P2P ファイル共有システムにおける鍵管理効率化手法の実装評価

大津一樹[†] 井上亮文[††]
宇田隆哉[††] 松下温[††]

本論文は P2P ファイル共有システムにおいて，効率的な鍵管理手法によるユーザマネジメント機能の実現とおよび実装・評価について述べている．P2P ファイル共有システムとは小規模なグループを対象に個人の使用する PC を，P2P 技術を用いて連携することで，ファイルサーバなどを導入するコストをかけずにファイルサーバと同等の機能を有し，情報漏洩に対する耐性も兼ね備えたシステムである．この P2P ファイル共有システムにおいてファイル共有を行う際に，従来手法では公開鍵を用いていたが，本論文では共通鍵を適用することで，ユーザマネジメントや鍵更新といった新たな機能を実装し，実行時間などの評価を得ることで，本手法の有効性を確認している．

An Implementing Evaluation of an Efficient Key Management Method in a P2P File Sharing System

Kazuki Ohtsu,[†] Akifumi Inoue,[††] Ryuya Uda[††]
and Yutaka Matsushita[††]

This paper is an implementing evaluation of an efficient key management method in a P2P file sharing system. P2P file sharing system is a system that has a function same as a file server and tolerance of leaking an information with low cost by P2P technology. Sharing a file in an existing system is used a RSA-key. A new system has a new method which is using an AES-key, so it can manage to a user and update an AES-key. And, this paper is described effectiveness of this method.

1. はじめに

PC の普及とネットワーク技術の進歩により個人が持つファイルをネットワーク上のファイルサーバに保存し，複数のユーザ間で共有するということが広く普及してきた．しかしながら，ファイルサーバはいまだ高価なものであり，導入後も信用のおける管理者が日常的にメンテナンスを行う必要があるなど，人件費の面からみてもコストの増加につながってしまう．

企業や大学などの比較的大きな組織単位であればファイルサーバを利用することも可能であると考えられるが，最近では，ファイルサーバ内の情報が一括して流出してしまうなどの問題も多い．また，2005 年 4 月より個人情報保護法が施行され，企業の情報管理が厳しく問われるようになってきた．しかしながら，サーバ管理者による不正や，人為的ミスによる情報の流出などがいまだ増加していることはいうまでもない．

このように，ファイルサーバを用いて情報の一括管理を行うことが現在では一般的な情報共有方法として利用されているが，ファイルサーバを用いるための管理費・人件費などを考えると小規模なグループでは安易に導入することは難しいといえる．

このような背景の下，小規模なグループ内の個人が使用する PC を P2P で連携させ，各 PC の余剰ディスクスペースにファイルを分散保存する．そして，保存されたファイルを安全に共有することを可能にする P2P ファイル共有システムが提案・実装された[1)~4)]．また，DICOMO2005 において本システムの鍵管理効率化手法を提案しその有用性をあげた[5)]．

そこで，本論文では，P2P ファイル共有システムにおけるファイルの共有時の鍵管理効率化手法を実装する際に従来の手法とどの程度の実行速度の差があるのかを求め，テストを行い，実装可能性を確かめることにする．

† 東京工科大学大学院
Graduate School, Tokyo University of Technology
†† 東京工科大学
Tokyo University of Technology

2. 関 連 研 究

2.1 完全分散型ファイル共有システム

現在発表されているファイルの分散管理手法[6),7)]は，完全分散型と呼ばれるファイル共有システムにあたる．完全分散型ファイル共有システムとは，ネットワーク内に，ファイルサーバとなる PC を数台配置し，それらの PC に対して分割したファイルを保存するというもので，ファイルサーバを用いずに，同等の機能を果たすという，本システムの目的と同一のものとなる．この完全分散型システムと本システムとの差異は，ファイルサーバとなる PC が特別な PC（サーバとなる PC）かどうかである．つまり，完全分散型システムではファイル保存対象 PC が特定であり，それらの PC は通常のファイルサーバと同様つねに電源を入れておく必要があり，ネットワークへも常時接続しておかなければならない．しかしながら，本システムにおいて，ファイル保存対象 PC は個人が通常利用している PC で，PC の状態はユーザの利用形態に完全に依存することになる．結果として，P2P ネットワークを構築するための仕組みを考慮する必要が生じるが，特定の PC の導入や管理などを行う必要がなくコストもかからない．また，本システムの利用者は今までどおりに PC を利用することが可能となり，ユーザにとっての利便性向上につながると考えている．

ファイル共有の観点からしてみれば，完全分散型のシステムでは，保存されている PC がつねに一定になるため，その PC に接続する際にユーザ認証を行い，各ユーザの権限を確認するだけでファイル共有が行える．しかしながら，本システムでは，サーバ管理者などは存在せず，システムを利用する際の管理者や管理サーバによるユーザ認証は原則として行っていない．よってファイル共有を行う際には別の仕組みを構築している．その部分に既存システムとの大きな差異があると考えている．

2.2 ファイルの動的再配置

本システムではグループ内のユーザの PC にファイルの一部を保存することになるが，各ユーザ PC の稼働状況などは各ユーザに委ねられている．つまりファイルがそのときの状況により取得できない可能性も生じる．

そこで，現在検討している手法としてファイルの動的再配置というものがある．その手法についての概要を説明する．

まず，本システムを利用する際にユーザは公開鍵を用いた認証を行いログインすることになるが，その際にそのユーザが利用した PC の ID をもとにログイン時間をシステムに参加している各 PC が記録する．そして，そのユーザがログアウトする際にログアウト時間を同じく記録する．このようにシステムに接続してきた PC の ID をシステム中の PC それぞれがログとして記録していくことで，各 PC のトータル稼働時間や，時間帯における稼動率，が求められることになる．

このログはすべての PC が相互に記録し合うものなので，自分が稼動している時間帯にどの PC が稼動しているかということを求めることができる．お互いに稼動しあっている状況の多い PC を相性の良い PC として扱うことで，相性の良い PC に保存されたデータの取得確率が上昇するのである．しかしながら，すべての PC が長い期間まったく同じペースで利用されるとは限らず，この方法でも長い期間継続して行くと相性の良い PC が変わってくる可能性がある．そこで，つねにログを記録しつつ，相性の良い PC のランクが変動するたびに保存されているファイルを上位の PC に動的に保存しなおすことを行う．

こうすることで，ファイルの取得確率が上がるだけでなく，一部の PC にファイルが集中するといった問題の解決にもつながり，全体的なシステムの効率化が図れるのである．

