社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－DPS－138 (40)
2009－CSEC－ 44 (40)
2009／3／6

ヒステリシス署名による改竄防止機構を備えたログ用ファイルシステムの提案

鶴田浩史† 齋藤彰一†
上原哲太郎†† 松尾啓志†

インターネットの普及により，コンピュータを用いた犯罪による訴訟が増えている．訴訟の際，アクセス記録などを保存しているログファイルの信頼性が問題となる．そのため，ログファイルなどのデジタルデータの証拠性を高める技術であるデジタルフォレンジックが重要となっている．本論文では，ログを追記のみにするファイルシステムとヒステリシス署名を用いることで，ログファイルの改竄を防止し，かつ，改竄を検出するシステムを提案する．これにより，マルウェアの攻撃やユーザによるログファイルの改竄を防止し，ログファイルの改竄を検出することができる．これにより，管理者が不正を行っていないという証拠を第三者が証明することができる．また，プロトタイプシステムにより，ログファイルの正当性と署名ファイルの正当性を確認した．

Proposal of Filesystem for Log with Falsification Detection by Hysteresis Signature

Koji Tsuruta ,† Shoichi Saito ,† Tetsutaro Uehara †† and Hiroshi Matsuo†

A lawsuit caused by a crime with computers has increased by the spread of the Internet. Reliability of log files which are managed by a system administrator is important at the lawsuit. Digital forensics technology is widely used for protecting evidences in digital data of the log files. In this paper, we propose a new system to prevent falsifying log files, and to detect falsification. The proposed system uses an appending only filesystem and a hysteresis signature. It can prove the system manager not to modify the log files. Moreover we build a prototype system, and verify the log files and a signature file by the prototype system.

1. はじめに

インターネットの普及にともない，コンピュータを用いた犯罪による訴訟が増えている．訴訟の原因として，ネット詐欺，誹謗中傷の書き込み，不正アクセス，個人情報などの機密情報の漏洩が挙げられる．訴訟に対して，コンピュータの操作記録やアクセス記録などのデジタルデータが証拠として扱われる．このデジタルデータを以後、ログと呼ぶ．ログを訴訟における証拠とするためには，ログを保全し，ログに対して不正がされていないことを証明する必要がある．しかし，デジタルデータのログは加工が容易であるため，不正された痕跡を消去することも容易であり，証拠性が失われやすい．

† 名古屋工業大学
Nagoya Institute of Technology
†† 京都大学
Kyoto University

そのため，ログを保全し，法的な証拠性を保証するための技術であるデジタルフォレンジックが重要になる．デジタルフォレンジックは，コンピュータに関連する訴訟が多くなってきていることにより注目され始めている技術である．最近では，事件が起きた際の法的な証拠性を証明するだけでなく，ネットワーク上の全パケットを記録し分析する技術や，コンピュータ操作をモニタリングし事件の予兆の発見や防止を促すことで事件を抑止することもデジタルフォレンジックとして扱われている．

訴訟の際，ログの内容が記録されているログファイルの証拠性を保証する必要がある．これまで，ログファイルを保護する研究が盛んに行われてきた．提案されてきた手法には，暗号を用いてログファイルが改竄されていないことを証明する手法[1)]やログファイルのバックアップを複数作成することで改竄を検出する手法[2)]がある．しかし，これらの手法には，ログファイルを改竄することはできないが消去すると消去したこと自体を検出できない問題や，改竄を検出すること

はできるが改竄自体を防止できない問題がある．

本稿では，ログファイルの信頼性を高める手法を提案する．本手法を用いることによりログの改竄を防止できる．さらに，ディスクを持ち出された場合やマルウェアの攻撃などによる改竄を検出することができる．この結果，システムの管理者（以下，管理者と呼ぶ）がログファイルに対して不正をしていないことを証明する．提案手法では，マルウェアとユーザによる改竄の防止として追記のみのファイルシステムを用いる．また，ログファイルの消去と改竄の検出としてヒステリシス署名を用いる．

