2008－UBI－20 (3)
2008／11／13

自己暗号化法によるモバイル端末向け セキュアファイルシステムの実装

遠藤大礎[†1] 川原圭博[†1] 浅見徹[†1]

携帯電話の高機能化，ストレージの大容量化が進むにつれて，携帯電話にはより多くの情報が蓄積されるようになっている．携帯電話は紛失されやすいため，内部情報の保護は大きな課題である．そこで我々は一般に処理能力が貧弱な携帯電話における高速暗号化・復号手法として，自己暗号化法を用いた分散ストレージシステムを提案してきた．この手法は近年高速化しつつある無線通信を考慮し，通信総量が大きいが処理が単純な高速暗号を使うところに特徴がある．本稿では暗号化・復号を自動的に行うセキュアファイルシステムをFUSE（Filesystem in Userspace）を用いてLinux端末に実装した．またWILLCOMの端末開発プラットフォーム“Sandgate WP”において行った暗号化・復号処理実験の計測結果について示す．

An Inplementation of Self-Encryption Based Secure File System for Mobile Handsets

Hiroki Endo,[†1] Yoshihiro Kawahara[†1] and Tohru Asami [†1]

People have had more opportunities to preserve important information properties in mobile handsets. The security for mobile handsets will become essential in cases of the loss and the theft. We have proposed an architecture for the distributed storage system, which gives the high speed encryption to mobile handsets, by light-weight processing called self-encryption, in the high-speed wireless communication environment today and in the near future. In this paper, we implement the secure file system that automatically processes encryption and decryption working on Linux PC with FUSE (Filesystem in Userspace), and show the measurement result of the processing time at encryption and decryption by the terminal development platform “Sandgate WP ”of WILLCOM.

1. はじめに

現在ではノートPCはもとより，携帯電話やPHSなどのモバイル端末が動画撮影ビデオカメラ，数100万画素のデジタルカメラ，電子書籍，業務用アプリケーション，ゲームアプリなど挙げきれないほどのさまざまな機能を備えるようになっている．この結果内部メモリ・外部メモリとして，ともにギガバイト超の大容量ストレージが装備され，個人情報，業務情報などの重要な情報資産が保存されている．しかし2007年一年間に東京都だけで124,863件の携帯電話の遺失届が提出されていることからもわかるように，携帯電話はとても紛失されやすいものである．東京都の人口がおよそ1,280万人であることを併せて考えるとおよそ1%の人間が携帯電話を紛失していることになり，さらにこの件数は近年増加傾向にある[1),2)]．

†1 東京大学 大学院情報理工学系研究科
The University of Tokyo, Graduate School of Information Science and Technology

このような携帯電話の紛失の状況に加えて，身分証明書やモバイルSuica，iDなど電子マネーの機能も担い始めていることから今後盗難のターゲットになっていくことも考えられるため，紛失・盗難に対するモバイル端末の内部情報漏えい対策は急務である[3)]．実際に各通信キャリアは暗証番号やバイオメトリクスによるユーザ認証，紛失に備えた端末のリモートロック，遠隔データ消去などの手段を提供しているが，これらの対策にはそれぞれ課題が残されている．暗証番号による対策には操作のたびに暗証番号を入力する必要がありユーザに大きな負担となることや，4桁の暗証番号では最悪1万回の施行で認証を突破されてしまう脆弱性がある[4)]．バイオメトリクス認証については顔認証，指紋認証が端末に搭載された事例があるが，いずれも本人の写真や残留指紋から作られたゼラチン製の疑似指により認証を破ることができると報告されている[5)–12)]．一方リモートロック，遠隔データ消去などのサービスでは端末が基地局からの命令を受信する必要があるので，端末を拾得・盗難した攻撃者が端末のバッテリーを抜き取ったり，電波の届かない場所に運

搬することで操作を受け付けなくすることが可能である[13),14)].