3. P2P ファイル共有システム

本章では，参考文献 1)〜5) にあげる P2P ファイル共有システムにおける，ファイルの保存・復元・共有について概要を説明する．

3.1 ファイル保存

本システムでは，保存されるファイルはすべて共通鍵を用いて暗号化された後，冗長性を持たせつつ複数に分割されネットワーク上の PC へ保存される．このネットワークとは複数の個人それぞれが使用している PC 十数台で構成される LAN などで，ネットワークへの参加離脱が自由な環境を想定している．ファイルを暗号化するための共通鍵はファイルの保存を行うたびに生成されるもので，ファイルを取得するためのプロパティファイル（復元情報ファイル）として保存される．このプロパティファイルにはオリジナルファイル（保存されたファイル）の部分データが何というファイル名で，どの PC に保存されているか，などといった情報も含まれる．ファイル名を記録しておく理由は，本システムでは，保存されるファイルすべてに 128 文字のランダムな名前が付けられることになるからである．

次にこのプロパティファイルも部分データと同様に

ネットワーク上の PC に保存されることになるが，ここではオリジナルファイルの所有者個人の公開鍵を用いて暗号化し，保存する．個人の公開鍵とはユーザがシステムを初めて利用するときに公開鍵暗号方式に基づき生成されるもので，ユーザの認証に利用する．プロパティファイル名も 128 文字のランダムな文字列で構成されるため，各ファイルのプロパティファイル名と保存先 PC を列挙したものをリストファイルと呼ぶファイルに記録している．

リストファイルはオリジナルファイルに 1 対 1 対応するプロパティファイルとは異なり，ユーザに 1 対 1 対応し，所持者であるユーザが現在保存しているファイルの一覧が記録されるものとなる．このリストファイルも所有者個人の公開鍵を用いて暗号化されネットワーク上に保存される．リストファイルにも他のファイル同様に，保存される際には 128 文字のランダムなファイル名が付けられるが，リストファイルはユーザが本システムを利用する際に，最初にアクセスしなければならないファイルである．リストファイルが得られなければ，プロパティファイル名を認識することができず，オリジナルファイルの復元が不可能となってしまう．よってランダムなファイル名が付けられていては，つねにリストファイルに最初にアクセスすることが不可能となってしまうのである．

そこで本システムでは次の方法でこの問題を解決している．ユーザの持つ公開鍵・秘密鍵のペアに着目し，まず，公開鍵データを，秘密鍵を用いて暗号化する．得られた暗号化済みデータをインクリメントした値からハッシュ値（SHA-1）を求める．これらの処理を，インクリメントを繰り返しながら 7 回行うと，20 Byte（= 160 bit）のデータが 7 個得られる．その 140 Byte のデータのうち，前半の 128 Byte を Base64 変換して文字列に変えたものをリストファイル名とするのである．この方法を用いることで，公開鍵・秘密鍵から一定の名前を求めることが可能となり，リストファイル名を一定とすることが可能となる．

3.2 ファイル復元

3.1 節で述べたように，ユーザはリストファイルをつねに取得可能となり，リストファイルの内容も公開鍵とペアになる秘密鍵を用いて復号化することで，得ることができる．得られた情報より，現在保存されているファイルの一覧が把握でき，その中からアクセスしたいファイルと対応するプロパティファイル名が認識可能となる．ファイルの復元時には対応するファイル名のプロパティファイルを探しネットワーク上より取得する．得られたプロパティファイルを，秘密鍵を用いて復号化する．そしてプロパティファイルの内容である，ファイル名と PCID より部分データファイルを集め，復元する．最後にプロパティファイル内に含まれる共通鍵を用いて復号化しオリジナルファイルを復元する．

3.3 ファイル共有

ファイルの共有については 3.1，3.2 節の仕組みを基に行われる．

本システムにおいて，保存されているファイルにアクセスするためには，プロパティファイルを取得することが条件となる．しかしながら，このプロパティファイルは個人の持つ公開鍵で暗号化されるため，共有を行うためには，共有を許すユーザの公開鍵を用いてファイルの暗号化を行う必要がある．つまり**図 1** に示すとおり，共有ユーザの人数分暗号化されたプロパティファイルが存在することになり，負担となってしまう．また，共有ファイルの編集が行われた際に，共有ユーザ全員に対し，編集が行われたことを伝え，プロパティファイルを更新させる必要もある．

編集が行われたことを通知する理由は，ファイルが保存される際に部分データが保存される先の PC はつねに一定ではないからである．共有ファイルを保存・取得する際に同一の PC がつねにオンラインにいるとは限らず，また，保存する PC との相性の関係により，元々部分データを保存していた PC とは別の PC を選択した方が好ましい場合もあり，そういった理由から，共有ファイルの編集が行われた際には，必ず新しい保存先を全員に知らせることが必要となる．それにともない共有ユーザ全員分のプロパティファイルを書き換える必要性が生じるのである．さらに，ファイルの暗号化につねに同一の鍵を使い続けることは安全性の観点から好ましくないため，本システムではファイルを保存しなおすたびに，ランダムに生成した鍵を暗号化に利用している．よって，共有ファイルにおい

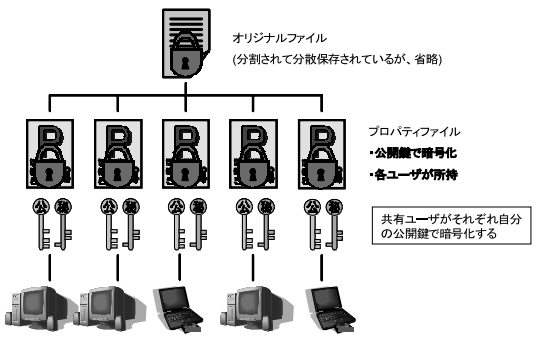


図 1 従来のファイル共有手法
Fig. 1 Existing method of file sharing.

ても，ファイルが再保存される際には，新しい鍵をプロパティファイルに再記述することも必要となるのである．

この方法では，共有ユーザ数に比例してファイル数や演算処理回数が増加してしまい，システム効率の低下につながってしまう．また，前述のとおり，共有ファイルを保存し直すたびに共有ユーザの公開鍵を用いて全員分のプロパティファイルを再記述する必要があるが，他人の公開鍵をシステム中で安全に受け渡す方法が未定であるため，システムとして現実的ではない．さらに，1 つの共有ファイルに対応するプロパティファイルが複数存在するため，排他制御が困難であるといった問題も生じる．

4. 提案手法

本章では，ファイル共有の提案手法について説明を行う．この手法は 3 章で述べたシステムの仕組みを前提とするので，プロパティファイルを暗号化し共有することで，ファイルの共有を実現するという方法は従来の手法[3] と同等になる．

4.1 共有用共通鍵

本手法では，プロパティファイルを暗号化する際の鍵を，従来の手法[3] である公開鍵ではなく，共通鍵暗号方式に基づく共通鍵とする．この共通鍵はオリジナルファイルを保存する際の暗号化に用いるものとは別のもので，ファイル共有用の共通鍵である．この共通鍵は共有ファイルが保存される際に自動的に生成され，共有ユーザ全員が所持することになる．以後この共通鍵を共有用共通鍵と表すことにする．