本稿では，2 章で既存手法について述べ，3 章で提案手法について述べる．4 章では提案手法によるシステムの実装とシステムの評価について述べる．5 章でまとめと今後の課題について述べる．

2. 既存手法

ログファイルの保護手法は，今日までにさまざまな方法が提案されている．Schneier らが提案した手法[1]は，ログデータの Message Authentication Code を作成する際に用いる署名用の鍵を，処理毎に破棄し，更新する手法である．このため，鍵が盗まれても過去の鍵は存在しないため，過去に署名したデータを改竄することはできない．しかし，検証者が最初の鍵を持っていなければならない．また，検証者が鍵を使ってログデータを改竄してしまうと，その改竄自体を発見することができない問題がある．そのため，管理者が不正をしていないという保証ができない．

高田らが提案した手法[2]は，定期的にログファイルのバックアップを複数作成することで，ログの改竄と消去を検出しログファイルの内容を復元可能とする手法である．この手法では，バックアップを隠蔽することで，ログファイルの改竄を困難にするという特徴がある．また，バックアップを含めたすべてのログファイルを定期的に監視することで，改竄を検出することができる．しかし，バックアップを頻繁に作成するため負荷が増大する問題がある．また，バックアップを含めたすべてのログファイルを攻撃者に知られると，改竄や消去が行われる問題がある．

3. 提案手法

本章では，提案手法であるログファイルの証拠性を保証するヒステリシス署名を用いた追記のみのファイルシステムについて述べる．

3.1 概　　要

ログファイルは，事件が発生したとき原因を特定するための重要な情報である．しかし，一般にファイルは任意の場所を書き換えることができる．そのため，不正をした痕跡を消去するためにログファイルを書き換えることが考えられる．もちろん，ログファイルを書き換えるには管理者権限が必要であるが，セキュリティホールなどの脆弱性を利用した攻撃により管理者権限が奪われて，ログファイルの改竄が可能になる可能性がある．このような状況において，改竄を防止するには，管理者権限でもログファイルの書き換えを不可能にすればよい．

一般にログファイルは，追記されるのみであるため，ユーザが利用するファイルのように任意の場所を書き換えることを想定する必要はない．したがって，追記のみのファイルを作成すればよい．しかし，ログファイルを追記のみに設定する方法が問題になる．もし，ログファイルのアクセス権限を管理者が任意に設定できるとすると，再び管理者権限の乗っ取りが問題になる．そこで，ファイルシステム全体を追記のみとすることで管理者によるファイル単位での設定を不要とする．

ファイルシステム全体を追記のみにするとすべてのファイルは任意の場所を書き換えることができなくなる．そこで，ログファイル専用の追記のみのファイルシステムを提案する．ファイルシステムとは，記憶装置に記録されているデータを管理するために，オペレーティングシステムが持つ機能の 1 つである．これは，記憶装置上のデータを読み書きするために使われるファイルというインタフェースを提供する仕組みのことである．また，ファイルの所有者と各ユーザに対するアクセス権限をファイルごとに管理することができる．このため，ログファイルに対するアクセス権限を読み出しと追記のみにすることができ，ログファイルの改竄を防止することができる．

しかし，ディスクを持ち出したり，ファイルシステムの脆弱性を利用した攻撃により改竄される可能性がある．改竄された場合に対応するために，改竄を検出するシステムが必要である．改竄を検出するためには，デジタル署名を利用する．デジタル署名を用いることで，ログファイルが改竄されていないことを証明することができる．また，署名は秘密鍵が安全に格納できる耐タンパデバイスで行う．

しかし，ログファイルは長期間に渡って保管される場合が多い．デジタル署名は時間が経つと署名の安全性や有効性が危殆化してしまったり，有効期限が過ぎてしまうといった問題がある．このために，ヒステリシス署名[3]を用いる．ヒステリシス署名とは，署名する際，過去に署名したデータを署名対象データに含めて署名を行う方式である．過去の署名を用いることによって，過去から現在までされてきた署名を連鎖することができる．ヒステリシス署名を用いることで，長期間にわたり安定してログファイルの改竄を検出できる．具体的なヒステリシス署名による署名の作成の流れは 3.2 で述べる．