またそもそも携帯電話のセキュリティに関心が低い人間も多い.「2007 年度携帯電話の利用実態調査」によると，携帯電話のセキュリティを掛けている人は42.8%で，そのうち 99.2%が暗証番号セキュリティを利用している[15)]．また「2005 年第 18 回携帯電話コンテンツ/サービス利用者調査結果（下)」より「実施する必要がない」「実施したいがやり方がよくわからない」を合わせると 75.8%にのぼることがわかる[16)]．携帯電話に蓄積される情報は個人の所有物だけではなく，他ユーザとの共有情報であるものも多い．もしセキュリティの意識が高いユーザでも他ユーザの端末からの情報漏えいまでは防ぐことはできない．たとえばメールのメッセージにしても送信相手の端末から取得されてしまう可能性もある．このようにセキュリティを考えるには攻撃者に対して社会全体で立ち向かうことを考える必要があり，このときは基準を 8 桁の暗証番号を厭わない意識の高いユーザにおくのではなく，「セキュリティに関心が薄く」,「難しいことはできない」,「めんどうくさがり」で「わがまま」なユーザを想定しなければならない．

以上から，私たちはモバイル端末向けのセキュリティシステムに対して 3 つの条件をあげている[17)]．

端末紛失に対して有効であること

モバイル端末は常にログインされた状態にあり，とても紛失されやすいことからモバイル端末内部情報を常に暗号化された状態におく設計が必要である．このとき復号に必要な情報はネットワークを介してリモートサーバへと保存し，紛失の際にはリモートサーバへのアクセスをブロックすることで情報の漏えいを防ぐ．またファイル名やファイルサイズ，変更日時などのファイル管理情報自体が重要な情報となりうるため，データだけでなくメタデータも暗号化の対象とする必要がある．

モバイル端末特有の機能の活用

一般にモバイル端末は固定端末と比較して CPU 処理能力が貧弱なため，ユーザに待ち時間を嫌われた結果セキュリティ処理を使ってもらえないことも考えられる[18)]．そこで本システムでは，モバイル端末の通信速度が年々高速化しつつあり，現在でも HSDPA や CDMA2000 1xEV-DO Rev.A 実測値で数 Mbps，数年後には LTE や次世代 PHS などさらに高速な通信方式も採用が予定されていることに注目し，通信容量を大きくし，CPU 処理を簡略化することによる最適化を考える[19),20)]．本システムでは暗号化アルゴリズムとして自己暗号化法を採用し，暗号鍵を大きくすることでモバイル端末における暗号化・復号処理の負荷を軽減，高速な処理によりわがままなユーザにも使ってもらうことを目指す．

セキュリティシステムの自動化

セキュリティにあまり関心がないユーザに対して内部情報保護のために余計な手続きを要求することは難しい．そのためユーザの手続きを必要としない端末による自動処理が必要である．暗号化・復号処理を行う階層にはアプリケーションからストレージデバイスドライバまでいくつか考えられるが，本システムではメタデータを暗号化の対象とすることや，ファイル単位での処理を行い柔軟なアクセス設定を可能にするため，ファイルシステムにおいて暗号化・復号処理を行うものとする[21)]．この実装によりアプリケーションに対する変更は一切必要なくなり，従来のアプリケーション資源をそのまま活用することが可能になる．

本稿では以上の 3 つの条件を満たすシステムとして自己暗号化法によるセキュアファイルシステムを提案する．このシステムは端末内の機密ファイルを暗号化し，端末とリモートストレージへ分散配置するクライアント・サーバ方式のシステムである．そしてその構成に必要となる内部暗号化手続きとして，暗号化対象ファイル「自身」を材料に自動的に暗号鍵を生成し，その鍵を用いてファイルを暗号化する手続き「自己暗号化法」を提案する．まず 2 節では，提案する分散ストレージシステム構成を示し，内部暗号化手続きとして用いる自己暗号化法について述べる．3 節では，提案セキュアファイルシステムの Linux PC における FUSE を用いた実装について述べる．最後に 4 節ではウィルコムの PHS 通信モジュール “W-SIM” に対応した Linux OS 搭載の端末開発プラットフォーム “Sandgate WP” において行った暗号化・復号処理実験の計測結果により自己暗号化法と既存暗号化アルゴリズムとの処理負荷比較を行う．

2. 自己暗号化法による分散ストレージシステム

自己暗号化法とは暗号化対象ファイルを 2 分割し，一方のデータ片から生成した暗号鍵を用いてもう一方のデータ片に暗号化を施す手法である．本手法においては暗号化されたデータを端末内に保存し，暗号鍵を生成するための元データをリモートサーバへアップロードすることを前提としている．これにより端末を紛失したとしてもリモートデータへのアクセスをブロックすることで，攻撃者による元データの復元を防ぐことが可能になる．

2.1 システム構成モデル

携帯電話網や，SSL/TLS などによりセキュリティが保たれた通信路の存在を前提に，モバイル端末で利用するデータを暗号化し，暗号化データとリモートデータをローカルストレージとリモートストレージ分散配置するファイルシステムの構成を図 1 に示す．