あるファイルを複数のユーザ間で共有する場合，通常のファイル保存過程と同様に共有ファイル（共有されるオリジナルファイル）は共通鍵により暗号化され，分割された後ネットワーク上の PC へ保存される．そして，システムにより生成された，ファイルの暗号化を行った共通鍵や，部分データが保存されたノード情報を記録したプロパティファイルを，共有用共通鍵を用いて暗号化し，他のプロパティファイルと同様に，ネットワーク上の PC に保存する．さらに，共有ユーザすべてにリストファイルの更新を依頼し，共有ファイルの保存を完了する．

本システムでは，ネットワークを構築する PC は各個人が利用する一般の PC であることから，ネットワークへの参加離脱は各個人の自由に行われる．よって，ファイルが保存された時点で共有ユーザすべてがオンライン（ネットワークに参加している状態）であるとは限らず，オフラインユーザに対してもリストファイルの更新を伝える必要が生じる．オフラインユーザへ情報伝達手法としての書置きという方法を利用する．**図 2** に詳細を示す．

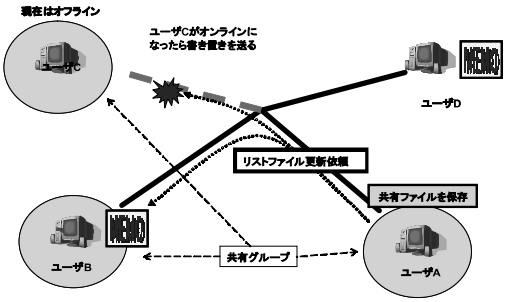


図 2 書き置き手法
Fig. 2 Method of leaving a message.

図 2 のようにユーザ A が共有ファイルを保存したとき，ユーザ A は共有ユーザであるユーザ B，C に対してリストファイルの更新を依頼する．しかし，オフラインのユーザ C には更新依頼を伝えることができないため，ユーザ A はその時点でオンラインであるユーザ B，D に対してユーザ C 宛の書置きを残す．この書置きにはリストファイルの更新内容である，共有ファイル名と共有ファイルを復元するためのプロパティファイル名さらに，プロパティファイルの保存先ノード情報が記録されることになる．そして，ユーザ C がオンラインとなった際に，ユーザ B，D は書置きがあることをユーザ C に伝え，書置きを送信する．書置きを受け取ったユーザ C は内容を解析し処理を行うことで，リストファイルの更新を行う．

共有ファイルにアクセスする際には，各共有ユーザは共有用共通鍵を用いてプロパティファイルを復号化する．プロパティファイルを保存しているノードは自分自身のリストファイルを参照することで認識可能であるので，自力で取得してくることが可能である．

本手法を用いることで，保存されるプロパティファイルの数が共有ユーザ数に関係なくある程度少数に抑えることが可能となる．具体的な値は 7.3 節で示すことにする．また，処理負荷の多い公開鍵暗号方式の演算を行う必要がなくなり，3.3 節で述べた，保存されるファイル数と演算処理回数が増大してしまう問題を解決することが可能となる．

4.2 ハッシュ連鎖を用いた共有用共通鍵の更新

提案手法ではプロパティファイルを共有用共通鍵で暗号化することで，ファイル共有を実現しているが，1 つの共有グループがファイルの暗号化を行う際に，つねに固定の鍵を利用し続けることは，安全性の観点から好ましくない．保存されるプロパティファイルは，

毎回異なる 128 文字のランダムなファイル名が付けられるため，プロパティファイル自体を特定することは困難であるが，共有ユーザ全員が共通のプロパティファイルにアクセスするため，特定される危険性は高いといえる．また 1 つのプロパティファイルの暗号化が破られてしまった場合に，他のプロパティファイルすべての暗号化も破られてしまうという問題も生じる．

そこで，共有用共通鍵については一定時間ごと，もしくは新たなファイルが保存されるたびに鍵の更新を行うことにする．本システムでは前述のようにすべてのユーザがネットワークへの参加・離脱を自由に行うことが可能であるので，鍵の更新を行う時点ですべての共有ユーザがオンラインであるとは限らず，オフラインのユーザに対しても新しい鍵を知らせることが必要となる．そこで，鍵の更新方法としてハッシュ連鎖とタイムスタンプを利用する．鍵生成からある時間 t が経過したら現在の鍵のハッシュ値を求め，得られたハッシュ値を次の鍵として利用することにすれば，その時点でオフラインのユーザであってもオンラインになった後に自ら鍵の更新が行えるようになる．

このハッシュ連鎖を用いるもう 1 つの理由として，共有ユーザの追加が効率的に行えるということもあげられる．その詳細を**図 3** に示す．

図 3 のようにファイル A，B，C を共有するユーザグループが存在するとする．共有ユーザ達が持つ共有用共通鍵はすでに数回の更新が行われていて，更新が行われたタイミングを，それぞれ t1，t2，とし，その時点での鍵をそれぞれ K1，K2 とする．そして，ある時点 T において新たな共有ユーザ N がこの共有グループに加わり，ファイル D の共有を行いたいとすると，まず共有ユーザたちは共有用共通鍵の更新を行う．そして，更新済みの共有用共通鍵 K3 を新規共有ユーザ N に渡す．そしてファイル D を保存した際に生成されるプロパティファイルに関して，K3 を用いて暗号化を行う．わざわざ鍵の更新を行った後にファイルの保存を行う理由は，以前から保存されているファイル A，B，C は新規ユーザ N に対し公開してはならない場合が考えられるからである．新規ユーザ N が持つ鍵は K3 であるから，今後この共有グループで共有されるファイルは K3 以降の鍵 $K4, K5, \cdots$ で暗号化されることになる．よって新規ユーザ N も鍵の更新が可能となるが，ハッシュ関数の特性である，一方向性により，K3 以前の鍵は求めることができない．よって新規ユーザ N が共有グループに加わる以前に保存されているファイル A，B，C に関してはアクセスが不可能なのである．もちろん，新規ユーザ N に対してファイル A，B，C を公開してもかまわないのであれば，K1，K2，などの鍵を教えることで，以前に保存されたファイルへのアクセスを段階的に許可することも可能である．このようにハッシュの連鎖による鍵更新方法を用いれば，動的にユーザの追加が行えることになる．

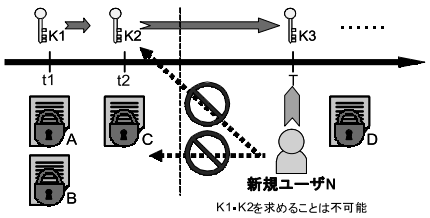


図 3 ハッシュ連鎖によるアクセスコントロール
Fig. 3 Effective access controls by hash chain.

