファイルを復号すれば良い。

この方式で、読み出し権のあるユーザを一人一人指定してアクセス制御する事ができる。ただし、複数のユーザ（グループ）に対する読み出し権の制御としては、これだけでは不十分であり、次のような問題が残っている。

- 読み出し権を持つユーザが多くなると、ACL が長くなるため、ファイルの書き込み性能が低下する。
- 読み出し権は ACL 作成時に決定するので、読み出し権を持つユーザを追加または削除しようとする時は、ファイルごとに ACL を修正しなくてはならない。削除時には、さらに各ファイルの暗号化もやり直す必要がある。これらには大きな処理コストがかかる。

そこで、さらにこの問題も解決するセキュリティ制御方式を次章で提案する。

3 ファイル鍵サーバによるグループ単位のアクセス制御

3.1 システムの構成とセキュリティ上の前提

図 1 のようにシステムの構成として、

- ファイルを書き込む時に書き込むユーザが使用するクライアント計算機（以下「書き込み端末」と呼ぶ）
- ファイルサーバ
- ファイルを読み出す時に読み出すユーザが使用するクライアント計算機（以下「読み出し端末」と呼ぶ）

に加えて、

- ファイル鍵サーバ

という計算機を設ける。ファイル鍵サーバは、個人単位、あるいはワークグループ単位などで設け、それ自身の公開鍵と秘密鍵を持つ。

ここで、次の事柄を前提とする。

1. 書き込み端末は、書き込むユーザにとって安全である。
2. 読み出し端末は、読み出すユーザにとって安全である。
3. ファイル鍵サーバは、ファイルを所有するユーザにとって安全である。

これらの前提の妥当性については、第 4 章で検証する。

本方式では、ファイルサーバ上のデータやネットワーク上を流れるデータのセキュリティは前提としない。（すなわち、これらは改竄や盗み見をされても構わない。）

3.2 本システムの概要

本システムでは、下記のものをファイルサーバに保存する。

- 乱数を鍵としてファイルを暗号化したもの（この鍵をファイル鍵と呼ぶ）
- ファイル鍵サーバの公開鍵でファイル鍵を暗号化したもの
- ファイルの読み出し権を持つユーザとグループのリスト

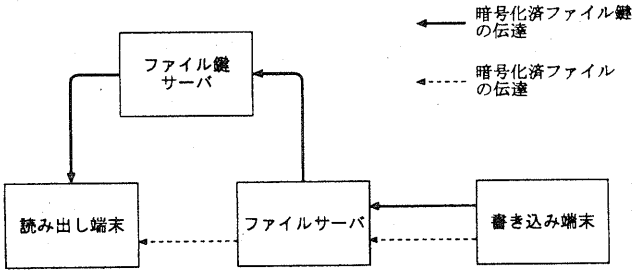

図 1: システム構成と情報の伝達

- 前項のリストにファイル鍵で署名した署名情報

ファイル鍵サーバは読み出し権を持つユーザとグループのリストを参照して、それに含まれるユーザにのみファイル鍵を安全に知らせる。

この仕組みにより、スーパーユーザの管理下にあるファイルサーバからファイルの所有者自身の管理下にあるファイル鍵サーバにアクセス制御の権限を移す事になり、スーパーユーザ等による盗み見を防ぎつつグループ単位でのアクセス制御が可能になる。

3.3 動作の詳細

3.3.1 書き込み処理

ファイルの書き込み時には、書き込み端末で次のような処理を行なう（図 2)。

1. 乱数を発生し、これを鍵としてファイルを暗号化する。(この鍵を「ファイル鍵」と呼ぶ事とする。)
2. ファイル鍵サーバの公開鍵でファイル鍵を暗号化する。
3. 読み出し権のあるユーザの識別子および読み出し権のあるユーザのグループの識別子(共に 0 個以上任意の個数) を連結したデータを作る。(本論文ではこれを「アクセスリスト」と呼ぶ事とする。)
4. 共通鍵暗号を用い、アクセスリストにファイル鍵で署名する。
5. 1 の暗号化済みファイル、2 の暗号化済ファイル鍵、3 のアクセスリスト、4 の署名を、ファイルサーバに送る。