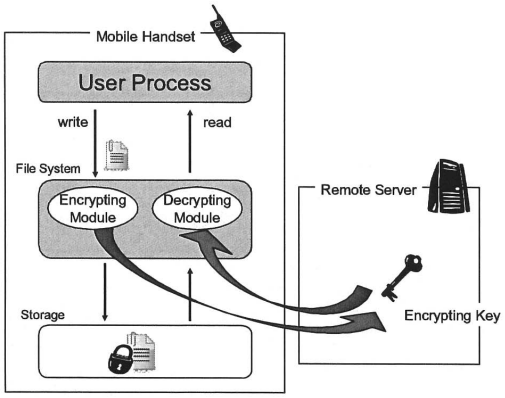

図 1 Encrypted File System for Mobile Handsets

2.2 自己暗号化法

自己暗号化法では，まず暗号化対象ファイル M を同じサイズのローカルデータ m_L とリモートデータ m_R に 2 分割する．次に m_R を材料に m_R と同サイズの暗号鍵 K を生成し，K を用いて排他論理和により m_L に暗号化を施す[22]．図 2 に自己暗号化法による暗号の手順を示す．

(1) M を m_L，m_R に 2 分割する
(2) m_R から鍵生成アルゴリズム F を用いて暗号鍵 K を生成する
(3) K を用いて排他論理和により m_L を暗号化し，c_L を生成
(4) c_L をローカルストレージ内に保存する
(5) m_R をリモートサーバにアップロードする

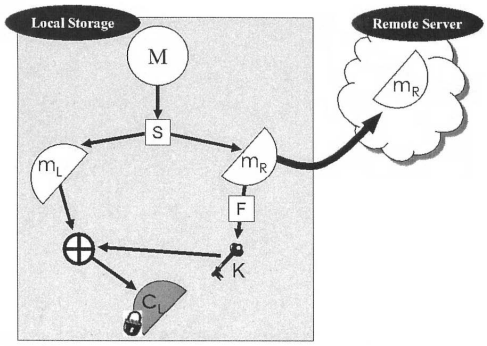

図 2 自己暗号化法 暗号化モデル

図 3 に復号の手順を示す．

(1) リモートサーバから m_R をダウンロードする
(2) m_R から F を用いて暗号鍵 K を生成する
(3) K を用いて c_L を排他論理和により復号し，m_L を生成する
(4) m_L，m_R を結合し，元ファイル M を得る

自己暗号化法を表 1 のように定式化する．

ここで高速にでランダムな K を生成する鍵生成アルゴリズム F として，m_R をある長さのブロック

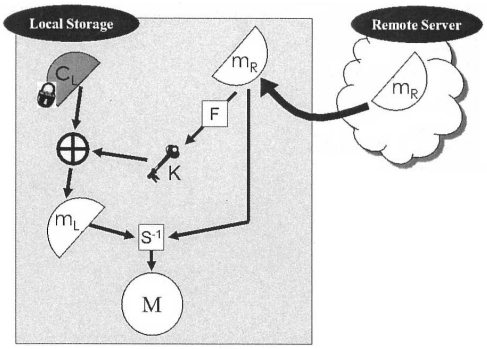

図 3 自己暗号化法 復号モデル

表 1 自己暗号化法の定式化

暗号化手続き	式
M の m_L, m_R への分割	$\{m_L, m_R\} = S(M)$
m_R から K の生成	$K = F(m_R)$
K を用いた m_L の暗号化	$c_L = m_L \oplus K$
復号化手続き	
m_R から K の生成	$K = F(m_R)$
K を用いた c_L の復号化	$m_L = c_L \oplus K$
m_L, m_R から M の合成	$M = S^{-1}\{m_L, m_R\}$

ごとに分割し，ランダムなバイナリ列 IV を用いて，$k_1 = IV \oplus m_1$, $k_i = k_{i-1} \oplus m_i$，とし，$k_1, k_2, \ldots$ とつなげたものを K とするスクランブリングを用いる．この攪拌アルゴリズムによりランダムな鍵 K を高速に生成することができる．排他論理和によるスクランブラのアルゴリズムを，図 4 に示す[23]．