共有ユーザの追加とは逆に共有グループからある共有ユーザを除きたい場合は，元の鍵自体を変更する必要がある．つまり，ある共有用共通鍵 Ka から派生する鍵は $Ka1, Ka2, \cdots$ となるが，共有グループから除かれた共有ユーザはこれらの鍵を知っているので，今後更新した鍵も求めることができてしまうのである．よって，このような場合には元の鍵自体を変更し新たな鍵の連鎖 $Kb1, Kb2, \cdots$ を作る必要がある．共有グループから離脱したユーザに対して，過去に共有されていたファイルへのアクセスを許可するのならば，Ka で暗号化されている共有ファイルはそのまま残すことで過去のファイルへのアクセスは可能となり，アクセスを許可しないのであれば，新たな鍵の連鎖 Kb を用いて過去のファイルを保存（暗号化）することでアクセスを不可能にする．また，ファイルの更新が行われた後に Kb を用いて暗号化を行うことで，最新のファイルにはアクセスを許さないなどの段階的なアクセス制御が可能となる．

4.1 節で述べた書置き手法を用いれば，オフラインユーザに対し鍵の更新を知らせることも可能となるが，鍵の更新のたびに書置きを残す必要があり，システム全体の負荷の増大につながる．ハッシュ連鎖の利用により，システムの効率化が図れ，さらに，ユーザの追加や削除といった機構にも対応可能となることから，本手法が有効であると考える．

4.3 提案手法におけるファイルの編集

3.3 節で述べているように従来手法においては共有ユーザ全員分のプロパティファイルを書き換える必要性が生じていたが，提案手法においては，その必要がなくなる．

なぜならば，提案手法において，保存された共有ファ

イルにアクセスするために必要なプロパティファイルは 1 つとすることが可能となる．つまり，保存された共有ファイルの編集を行った場合，編集後のファイルを保存し直して，部分データが新たな PC に保存されたとしても，新しい情報を書き込む必要があるプロパティファイルは 1 つとなるからである．また，ファイルが保存される際に暗号化に使われるランダムに生成される鍵情報に関しても，記述するプロパティファイルは 1 つで済むことになる．

5. 実　　装

今回実装した機能は，4.2 節で述べたハッシュ連鎖による鍵更新機能である．実装の詳細としては以下で述べることにする．

5.1 ハッシュ関数

ハッシュ連鎖に用いるハッシュ関数のアルゴリズムとしては SHA-1 を採用した．これは既存のシステム中で用いられているアルゴリズムで，容易に拡張が可能であったからである．

また，本システム中で用いられている共通鍵アルゴリズムは AES であるが，AES の鍵長は 128 ビットを選んでいる．つまりハッシュ関数の出力値のビット数は 20 バイトとなり，AES の鍵長は 16 バイトとなる．そこで本実装では**図 4** のとおりにハッシュ連鎖を行うこととした．

図 4 のようにハッシュ連鎖を用いることで，ハッシュ関数 SHA-1 の最大の強度を利用できることになり，出力値から元の値を求められる可能性を最低限に抑えている．

5.2 ハッシュ連鎖

次に実装システムで用いられるハッシュ連鎖の回数であるが，これは共有用共通鍵を更新する頻度に比例して増加することになる．また，セキュリティ上の観点から見て鍵を定期的に更新する必要があると考えられるので，ファイルが更新されていなくても自動で鍵を更新することも想定した．

そこで問題となるのが，どの程度の頻度で鍵を更新するかということである．たとえば，1 日ごとに鍵を更新することにした場合，その日のうちに保存されたファイルはすべて同じ鍵で暗号化されることになる．また，共有ファイルであることから，複数人が頻繁にファイルの更新を行うことが考えられるので，1 日のうちに何十回もファイルの更新が行われる可能性もある．しかしながら，従来の手法である個人の公開鍵を用いてファイルを共有した場合と比較するとハッシュ連鎖の回数によっては実行速度に差が認められるということが考えられ，そのことによりファイル共有時のシステム負荷が増加してしまってはシステム全体としての価値を損なってしまうことにつながる．

そこで 6 章で，ハッシュ連鎖回数別の実行時間と公開鍵暗号方式を用いた従来手法の実行時間との差を計算し本提案手法が従来の手法よりも優れていることを確認する．

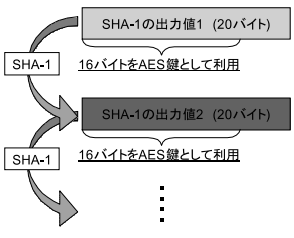


図 4 ハッシュ連鎖の方法
Fig. 4 Method of hash chain.

6. 評　　価

これまで，提案手法の機能について説明してきたが，この章では，提案手法によるファイル保存・復元が従来手法によるファイル保存・復元と，どの程度，処理性能に違いがあるのかを求め，評価した．この評価から得られる結果により，提案手法を実装しシステムに組み込む意義があるのかどうかを定量的に判断することが可能となる．

ここで，求める評価としては，本提案手法はファイル共有時の暗号化・復号化に用いられる手法であることから，従来手法である公開鍵を利用した手法と提案手法の共通鍵＋ハッシュ連鎖を利用した手法とで，実際にファイルの暗号化・復号化にかかる処理時間を測定した．

評価環境としては**表 1** のとおりである．

図 5 で示すグラフは従来手法，提案手法のそれぞれにおける暗号化にかかる処理時間でファイルサイズが 1 KB〜10 MB まで，提案手法でのハッシュ連鎖回数が 1,000，5,000，10,000 回の場合のデータを表していて，4 つのデータのうち “RSA” が従来手法を，“AES ＋ hash” が提案手法を示している．同様に，**図 6** に関しては，従来手法，提案手法のそれぞれにおける復号化にかかる処理時間を図 5 と同様の条件で表してい

表 1 評価環境
Table 1 Evaluation environment.

CPU	Pentium4 3.00GHz
MM	512MB
OS	WindowsXP Pro. SP2

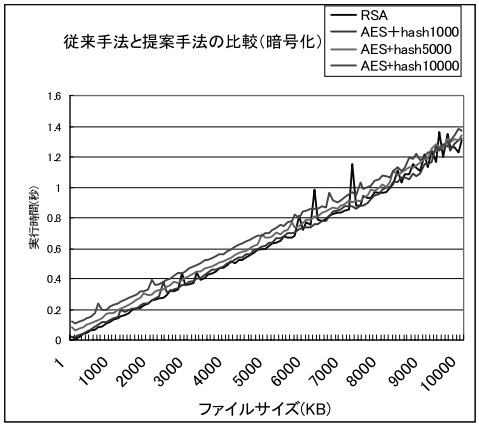


図 5 暗号化にかかる処理時間の比較
Fig. 5 Comparison of processing time of encryption.

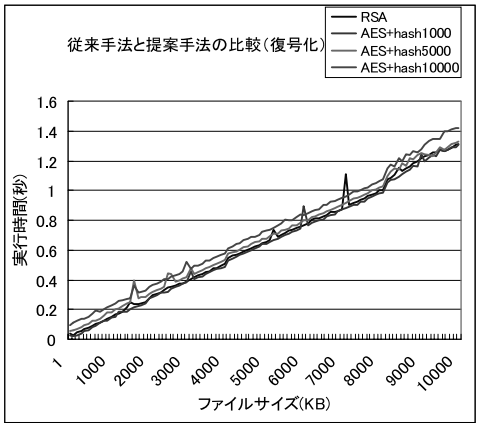


図 6 復号化にかかる処理時間の比較
Fig. 6 Comparison of processing time of decryption.

