情報処理学会論文誌　Vol. 51　No. 9　1507–1519 (Sep. 2010)

不正プログラムの起動制御機能を持つDFシステムの提案と評価

藤田圭祐[†1]　芦野佑樹[†1]
上原哲太郎[†2]　佐々木良一[†1]

近年，ITへの依存度の高まりにともない，裁判などにおいて，証拠としてデジタルデータが用いられる場面が増加してきている．今後は日本においても訴訟の増加が予想されるため，訴訟に持ち込まれても問題なく対応することのできる証拠能力を持ったデジタルデータの確保が重要になると考えられる．そこで著者らは，スタンドアロン環境下でPCを操作した内容をログデータとして蓄積するだけでなく，操作した本人であってもログデータの改竄を防止するDF（Digital Forensic）システムの開発を進めてきた．従来は，プログラムの不正な改竄がないと仮定していたが，現実にはこの仮定が成立しない場合も少なくない．そこで著者らは，APIHookを用いた不正プログラムの起動制御機能を持ち，PCなどのマザーボード上に実装されているセキュリティチップであるTrusted Platform Module（以下，TPM）を用いて信頼性を向上させたプログラムによって，正しいプログラムのみが稼働しているように制御する方式を提案する．あわせてそのプロトタイププログラムを開発し，機能実験を行ったので実験結果の報告も行う．なお，提案技術は，PCの所有者が不正を行おうとしても，正しいプログラムのみが稼働しているように制御する方式として，DFシステム以外にも広く利用が可能であると考えている．

Proposal and Evaluation of DF System with Boot Control Function against Unauthorized Programs

Keisuke Fujita,[†1] Yuki Ashino,[†1]
Tetsutaro Uehara[†2] and Ryoichi Sasaki[†1]

In recent years, society has become more dependent on information technology, and so there have been more cases of digital data used as evidence in courts, etc. Since it is also predicted that there will be more court cases in Japan, we expect it will become important to secure digital data that is legally admissible in lawsuits. With this background, the authors of this paper have been developing a Digital Forensic (DF) system that accumulates the operation contents of a PC in a standalone environment as log data and also prevents alteration of the log data, even by those who conducted the operations recorded in them. Although traditionally, no illicit alteration to the program has been assumed. In reality, however, this assumption does not necessarily hold true. Therefore, we propose in this paper a DF system that enables programs to start correctly and prevents tampered programs from starting with APIHook function and the Trusted Platform Module and its related program. Moreover, we report the results on the functional experiments of the proposed system using the proto type program. The method in the proposed system can be used to control unauthorized programs not only in the DF system, but in other systems.

1. はじめに

インターネット社会の進展にともない，ほぼすべてのデータはデジタル化されて扱われるようになってきた．また，日本でも，従来は考えられなかったような場合にも訴訟が行われるようになってきている．このような状況から，デジタルデータの証拠性を確保し，訴訟などに備えるための技術や社会的仕組みが要求されるようになってきた．これが，デジタル・フォレンジック（Digital Forensics：以下DF）と呼ばれているものである[1)]．

第三者の不正の痕跡や証拠性を確保するためのDFツールはすでにForensic Toolkit[2)]などが発売されており，DFに関する研究も増加しつつある[3),4)]．しかし，PCの利用者自身が不正を行っていないことを証明する方式の開発は従来行われてこなかった．

このような機能を実現するシステムとして，著者らはスタンドアロン環境下で用いるDFシステム“Dig-Force（Digital Forensic system with Chaining signature for Evidence）”の提案と開発を行ってきた[5)]．

しかし，著者らが提案したシステムでは，PCのユーザによってプログラムの改竄ができないという前提条件をおいていた．この前提条件は実際に適用する場面を想定すると成立が困難な場合もある．また，ここで本方式はOSとしてはシェアの大きさからWindowsを前提とする．このとき，Windows-PCの利用者が既存のプログラムを改竄したり，別のプログラムを挿入したりすることはそれほど困難ではない．そのため，正しいプログラムのみがWindows-PC上で稼動していることを確認できる方式が必要になった．

†1 東京電機大学
Tokyo Denki University
†2 京都大学
Kyoto University

そこで，著者らは正しいプログラムのみが稼働している環境を構築するために 2 つの方式を考えた．

1 つ目の方式はホワイトリストを用い，そこに登録された以外の不正プログラムが起動しようとすると APIHook 機能を用いて起動を防止する方式である．

2 つ目の方式は上記の起動制御プログラム自体の改竄を防止するためのもので起動制御プログラムが入ったハードディスク（以下 HDD）を外付け HDD として別の PC に接続して修正するのを防止するための方式である．対象となる PC のマザーボード上に実装されているセキュリティチップである Trusted Platform Module（以下，TPM）を用いて HDD の暗号化を行うことで，他の PC では復号できないため目的とする改竄が不可能となる．

この 2 つの方式を組み合わせることで，既存方式[5] である DF システムの証拠性保全機能の有用性を保ちつつ，不正プログラムの起動を防止することで安全性を向上させることが可能となった．