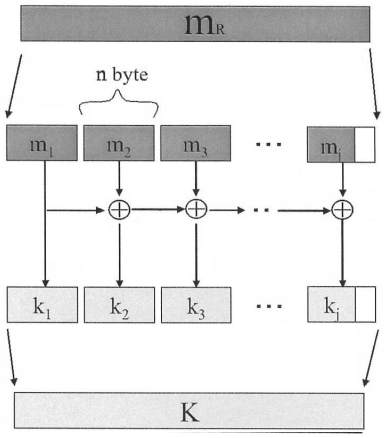

図 4 排他論理和に基づく
スクランブラによる鍵生成

2.3 安全性の評価

提案手法の安全性は排他論理和に用いる鍵 K のラ

ンダムネスに依存する．そこで提案アルゴリズムFにより生成されるKがランダムなビット列になっているかを調べた．

2.3.1 評価方法

鍵材料の対象ファイルとして，日本語テキストファイル (青空文庫，1000 ファイル) とその zip 圧縮ファイル (1000 ファイル) を扱い，検定ツールの都合上，生成した鍵の先頭 64,000 ビットに対して擬似乱数検定を行った．この際，入力データを分割するブロック長を 64, 128, 256, 512[byte] とした前述の鍵生成スクランブラを用いて鍵を生成する．次に，生成した鍵について，NIST の SP800-22 が定める擬似乱数検定のうち，頻度検定，連検定，累積和検定，2 値行列ランク検定を行った[24]．検定には NIST の提供している擬似乱数検定ツール NIST Statistical Test Suite を使用した．この検定においては，各検定ごとに p-value が得られる．p-value とは，真の乱数生成器が検定を行っている系列よりも乱数らしからぬ系列を生成する確率のことであり，この p-value が，p-value < 0.01 のときに，良い乱数ではないと判断する．この検定を，複数の標本系列 (NIST では 1000 程度が推奨されている) に対して行い，

(1) p-value が 0.01 より大きい割合 (Proportion)
(2) p-value の一様性 (Uniformity)

の二つから乱数列の評価を行う．(1) では，標本の数を m としたとき，0.01 以上となる p-value の数の割合が $0.99 \pm 3\sqrt{\frac{0.99 \times 0.01}{m}}$ の範囲に入っている場合に，よい乱数であると判断する．また，(2) では，得られた p-value が区間 [0,1) で一様に分布しているかを調べるために，カイ 2 乗検定にて検定を行う．カイ 2 乗検定により得られた p-value が 0.0001 以上ならば，よい乱数であると評価する．

2.3.2 評価結果

検定結果について，p-value が 0.01 より大きい割合 (Proportion)，p-value の一様性 (Uniformity) を計算し，それぞれ検定に合格しているかを表 2 と表 3 に示した．合格した値のセルの背景色をグレーで示す．この結果からテキストファイルに比べて zip 圧縮ファイルを材料としたときのほうがランダムな鍵が生成される．そのため，テキストファイルなどのビットに偏りがあるものをそのまま鍵材料として使用するとランダムな鍵の生成は期待できないことがわかる．これは，スクランブラが単純な排他論理和のみで構成されているため，鍵材料のビット列に偏りがあると，それが大きく反映されるためと考えられる．一方ブロック長を短く設定すると出力がより乱数列に近づいた．これは，短くすればするほどビット列の攪拌回数が大きくなるためだと考えられる．今回の実験では 8000byte のファイルを入力として扱ったことから，ブロック数がは 62.5(= 8000/128) 以上ならば本実験の 4 検定に対して十分な攪拌回数であると考えられる．

表 2 p-value が 0.01 よりも大きくなる割合

block	M	頻度	連	ランク	累積和
64byte	txt	0.540	0.436	0.514	0.713
	zip	0.992	0.987	0.993	0.985
128byte	txt	0.565	0.567	0.539	0.982
	zip	0.988	0.991	0.987	0.992
256byte	txt	0.489	0.578	0.383	0.960
	zip	0.995	0.983	0.988	0.984
512byte	txt	0.243	0.294	0.159	0.511
	zip	0.940	0.973	0.899	0.989

表 3 p-value の一様性

block	M	頻度	連	ランク	累積和
64byte	txt	0	0	0	0
	zip	0.850	0.13188	0.738	5.4E-02
128byte	txt	0	0	0	6.1E-13
	zip	0.083	0.38041	0.010	1.2E-03
256byte	txt	0	0	0	0
	zip	0.070	0.00353	0	1.4E-02
512byte	txt	0	0	0	0
	zip	0	7.3E-08	0	1.3E-06