る．**図 7** で示すグラフは，ファイルサイズが 10 MB〜30 MB のときの両手法における処理時間を表している．図 7 ではハッシュ連鎖回数を 1,000 回，10,000 回の 2 つのみを載せている（各グラフで用いられている AES+hashXXX という表現は，ハッシュ連鎖の回数を示している）．

3 つのグラフとも，横軸は暗号化・復号化される対象のファイルサイズ（KB）を，縦軸は，暗号化・復号化にかかる処理時間（秒）を，それぞれ表している．

ここで，暗号化・復号化について説明を加えると，本システムにおいては，ファイルを暗号化する際の手順として，まず，暗号化するごとにランダムに生成した共通鍵による，ファイル自体の暗号化を行い，次に，従来手法の場合公開鍵を，提案手法の場合共通鍵にハッシュ連鎖を行い得られた鍵を，それぞれ利用してファイル自体の暗号化に利用した共通鍵を暗号化し，

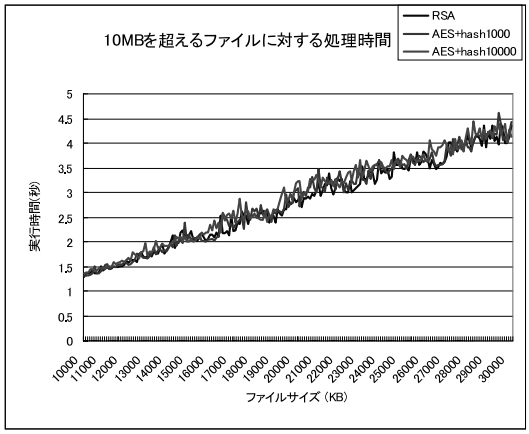


図 7 大きなファイルに対する処理時間
Fig. 7 Processing time to big file.

暗号化済みファイルデータの後ろに付加するという処理を行う．また，復号化処理に関してはこの手順の逆をたどることになる．これらの処理を総じて暗号化・復号化処理として表現している．

図 5，図 6，図 7 を見ると実行時間はともにファイルサイズに比例することが分かり，特にハッシュ連鎖回数が 1,000 回のデータ（図 5，図 6 において，原点付近から伸びているデータ）に限れば，両者は，グラフ上同程度の値を示していることが分かる．具体的な値としては，提案手法におけるファイルサイズが 5 MB 以上 10 MB 以下で，ハッシュ連鎖回数が 1,000 回の場合と，従来手法におけるファイルサイズが 5 MB 以上 10 MB 以下の場合とを比較すると，およそ 1.5%処理速度が高速化することも認められた．ただし，逆にファイルサイズが 5 MB 以下の場合は，従来手法におけるファイルサイズ 5 MB 以下の場合と比較し，ハッシュ連鎖回数が 1,000 回のとき，およそ 4%遅延してしまうことも明らかとなった．

これらのことから，ハッシュ連鎖回数を 1,000 回以下に抑えるよう更新頻度を制限すれば，ファイル共有時の手法として，従来手法と同程度の処理性能となるということがいえる．そこで，ハッシュ連鎖回数を最大で 1,000 回までとし，1,000 回を超えた時点で新しい共有用共通鍵を採用しなおすことで，1,000 回以上の鍵更新を行わないよう制限することが好ましいと考えられる．

また図 7 から，ファイルサイズが 20,000 KB（≒ 20 MB）の時点で実行時間がほぼ 3 秒となっていることが分かる．つまり，従来手法・提案手法両方において，ファイルサイズが 20 MB 以上ともなれば，暗号化・復号化の処理に 3 秒以上かかることから，ユーザ

にとっては多少のストレスが感じられるのではないかと考えられる．

このように，ハッシュ連鎖回数を 1,000 回以下に制限することで，従来手法と提案手法はほぼ同等の処理性能を示すこととなり，その他の機能的な面で有効性が認められる本提案手法をシステムに組み込む意義はあると考えられる．

7. 考　　察

7.1 ファイル共有時の計算量

本節では，従来手法と提案手法のそれぞれについて，ファイル共有時の計算量についての考察を示す．

まず，従来手法の場合，新たに共有ファイルが保存されたときに全員分のプロパティファイルを作成し全員分の公開鍵を用いてそれを暗号化する必要が生じる．それに対し，提案手法では，共有ユーザ間で共通のプロパティファイルを持つことになり，それを暗号化している鍵も全員が所有し合うため，行わなければならない処理は，1 つのプロパティファイルの暗号化のみである．

6 章で，ファイルの暗号化にかかる処理時間を計測しているが，図 5，図 6，図 7 において，ハッシュ連鎖回数が 1,000 回で，ファイルサイズが 5〜10 MB の範囲に注目すれば，提案手法の方が優位であることが分かり，それ以外の範囲の場合は，ほぼ同一の処理性能を示していることから，従来手法である RSA を用いた場合と AES ＋ハッシュ連鎖を用いた場合とでは，処理性能は等しいといってかまわないと考えられる．

よって，両手法の計算量は次のとおりとなる．前述のとおり，従来手法の計算量は暗号化演算の共有ユーザ数倍となり，逆に提案手法では暗号化処理はつねに 1 回でよい．そこで，次のような条件で実際にどの程度計算量が削減できるのかを求める．

まず，本システムの想定利用環境は小規模なグループとしているが，具体的には 15 人以内のグループと考えている．よって，1 つのファイルを共有する人数を最大 15 人，そして共有するファイル数を 10 ファイルとした（各共有ファイルのサイズは 10 MB 以下とする）．

たとえば，上記のような条件のもと，共有ユーザ数が 10 人であったとすると，従来手法では，10 回の暗号化演算を行わなければならないが，提案手法では 1 回の暗号化演算 ＋ ハッシュ連鎖となり，計算量としては，従来手法の 10%の量で済むことになる．しかしながら，共有ユーザ数が 2 人のように少人数である場合に関しては従来手法においても提案手法のおよそ 2 倍の処理負荷しかかからず，提案手法でも計算量を半減するだけの効果しか得られないことになる．

これらのことから，なるべく多くの共有ユーザが存在している状況の方が提案手法の優位性が示せることが分かる．

7.2 共有ファイル編集時の通信回数

4.3 節にあるとおり，従来手法では共有ファイルの編集を行った際に，共有ユーザ全員分のプロパティファイルを再記述しなければならない．つまり，再記述したプロパティファイルを再びネットワーク上に保存しなおすということになる．

