情報処理学会論文誌 Vol. 50 No. 6 1561–1574 (June 2009)

デジタルフォレンジックを目的としたファイル分散保存システムの実装および評価

東森 ひろこ[†1,*1] 手塚 伸[†2] 宇田隆哉[†1]
伊藤雅仁[†1] 市村哲[†1]
田胡和哉[†1] 星徹[†1]

本稿では，大学の研究室や企業のプロジェクトチームなどの小規模なネットワーク内における PC のディスク余剰領域と通信帯域を利用してファイルを保存するシステムについて述べる．システムの利用者はユーザごとの公開鍵に基づいた認証を行ってネットワークに参加し，共通鍵暗号方式を利用してファイルを保存するため，ファイル情報の秘匿性が保たれる．また，ファイルの保存・削除時は，ファイル更新履歴に連続性を持たせた署名データを生成してシステム参加 PC どうしで交換し，交換した署名データを交差して加えたものを更新履歴の署名とする．これにより，システム内で利用する情報が交差して，かつ連続性を持つことになるため，第三者による不正なデータの削除に対して，ファイル情報の信頼性を高めることが可能となる．

An Implementation and Evaluation of a Distributed File Backup System for Digital Forensics

Hiroko Tomori,[†1,*1] Shin Tezuka,[†2] Ryuya Uda,[†1]
Masahito Ito,[†1] Satoshi Ichimura,[†1] Kazuya Tago[†1]
and Toru Hoshi[†1]

In this paper, we make an observation on a file backup system that uses surplus disk space and communication band on each pc in a small-scale network such as a laboratory of university or a project team of enterprise. Confidentiality of information of files is kept, because each user on the system joins the network after authentication process based on public key cipher, and because files are encrypted with common key cryptography. Operation logs are securely saved in the system and each log is chained one by one. Moreover, the log is crossly exchanged several PCs, in order to make the system safer than existing one. In this system, reliability of files is high, since no one can delete the log without keeping all signatures for logs are correct.

1. はじめに

近年，高度情報化社会の進展に呼応して PC の普及率が高まり，それにともなってネットワーク技術が飛躍的に進歩してきている．また，個人ユーザにおいてもデータを電子的に取り扱う機会が増えたため，専用のファイルサーバや共有ソフトを用いて，複数のユーザとファイルを交換・共有する技術が進歩し，普及してきた．しかし，既存のファイル共有ソフトはファイルを一元管理する目的で作成されておらず，導入に際してユーザへの負担が大きい．また，ファイルサーバにおけるファイルの安全性は管理者のスキルに完全に依存しており，記憶媒体の破損やデータの改竄などの問題が絶えない．

このような背景の下，SOHO や大学の研究室内などの小規模なネットワーク内で，安全性を考慮した P2P 技術で各 PC を接続し，ディスク余剰スペースと通信帯域を利用した管理者不在でのファイル共有を実現するファイル分散保存システムが提案・実装された[1)–3)]．これらのシステムでは，保存されたファイルが暗号化されているだけでなく，ファイル保存時のログにディジタル署名が付与されており，第三者によるログの改竄や捏造が理論的に不可能となっている．しかしながら，前述のシステムでは，ファイル作成者本人によるファイルの捏造や削除に耐性がなく，今日問題となっている職員による証拠隠滅のような内部の犯行には対応できない．

そこで本稿では既存システム 1)–3) のファイル保存時に，ファイルの更新履歴を，ヒステリシス署名を用いたログとして記録する手法を提案し，これにより証拠隠滅を阻止するデジタルフォレンジックを実現する．

本提案システムは，すべての PC の管理者権限を特定の人物が持たないことを前提とした小規模なオフィスなどに適用可能である．1 人のユーザが複数台の PC の管理者権限を持っていてもよく（ただしすべての PC の管理者権限を持っていてはいけない），1 台の PC が複数人のユーザ（すべてのユーザでもよい）によって利用されてもかまわない．

本提案では，ファイル保存時のログを複数の PC に保存しており，その中に管理者権限を持たない PC も含まれることから，消去や改竄が行えないログが存在するようになってい

†1 東京工科大学大学院
Graduate School, Tokyo University of Technology
†2 慶應義塾大学大学院
Graduate School, Keio University
*1 現在，KDDI 株式会社
Presently with KDDI Corporation

る．また，ログには本人の署名が付与されているため，発見されたログに対しての否認も不可能である．具体例をあげると，ログが 5 台の PC へ保存された場合，そのうち 4 台の PC の管理者権限を持つ不正者は，残りの 1 台の PC のログへアクセスできないために，対象のログを削除することができず，不正の痕跡が残る．

さらに，証拠としてログを提示する者や捜査に関わった者が不正を行う場合も想定し，第三者がログの一部を削除したり，ログの順序を変更したりできないよう，ログにはヒステリシス署名が施されている．また，ヒステリシスとなっている一連のログが最初の部分から捏造されることを防ぐため，複数の PC におけるログへの署名を合わせて全体の署名部分が作成される手法を導入している．管理者権限を持たない PC の署名は作成できないため，単独犯が一連のログを捏造することは不可能である．

既存システム 1)–3) では，企業の部署などの小規模ネットワーク内において，ファイルサーバの管理者を必要とすることなく，複数台の PC に対してログインできる権限を持つユーザが，ネットワークに自由に参加・離脱を行いながらファイルを安全に保存・管理できる．本提案システムはこのシステムをベースにしており，これらの技術の利点を阻害しないで組み込むことが可能である．

このようにして，既存システム 1)–3) のファイル保存手法や，従来の単純な分散保存方式[4]，秘密分散方式を用いて実装を行ったシステム[5] に比べ，証拠隠滅を目的とした内部犯による不正なファイル削除を否認できないようにすることで，デジタルフォレンジックの観点から有効な改善策を実装し評価した．

2. 関 連 技 術

本提案手法では，ヒステリシス署名技術を用いる．これは長期的に利用できる電子署名技術の 1 つであり，過去の署名が連鎖的に次の署名に反映される手法である[6),7)]．

ヒステリシス署名においては，n 番目の署名を作成する際に，$n-1$ 番目の署名を n 番目のデータに追加する操作を行う．この操作により，1〜n 番目のデータとそのデータに対応する n 個の署名が存在するとき，そのうちの 1 組でも削除してしまうとその部分で連鎖がとぎれてしまうので，ヒステリシス署名は部分削除が行えないという特性を持つ．この特性は，署名作成に必要な秘密鍵を所持している署名者本人にも適用されるが，署名者本人は部分削除を行った後に新たなヒステリシス署名を 1 番目のデータから再度完全に作成することが可能であり，これがヒステリシス署名の弱点でもある．

現在，この署名技術を利用して，ユーザが PC を操作した履歴を，ヒステリシス署名を施したログとして記録する，デジタルフォレンジック保全装置[8] が提案されている．このシステムでは，検証用ログデータと署名生成時に必要な秘密鍵を USB メモリの耐タンパ領域に保存することによって，履歴の整合性が確認でき偽造の検知を可能にしている．

しかし，文献 8) で利用されているログファイルは，PC 上に通常のファイルと同様に追記・編集・削除が可能な状態で保管されている．そして，操作者は不正を行う可能性があるが，管理者は不正を行わないことが前提になっているため，操作者が管理者である場合はデータの捏造が自由に行えてしまう．これでは，各人が管理するノート PC などをネットワーク内には持ち込めないことになる．こうした問題点を解決するためには，ログファイルを安全な場所に保管するか，人間が触ることのできない装置を通してログファイルを作成する必要がある．

また文献 8) ではログファイルを任意に人間が作成できないよう，セキュリティデバイス（実装には e-Token Pro[9] が使用されている）を単なる耐タンパな鍵の管理場所および署名装置として使用するだけでなく，連鎖用データの保存場所としても使用しており，不正対策が完全にハードウェア依存となっていることが問題である．入力データの署名を返すだけの装置と比較すると，このようなハードウェアは汎用性がなく，コストの面からも現実的に導入するのは非常に難しいといえる．