ファイルサーバは、5 で書き込み端末から受け取った情報を保存する。

上記の処理を式で表すと、以下のようになる。

共通鍵暗号 c において鍵 k でデータ d を暗号化したものを

$$c(k,d)$$

と表記することとし、公開鍵暗号 C において鍵 k でデータ d を暗号化したものを

$$C(k,d)$$

と表記することとし、また共通鍵暗号を用いて鍵 k でデータ d に署名した際の署名情報を

$$s(k,d)$$

と表す事とする。[2]

書き込むファイルを f、そのファイル鍵を k_f、ファイル鍵サーバの公開鍵を p_{KS}、アクセスリストを L とすると、ファイルサーバに送って保存するものは、暗号化済ファイル

$$c(k_f, f) \qquad \text{(式 1)}$$

[2] 共通鍵暗号 c において鍵 k でデータ d に署名することは、署名の検証者と署名者が共に鍵 k を知っているという前提の下に、例えば次のようにして実現できる。すなわち、d を MD5 などのハッシュ関数 h でハッシュし、これを鍵 k で暗号化した情報 $c(k, h(d))$ を、署名情報として d に添付して署名の検証者に送る。署名の検証者は、受け取った d について同様に署名情報を計算し、その結果が受け取った署名情報と一致していれば、データ d は鍵 k を知っているものによって署名されたと認められる。

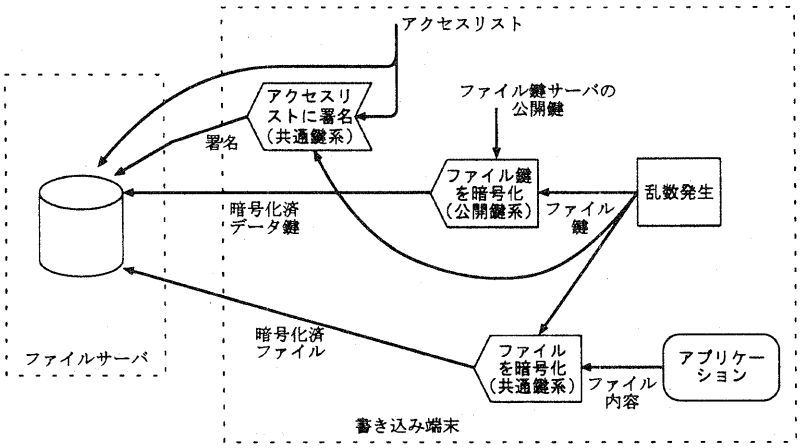

図 2: ファイルの書き込み処理

暗号化済ファイル鍵

$$C(p_{\mathrm{KS}}, k_f) \qquad \text{(式 2)}$$

アクセスリスト

$$L \qquad \text{(式 3)}$$

アクセスリストの署名

$$s(k_f, L) \qquad \text{(式 4)}$$

である。

3.3.2 読み出し処理

ファイルの読み出しにおいては、以下のような処理を行なう（図 3)。

まず読み出し端末は、読み出すべきファイルの識別子（ファイルハンドル等）と読み出し者の識別子を、ファイル鍵サーバに送る。これに対してファイル鍵サーバは、以下の処理を行なう。

1. 読み出すべきファイルに対応する暗号化済ファイル鍵（式 2)、アクセスリスト（式 3)、アクセスリストの署名（式 4）を、ファイルサーバから読み出す。

2. 暗号化済ファイル鍵を復号して、ファイル鍵 k_f を得る。

3. アクセスリストが正当である事を確認する。すなわち、ファイルサーバから得た署名が、前項で得たファイル鍵 k_f でアクセスリストを署名したものに一致する事を確認する。署名が正しくなければ、エラーとする。