しかしながら，提案手法においては，1 つのプロパティファイルの再記述のみでファイル編集が行えることになり，ネットワーク上に保存するプロパティファイルは 1 つということになるので，それぞれの手法における通信回数は次の式 (1)，式 (2) で表すことができる．

$$C = User \times Pro \tag{1}$$

$$C' = 1 \times Pro \tag{2}$$

C，C'：総通信回数
$User$：共有ユーザ数
Pro：プロパティファイル数

ここで，式 (1)，式 (2) におけるプロパティファイル数 Pro を 1 とし，共有ユーザ数を 1，3，5，10，15 人としたときのそれぞれの手法の通信回数を**表 2** に示す（共有ユーザ数は 7.1 節で示した本システムの想定している利用環境から考えられる条件をもとにしている）．

通信回数の削減率に関しては，提案手法と従来手法の商を求め 1 から引いたものを百分率で表したものとした．なお，オフライン共有ユーザがいた場合の書き置きを残す処理に関する通信も 1 回として求めている．

表 2 を見ると，たとえば，共有ユーザが 15 人いた場合には，提案手法の方が通信回数を約 93%削減する

表 2 通信回数の比較
Table 2 Comparison of communication frequencies.

共有ユーザ数	従来手法通信回数	提案手法通信回数	通信回数削減率
1 人	1	1	0%
3 人	3	1	67%
5 人	5	1	80%
10 人	10	1	90%
15 人	15	1	93%

（単位：通信回数［回］）

ことが可能となることが分かる．

さらに，5 人の共有ユーザ数であっても，80%も通信回数が削減できることになり，提案手法の有効性が確認できる．

7.3 保存されるデータ量

この節では，従来手法と提案手法のそれぞれでどの程度プロパティファイルのデータ量に差が生じるのかを示す．

まず，従来手法で保存された場合の総プロパティファイルデータ量 D と提案手法によるもの D′ は次の式 (3)，式 (4) となる．

$$D = User \times data_{pro} \times file \quad (3)$$

$$D' = 1 \times data_{pro} \quad (4)$$

$$\left[\begin{array}{l} D，D'：総データ数 \\ User：共有ユーザ数 \\ data_{pro}：プロパティファイルサイズ \\ file：共有ファイル数 \end{array}\right]$$

式 (3) に具体的な値を代入して得られた結果を次に示す．ここでは，オリジナルファイルを 5 個に分割し，冗長性 6 割の場合のプロパティファイルサイズをもとに計算を行う（参考文献 3) より，760 バイト）．

ここでも，7.1 節で示した条件をもととして，共有ユーザ数 2，5，10，15（人），プロパティファイルサイズは 760（バイト），共有ファイル数 1，5，10（個）として計算を行った．結果は**表 3** に示すとおりとなる．表 3 の列は共有ファイル数，行は共有ユーザ数をそれぞれ示している．

提案手法となる式 (3) の場合は，総データ量はプロパティファイル数に比例するので，データ量としては最低で 760 バイト，最大で 7,600 バイトとなる．よってこのことから，15 人で 10 個のファイルを共有した場合で約 94%，2 人で 1 個のファイルを共有した場合でも 50%のデータ量が削減できたことになり，提案手法の有効性が確認できたといえる．

7.4 共有グループからのユーザ離脱

本提案手法の問題点として考えられることは，共有グループからユーザが離脱した際の処理である．前述のとおり共有グループに新たなユーザが追加された場合においては，ハッシュチェーンを利用することで効率的なアクセスコントロールを可能としユーザ管理が容易に行えるが，共有グループからユーザが離脱した場合には同じ鍵系列を用いてしまうと離脱したユーザもその鍵系列を求めることが可能となってしまい，本来ならば閲覧する必要のない（権限のない）ユーザがファイルを閲覧することが可能となってしまうのである．

表 3 従来手法におけるデータ量
Table 3 File size of Existing method.

	1 個	5 個	10 個
2 人	1520	7600	15200
5 人	3800	19000	38000
10 人	7600	38000	76000
15 人	11400	57000	114000

(単位：［バイト］)

この問題点を解決する方法として，鍵系列自体を交換してしまうということを考えている．鍵系列の交換とは，その共有グループが現在保存している共有ファイルを新たな root 鍵（ハッシュ連鎖のもととなる鍵）を用いて再暗号化することである．この再暗号化はシステムがバックグラウンド処理として自動で行うことになるが，実際にどの程度の負荷がシステムにかかるのかを以下の条件をもとに考える．

まず，7.1 節で示しているとおり，本システムの想定利用環境は企業などのプロジェクトグループで 15 人以内のグループと考えている．そこで，企業などのプロジェクトグループにおいてプロジェクト進行中に，多数のグループメンバが入れ替わることは考えにくいということから，最大でも 15 人中 3 人が離脱するという条件で考察した．

この条件でユーザが離脱した場合にアクセスコントロールを行うために鍵系列を再構築しなければならない最多回数は，ユーザが離脱する回数と等しくなり 3 回となる．ただし，限りなく短い時間に連続してユーザが離脱しない限り，鍵系列の再構築は十分な間が空いて行われる処理となるので，1 回にかかる処理負荷を求めることでユーザ離脱に関する処理負荷とすることができる（短い時間に連続してユーザが離脱した場合まとめて処理を行うことが可能となるので 1 回と見なしても問題ない）．

鍵系列を 1 回再構築するために必要な演算回数はプロパティファイルの再暗号化にかかる処理となるので，旧共有用共通鍵によるプロパティファイルの復号化と新共有用共通鍵によるプロパティファイルの暗号化の回数のプロパティファイル数倍となり，前述の条件では 20 回となる．ここで，ファイルの暗号化・復号化時にハッシュ連鎖を行うことになるが，6 章の評価よりハッシュ連鎖回数が 1,000 回以内であれば，ほとんど実行時間に差が出ないことから，この条件でシステムにかかる処理負荷は 20 回のファイルの暗号化・復号化にかかる処理負荷とほぼ等しくなる．20 回の暗

号化・復号化にかかる処理時間は 6 章の図 5，図 6 からも分かるとおり，ファイルサイズにほぼ比例することになるが，ファイルサイズが 10 MB 以下の場合（6 章の図 5，図 6 を参照），処理時間はおよそ 1.4 秒以下となるので，処理時間の合計は最大でも 28 秒ということになる．この処理時間はシステムがアイドル中にバックグラウンドで処理する分には十分問題ない値であるといえ，このことから提案手法におけるユーザ離脱にかかる処理負荷は，小さいものであると考えられる．

また，この鍵系列の交換を繰り返し行った場合でも共有グループに残っているユーザ（離脱していないユーザ）が管理する鍵数は変わらないことも本提案手法の利点である．つまり，共有ユーザにとっては単に所有する root 鍵が変わるのみで，その他に新たに管理する鍵や，パスワードなどの情報は必要ないのである．これらのことからも，提案手法によって鍵管理を効率化できることが分かる．