上記のような専用のストレージを利用する方法以外に，Schneier らが提案したセキュアログ方式[10] がある．この方式ではデータの正当性を証明するために，秘密鍵を用いた Message Authentication Code を使用している．しかし秘密鍵を所持した者は，すべてのログデータを捏造することができるため，検証者を慎重に選ぶ必要があり，現実的ではないといえる．

本提案手法は，管理者不在下の P2P を用いた環境での利用を想定しているため，一貫性を持たせた検証用ログデータに複数台の PC の署名を交差して加え，さらにこのログデータをシステム参加 PC すべてが保持することで，耐タンパ性を有するモジュールを利用せずに履歴の整合性を検証できる点で大きな差異がある．特に，署名が複数台の PC にわたって行われていることで，悪意のあるユーザは，過去の署名履歴に一貫性を持たせて偽造しなければならないだけでなく，複数の PC に侵入して，対象 PC の鍵を手に入れたうえで署名を作成しなければならないため，非常に手間がかかり，署名の偽造は困難となる．

3. P2P ファイル分散保存システム

本章では，文献 1)–3) における，P2P ファイル分散保存システムにおけるファイル保存・取得・削除の概要を示す．ファイルの暗号化には共通鍵暗号方式の Advanced Encryption

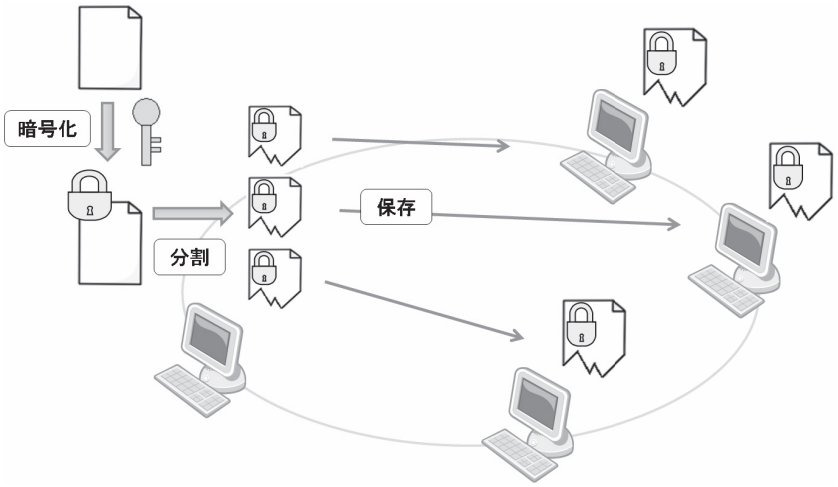

図 1　システム概要
Fig. 1　An outline of system.

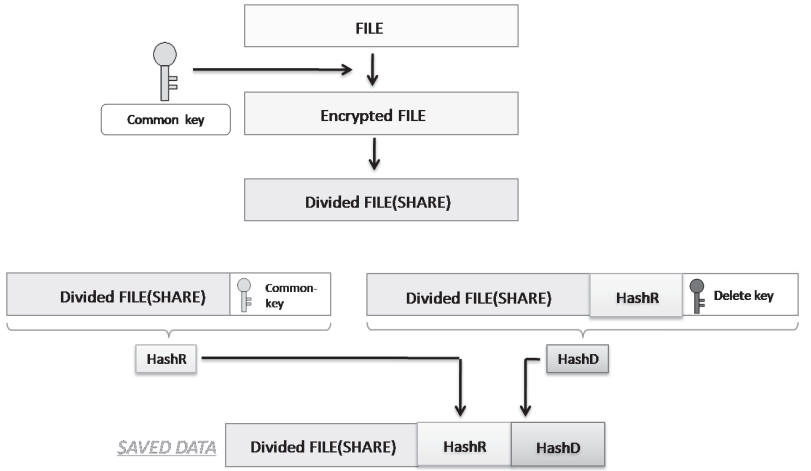

図 2　シェア（分割ファイル）保存形式
Fig. 2　A form for saving SHAREs (divided files).

Standard（AES）[11] を，また暗号化に用いた共通鍵は公開鍵暗号方式の RSA [12] を用いて暗号化することで，ファイル情報の秘匿性を高める．

3.1　システム利用環境

本システムがベースとしている P2P ファイル分散保存システム（以下これを既存システムとする）1)–3) は **図 1** のように，大学や SOHO 内に設置してある小規模なローカルエリアネットワークを対象に，各 PC を P2P で連携し，それらの PC にファイルを保存するシステムである．

保存対象のファイルはランダムに生成される共通鍵を用いて暗号化されたのち，冗長性を持たせてシェアと呼ばれる複数のファイルに分割され，ネットワークに参加している PC に保存される．

また，既存システムは IP アドレスが浮動の場合にも対応しており，PC には PCID が，ユーザにはユーザ ID および公開鍵ペアがシステム初回起動時に設定されるため，PC およびユーザは IP アドレスに依存することなく管理される．さらにユーザは使用 PC を制限されることなくシステムを利用できる．これら ID は次回以降の利用で同一の ID が使用される．PCID はユニークな 16 byte の文字列である．

本提案手法ではファイルの保存・復元は既存システム 1)，2) を，共有は既存システム 3) の仕組みを踏襲している．

3.2　ファイル保存

ファイルの保存時にユーザは，毎回生成されるランダムな共通鍵（**図 2** 内 Common key）を用いてファイルを暗号化し，ファイルの重要度に応じた任意の冗長性を持たせて複数に分割する．ファイルの分割には (n, k) 閾値秘密分散法の Information Dispersal Algorithm（IDA）[13] を用いる．分割されたそれぞれのシェアにはランダムな 128 byte のファイル名がつけられる．次にこの共通鍵を，分割した個々のシェアに付加し，そのハッシュ値（SHA-1）[14] を計算する．このハッシュ値はシェアのデータ破損検出に用い，これを閲覧ハッシュ（図 2 内 HashR）とする．また先の共通鍵と同様にランダムに生成される，別の共通鍵（図 2 内 Delete key）と閲覧ハッシュを付加し得られたハッシュ値を削除ハッシュ（図 2 内 HashD）とする．この鍵はファイル削除用の鍵（削除鍵）であり，ファイル共有時にユーザの削除権限の有無を確認するためにも用いられる．各シェアに閲覧ハッシュ，削除ハッシュを付加したものが，シェアの保存形式となる．

これら共通鍵は，ユーザの公開鍵で暗号化され，シェア名，分割・復元数，ファイルの署名，タイムスタンプ，保存先 PCID などファイルを取得するためのメタ情報としてプロパ

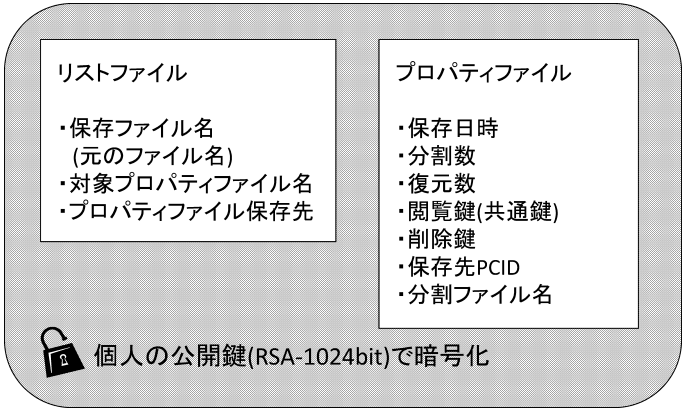

図 3　リストファイルとプロパティファイル内のデータ
Fig. 3　A data of List File and Property File.