4. 読み出し者の識別子がアクセスリスト L 内に直接含まれるいずれかのユーザ識別子と一致すれば、読み出し者が読み出し権を持っているものと判定する。さもなくば、グループの定義（各グループにどのユーザが属しているかという情報）を参照し、L に含まれるいずれかのグループに読み出し者が属していれば、読み出し者はやはり読み出し権を持っているものと判定する。どちらでもなければ、読み出し権を持っていないものと判定して、エラーとする。

5. 読み出し端末から示された読み出し者の識別子に対応するユーザの公開鍵（p_R とする）を取得し、ファイル鍵 k_f をこれで暗号化し（$C(p_R, k_f)$ を得る)、その結果を読み出し端末に送る。

そして読み出し端末は、読み出し者の公開鍵で暗号化されたファイル鍵 $C(p_R, k_f)$ をファイル鍵サーバから受け取り、対応する読み出し者の秘密鍵でこれを復号してファイル鍵 k_f を得、

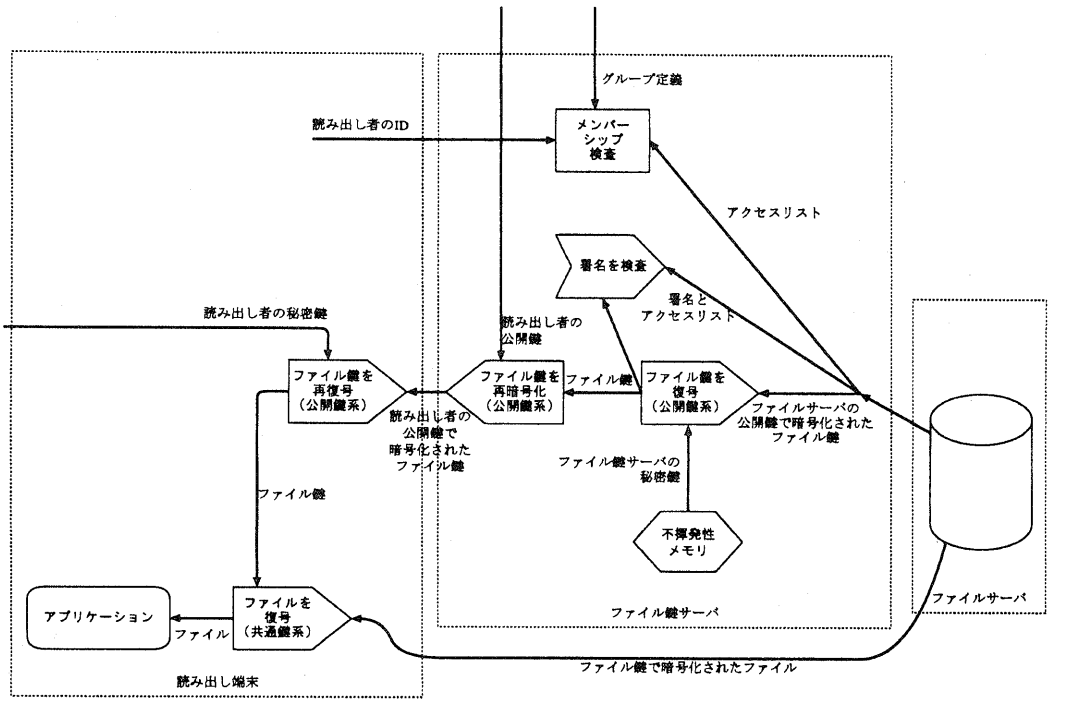

図 3: ファイルの読み出し処理

ファイルサーバから暗号化済ファイル（式 1）を取得して、ファイル鍵 k_f でこれを復号して平文のファイル f を得る。

3.4 アクセス権設定とファイル書き込みの独立実行

以上では、ファイルのアクセス権の設定とファイルの内容の書き込みを一つの計算機上で同時に行なう事としたが、これらは別々の計算機上で、あるいは別々の時点で行なうこともできる。