3. セキュアファイルシステムの実装

本節では暗号化・復号処理を端末が自動的に行うセキュアファイルシステムの実装について述べる．本ファイルシステムの実装には FUSE (Filesystem in Userspace) ライブラリおよび WebDAV プロトコルを用いる[25),26)]．FUSE とはユーザーモードのアプリケーションとしてファイルシステムを作成することを可能にするライブラリであり，これを用いることでファイルシステムの開発が容易になる．FUSE は Linux 向けに開発されたものだが，FreeBSD および Mac OS X，Windows とさまざまな OS に移植されている．本稿では現在携帯電話において採用が進み始めている Linux における実装を行った．一方 WevDAV は Web 上のリソースを操作するために HTTP1.1 を拡張したプロトコルである．同様の機能を持つプロトコルには FTP や NFS，Samba などがあるが，WebDAV には Web サーバにおいて新たなポートを公開することなく SSL によりセキュアな通信が可能であること，またファイルのロックなどによりコンテンツの不整合を防ぐことやコピーメソッドによりリモートサーバ上においてファイルの複製が可能になりトラフィックを制限することが可能なことなどの利点がある．本システムでは WebDAV プロトコルを用いてリモートサーバと m_R の通信を行う．

3.1 Self-Encryption File System

書き込み処理は 2 節で述べたように対象ファイルを m_L と m_R に分割する split モジュール，m_R から暗

号鍵 K を生成する keygen モジュール，生成した K を用いて m_L に暗号化を施し c_L を得る encryption モジュール，そして m_R をリモートサーバにアップロードする upload モジュールからなる [図 5]．一方，読み込み処理はリモートサーバから m_R をダウンロードする download モジュール，keygen モジュール，c_L を復号し m_L を得る decryption モジュール，m_L と m_R を結合し読み込み対象ファイルを得る combine モジュール，そしてそれらのファイルを一時的に格納する cache モジュールからなる [図 6]．

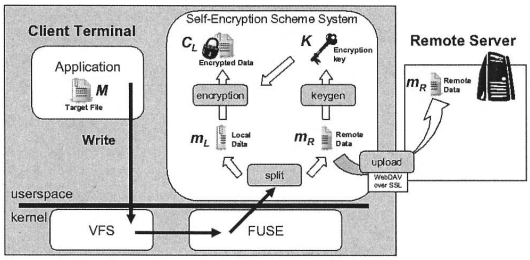

図 5 ファイルシステムにおける WRITE

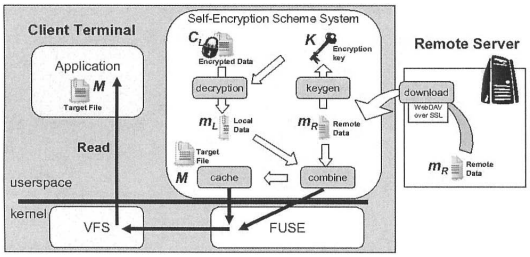

図 6 ファイルシステムにおける READ

今回以上に述べた各モジュールの機能を持つファイルシステムをマウントするプログラムとして，Self-Encryption File System(sefs) を作成した．実装環境として端末に Linux PC[プロセッサ : Intel Pentium 4 (2.8GHz) OS : Fedora Core 5，メモリ : 512MB] を，リモートサーバには WebDAV サーバを動作させた Linux PC[プロセッサ : Intel Pentium M (900MHz) OS : Fedra 8，メモリ : 512MB] を用いた．以下に示すように，ディレクトリを引数として与えて sefs プログラムを実行することで，そのディレクトリに対して書き込み・読み込みを行う際にファイルシステムが自己暗号化法による処理を自動的に行う．今回の実装においてはファイル分割，鍵生成，暗号化，WebDAV プロトコルによるアップロードの各動作を実現するプログラム “enc_fuse” を対象ファイルを fuse が書き込み終わった際に sefs がシステムコールで呼び出し処理することで実装をした．読み込み処理も書き込み処理と同様に，各モジュールの動作を実現するプログラム “def_fuse” を fuse からシステムコールで呼び出し，対象ファイルを復号した後 fuse へと読み込ませることで実装をした．