ティファイル（復元情報ファイル）（**図 3**）に保存される．このプロパティファイルは，オリジナルファイルを保存するユーザ個人の公開鍵によって暗号化された後，シェアと同様にネットワーク上の PC に保存される．プロパティファイル名もまたランダムな 128 byte の文字列で構成されるため，各ファイルと対のプロパティファイル名と保存先 PCID が，リストファイル（保存ファイルの一覧）に記録される．

リストファイルもまた他のファイルと同様に 128 byte のファイル名がつけられ，ユーザの公開鍵で暗号化された後にネットワーク上の PC に保存される．しかしリストファイルは，保存したファイルに 1 対 1 対応しているプロパティファイルとは違い，ユーザが保存したファイルの一覧が列挙されているので，ユーザに 1 対 1 対応したファイルとなる．そのため，リストファイル名をランダムに生成してしまうと，システム参加時にユーザは必要なリストファイルを特定できず，オリジナルファイルの復元が不可能となってしまう．

そこで，ユーザの持つ秘密鍵・公開鍵ペアに着目してこの問題を解決する．まず，秘密鍵を公開鍵で暗号化し，得られた暗号化済みデータをインクリメントしながらハッシュ計算を 7 回繰り返し，それらのハッシュ値を連結する．結果として得られた 140 byte のデータを BASE64 変換し，その先頭部分から 128 byte 抽出したものをリストファイル名とする．この方法を用いることで，ユーザの鍵ペアから一意のファイル名を持つリストファイルを生成することができる．

ファイルを複数人で共有する場合は，ユーザの鍵ペア以外にグループ専用の鍵ペアとリストファイルを作成し，グループの公開鍵を用いてプロパティファイルとリストファイルを暗号化する．グループの秘密鍵はそのグループに属する各個人の鍵で暗号化された状態でネットワーク上に保存される．

なお，理想的には，各ユーザが持つ秘密鍵，また各ユーザが管理者権限を持つ PC の秘密鍵は耐タンパデバイス内で管理され，なおかつ公開鍵暗号の演算もそのデバイス内で完結すべきであるが，実装の都合により既存システムでは，各人が持つ USB メモリ内で秘密鍵を管理している．本システムにおいても，同様に実装上の都合で一般的な USB メモリ内でユーザの秘密鍵を管理しているが，実運用の際には耐タンパデバイスで管理すべきである．ただし，この耐タンパデバイスは，文献 8) のように特殊な機能を有したデバイスである必要はなく，入力データに対して署名をして出力するだけの汎用的な機能を有したものであればよい．

3.3　ファイル復元

3.2 節で述べたとおり，ユーザはつねにリストファイルを取得できるため，秘密鍵を用いてリストファイルの内容が復号できる．まず，ユーザは得られたリストファイルの内容から復元したいファイルを選択し，次に，対応するプロパティファイルをネットワーク上の PC から取得し，秘密鍵で復号する．そして，得られたプロパティファイルの内容であるシェア名と保存先 PCID より，シェアを閾値分取得し，結合したのち共通鍵で復号して，オリジナルファイル情報を得る．

3.4　既存システムの問題点

3.4.1　汎　用　性

既存システムは，windows のアプリケーションプロセスとして実装されている．そのため，ユーザは windows 利用者に限定され，かつ word や excel など他のアプリケーションを通してシステムを利用することができないため，汎用性に乏しいといえる．また，ユーザはファイルを 1 度ローカルで保存した後でシステムにアップロードしなければならないので，ローカルに残されたファイルキャッシュからファイル情報が漏えいする恐れもある．これら 2 点をふまえ，まず OS における汎用性を考慮した，Linux における Hybrid-P2P 型分散ファイルサーバシステム[15] が提案・実装され，その後文献 15) をファイルシステムに組み込んだ，分散仮想ファイルシステム[16] が提案・実装された．しかし，文献 15)，16) は比較的大規模な組織間での運用を目的としているため，本提案が主眼とする小規模な LAN 内のみでの環境には不向きであるといえる．

3.4.2　デジタルフォレンジックの有効性

既存システム 1)，2) や単純な分散保存方式 4)，秘密分散方式を用いて実装を行ったシステム 5) には，ファイル保存時の更新履歴がなく，また文献 3) は更新履歴への署名を行っていないため，第三者は容易にファイルと履歴を削除でき，不正の証拠隠滅を謀れる．よって，ファイル情報に対する信頼性が低いといえる．特に，文献 3) では誰でも簡単に更新履歴を削除できるため，不正が行われたことを証明する手段がなく，ファイル情報を検証することも不可能である．そのため，利用者に対してのファイルの機密性や対改竄性のみを重視している上記システムは，ファイル操作の不正に対して，証拠隠滅を容易に行える点で，デジタルフォレンジックの観点から望ましくないといえる．

3.4.3　ファイル取得率の問題

通常既存システムや，文献 15)，16) はシェアを，我々がふだん利用する PC の余剰ディスクスペース上に保存しているため，電源が入っていないオフライン時にシェアが回収不可能となる恐れがある．これらの論文のシステムでは，分割されたシェアに冗長性を持たせ，回収率が閾値を上回ればファイルを復元できるように工夫している．

また，文献 15)，16) では，参照頻度の高いファイルを，ネットワーク内の PC を管理するファイルサーバゲートウェイ上にキャッシュすることによって，ファイルの復元率を高めている．その際の評価として文献 16) では，研究室と企業の起動ログを回収して，ファイル取得のシミュレーションを行い，ファイルの復元率が 90%となることを確認している．しかしファイルサーバを持たずにシステムを構成する本提案システムにおいては，キャッシュ機構を持つことができないため，ファイル復元率がさらに低下する恐れがある．

そこで本システムでは，起動時間帯が類似している PC から順に優先的にシェアを保存するとともに，閾値を上回るシェアを回収できない場合には，シェアを保持する PC の中で起動時間帯が類似している PC から順に Wake-On-LAN を用いて強制的に PC を起動させることで，ファイル復元率を 100%としてこの問題を解決する．なお，シェアの保存先選択アルゴリズムについては，既存システムと同様であり，本稿が主眼とする点ではないため説明を割愛する．

なお，本提案手法においてデジタルフォレンジックを実現するためには，「シェアを分割保存する PC の数 $>$ 1 人のユーザが管理者権限を持つ PC の台数の最大値」でなければならない．これは，シェアが分割保存された PC すべての管理者権限を持つユーザが存在してしまった場合，そのユーザはヒステリシス署名を崩すことなくその履歴を抹消することが可能となってしまうためである．

4. 提 案 手 法

本章では，3.4.1，3.4.2 項であげた問題点を解決する手法を説明する．本提案手法では 3 章で述べたシステムの仕組みを前提としている．

4.1　ファイルシステムへの導入

保存ファイルを，ファイルシステムから直接利用できるようにするためには，従来のプロパティファイルの内容以外に，ファイルやディレクトリのシステム内における階層構造を示すための，ディレクトリエントリが必要になる．また 3.4.2 項で述べたとおり，更新履歴や署名を含めてファイルの信頼性を高める必要がある．そのため，従来よりもファイルを復元する際に必要となる情報が多くなり，効率的な情報の管理が求められる．そこで本提案手法では，文献 16) と同様の手法でデータベースを用いてファイルの復元情報を一括管理する．

従来のプロパティファイル内の情報，および仮想ファイルシステムとして利用するためのディレクトリエントリは，ファイルを復元する際に必要となるため，本システムではこれらをまとめてファイルプロパティとして保存する．ファイルプロパティには，システムを通して保存したファイルの復元情報が記録されるため，全 PC がファイルプロパティを共有しあうことになる．この共有は，5 章で詳述する DB の同期により実現する．

また，各 PC が保持するシェアについてはシェア一覧として，それぞれの PC がその PC に保存されるシェア情報を，その PC のファイルリストに記録する．ファイルリストは PC ごとに異なる内容であるため DB の同期は行われない．

本システムを用いてのファイルの保存手順を**図 4** に示す．

ファイルを保存するユーザは 3.2 節で述べたとおり，まず（図 4 ①）対象のファイルを共通鍵で暗号化し，（図 4 ②）閾値を持たせて複数のシェアに分割して，（図 4 ③）シェアをネットワーク内の各 PC に保存する．各 PC は（図 4 ④）受信したシェアに関する情報をそれぞれのファイルリストに記録する．そして，（図 4 ⑤）ファイルプロパティをユーザの公開鍵で暗号化してファイルプロパティを更新する．その後，（図 4 ⑥）暗号化済みのファイルプロパティを全 PC に送信して，各 PC はその PC に保存されている暗号化されたファイルプロパティを更新する．

次にファイル保存時に記録されるログの生成方法を説明する．4.2 節で後述する情報 $LogPL_n$ を，ファイルプロパティの更新履歴とする．通常のシステムでは，この更新履歴がログとして扱われるが，本システムでは更新履歴 $LogPL_n$ にヒステリシス署名を付与したデータ（4.2 節の $SignedLog_n$）をプロパティログと呼び，記録する．