7.5 共有グループごとの鍵系列管理

提案手法における共有用共通鍵の管理について，共有グループが増加するとともに，共有グループごとに管理しなければならない鍵が増加してしまうという問題がある．たとえば，あるファイル F1 を共有するグループ A にユーザ $\{a1, a2, a3\}$ が属しているとする．また，別のファイル F2 を共有するグループ B にユーザ $\{b1, b2, b3, a1\}$ が属しているとすると，それぞれのユーザが管理しなければならない可能性がある共有用共通鍵は，ユーザ a2，a3，b1，b2，b3 に関してはそれぞれが属する共有グループ用の共有用共通鍵 KeyA，KeyB となるが，ユーザ a1 に関しては，KeyA，KeyB 両方の鍵を所有する必要が生じてしまうのである．

さらに，提案手法を用いて新しい共有グループを構築する場合に，既存の鍵を派生させて新共有グループ用の共有用共通鍵を生成することになるが，その際にグループ固有のパスワードのような値をハッシュ連鎖に組み込み，グループ外のユーザがその鍵を生成することを不可能にすることが必要となる．

これらのことが複雑に絡み合ってくると，ユーザが管理しなければならない鍵や，記憶しなければならないパスワードなどが膨大な量となってしまう可能性がある．

しかしながら，本システムの想定している利用環境においては，7.4 節で述べているように 15 人以内のグループと考えている．さらに，共有するファイル数は 10 個程度ということを考慮し，同じ目的を持ったプロジェクトメンバなどで構成されるグループであることを前提とすると，ユーザごとに複雑なグループ分けをすることは考えにくいといえる．よって，本システムが利用されるグループ内において構成されるであろう共有グループは最大でも 5 グループと考えることで，個人が管理する必要のある鍵は 5 個程度に抑えられることになり，膨大な数の鍵を管理しなければならないという問題を回避することが可能であると考える．

また，同一の理由から新グループ用鍵を生成する際に必要なパスワードに関しても，5 個程度で済むことになり，ユーザにとっての大きな負担にはならないと考えられる．このパスワード指定するユーザは共有グループにおけるリーダ的な役割を担うユーザとなってしまうことが考えられるが，このパスワードをどのユーザが決めても，そのユーザに対して特別な権限が与えられるわけではなく，共有グループを構成する際に一度だけ必要な処理となるので，システム上大きな問題になるとは考えにくく，管理者を設ける必要がないという本システムの特徴を維持することが可能となる．

しかしながら，利用形態によっては，管理すべき鍵系列増加してしまうという問題は，本システムでの利用形態に限定すれば前述のとおり問題とならないとしてあるのみで，仕組みとして根本的な解決はしていない．そこで，今後の課題として，ブロードキャスト暗号技術を導入し共有ユーザごとの鍵系列数自体を効率良く削減する方法についても検討していきたいと考えている．

7.6 共有時における鍵の受け渡しの必要性

従来手法では 3.3 節で述べたようにファイルを保存しなおすたびに他人の公開鍵を用いてプロパティファイルを暗号化しなければならず，共有ユーザ全員分の公開鍵を集める必要があり，さらにオフラインの共有ユーザがいた場合にそのユーザの公開鍵を得ることができないことから，システム上現実的ではないといった問題があったが，本提案手法では次の理由でこの問題を解決することができる．

共有ユーザは共有ファイルを取得するために必要なプロパティファイルを，共有用共通鍵を用いて暗号化している．よって，ファイルが保存・更新などされた時点でオフラインのユーザが存在したとしても，そのユーザは同一の共有用共通鍵をすでに所有していることになるため，ハッシュ連鎖を正しく行うことで保存された共有ファイルにアクセスすることが可能となるのである．また，新しい共有ファイルが保存された場合にも，書置きを利用することでオフラインユーザのプロパティファイルを再記述するといったことに対応

することが可能となる．

このように，提案手法を用いることで，個人の持つ公開鍵・秘密鍵ペアは，ファイル共有時に利用することはなくなり，個人の所有するファイルに関する暗号化・復号化とシステムを利用する際のユーザ認証のみに用いることになるので，システム中でどのように安全に受け渡すかといった方法を考慮する必要がなくなる．本システムでは，ユーザ全員が公開鍵ペアを所有することから，ユーザの判別のために公開鍵より求められる ID（ユーザ ID：参考文献 3) を参照）をログイン時のユーザ認証に利用している．

新たに生じる問題点として考えられることは，共有グループよりユーザ離脱の際に鍵系列の再構築を行うことになるが，新規共有鍵系列を享有ユーザ同士が受け渡す必要性が生じることである．ただし，この問題に関しては，7.4 節で説明しているとおり，本システムの想定している利用環境から考慮して，頻繁に起こる処理ではないと考えられる．よって，システム上深刻な問題にはならないといえる．

7.7 共有用共通鍵の正当性

本論文で述べてきた提案手法の場合ファイル共有を行う際にハッシュ連鎖のもととなるルート鍵を配布することになるが，その鍵が正しい共有ユーザが生成したものであるかの正当性を示す必要が生じる．

しかしながら，本システムでは，同一の目的を持ったプロジェクトグループなどの小規模なグループでの利用を想定しているため，まったく知らない人物とファイルを共有するといったことを考慮する必要がない．つまりファイル共有を行うユーザは，必ず面識があり，信頼のおけるユーザであることに加え，共有用共通鍵を手渡しなどの物理的な手法で受け渡しを行うことで，本システムにおける共有用共通鍵の正当性の確認は特に必要ないと考えられる．

物理的な手法で共有用共通鍵を受け渡すとあるが，共有用共通鍵はハッシュ連鎖のもととなる root 鍵のみを受け渡しさえすれば，それ以降の鍵はハッシュ連鎖により自動的に生成することが可能となるので，ただ 1 回の受け渡しに関して物理的な方法を用いることに関してはユーザにとって負担にはならないと考える．むしろ，確実な受け渡し方法であることから，より正当性を保つことが可能となるといえる．

7.8 ファイルの動的再配置手法の適用

2.2 節において，関連研究としてのファイルの動的再配置についての概要を述べたが，本論文で提案しているファイル共有手法にも，その動的再配置手法は適用可能である．どのように動的再配置が行われることになるか，2.2 節で説明した概要をもとに，提案手法への適用方法について説明する．

提案手法において，保存される共有ファイルは，ネットワーク上の PC に分散して保存されるという，本システムの基本機能を周到している．そのため，保存されたファイルに関しては，共有ファイル以外の個人が所有するファイルと同等の扱いを受けることになる．

つまり，ネットワーク上の PC に共有ファイルの部分データが保存されるが，その部分データへのアクセス頻度や，保存された PC とアクセスしてくる PC との相性を考慮することで，より取得確率の高い PC に部分データを動的に再配置することが可能となるのである．部分データが再配置された場合には，プロパティファイルの内容を書き換えればよく，共有ユーザからは，何 1 つ変わることはない．従来手法では，1 つの共有ファイルに対し，複数のプロパティファイルが存在していることから，部分データの再配置が起こった場合に，すべてのプロパティファイルを更新しなければならず，共有ユーザ数分の処理が必要となる．その点から見ても，従来手法よりも，提案手法を取り入れることで，より発展したシステムとなることがいえる．