3.2 署名を用いた追記のみのファイルシステム

ヒステリシス署名を用いた追記のみのファイルシステムについて述べる．追記のみのファイルシステムで

表 1　システムの特徴

	ログファイルの改竄防止	ディスクの持ち出しやコピー後の改竄検出
追記のみのファイルシステム	○	×
署名システム	×	○
提案システム	○	○

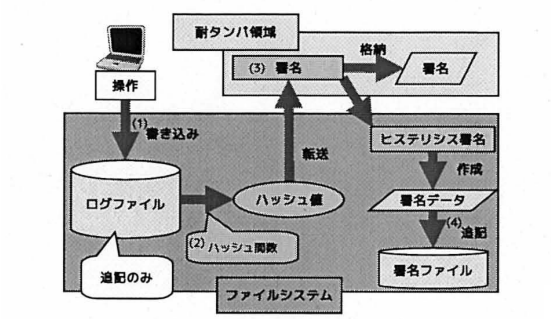

図 1　提案システムの概要

改竄を防止し，ヒステリシス署名で改竄を検出する．2 つの手法を組み合わせることで，それぞれの問題点を補うことができる（表 1 参照）．

提案システムでは，読み出しと書き込みに関するシステムコールが発行された時にログをログファイルに追記する．提案システムの概要を図 1 に示し，処理の流れを以下に述べる．なお，図 1 の番号と以下に示す番号はそれぞれ対応する．

(1) 追記のみのファイルシステム

追記のみのファイルシステムの実現手法を述べる．ログファイルを追記のみにするには，ログファイルのアクセス位置によって，書き込み権限を決定する必要がある．ログファイルのアクセス位置がログファイルの末尾を指している場合のみ，書き込みの許可を与える．しかし，アクセス位置がログファイルの末尾以外を指しているときは，読み出し権限のみを与える．読み出し権限に対しては特に制限はない．アクセス位置によって書き込み権限を制限することによって追記のみのファイルシステムを実現する．この手法により，ログファイルの改竄を防止する．

(2) ハッシュ値作成

ログファイルのハッシュ値を作成する．このハッシュ値を基に署名を行う．この処理もファイルシステム内で行う．

(3) ヒステリシス署名

ファイルシステムで作成されたハッシュ値を用いて，ヒステリシス署名用のデータを作成し，耐タンパデバイスの秘密鍵を用いて署名する．また，作成した署名を耐タンパデバイスに格納する．その後，署名をファイルシステムへ送り返す．

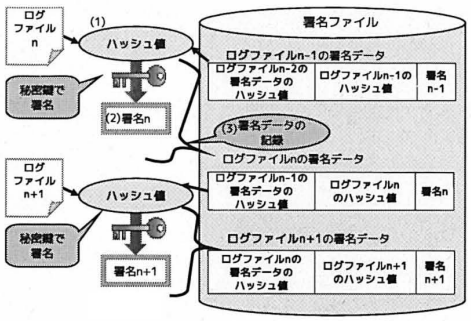

図 2　ヒステリシス署名の流れ

表 2　ヒステリシス署名の構成要素

n 番目の署名データ	SignData（n）
n 番目の署名を行う時点でのログファイル	LogFile（n）
n 番目の署名	Sign（n）
ハッシュ関数	Hash（）

ヒステリシス署名[3] では，最新の署名に 1 つ前の署名データが反映されることにより，署名を連鎖させる．署名データとは，1 つ前の署名データのハッシュ値とログファイルのハッシュ値と署名を一括りにまとめたものである．署名の連鎖を繰り返すことで，過去の署名データが改竄された場合でも，最新の署名を用いることで，改竄を検出することができる．ヒステリシス署名の流れを図 2 に示し，新しい署名データ（n 番目とする）を作成する手順を以下に述べる．なお，以下で用いる各要素を表 2 に示す．