“./dir” が sefs でマウントされているときには，ファイル M の書き込みは，実際には一時ディレクトリ “/tmp/fuse” に対して行われる．ファイルシステムは書き込まれたファイル M からプログラム “enc_fuse” によって c_1 である $M.loc$ と m_2 である $M.rem$ を出力し，curl コマンドによって https 経由で $M.rem$ は WebDAV サーバにアップロードされる．M および $M.rem$ は “/tmp/fuse” から消去され，$M.loc$ は M にリネームされる．一方 “./dir” ディレクトリ下のファイル M の読み込みがあったとき，ファイルシステムは一時ディレクトリ “/tmp/fuse” 下に WebDAV サーバから $M.rem$ を https 経由でダウンロードする．その後プログラム “dec_fuse” にを実行し，“/tmp/fuse” 下の M（内容は $M.loc$）と $M.rem$ から M を復号してユーザプロセスに返す．

マウント

```
$ sefs ./dir
```

アンマウント

```
$ fusermount -u ./dir
```

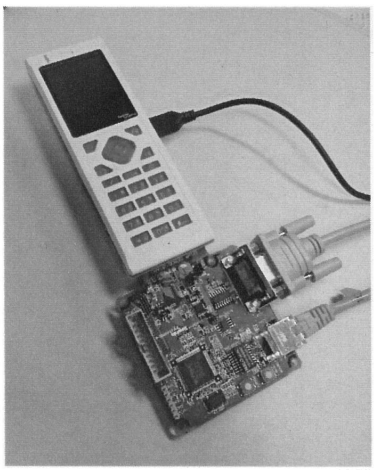

図 7 Sandgate W-SIM Phone

sefs.c

```
#include ⟨fuse.h⟩
                    .
                    .
static int sefs_write(const char *path, const char *buf,
    size_t size, off_t offset, struct fuse_file_info fi){
                    .
                    .
// 書き込んだ M を暗号化
        sprintf(encrypt, "enc_fuse %s", sM);
        system(encrypt);
// m2 をアップロード
        sprintf(curl_put, "curl -T %s
              https://largend.net/pub/ -k", sm2);
        system(curl_put);
// m2 を消去
        sprintf(rm_rem, "rm %s", sm2);
        system(rm_rem);
// M を消去
        sprintf(rm_M, "rm %s", path);
        system(rm_M);
// c1 を M にリネーム
        sprintf(rn, "mv %s /tmp/fuse/%s", sc, fn);
        system(rn);
                    .
                    .
        return res;
}

static int sefs_read(const char *path, char *buf, size_t
    size, off_t offset, struct fuse_file_info *fi){
                    .
                    .
// m2 をダウンロード
        sprintf(curl_get, "curl -o /tmp/fuse/%s.rem
              https://largend.net/pub/%s.rem -k", fn,
fn);
        system(curl_get);
// M を復号
        sprintf(decrypt, "dec_fuse %s", sM);
        system(decrypt);
                    .
                    .
        return res;
}

static struct fuse_operations sefs_oper = {
                    .
                    .
        .read = sefs_read,
        .write = sefs_write,
                    .
                    .
};

int main(int argc, char *argv[]){
        umask(0);
        return fuse_main(argc, argv, &sefs_oper, NULL);
}
```

4. Sandgate WP における暗号化・復号処理時間計測

3 節では FUSE ライブラリを用いた PC Linux へのセキュアファイルシステムの実装を示した．本節では本来の実装の対象目標である携帯電話における性能評価のために，ウィルコムの端末開発プラットフォーム "Sandgate WP" において自己暗号化法によるファイルの暗号化・復号の処理時間を計測し，AES と比較して処理負荷を調べた[27]．

4.1 方　　法

自己暗号化法および AES 256bit による暗号化を組み込み Linux 端末で処理時間計測し比較した[28),29)]．10[byte] から 20[MB] までのファイルを暗号化の対象として，

- 64 ブロック分割のスクランブラによる鍵生成に基づく自己暗号化法
- 鍵長 256 ビットの AES による暗号化

のそれぞれについて 5 回ずつ暗号化を行い，ファイル M の読み込み，暗号化処理，c_1 および m_2 のストレージへの書き込みにかかった時間の平均を求めた．

4.2 実験環境

Sandgate WP [プロセッサ：マーベル PXA270 (416MHz), OS : Linux (VER.2.6), メモリ : SDRAM 64MB, Flash ROM 128MB] を使用して実験を行った [図 7] 今回の実験においては Sandgate WP のルートファイルシステムとして LinuxPC の NFS サーバをマウントしてある．プログラムは C 言語により記述した．