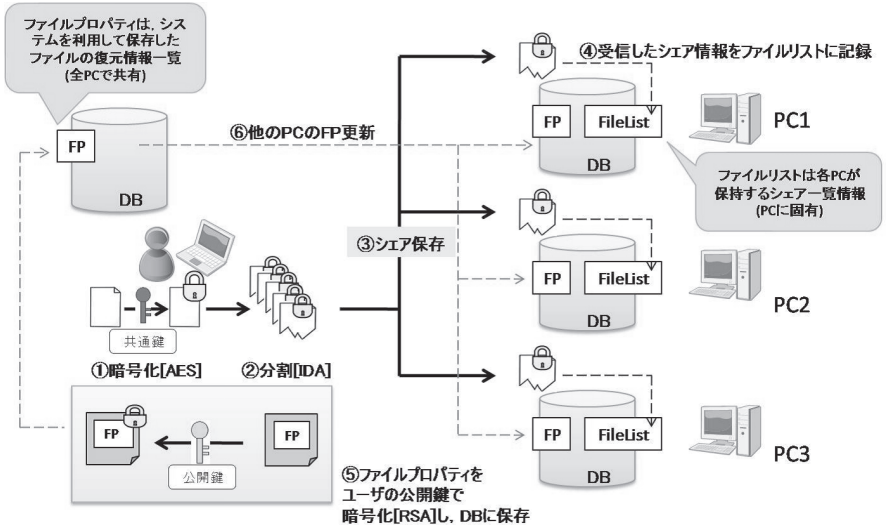

図 4　ファイルの保存手順
Fig. 4　A Process of saving failes.

また，シェアを受信した各 PC はそれぞれが持つファイルリストに，新たにシェアに関する情報（シェア名・ファイルサイズ）を追記する．この追記された情報に対して生成された履歴（4.3 節 $LogPC1_l$ 参照）を，ファイルリストの更新履歴とする．ファイルプロパティと同様，通常のシステムではこの更新履歴をログとして扱うことが多いが，本システムでは更新履歴 $LogPCi_l$ にヒステリシス署名を付与したデータ（4.3 節の $SignedLog_l$）をファイルログと呼び，記録する．

2 章で述べたとおり，ヒステリシス署名方式を用いたとしても，検証用のログデータが安全に保持される必要がある．文献 8) では PC，あるいは共有ファイルに対してのアクセス権を不特定のユーザが持っているため，3.4.2 項で述べた証拠隠滅の問題が起こる可能性がある．

そこで本手法では，4.2 節で後述するように，ファイルプロパティの更新履歴に，各シェアを保持する PC のファイルリストの更新履歴を追加して，ヒステリシス署名を生成する．また同様に 4.3 節で後述するように，各 PC が持つファイルリストの更新履歴に，ファイルプロパティの更新履歴を追加してヒステリシス署名を生成する．

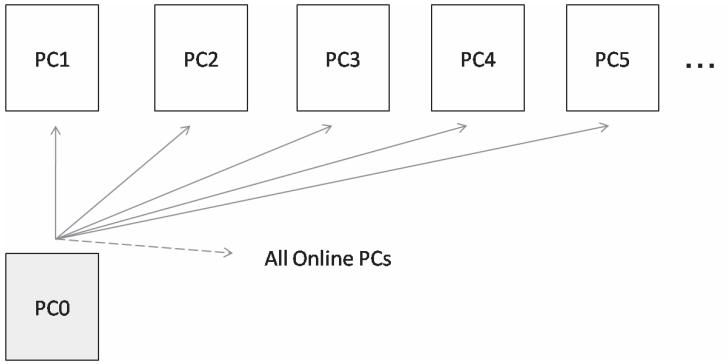

図 5　利用環境略図
Fig. 5　Deployment of PCs.

このように，システムに参加している PC それぞれの鍵で署名したデータを，互いに交換して署名を生成することで，相互に内容を証明することが可能となる．またユーザは，システムに参加する全 PC の管理者権限を持たないことを前提としているため，そのユーザの署名が施されたログがいずれかの PC から 1 つでも発見されると，そのログの内容に対する否認は不可能となり，そのログの一連のヒステリシス署名が有効であることが確認されれば，ログ全体の部分削除や捏造がないことが判明する．

さらに，1 回のファイル保存に対する記録が複数台の PC にわたって，署名つきで保存されているため，対象のログすべてに適切な改竄を加えなければ任意のプロパティログやファイルログを捏造できず，管理者権限のない PC 上のログ操作は不可能であるため，証拠隠滅を目的としたファイルの改竄や削除を完全に防止できる．

4.2　プロパティログ

本提案手法の利用環境概略を**図 5** に示す．図 5 は小規模な LAN 内を想定して描かれており，ファイルの保存者は PC0 を用いてオンライン PC1〜PC5 へファイルを保存する．

PC0 の利用者がファイルを保存した際，全 PC には新たなファイルプロパティが，シェアが保存された各 PC のファイルリストには新たに保存されたシェアに関する情報が記録される．このとき PC1 が持つプロパティログの内容は**図 6** のようになる．ファイルプロパティの n 番目のレコードの履歴 $LogPL_n$ に対し，各 PC のファイルリストの最新レコードの履歴は $LogPCiN_l$ となる（今回は図 5 を仮定しているため，$i = 2$〜5 となる）．このときに作成されるプロパティログを，$SignedLog_n$ としたときに，$SignedLog_n$ に含まれる情

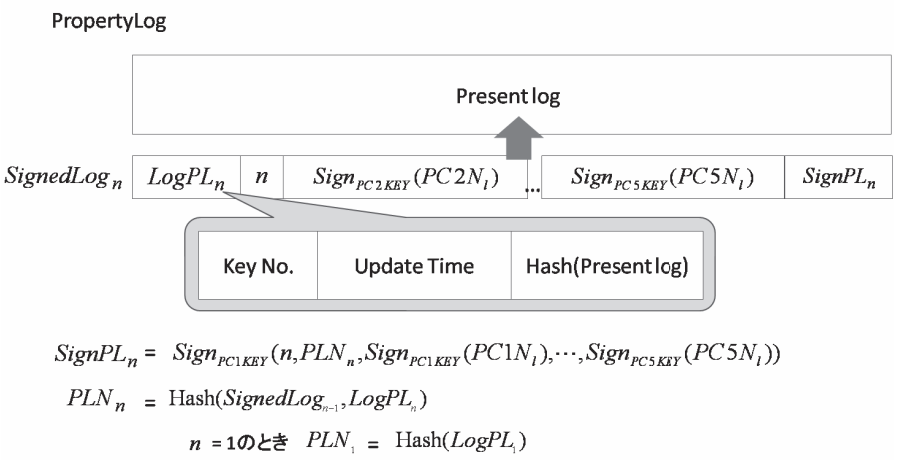

図 6　プロパティログ詳細
Fig. 6　Details of Property-log.

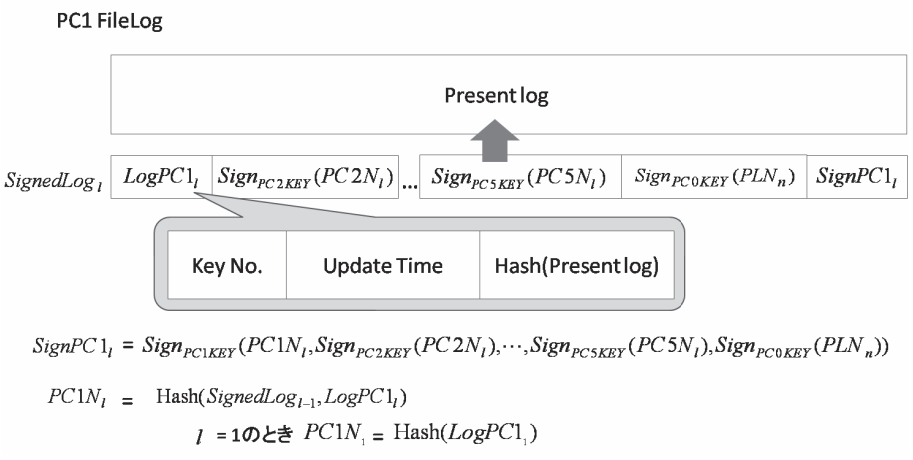

図 7　ファイルログ詳細
Fig. 7　Details of File-log.