まず，署名ファイル上の最新（n-1 番目）の SignData（n-1）には，Hash（SignData（n-2））と Hash（LogFile（n-1））と Sign（n-1）が登録されている．署名ファイルとは，過去に署名した署名データをすべて記録したファイルである．

新しい SignData（n）を作成する手順は次のとおりである．

（1） Hash（LogFile（n））と Hash（SignData（n-1））を連結し署名用の値を作成する．

（2） （1）で作成した値を耐タンパデバイスに格納されている秘密鍵で署名し Sign（n）を作成する．

（3） Hash（SignData（n-1））と Hash（LogFile（n））と Sign（n）を SignData（n）として署名ファイルに追記する．

以上によって，SignData（n-1）と SignData（n）の間の署名が連鎖する．

しかし，署名ファイルに署名データを追記するだけでは，ディスクを持ち出した場合に，改竄を検出することができない．このような攻撃を防ぐために最新の署名を耐タンパデバイスにも保管する必要がある．これにより，署名を検証することでディスク持ち出し攻撃による改竄を検出できる．署名検証方法は 3.3 で述

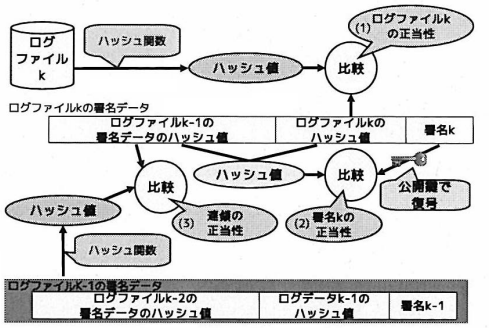

図 3 署名検証の流れ

べる．

(4) 署名データの追記

耐タンパデバイスで作成した署名データを署名ファイルに追記する．この処理は，ファイルシステムで行う．追記のみのファイルシステムで行うことで，ログファイルを追記のみにする手法と同等の効果があり改竄を防止できる．

3.3 署名検証

検証の概要を図 3 に示し，ログファイル（k 番目）の正当性と署名ファイルの正当性を検証する手順を以下に述べる[3]．提案システムで作成した署名における，署名ファイル上の最新（k 番目）の Sign（k）と耐タンパデバイスに格納されている最新の署名の一致を検証する．その後，ログファイルと署名ファイルの正当性を検証する．以下の番号は，図 3 の番号にそれぞれ対応する．

(1) ログファイルの正当性検証

LogFile（k）が改竄されていないことを検証する．この検証では，k 番目の署名を行った時点におけるログファイルからハッシュ値を求め，署名ファイル中の Hash（LogFile（k））の値を比較し同一であるか判定する．もし，一致しなければログファイルが改竄されていることを検証者に示し，検証を終了する．

(2) 署名データの正当性検証

SignData（k）が改竄されていないことを検証する．この検証では，Hash（SignData（k-1））と Hash（LogFile（k））から作成した署名用の値と Sign（k）を耐タンパデバイスに格納されている公開鍵で復号した値を比較し，同一であるかを判定する．もし，一致しなければ署名データが改竄されていることを検証者に示し，検証を終了する．

(3) 署名データ間の連鎖の正当性検証

最後に，SignData（k）と SignData（k-1）の間で署名が連鎖しているかを検証する．この検証では過去から現在に至るまで連鎖して署名しているか検証する．SignData（k）上の Hash（SignData（k-1））と SignData（k-1）から作成したハッシュ値を比較し同一であるか判定する．もし，一致しなければ SignData（k）と SignData（k-1）の間に連鎖がないことを意味する．よって，1 つ前の署名を利用して署名を作成していないことを検証者に示し，判定を終了する．一致した場合は，署名データ間で連鎖の検証を過去に遡って順に署名ファイルの先頭まで検証する．

以上の手順で，ログファイルと署名ファイルに対する検証を行う．すべての手順で検証が正しい場合は，ログファイルの正当性と署名ファイルの正当性を確認することができる．しかし，いずれかの場合で検証が正しくない場合，ログファイルもしくは，署名ファイルが改竄されていることを意味する．