例えば、既にアクセス権を設定された（すなわち、アクセスリスト、その署名、暗号化済ファイル鍵を既に設定された）ファイルに対してあとから上書きや追記をしようとする時は、書き込み端末は読み出し端末と同様にファイル鍵サーバからファイル鍵を得て、書き込むべき内容をこの鍵で暗号化してファイルサーバに送る。

また、既存のファイルのアクセス権を変更しようとする時は、同様にファイル鍵サーバからファイル鍵を得て、新しいアクセスリストを作ってファイル鍵で署名し、新しいアクセスリストと新しい署名をファイルサーバに保存する。この際同時に、ファイル鍵を変更してファイルを暗号化し直しておくと、読み出し権を剥奪されたユーザが過去に得たファイル鍵でファイルを読み出す事を防ぐ事ができる。

3.5 ファイル鍵サーバが参照する情報の正当性

ファイル鍵サーバはいくつかの情報を参照してアクセス制御を行なうので、これらの情報が偽造されると、正しいアクセス制御ができなくなってしまう。本節では、これらの偽造を防ぐ方法を述べる。

3.5.1 グループ定義

読み出し処理の 4 項めにおいて、ファイル鍵サーバはグループの定義を参照する。このグ

ループ定義が偽造されると、読み出し者が本当に読み出し権を持っているかどうかをファイル鍵サーバが誤って判別してしまう危険がある。しかし、例えばファイルの所有者が自分のファイル鍵サーバ内に予めグループ定義を書き込んでおいて、読み出し時にこれをファイル鍵サーバが参照するようにすれば、ファイル鍵サーバが参照するグループ定義が正当である（定義すべきユーザが定義したものである）事を保証する事は容易である。

ただし、ファイル鍵サーバは多数存在し得るので、「グループ定義サーバ」とでも言うような別の計算機でグループ定義を集中管理して、ここから各ファイル鍵サーバに提供するような形態の方が便利であると考えられる。このような場合のグループ定義情報の正当性の保証に関しては、稿を改めて論じる予定である。

3.5.2 読み出し者の公開鍵

読み出し処理の5項めにおいて、ファイル鍵サーバは読み出し者の公開鍵を参照する。ここで、読み出し端末が読み出し権のあるユーザの識別子を示しながら読み出し権のないユーザの公開鍵をファイル鍵サーバに参照させることができると、読み出し権のないユーザがファイル鍵を復号することが可能になってしまう。しかし、例えばWWWなどで利用されている証明書を導入し、しかるべきcertification authorityによって証明された公開鍵を読み出し端末などから受け取るようにすれば、公開鍵の正当性を保証する事は容易である。

4 妥当性の検証

第3章で提案したグループ単位でのアクセス制御の方式の、妥当性を検証する。

4.1 前提の妥当性

まず、3.1節で示した3つの前提の妥当性を考える。

4.1.1 書き込み端末と読み出し端末の安全性

第1および第2の前提（書き込み端末と読み出し端末の安全性）は、次のようにして成り立たせる事ができる。

- 書き込み端末（または読み出し端末）のユーザ自身、またはそこで扱う情報に対するアクセス権を持つ人物のみを、その端末のスーパーユーザとする。

またそうしない場合でも、秘密の鍵等をこれらの端末内にファイルとして記録しなければ、これらの前提をある程度仮定する事ができる。

4.1.2 ファイル鍵サーバの安全性

さらに第3の前提（ファイル鍵サーバの安全性）は、例えば次のようにして成り立たせる事ができる。

- ファイル鍵サーバをファイルの所有者ごとに設け、本人以外はそれを操作できないようにする。

鍵の設定等の操作をネットワーク経由で行なうようにすれば、I/Oとしてはネットワーク・インタフェースを備えるだけで済む。永続的に記憶する必要があるのはファイル鍵サーバ自身の秘密鍵とプログラムくらいであるので、ハードディスクも必要なく、ROMと小容量のEEPROMがあればよい。