報を以下に説明する．なお，ファイルリストの履歴の番号は PC ごとに異なるため，n 番目という表記はせず，latest（最新の）の意として l を用いて $LogPC1N_l$ のように表記した．$LogPC1N_l$ は PC1 における最後に追加された履歴を意味する．

- $LogPL_n$
  ファイルプロパティの更新履歴．内容は以下のとおりである．
  ・更新レコードのキー（Key No. (4 bytes)）
  ・更新時間（Update Time (4 bytes)）
  ・更新レコードのハッシュ値（Hash(Present log) (20 bytes)）
- n
  1〜n 番まで存在する，追記されるログの番号
- $Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$
  その PC における最後から 2 番目（$l-1$ 番目）のファイルログと更新履歴のハッシュ値 $PCiN_l$（ファイルログの連鎖データ）に対しての，各 PC（PC2〜PC5）の署名．$PCiN_l$ の作成方法については **図 7** 内にて記述する．なお，今回は図 5 を仮定しているため，$i = 2$〜5 となる．
- $SignPL_n$
  追記されるログ番号 n に加え，ファイルログの連鎖データ $PCiN_l$ に対しての各 PC の署名（$Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$）と，$n-1$ 番目のプロパティログと更新履歴のハッシュ値 PLN_n（プロパティログの連鎖データ）に対しての，PC1 の署名．

上記をふまえ，PC1 のプロパティログの作成手順を説明する．

（1）　ファイル保存 PC（PC0）は，ファイルプロパティから更新履歴 $LogPL_n$ と追記されるログの番号 n，プロパティログの連鎖データ PLN_n を生成し，シェア受信 PC（PC1〜PC5）に送信する．
（2）　シェア受信 PC（PC1〜PC5）は，その PC に保存されているファイルリストから $Sign_{PC1KEY}(PC1N_l)$〜$Sign_{PC5KEY}(PC5N_l)$ を生成する．
（3）　PC1 は他の PC（PC2〜PC5）からそれぞれ，$Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$ を受信する．
（4）　（1）より得られた n・PLN_n，（2）（3）より得られた $Sign_{PC1KEY}(PC1N_l)$〜$Sign_{PC5KEY}(PC5N_l)$ から $SignPL_n$ を生成する．
（5）　（1）より得られた $LogPL_n$・n，（3）より得られた $Sign_{PC2KEY}(PC2N_l)$〜

$Sign_{PC5KEY}(PC5N_l)$，(4) より得られた $SignPL_n$ を連接したものを PC1 が保持するプロパティログ $SignedLog_n$ とする．

4.3 ファイルログ

ファイルリストも，ファイルプロパティと同様の手順でログが生成される．

PC1 が持つファイルログは図 7 のようになる．前節と同様に，ファイルリストにおける最新の履歴 $LogPC1_l$ に対して図 7 に示す方式で署名したものを，latest（最新の）の意として l を用いて，$SignedLog_l$ と表現している．$SignedLog_{l-1}$ はその PC における最後から 2 番目のログという意味である．$SignedLog_l$ の内容を以下に説明する．

- $LogPC1_l$
  ファイルリストの更新履歴．内容は以下のとおりである．
  ・更新レコードのキー（Key No. (4 bytes)）
  ・更新時間（Update Time (4 bytes)）
  ・更新レコードのハッシュ値（Hash(Present log) (20 bytes)）
- $Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$
  その PC における最後から 2 番目（$l-1$ 番目）のファイルログと更新履歴のハッシュ値 $PCiN_l$（ファイルログの連鎖データ）に対しての，各 PC（PC2〜PC5）の署名．
- $Sign_{PC0KEY}(PLN_n)$
  $n-1$ 番目のプロパティログと更新履歴のハッシュ値 PLN_n（プロパティログの連鎖データ）に対しての，PC0 の署名．
- $SignPC1_l$
  ファイルログの連鎖データ $PCiN_l$ に対しての各 PC の署名（$Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$）に加え，プロパティログの連鎖データ PLN_n に対しての PC0 の署名（$Sign_{PC0KEY}(PLN_n)$）と，PC1 における最後から 2 番目（$l-1$ 番目）のファイルログと更新履歴のハッシュ値 $PC1N_l$（PC1 のファイルログの連鎖データ）に対しての，PC1 の署名．

上記をふまえて，PC1 のファイルログの作成手順を説明する．

(1) シェア受信 PC（PC1）は，ファイルリストの更新履歴 $LogPC1_l$ と，PC1 のファイルログの連鎖データ $PC1N_l$ を生成する．
(2) PC1 は他の PC（PC2〜PC5）からそれぞれ，$Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$ を受信する．
(3) ファイル保存 PC（PC0）は $Sign_{PC0KEY}(PLN_n)$ を生成して，シェア受信 PC（PC1〜PC5）へ送信する．
(4) (1) より得られた $PC1N_l$ と，(2) より得られた $Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$，(3) より得られた $Sign_{PC0KEY}(PLN_n)$ から，$SignPC1_l$ を生成する．
(5) (1) より得られた $LogPC1_l$ と，(2) より得られた $Sign_{PC2KEY}(PC2N_l)$〜$Sign_{PC5KEY}(PC5N_l)$，(3) より得られた $Sign_{PC0KEY}(PLN_n)$，(4) より得られた $SignPC1_l$ を連接したものを PC1 が持つファイルログ $SignedLog_l$ とする．

以上のように，図 6 内プロパティログ $SignedLog_n$ と図 7 内ファイルログ $SignedLog_l$ は，それぞれ過去のログ情報が最新のログに反映された状態となっている．また，プロパティログとファイルログは，相互の情報を含んだうえで署名が施されている．このため，署名が正しく検証できるように保ったうえでログの部分削除や改竄を行うには，そのログに関連するすべてのログを保持する PC へアクセスし，これらのログに対応する署名を 1 つめのログから捏造し直すことが必要となる．

ヒステリシス署名には，ログデータが部分的に削除された場合に，それより以前のログデータの信頼性が損なわれてしまうという問題がある．また，署名生成者によって，削除部分から一連のログデータを捏造されたり，文献 8) のように最初の部分からログデータを捏造されたりする恐れもある．しかし本手法では，それぞれの PC が持つログに含まれる署名が，複数の PC から集められた署名に基づいて生成され，各 PC に保存されている．よって一部に不正なログが検出された場合でも，他の PC のログから必要な情報を適宜補完することで，不正部分の検出あるいは検証が可能となる．さらに，保存ファイルに対しての不正の証拠隠滅を謀る場合は，複数の PC に分散して保存されているすべてのシェアとファイルログ，プロパティログに変更を加えなければならず，他ユーザによる操作もログに影響を与えるため，組織内で利用される情報に対する内部犯による改竄および消去が困難となる．

本手法では，ファイルサーバの管理者を置かない小規模な組織で，各人が 1 台もしくは複数の PC を管理する環境でのシステム利用を前提としている．そのため，ユーザが利用できる PC の台数が限られ，不正侵入をしない限り，管理者権限を持たない PC に保存されているログに，容易にアクセスすることができない．また，各署名がヒステリシスであるため，連鎖性を保持したままで 1 台の PC のログだけを改竄することは不可能となり，他の署名との整合性もとれなくなってしまうので，ログの部分的な削除も困難となる．そのため，ファイルリスト・ファイルプロパティそれぞれの署名の整合性が互いにとれない場合は，残された PC のログを可能な限り集めることで，改竄を行った人物を絞り込むことが可能とな

るとともに，残存している正しいログをたどり，不正に削除されたファイルを復元することが可能となる．

4.4　不正検知の仕組み

提案のシステムにおいて改竄，削除などの不正が行われた場合の検知方法について説明する．提案手法の特徴は内部犯による証拠隠滅を防止する点にある．ユーザ A を内部犯と仮定して，本人が正しく作成した一部のファイルの削除を試みる場合の不正検出について説明する．