4. 予備評価とプロトタイプシステム

本章では，提案システムのプロトタイプシステムについて述べる．まず，システムを実装するシステム環境と予備評価について述べる．次に，予備評価に対する問題を考慮したプロトタイプシステムの実装方式とその評価を述べる．

4.1 システムの環境

本提案システムの有効性を検証するためにプロトタイプシステムを実装した．追記のみのファイルシステムを実装するためには，Filesystem in Userspace[4]（以下，FUSE と呼ぶ）を用いた．また，耐タンパデバイスには eToken PRO[5]（以下，eToken と呼ぶ）を用いた．これらと検証用プログラムの各システムについて述べる．

4.1.1 FUSE

Unix 系オペレーティングシステムのカーネルモジュールの一種で，一般ユーザがカーネルコードを修正することなく，独自のファイルシステムを作成できる機能を提供する．FUSE を用いることで，ファイルシステムをユーザ空間で実行することができる．FUSE を用いて，追記のみのファイルシステムを実装した．

4.1.2 耐タンパデバイス

アラジン社の eToken PRO[5]（以下，eToken と呼ぶ）を用いた．eToken は，耐タンパ領域を備えており，その内部に秘密鍵と公開鍵を保存することができる．また，秘密鍵を外部に取り出すことができないセキュアデバイスである．RSA の公開鍵暗号機能を利用することができる．さらに，標準 Crypto API および，PKCS#11 をサポートしているためアプリケーションから暗号トークンインタフェースを利用することで eToken にアクセスすることができる．

4.1.3 検証プログラム

プロトタイプが生成したログファイルと署名ファイルの正当性を検証するプログラムを作成した．本検証プログラムは，3.3 で述べた署名検証の手順で検証を行う．

表 3　ハッシュ値計算時間

ログファイルの大きさ [MB]	ハッシュ値計算時間 [秒]
10	0.29
100	1.92
200	4.07
500	9.42
1000	18.73

4.2 予備評価

予備評価としてファイルサイズに対するハッシュ値の計算に要する時間と，eToken における署名に要する時間をについて計測し，計測時間に対する考察を行った．

4.2.1 ハッシュ値の計算時間

1 日のログファイルの大きさとして，約 1GB のログファイルを想定する．この大きさは，名古屋工業大学におけるファイルサーバアクセスログの大きさを参考にした．まず，1 日単位でヒステリシス署名を行う場合を想定すると，最大で 1GB のログファイルに対するハッシュ値を計算する必要がある．そこで，1GB のログファイルに対するハッシュ値を計算するために要する所要時間を計測した．20 回の計算から平均値を求めた結果，18.7 秒であった．そのため，1GB のログファイルの場合では，平均 18.7 秒の間，ファイルシステムがハッシュ値を計算することに専念するために停止し，運用ができなくなると考えられる．そのため，ハッシュ値を作成する時間を減らしシステムの負担を軽減する必要がある．

負荷を軽減する方法として，ハッシュ値を計算する対象となるログファイルの大きさを制限する．そこで，システムを運用する際にどの程度まで許容できるか把握する必要があるため，ログファイルの大きさによるハッシュ値の計算時間を計測した．結果を表 3 に示す．測定結果から，ログファイルの大きさに比例してハッシュ値の計算に要する時間が長くなっている．ハッシュ値を計算している時間は，システムが停止し運用ができなくなる問題が発生する．そのため，ハッシュ値を計算する時間を極力小さくする必要がある．そこで，ログファイルの大きさをファイルシステム内部で分割する手法を提案する．詳細は，4.3 で述べる．

4.2.2 eToken の署名時間

eToken で署名を作成する時間を計測した．eToken で署名を作成し，署名を eToken 内部に格納し署名した結果を返すため時間は 1.7 秒である．しかし，ファイルシステムで処理する機構と eToken で処理する機構は，独立している．このため，ファイルシステム中の処理と eToken の処理をオーバラップさせることでシステムを停止させる必要はない．