ホワイトリストと APIHook を用いた正当性チェック方式や TPM を用い HDD を暗号化する方式はすでに知られている．また，TPM を利用した研究はネットワーク環境下でプログラムの正当性検証を行う方式の研究や[6),7)]，その TPM を用いたプログラムの正当性検証を DRM に応用する方式の研究[8] などがすでに検討されている．これらの方式は OS に Linux を用いて BIOS の改竄不可能な領域を信頼の拠点とするプログラムの正当性検証機能を検討している．この機能ではプログラムの計測値を TPM に格納し，その値をもとに正当性検証を行う．

しかし，Windows にはプログラムの計測を行い，TPM に格納するという機能がないため上記の方式を用いることができない．そのため，起動制御プログラムの入った HDD を別の PC に接続し，改竄することを防止する機能が必要となり，前記した 2 つの方式を組み合わせるアプローチを考案した．

このような 2 つの方式を組み合わせスタンドアロン環境であっても正しいプログラムのみが稼働しているように制御する方式は少なくとも Windows に関しては，DF 分野をはじめとして現在まで提案されていないと考えている．なお，Windows-PC 所有者の不正を防止し，正しいプログラムのみが稼働しているように制御する方式が必要なシステムは，建築士が行う構造計算書の作成や経理担当者が行う財務データの作成などほかにも数多く存在する．よって DF の分野だけでなく，これらの問題を解決するためにも本稿で提案する方式を様々な分野で用いることができると考えている．

本稿では，提案方式を記述するとともに，この方式のプロトタイププログラムを開発し，機能実験を行ったので報告を行う．

2. Dig-Force の概要

以下に Dig-Force の概要を述べる[5]．

2.1 Dig-Force の機能

- PC 操作に関する情報をログデータとして確実に収集する．
- 操作者が行うログデータの改竄であっても検知することができる．

2.2 Dig-Force の構成

Dig-Force は以下の 3 つのサブシステムから構成されている（**図 1**）．

(1) セキュリティデバイス
- 耐タンパ領域を持ち，そこに，署名鍵，そして最終署名履歴である連鎖用データを保存し，改竄を防止する．
- 2.4 節で説明するヒステリシス署名の署名演算を行う．

(2) ロガープログラム
- 操作者の証拠となる操作情報をデジタルデータ化し，それをオペレーションデータとしてログストレージプログラムに転送する．
- セキュリティデバイスと連携し PC ロック機能を持つ．すなわち，セキュリティデバイス装着時のみ PC が動作可能となっている．

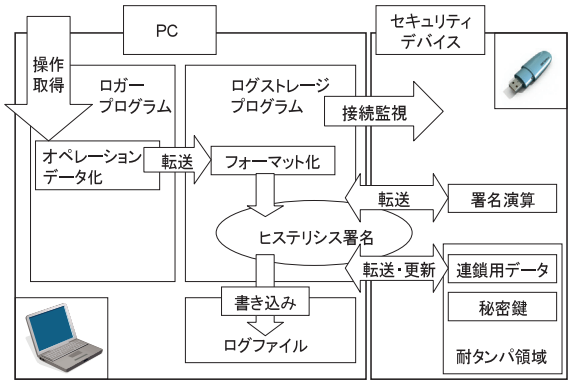

図 1　Dig-Force の構成
Fig. 1　Composition of Dig-Force.

(3) ログストレージプログラム

- 蓄積されたログデータにセキュリティデバイスと通信し，ヒステリシス署名を行う．

2.3 Dig-Force の処理フロー

Dig-Force の処理フローには導入，運用，検証の 3 つのフェーズが存在する．ここには，システムに関わる登場人物として初期設定を行う管理者，PC の操作者，PC に蓄積されたログの検証を行う検証者の 3 者が存在する．

A) 導入フェーズ

管理者はセキュリティデバイスで秘密鍵，公開鍵ペアを生成し，セキュリティデバイス内の耐タンパ領域に秘密鍵を格納する．また最初の連鎖用データとなる初期値を決定しセキュリティデバイス内の耐タンパ領域に格納する．

その後，検証者にセキュリティデバイスで生成した公開鍵，初期値を送付する．また，管理者は操作者にロガープログラムとログストレージプログラムがインストールされた PC とセキュリティデバイスを渡す．

B) 運用フェーズ

操作者は管理者から渡された PC で作業を行う．ロガープログラムはセキュリティデバイスの装着を監視し，装着を確認したら操作者の操作情報をログデータとして取得しログストレージプログラムに転送する．

ログストレージプログラムは次の処理を行う．

(1)　ログデータを蓄積する．
(2)　セキュリティデバイスと通信する．
(3)　ログデータにヒステリシス署名を施す．
(4)　連鎖用データ，署名データ，ログデータをログファイルに書き込む．
(5)　連鎖用データのハッシュ値をセキュリティデバイス内に格納する．