ユーザ A は PC0 を使用して当該ファイルを保存し，ファイルのシェアは PC1，PC2，PC3，PC4，PC5 に保存されたと仮定する．また，1 章のルールより，ユーザ A が PC1〜PC5 すべての PC の管理者権限を持つことはないため，PC1〜PC4 の管理者権限のみ持ち，PC5 のログにはアクセスできないと仮定する．

ユーザ A は PC1〜PC4 の DB にアクセスし，DB 内に記録されている当該ファイルのファイルプロパティを削除するとともに，PC1〜PC4 のプロパティログおよびファイルログを捏造する．これらのログの捏造には 5 台の PC が必要であるため，PC5 の代わりに適当な PC を用意し，偽の署名を付与する．

本システムでは 4.1 節で述べたように DB の同期をとっており，先ほどのファイルが保存された時点以降に電源が入っていた PC には当該ファイルのファイルプロパティおよびプロパティログが記録されている．ユーザ A が不正操作を行った後にこれらの PC のいずれかがシステムのネットワークに参加すると，再び DB の同期がとられ，PC1〜PC4 とは DB の最終更新時刻以前のデータが一致しないため，不正操作が発覚する．

不正が発覚した時点で，各 PC のファイルプロパティを比較する．この時点で，PC1〜PC4 には存在せず，他の PC の記録には存在する当該ファイルのファイルプロパティが発見される．このファイルプロパティには，ユーザ A が当該ファイルの保存を行った記録および PC1〜PC5 に当該ファイルのシェアが保存されているという記録がユーザ A の署名つきで記録されており，ユーザ A はこれを否認できない．次に，PC5 のプロパティログのヒステリシス署名を検証すると正しいことが判明するので，先ほど発見されたファイルプロパティの保存順序には入れ替えのないことも確認される．これにより，当該ファイルが複数のユーザに共有されていた場合でも，このファイルを書き換えたユーザの順番も確認される．

もし，PC1〜PC4 に保存された当該ファイルのシェアが削除されていなければ当該ファイルは復元される．削除されていたとしても，閾値以内のシェアが残っていればファイルは復元できるが，閾値を下回った場合は復元できない．ただし，当該ファイルの証拠隠滅が図られた事実は否認できない．

以上のことから，本手法ではファイルの不正な改竄や削除を容易に検知することが可能であり，既存システムや単純な分散保存方式 4)，秘密分散方式を用いて実装を行ったシステム 5) では実現不可能であった証拠隠滅にともなう否認を防止することができる．さらに，整合性がとれたログの捏造が非常に困難であるため，改竄や削除の抑止も期待できる．

5. 実　　装

本システムで利用したアルゴリズムを**表 1** に示す．

共通鍵暗号をファイルの暗号化と，ファイルプロパティの暗号化に利用し，閾値秘密分散法の IDA をファイルの分割時に使用した．これらの暗号モジュールは，いずれも crypto++ [17] のライブラリを利用した．これらアルゴリズムは，既存システムで用いられているものと同様である．

本システムでは，ユーザに従来通りの操作性を提供するため，システムそのものをファイルシステムとして実装している．ユーザは，ファイルが分散保存されていることを意識しなくても，従来と同様の操作で本システムを利用できる．なお，ファイルシステムとしての実装を実現するため，FUSE（Filesystem in Userspace）[18] を用いている．**図 8** に，本システムのモジュール構成を示す．

各部の説明は以下のとおりである．

- Userprocess Module
  一般的なアプリケーションと同様，ファイルの作成，保存などのファイルシステム系のシステムコールを，OS に対して発行するプロセス．
- Distributed Filesystem Client（DFC）
  FUSE から渡されたファイルシステムの操作リクエスト（ファイル作成・読み書き・削除など）を受け取り結果を返す，本システムの中核を成すプロセス．この DFC 内で一

表 1　実装に用いたアルゴリズム
Table 1　Algorithms for implementation.

機能	種類
公開鍵暗号	RSA（1,024 bit）
共通鍵暗号	AES（128 bit）
閾値秘密分散法	RabbinIDA
ハッシュ関数	SHA-1（160 bit）

般的なファイルシステム（FAT や ext3 など）における，ファイルシステムテーブルを管理する．ファイルシステムテーブルは，本システムではファイルプロパティとして扱われている．

また，保存ファイルに対して，IDA による分割・結合，暗号化，署名生成・検証処理を行い，それらを保存先 PC の DFC Daemon に送信する．本システムにおけるネットワークへの参加・離脱通知と DB の同期も DFC を通して行われる．

- DFC Daemon
  他の PC の DFC から要求を受け，応答するデーモン．他の PC のファイル保存時にシェアを受け取り，要求があれば保存されているシェアを送信する．また，各 DFC がシェアに対する署名を生成する際に最終履歴を送信する．

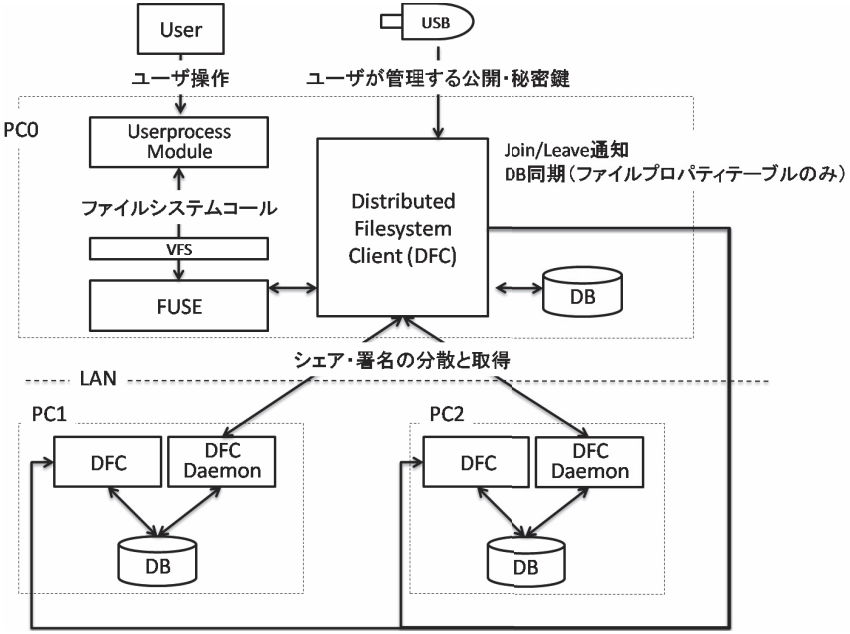

図 8　システムモジュール構成
Fig. 8　Module composition of the system.

- DB
  内部では，DFC が利用するファイルプロパティ（通常のファイルシステムテーブルの代用になる）や，DFC Daemon が利用するファイルリストなどを，DB のテーブルとして管理している．この DB は複数の PC 間で DFC を通してファイルプロパティの同期を行っている．ファイルリストは各 PC に保存されたシェアの一覧であり，各 PC に固有の情報であるため同期は行わず，各 PC で個別に管理される．

6. 評　　価

本システムを利用して，ファイルを保存・復元した際の評価を行った．

この評価においては，1 台の PC から 5 台の PC にシェアを分割保存し，3 台分のシェアが集まればファイルが復元可能であるように設定した．この評価を行う際，実際には PC は 6 台同時に必要であるが，6.1 節および 6.2 節の内容は単独の PC における実行時のパフォーマンス評価であるため，評価に必要な部分の処理のみを 1 台の PC で実行し，時間を測定している．評価に使用した PC のスペックは**表 2** のとおりである．

6.1　ファイルの暗号化と分割，復元評価

本システム上で，1 KB〜50 MB までのファイルを保存，復元した場合の処理時間に関する結果を**図 9** に示す．ファイルの暗号化と，分割にかかった時間を save とし，閾値分のファイルを統合・復号にかかった時間を restore とした．図 9 に示す評価では，ログデータの生成・検証は行っていない．

図 9 の結果より，数 MB 程度のファイルならば高速で処理を行えるが，50 MB の場合，ファイルの暗号化と分割だけで 180 秒かかることが分かる．これは，ファイルを閾値を持たせて分割する IDA の，ファイルサイズに比例して処理時間が増大するという特性に起因している．しかし，文献 3) によると，一般的な PC 内で利用されているファイルのおよそ 95%は 1 MB 程度のサイズであるため，本システムにおいて IDA の処理時間は問題にならないといえる．

表 2　評価環境
Table 2　Environment for evaluation.