ただし，ここで 1 つ問題となることがある．それは，個人の所有するファイルと異なり，共有ファイルでは，アクセスしてくるユーザ（PC）が 1 つではないということである．つまり，複数の共有ユーザからアクセスされるファイルを最適な位置に配置する必要があるということである．個人の所有するファイルであれば，1 対 1 の相性を求め，その順位を求めることで，最適な配置場所が求められるが，共有ファイルに関しては，1 対複数の場合での最適な保存位置を求める必要性が生じる．この問題に関しては，今後の課題としてさらに検討を重ねる必要があると考えている．

8. ま と め

これまで述べてきたように P2P ファイル共有システムにおいてファイル共有を行う際に従来手法では公開鍵暗号方式を用いて共有ファイルにアクセスしていたが，共有用共通鍵を導入し，その共有用共通鍵をファイルの保存時もしくは時間軸において定期的に更新することで，管理者不在の本システムにおいてユーザマネジメントを効率的に行うことが可能となり，ハッシュ連鎖を用いる本手法を実装しテストした結果から，従来手法と大差ない実行速度が得られることが確認できた．

このことから，本論文で述べてきた手法の有効性が証明され，システム利用面においても既存のシステム

と同様の感覚で利用できることが予想される．

謝辞 本研究の一部は，IPA（情報処理推進機構）2005 年度下期未踏ソフトウェア創造事業プロジェクトの一環として行われた．ここに記して謝意を表す．

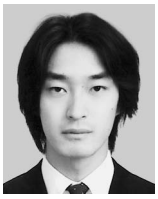
参 考 文 献

1) 大津一樹，宇田隆哉，伊藤雅仁，市村 哲，田胡和哉，星 徹，松下 温：アクセス制御機構を持つ P2P 共有ファイルシステムの提案，情報処理学会 DICOMO2004 論文集，pp.265–268 (2004).
2) 鹿島隆行，宇田隆哉，伊藤雅仁，市村 哲，田胡和哉，星 徹，松下 温：PC 共有による安全で低コストな P2P 分散ファイルシステムの提案，情報処理学会 DICOMO2004 論文集，pp.261–264 (2004).
3) 大津一樹，宇田隆哉，伊藤雅仁，市村 哲，田胡和哉，星 徹，松下 温：アクセス制御機構を持つ P2P ファイル共有システム，電子情報通信学会 SCIS2005 論文集，Vol.1, pp.13–18 (2005).
4) 鹿島隆行，宇田隆哉，伊藤雅仁，市村 哲，田胡和哉，星 徹，松下 温：PC 共有による安全で低コストな P2P ファイル分散システム，電子情報通信学会 SCIS2005 論文集，Vol.1, pp.1–6 (2005).
5) 大津一樹，宇田隆哉，伊藤雅仁，市村 哲，田胡和哉，星 徹，松下 温：P2P ファイル共有システムにおける鍵管理効率化手法の検討，情報処理学会 DICOMO2005 論文集，pp.609–612 (2005).
6) 平野仁之，中村康弘：分散ストレージにおけるセグメント欠落を考慮したデータ分割，情報処理学会研究報告 CSEC-2004，pp.83–88 (2004).
7) 石田祐子，井上 徹，佐久間隆，下田泰戈，金平恵実，宮前俊一，竹久達也，森本健嗣：データ分散保存システム，平成 13 年度情報処理振興事業協会セキュリティセンター：電子政府情報セキュリティ技術開発. http://www.ipa.go.jp/security/fy13/tech/crypto_vss/data_report.pdf

(平成 17 年 11 月 25 日受付)
(平成 18 年 6 月 1 日採録)

大津 一樹

2005 年東京工科大学工学部情報工学科卒業．現在，同大学大学院バイオ・情報メディア研究科コンピュータサイエンス専攻修士課程在学中．ネットワークセキュリティの研究に従事．2004 年情報処理学会 DICOMO2004 シンポジウムにおいて優秀プレゼンテーション賞を受賞．2005 年度下期未踏ソフトウェア創造事業採択（共同開発者）．



井上 亮文（正会員）

1999 年慶應義塾大学理工学部計測工学科卒業．2001 年同大学大学院前期博士課程修了．2005 年同大学院後期博士課程修了．博士（工学）．現在，東京工科大学コンピュータサイエンス学部助手．ネットワークセキュリティ，コンテンツ視聴技術，グループウェアの研究に従事．2005 年度下期未踏ソフトウェア創造事業採択．

宇田 隆哉（正会員）

1998 年慶應義塾大学理工学部計測工学科卒業．2000 年同大学大学院理工学研究科計測工学専攻前期博士課程修了．2002 年同大学院理工学研究科開放環境科学専攻後期博士課程修了．博士（工学）．現在，東京工科大学コンピュータサイエンス学部講師．ネットワークセキュリティの研究に従事．2002 年 IFIP/SEC 2002 Best Student Paper Award 受賞．電子情報通信学会会員．

松下　温（フェロー）

1963 年慶應義塾大学工学部電気工学科卒業．1968 年イリノイ大学大学院コンピュータサイエンス専攻修了．工学博士．1989〜2002 年慶應義塾大学理工学部教授，2002 年より東京工科大学教授，2003〜2005 年東京工科大学コンピュータサイエンス学部長．マルチメディア通信，コンピュータネットワーク，グループウェア等の研究に従事．情報処理学会理事，同学会副会長，マルチメディア通信と分散処理研究会委員長，グループウェア研究会委員長，電子情報通信学会情報ネットワーク研究会委員長，MIS 研究会委員長，バーチャルリアリティ学会サイバースペースと仮想都市研究会委員長，情報処理学会 ITS 研究会委員長等を歴任．郵政省，通産省，建設省，農水省，都市基盤整備公団，行政情報システム研究所等の委員長，座長，委員を多数歴任．特に国土交通省，住宅情報化標準策定委員会委員長，経済産業省総合エネルギー調査会電子計算機と磁気ディスク委員会委員長，経済産業省総合エネルギー調査会ルータ装置基準委員会委員長，最高裁判所専門委員を務める．『やさしい LAN の知識』（オーム社），『201x 年の世界』（共立出版）等著書多数．1993 年情報処理学会ベストオーサ賞，1995 年および 2000 年情報処理学会論文賞，2000 年情報処理学会 40 周年記念 90 年代学会誌論文賞，2000 年バーチャルリアリティ学会サイバースペース研究賞，2001 年情報処理学会功績賞受賞．情報処理学会フェロー，電子情報通信学会フェロー，人工知能学会，ファジイ学会，IEEE，ACM 各会員．