4.3 実験結果

実験結果を，横軸にファイルサイズ [KB]，縦軸に処理時間 [s] として暗号化処理時間を図 8 に，復号処理時間を図 9 に示した．

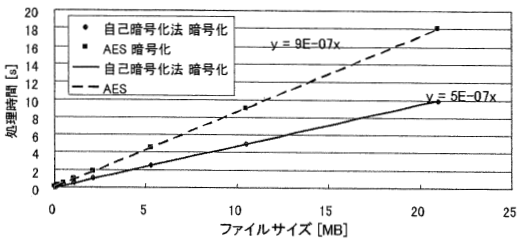

図 8　暗号化処理時間計測結果

4.4 考　　察

図 8，図 9 より，自己暗号化法は暗号化・復号の両方において AES のおよそ 1.8 倍の速さで処理を行うことがわかった．この処理時間の差よりも短い時間でリモートファイル m_2 をダウンロードできればアルゴ

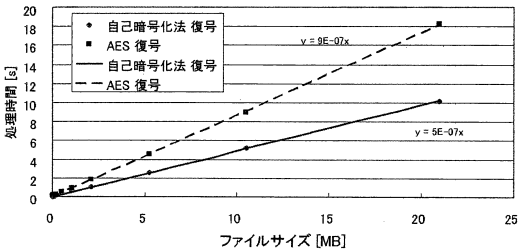

図 9　復号処理時間計測結果

リズムを用いるより高速にファイルを読み込むことができるが，表 4 に示すように，将来の 10Mbps 超の無線通信を仮定しても現在の実装においてはこの差は十分ではない．これは表 5 に示した自己暗号化法におけるファイルの読み込み，書き込み時間からわかるように，NFS サーバからの読み込み，書き込みに時間がかかっていることにも影響されており，ローカルストレージにおける追加実験も必要となるが，m_2 のダウンロードを終えてから復号処理をかける現在の実装にも問題がある．そのためダウンロードバッファごとに復号処理をかけるようなストリーミング処理を施す実装により，ファイルの読み込みとダウンロードの並行処理による高速化が考えられる．

表 4　処理時間差

サイズ [MB]	1	2	5	10	20
自己暗号化法 [s]	0.66	1.06	2.57	5.18	10.19
AES [s]	0.92	1.85	4.49	8.95	18.27
処理時間差 [s]	0.25	0.80	1.91	3.78	8.09

表 5　自己暗号化法における読み込み・書き込み時間

サイズ [MB]	1	2	5	10	20
自己暗号化法 [s]	0.66	1.06	2.57	5.18	10.19
Read/Write[s]	0.52	0.97	2.40	4.62	9.05

5. おわりに

ここまでモバイル端末の内部情報を保護する自己暗号化法によるセキュアファイルシステムの実装について述べ，実際の Linux PC における動作実験を行った．このセキュアファイルシステムにおいては自己暗号化法による暗号化・復号処理がユーザプロセスからの Read/Write にともなって行われる．またウィルコムの端末開発プラットフォーム "Sandgate WP" における自己暗号化法の処理実験を行い，自己暗号化法が従来の暗号化手法よりも高速にによる処理を行うことがわかった．

現在はセキュアファイルシステム実装上の処理効率の改善と本稿では PC 上で動作させたセキュアファイルシステムの移植をモバイル端末に対して行っている．

参考文献

1) 東京都総務局，"東京都の人口 (推計)，" http://www.toukei.metro.tokyo.jp/jsuikeijs-index.htm，2008.

2) 警視庁，"平成 19 年中遺失物取扱集計データ，" http://www.keishicho.metro.tokyo.jp/toukei/kaikei/kaikei.htm，2008.

3) 電子商取引推進協議会，財団法人日本情報処理開発協会 電子商取引推進センター，"モバイル EC に関するセキュリティガイドライン，" http://www.ecom.jp/results/h15seika/20_モバイル EC に関するセキュリティガイドライン.pdf，東芝ドキュメンツ株式会社，Mar. 2004.

4) 増井俊之，"インターフェイスの街角 (96) - マイ認証，" http://pitecan.com/UnixMagazine/PDF/if0603.pdf，UNIX MAGAZINE 2006.3，株式会社アスキー，2006.

5) 宇根正志，松本勉，"生体認証システムにおける脆弱性について：身体的特徴の偽造に関する脆弱性を中心に，" 金融研究第 24 巻第 2 号，日本銀行金融研究所，Dec.2005.

6) Ton van der Putte and Jeroen Keuning, "Biometrical Fingerprint Recognition: Don't Get Your Fingers Burned," Proceeding of IFIP TC8/WG8.8 Fourth Working Conference on Smart Card Research and Advanced Applications, Kluwer Academic Press, 2000, pp. 289-303.