OS	Vine Linux 4.2
CPU	core2Duo6400 2.13 GHz
MM	2 GRAM

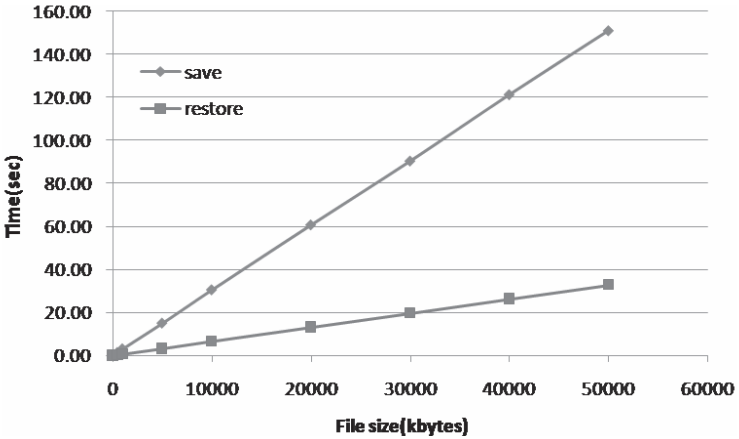

図 9　ファイル保存・復元時間評価
Fig. 9　Evaluation of file save and restore.

6.2　ログデータ生成，検証，サイズ評価

1,000 KB のファイルを，PC0 から 1〜1,000 回更新する際に生成されたプロパティログ，ファイルログのサイズを**図 10** に示す．

図 10 の結果から，1 回の更新におけるファイルログのサイズはおよそ 0.795 KB，プロパティログのサイズもおよそ 0.678 KB であることが分かる．また，プロパティログ，ファイルログそれぞれの署名生成時間は 1 回あたりおよそ 7 msec，署名検証時間はおよそ 0.9 msec であり，非常に高速に処理された．特に，本システムはファイルシステムでの透過的な利用環境で実装しており，ユーザはファイルの保存や復元時に意識して特別なコマンドを利用するわけではないため，ユーザがシステム運用上感じる遅延はほとんどないといえる．

次に，本システムを 20 人のユーザが各自 1 台の PC を利用した場合に生成されるプロパティログのサイズを検証する．なお，ファイルリストは 5 章で述べたとおり，各 PC に固有の情報であるため，同期を行っていない．そのため，更新のつど同期をとるファイルプロパティをもとに生成される，プロパティログのサイズを評価する．プロパティログ 1 レコードのサイズは先の結果より 0.678 KB である．ユーザがファイルを上書き保存するタイミングは性格によって異なるため，頻度が高い極端な場合を想定して 1,000 KB のファイルを 10 分に 1 回，逆に頻度が低い極端な場合を想定して 1 時間に 1 回更新すると仮定したうえで計算を行うことにする．このときに生成されるプロパティログのサイズを**表 3** に示す．

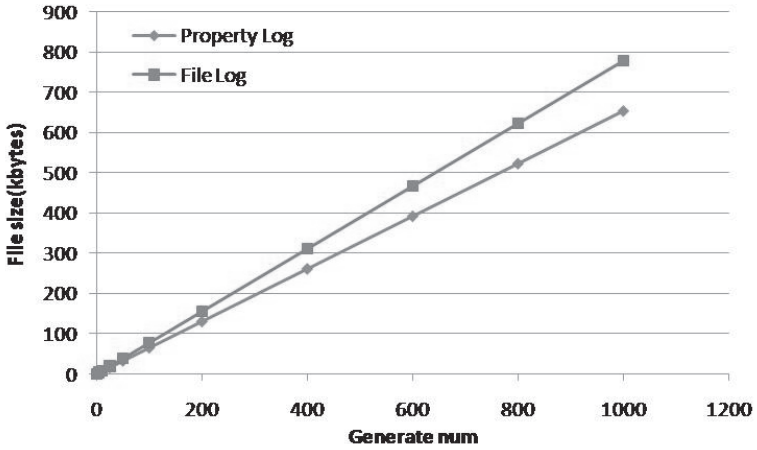

図 10　ログデータのファイルサイズ
Fig. 10　File size of a log.

表 3　更新回数とプロパティログのサイズ
Table 3　Renewal num and Property-log size.

	更新回数合計	プロパティログサイズ
10 分/回	19,200	13,017（KB）
1 時間/回	3,200	2,169（KB）

表 4　各保存方式におけるシステム全体の保存ファイルサイズ
Table 4　All File size for each save method.

	既存システム保存方式	差分保存方式
10 分/回	185,793,320（KB）	353,320（KB）
1 時間/回	9,660,000（KB）	65,333（KB）

また，1 回の更新で同じファイルが別名保存される既存システムの保存方式と，1 回の更新で差分のみが保存される方式を利用した場合に，システム全体に保存されるファイルサイズを**表 4** で比較する．文献 16) より平均的な各更新における差分は 10 KB であり，一般的な業務時間を 1 日 8 時間で 20 日間 PC が稼働すると仮定して計算する．

既存システム保存方式は，オリジナルファイルに差分を加えたものを新たなファイルとして別名保存するため，毎回の更新で保存するファイルサイズは大きくなる．一方，差分保存

方式の場合，差分のみを保存していくため，オリジナルファイル自体が増えることはなく，追加された差分のサイズのみが増加していくことになる．そのため，表 4 のとおり，システム全体に保存されるファイルサイズが，差分保存方式に比べ，既存システム保存方式の方が大きいことが分かる．このとき，各保存方式における，システム全体の保存ファイルサイズに対しての，プロパティログのサイズの割合を比較する．

表 3 と表 4 より，既存システム保存方式の更新頻度が 10 分に 1 回だったとき，全ファイルサイズに対してのログサイズの割合は，およそ 0.007%，1 時間に 1 回のときはおよそ 0.022%となる．また差分保存方式の更新頻度が 10 分に 1 回のときは，およそ 3.684%，1 時間に 1 回のときは，およそ 3.32%である．

7. 考　　察

7.1 各ログデータの計算量

本提案手法では，ファイルプロパティの署名に対して，シェアを保存する PC の台数分，つまりファイルの分割数分の，ファイルリストに対するヒステリシス署名が必要となる．同様に，ファイルリストの署名にも，分割数分のファイルリストのヒステリシス署名と，ファイルプロパティに対するヒステリシス署名が必要となってくる．しかしこれらの署名は，それぞれの PC が計算を行い，署名を必要としている PC に送信されるため，ファイル保存 PC あるいは受信 PC の計算量が極端に増大することはないといえる．

また署名の検証は 6.2 節に示したとおり非常に高速に行えるため，こちらも計算にともなう PC の負荷は大きくないといえる．

7.2 プロパティログのファイルサイズ

本提案システムは既存システムを踏襲しているため，ファイル保存時に 6.2 節で述べた既存システム保存方式を用いており，システム全体の保存ファイルサイズは表 4 のとおり，非常に大きくなる．このとき，プロパティログのサイズが全保存ファイルサイズに対して占める割合は，6.2 節のとおり，1%未満である．しかし今後は，差分保存方式を用いて，効率良くファイルを保存する必要がある．このとき，プロパティログのサイズが全保存ファイルサイズに対して占める割合は，およそ 3%である．以上の結果から，本システムでファイルの保存方式に，差分保存方式を用いた場合でも，保存ファイルサイズに対するプロパティログの割合は小さく，ユーザがシステム運用上，ログのサイズを意識することはないと考えられる．

また，単純にプロパティログのサイズのみを考えた場合，6.2 節での評価環境において，10 分に 1 回の更新時におよそ 13,017 KB，1 時間に 1 回の更新時におよそ 2,169 KB となる．しかし，現在クライアント用 PC として利用される HDD の容量は 1 TB 近くあるものが多い．このような環境下において本システムを利用する場合，160 時間（1 カ月）に 2 MB〜13 MB 程度のファイルサイズが保存されたとしても，1 TB あたり約 0.00075%であり，一般的な PC の耐用年数 5 年間の間に約 0.045%のディスク領域を使用したとしても，ユーザの PC 利用に大きな影響を与えることもないと考えられる．