4.3 プロトタイプシステム

ログファイルの大きさをファイルシステム内部で分割するプロトタイプシステムを実装した．プロトタイプシステムの概要を図 4 に示す．プロトタイプシステムでは，FUSE を用いて追記のみのファイルシステムを実現し，ファイルシステム内部でログファイルを分割する．実装したファイル分割方式の概要を図 5 に示す．ファイルシステム内部では，ログファイルの大きさを監視し，分割サイズを超えると自動的に分割する．ログファイルを分割することでハッシュ値の計算時間を短時間に抑えることができる．ただし，ユーザからは，ファイルシステム内部の分割を見ることはなく，通常の大きさのファイルとして扱うことができる．

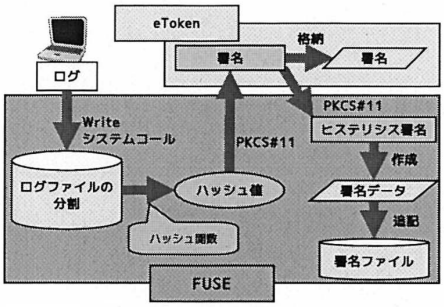

図 4　プロトタイプシステムの概要

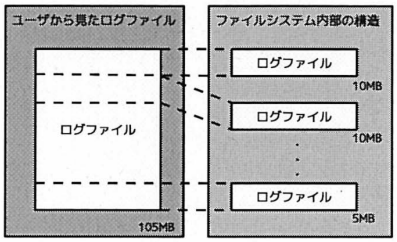

図 5　ファイルの分割

4.4 評　　価

ログファイルの正当性と署名ファイルの正当性の検証を行った．署名ファイルに含まれる各署名データは，ヒステリシス署名によって署名が連鎖している．また，ログファイルを用いて最新の署名データを作成しているため，ログファイルと署名ファイルを改竄しても検証プログラムにより，改竄を検出できた．

ファイルシステムでファイルを分割し，500MB のログファイルのハッシュ値を計算するのに要する時間を分割サイズを変えて計測した．計測した結果を表 4 に示す．測定結果から，分割した容量に比例して，ハッシュ値を作成する時間が長くなっている．ハッシュ値を計算するには，ファイルを一度走査する必要がある．ファイルシステムで分割した容量がログファイルのサイズを下回る場合，分割したサイズの部分だけハッシュ値を計算するため，表 4 のような結果になったと考えられる．

また，3.2 で述べたように，ログファイルにデータ

表 4　500MB のログファイルのハッシュ値作成時間

分割サイズ [MB]	10	100	200	500
作成時間 [秒]	0.21	2.13	4.32	10.81

を追記するのは，システムコールが発行されたときである．そのため，システムコールが長時間に渡って発行されない場合，ログファイルに記録が追記されず署名を作成できない．したがって，長時間システムコールが発行されない場合の対策を行う必要があると考えられる．

5. まとめと今後の課題

ヒステリシス署名を用いた改竄検出機構を備えたログ用ファイルシステムを提案し，プロトタイプシステムを実装した．ログファイルを追記のみにすることで，ログファイルに対する攻撃を防止することができる．また，ヒステリシス署名を用いることで，ログファイルの改竄を検出することができる．署名ファイルを確認することで，ログファイルの正当性と署名ファイルの正当性を確認することができる．

今後の課題として，システムの完成と時間間隔でログファイルを署名する方法を実装する．また，署名をする際にかかるオーバヘッドを短縮する方法を検討する予定である．

参 考 文 献

1) Schneier, B. and Kelsey, J.: Cryptographic Support for Secure Logs on Untrusted Machines, *The 7th USENIX Security Symposium Proceedings*, pp.53–62 (1998).
2) 高田哲司，小池英樹：逃げログ：削除まで考慮にいれたログ情報保護手法，情報処理学会論文誌，Vol.41, No.3, pp.823–831 (2000).
3) 芦野佑樹，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.999–1009 (2008).
4) Filesystem in Userspace: http://fuse.sourceforge.net/.
5) Aladdin Knowledge Systems: http://www.aladdin.com/.