7) T. Matsumoto, Hiroyuki Matsumoto, Koji Yamada, Satoshi Hoshino, "Impact of Artificial "Gummy" Fingers on Fingerprint Systems," Proceedings of the Conference OpticalSecurity and Counterfeit Deterrence Techniques IV, Part of IS&T/SPIE's Electronic Imaging 2002, pp. 275-289, 2002.

8) Aaron Ligon, "An Investigation Into the Vulnerability of the Siemens ID Mouse Professional Version 4," 2002.

9) Lisa Thalheim, Jan Krissler and Peter-Michael Ziegler, "Body Check: Biometric AccessProtection Devices and their Programs Put to the Test," c't, p. 114, 2002.

10) Johan Blommé , "Evaluation of biometric security systems against artificial fingers," http://www.diva-portal.org/diva/getDocument?urn_nbn_se_liu_diva-1145-1__fulltext.pdf, 2003.

11) Marie SandstrAöm , "Liveness Detection in Fingerprint Recognition Systems," http://www.divaportal.org/diva/getDocument?urn_nbn_se_liu_diva-2397-1__fulltext.pdf, 2004.

12) 堀内かほり，“濡れた指、乾燥した指—指紋認証の実際，” 日経バイト 2005 年 4 月号，p. 60-67，日経 BP 社，2005.
13) 國米仁 “奇怪論理と優良誤認に脅かされる情報セキュリティ，” http://www.mneme.co.jp/data/thesis3.html，JSSM 第 19 回全国大会，2006.
14) “「MagicConnect NDL」によるサーバ・ベース・コンピューティング (SBC)，” http://www.magicconnect.net/index.html，NTTIT，2007.
15) 情報通信ネットワーク産業協会，“2007 年度 携帯電話の利用実態調査，” http://www.ciaj.or.jp/content/pressrelease07/070718.html，2007.
16) 楽天リサーチ，三菱総合研究所，“2005 年第 18 回携帯電話コンテンツ/サービス利用者調査結果 (下)，” http://research.rakuten.co.jp/report/20051110/，2005.
17) 遠藤 大礎，川原 圭博，浅見 徹，“モバイル端末向け自己暗号化法による暗号化ファイルシステムの設計，” 2008 信学会ソ大，B-15-3，2008.
18) 椿山英樹，“au ケータイで実現する新しいオフィススタイル，” http://www.kddi.com/business/news/information/iexpo2004/pdf/officewise.pdf，KDDI 株式会社，Dec.2004.
19) “携帯電話のデータ通信速度の進化，” NTT ドコモレポート NO.40，Feb.2006.
20) 岩城俊介，““Rev.A” アップロードは 12 倍高速になるか—— W47T，” ITmedia +D モバイル，http://plusd.itmedia.co.jp/mobile/articles/0701/05/news 087.html，2007.
21) 川島潤，舩曳信生，中西透，竹内順一，石崎雅幸，Andre Caldas de Souza，“メモリカードのためのセキュアファイルシステム SAS の提案と実装，” 情報処理学会論文誌，Vol.47, No.8, pp. 2234-2395, 2206
22) 遠藤大礎，川原圭博，浅見徹，“自己暗号化法による分散ストレージシステムの検討，” 信学技報, IN, Vol.106, No.578(20070301) pp. 351-356, 2007.
23) Y. Kawahara, H. Endo, and T. Asami, “A Key Generation Scheme of Self-Encryption Based Mobile Distributed Storage System,” In Proceedings of The 6th International Workshop on Wireless Information Systems (WIS 2007), 2007.
24) Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray and San Vo, “NIST Special Publication 800-22,” http://csrc.nist.gov/rng/SP800-22b.pdf, National Institute of Standards and Technology, May 2001.
25) Filesystem in Userspace, http://fuse.sourceforge.net/
26) G. Clemm, J. Amsden, T. Ellison, C. Kaler, J. Whitehead, “Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning),” RFC3253, 2002.
27) “WILLCOM SIM STYLE 開発プラットフォーム Sandgate W-SIM Phone，” http://www.sophia-systems.co.jp/ice/intel/pxa27x/sgwp/index.html.
28) “Federal Information Processing Standards Publication 197,” National Institute of Standards and Technology, Nov.2001.
29) 総務省，経済産業省，“電子政府推奨暗号リスト，” http://www.soumu.go.jp/joho_tsusin/security/img/cryptrec01.pdf，2003.