8. ま　と　め

これまで述べてきたように，本提案手法を既存の P2P ファイル共有システムに導入することで，ファイルを効率的に管理でき，かつ管理者不在の環境において，ファイル情報の信頼性も確保できるといえる．

また，近年はデータの情報漏えいだけでなく，内部者によるデータの改竄や不正な記録の削除など証拠隠滅の問題が頻発しているが，このような事態となる前にデジタルフォレンジックが考慮されたシステムを導入することが望ましいといえる．しかし，デジタルフォレンジックを考慮した文献 8) は，ログの対改竄性を高めるために特殊なハードウェアを用いており導入コストが高い．

ヒステリシス署名においては，連続している署名を途中で改竄することは理論的に不可能であるが，署名を行う者が不正を行うと，一連の署名を先頭部分から捏造可能であるため，ログを自由に作成できてしまう．さらに，ログに対するヒステリシス署名には弱点があり，新しいログから順に削除していくと削除された部分の直前までは一連の署名が正しくつながっているため，改竄は困難でも証拠隠滅は容易である．文献 8) では，ヒステリシス署名を作成する連鎖用データの保存場所として耐タンパデバイスを必要としており，ログの正当性の根拠がハードウェア依存となってしまっている．

これに対して本手法では，ログを複数の PC に関連性を持たせたうえで署名を行っており，システム参加 PC に各署名の検証用ログファイルを保持することで，特殊なハードウェアを用いることなくログの部分削除を困難にしている．本システムの運用環境においてはすべての PC の管理者権限を持つユーザは存在せず，各人が数台の PC を管理することを想定しているため，内部者にとっても管理者権限を持たない PC に保存されたログの削除は困難である．さらに，各 PC に記録されるログに対しては，誰か 1 人でもファイルの操作を行えば，そのユーザが操作を行った PC の鍵で複数の PC へヒステリシス署名が記録されるため，証拠隠滅に賛同しない者が組織内にいる限り証拠隠滅は計れず，逆に不正操作の記

録が不正者の署名付きで残存することとなる．

よって本手法は，ファイル操作に対する証拠保全の面でデジタルフォレンジックに効果的であるといえる．

なお，本提案システムは Linux のオープンソースプロジェクトの一環として開発しており，実装したものは随時公開している．

謝辞　本研究は文部科学省オープン・リサーチ・センター整備事業（平成 16 年度〜20 年度）による私学助成を得て行われた．ここに記して謝意を表す．

参 考 文 献

1) 大津一樹，宇田隆哉，井上亮文，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実装評価，情報処理学会論文誌，Vol.47, No.8, pp.2464–2476 (2006).
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，電子情報通信学会 SCIS2005 論文集，Vol.1, pp.1–6 (2005).
3) 東森ひろこ，宇田隆哉：署名付き更新履歴を持つファイル分散保存システム，情報処理学会 CSS2007 論文集，Vol.2007, No.10, pp.121–126 (2007).
4) 平野仁之，中村康弘：分散ストレージシステム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 CSEC 研究報告，Vol.27, No.129, pp.83–88 (2004).
5) 井上亮文，石原礼男，大津一樹，鹿島隆行，手塚　伸：帯域・記憶領域共有による管理者不在のセキュア分散ファイル管理，2005 年度下期未踏ソフトウェア創造事業 (2005).
6) Haber, S. and Stornetta, W.: How to Time Stamp a Digital Document, *Advances in Cryptology-Crypto'90*, pp.437–455 (1991).
7) Schneier, B. and Kelsey, J.: Automatic Event-Stream Notorization Using Digital Signatures, *International Workshop*, pp.155–169 (1997).
8) 芦野佑樹，粉川寛人，佐藤　吏，佐々木良一：セキュリティデバイスとヒステリシス署名を用いたデジタルフォレンジックシステムの提案と評価，情報処理学会論文誌，Vol.49, No.2, pp.999–1008 (2008).
9) Aladdin Knowledge Systems, Ltd. http://www.aladdin.com/
10) Schneier, B. and Kelsey, J.: Cryptographic Support for Secure Logs on Untrusted Machine, *The 7th USENIX Security Symposium Proceedings*, USENIX Press (1998).
11) Daemen, J. and Rijmen, V.: AES Proposal: Rijndael document version 2 (1999).
12) Rivest, R., Shamir, A. and Adleman, L.: A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, *Comm. ACM*, Vol.21, No.2, pp.120–126 (1978).
13) Rabin, M.: Efficient dispersal of information for security, loadbalancing and fault tolerance, *J. ACM 38*, Vol.36, pp.335–348 (1989).
14) U.S. National Institute of Standards and technology: Secure Hash Standard (1995).
15) Shin, T., Ryuya, U., Akihumi, I. and Yutaka, M.: A SECURE VIRTUAL FILE SERVER WITH P2P CONNECTION TO A LARGE-SCALE NETWORK, *IASTED International Conference NCS 2006*, pp.310–315 (2006).
16) 手塚　伸，東森ひろこ，井上亮文，宇田隆哉：Linux におけるセキュアな分散仮想ファイルシステム，電子情報通信学会 SCIS2008 予稿集，4F1-3 (2008).
17) crypto++ Library5.2.1. http://cryptopp.sourceforge.net/docs/ref521/index.html
18) Filesystem in Userspace (FUSE). http://fuse.sourceforge.net/

(平成 20 年 9 月 4 日受付)
(平成 21 年 3 月 6 日採録)

東森ひろこ

2007 年東京工科大学コンピュータサイエンス学部卒業．2009 年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修了．2006 年度上期未踏ソフトウェア創造事業（未踏ユース）採択．現在，KDDI 株式会社勤務．

手塚　　伸（学生会員）

2006 年東京工科大学工学部情報工学科卒業．2008 年同大学大学院前期博士課程修了．現在，慶應義塾大学大学院理工学研究科開放環境科学専攻後期博士課程に在学中．慶應義塾大学 ITC 助教．ネットワークセキュリティの研究に従事．2005 年情報処理学会 DICOMO2005 シンポジウムにおいてヤングリサーチャ賞，2006 年情報処理学会 DICOMO2006 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）．

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．

伊藤　雅仁（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修士課程修了．2003 年同大学院理工学研究科後期博士課程修了．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．ユビキタスコンピューティング，情報家電の研究に従事．電子情報通信学会員．

市村　　哲（正会員）

1989 年慶應義塾大学理工学部計測工学科卒業．1994 年同大学大学院理工学研究科博士後期課程修了．博士（工学）．同年富士ゼロックス（株）入社．1997〜1999 年富士ゼロックスパロアルト研究所駐在．2002 年 4 月より東京工科大学．現在，同大学准教授．グループウェア，生体情報活用等の研究に従事．

田胡　和哉（正会員）

1986 年筑波大学大学院工学研究科博士課程修了．工学博士．筑波大学電子情報工学系助手，東京大学工学部助手，日本 IBM 東京基礎研究所を経て，現在，東京工科大学コンピュータサイエンス学部教授．オペレーティングシステムの構成方式に興味を持つ．1984 年情報処理学会論文賞．ACM 会員．

星　　徹（フェロー）

1969 年東京工業大学工学部電気工学科卒業．同年（株）日立製作所入社．戸塚工場，システム開発研究所，中央研究所等で，交換システム，マルチメディア LAN，CSCW，デスクトップ会議，CTI，IP テレフォニー等の研究開発に従事．この間，1975 年カリフォルニア大学ロサンゼルス校（UCLA）大学院コンピュータサイエンス専攻修士課程修了．2003 年 4 月より東京工科大学コンピュータサイエンス学部教授．2007 年 4 月より学部長．マルチメディアコミュニケーション，RFID タグ応用等，ユビキタスネットワーキングの研究に従事．工学博士．2006 年 3 月情報処理学会フェロー．電子情報通信学会，電気学会，IEEE，ACM 各会員．