ファイル鍵サーバの処理は単純であり、そのファイル鍵サーバが管理するファイルのアクセス頻度が高くなければ処理速度も低くて良いので、MPUやRAM等は安価なものでよい。

このような装置は例えば組込みOSによる小型（例えば数百グラム程度）のものにでき、ある程度大量生産すれば安価（例えば1〜4万円程度）にできると思われる。また、データのバックアップその他のシステム管理も必要ない。例えば、ファイル鍵サーバの秘密鍵をその所有者自身の秘密鍵と同じにして所有者が憶えておけば、ファイル鍵サーバがクラッシュした場合のその秘密鍵の再設定は容易である。

従って、ファイル鍵サーバをユーザごとに設ける事に困難はない。

そして、ケースに封印を施すか鍵のかかる引出し等に保管するなどの措置により、ファイル鍵サーバへの物理的な攻撃を阻止または抑止できる。またネットワーク経由の操作をパスワード等で保護すれば、ネットワーク経由での侵入も阻止できる。

従って、第 3 の前提を成り立たせる事にも特に困難はない。

4.2 耐障害性

本システムでは、ファイル鍵サーバに障害が起こっている時は、そのファイル鍵サーバが管理するファイルの読み出しはできない。しかし、ファイル鍵サーバの導入コストと運用コストは前述の通り小さいので、それを多重化する事や、予備のファイル鍵サーバの在庫を持つ事は容易である。多重化等により、ファイルをアクセスできなくなる事はほぼ防ぐ事ができ、従来の分散ファイルシステムとほぼ同様の耐障害性を得る事ができる。

5 まとめ

本論文で提案した、ファイル鍵サーバを用いたアクセス制御方式では、ファイルサーバには平文のファイルも平文の秘密鍵も渡されない。そして、アクセス権のあるユーザにのみファイル鍵サーバがファイル鍵を知らせる。これによって、

- ファイルサーバのスーパーユーザの信用を前提とせず、
- 個々のユーザが過剰な管理コストを負う事もなく、
- ユーザ間で秘密の鍵を共有する必要なく、
- ユーザ数に応じて ACL が長くなることもなく、
- グループのメンバーの追加変更の際にファイルごとに ACL を修正する必要もなく、

ユーザ間でファイルを安全に共有することができる。

ここで、旧来のファイルシステムにおけるセキュリティ・ドメインはファイルサーバ単位であったと言えるのに対し、本システムにおけるセキュリティ・ドメインはファイル鍵サーバ単位である。ファイル鍵サーバの導入と運用は、ファイルサーバに比べて人件費・設備費等を含めて極めて低コストに行なえるので、これはワークグループ単位や個人単位など自由にきめ細かく設ける事ができる。すなわち、ユーザの必要に応じて自由にきめ細かくセキュリティ・ドメインを定義する事が可能である。

謝辞

本論文の執筆に当たり暗号と電子署名の理論についてご教授くださった NEC C&C メディア研究所の宮野浩氏に感謝致します。

参考文献

[Blaze93] Matt Blaze, "A Cryptographic File System for Unix", In *1st ACM Conference on Computer and Communications Security, pp. 9–16*, Nov. 1993

[Blase94] Matt Blaze, "Key Management in an Encrypting File System", In *Proceedings of the USENIX Summer 1994 Technical Conference*, Jun. 1994

[Dusse98] Steve Dusse, P. Hoffman, B. Ramsdell, L. Lundblade, and L. Repka, "S/MIME Version 2 Message Specification", In *RFC2311, Internet Engineering Task Force*, Mar. 1998

[Gudes80] Ehud Gudes, "The Design of a Cryptography Based Secure File System", In *IEEE Transactions on Software Engineering Vol.SE-6, No. 5, pp 411–420*, Sep. 1980

[Linn93] John Linn, "Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures", In *RFC1421, Internet Engineering Task Force*, Feb. 1993

[斉藤 97] 斉藤紀、松永良太郎, "暗号ファイルシステムの試作" http://www.netlaputa.or.jp/%7Eossan/freebsd/securefs/, 1997 年 3 月