作業が終了し，検証者に自身の操作を報告する際に，作業中に作成したドキュメントとログファイルをメディアに書き込み，セキュリティデバイスとともに検証者へ提出する．

C) 検証フェーズ

検証者は提出されたログファイル内の最終連鎖用データのハッシュ値を計算し，セキュリティデバイスのものと比較を行う．その後，導入フェーズで受け取った初期値，ログファイルに保存されている連鎖用データ，署名データ，ログデータを基に署名検証を行う．

2.4 ヒステリシス署名

ヒステリシス署名は，通常の電子署名に直前の署名データを引き継いで署名を行う，署名間に連鎖構造を持たせた電子署名である[9]．

ヒステリシス署名を用いることにより，署名履歴に依存関係が存在することになる．そのため，データを改竄するためには，それに対応する署名だけでなく前後にあるすべての署名データを改竄する必要があるので安全性が高い．

Dig-Force では順次生成されるログデータごとに連鎖的に署名を行うため，ログデータの一部を削除する攻撃や，一部を変更するといった攻撃に対処可能である．またセキュリティデバイスと併用することによりヒステリシス署名の課題であったログファイルに蓄積されているログデータの末尾を消去し，新たに一連の操作を行い，ログファイルを更新するといった復元攻撃に対処可能となる[5]．

2.5 Dig-Force の前提条件

なお，Dig-Force が目的とする機能を実現し，安全を確保するための前提条件は以下のとおりである．

（前提条件 1）　PC 内の各プログラムは不正に変更されることはない．
（前提条件 2）　操作者は不正を行う可能性はあるが，管理者，検証者は不正を行わない．
（前提条件 3）　操作者はセキュリティデバイスを他人に渡さない．

以上の条件の下で，Dig-Force は，目的とする機能を実現できることを確認している．

3. Dig-Force の改良の必要性と対応策

3.1 Dig-Force の問題

上記の前提条件のうち（前提条件 1）は現実に成立しない場合が少なくない．プログラミング能力のあるユーザなら，Dig-Force に影響を及ぼす不正なプログラムを追加することが可能である．さらに能力のあるユーザならリバースエンジニアリングを行うことによるプログラムの改竄もありえないことではない．

この前提条件が成立しないならば，Dig-Force のプログラムを改竄するなどしてログデータの改竄を検知できなくするなどの不正が可能となる．

以上の点から，この問題点を解決する対応策が必要となった．

3.2 Dig-Force の問題に対する要件

ここで，3.1 節で述べた Dig-Force の問題点に対処するための要件を整理する．

まず Dig-Force の問題点を解決する提案方式に求められる要件は大きく下記のように整理できる．

（要件 A）　証拠性のあるデータを残したい，またそのデータに改竄が加えられても検知したい．

（要件 B）　確実にログデータを収集したい．
（要件 C）　不正なプログラムが稼動していない PC 環境で作業をしたい．

まず，（要件 A）に関しては Dig-Force の機能であるヒステリシス署名をログデータに施し，署名履歴を耐タンパ領域に安全に保管することで実現することができる．（要件 B）に関してもロガープログラムがセキュリティデバイス装着時のみ動作可能にすることによって実現することができる．

しかし，（要件 C）については 3.1 節で述べたように実現することができていない．また，Dig-Force の各プログラムが起動した後に不正に終了された場合，（要件 A）（要件 B）を実現することができない．

そこで（要件 C）を実現するためには，改竄が困難な，より信頼性のあるプログラムが監視を行い，正しいプログラムのみが起動できる環境を構築する必要があると考えた．また不正に Dig-Force の各プログラムが終了した場合に備えて，プロセスの監視を行い，不正に終了した場合は再度起動するといった機能を持たせる必要があると考える．

ここでいう信頼性のあるプログラムというのは，技術的に改竄が困難で，他のプログラムよりも先に起動している必要がある．

3.3　不正プログラムに対する起動制御方式の提案

正しいプログラムのみが起動できる環境を構築するために，信頼性のあるプログラムが，他のプログラムの起動制御を行う．この起動制御にはホワイトリスト方式を用いる．

ホワイトリストは起動を許可するプログラムのハッシュ値一覧を持つものであり，信頼性のあるプログラムはこのホワイトリストを参照し，ホワイトリストに登録されているプログラムのみ起動を許可する．

本稿では，このホワイトリストの不正な改竄を防ぐために，ホワイトリストに登録される各プログラムに電子署名を施すこととする．

3.4　信頼性のあるプログラムの実現方式の検討

ここでは，Microsoft Windows XP 環境下で検討する．Windows XP では，ハードウェアからアプリケーションまで複数の階層が存在している（**図 2**）．

3.2 節で述べた信頼性のあるプログラムの実現方式として以下の 3 つの方法が考えられる．

(1)　OS の機能を利用する：アプリケーションの管理を行っている OS がプログラムの起動制御を行う．
(2)　フィルタドライバを用いる：OS のカーネルモードに位置するフィルタドライバにプログラムの監視機能を追加する．

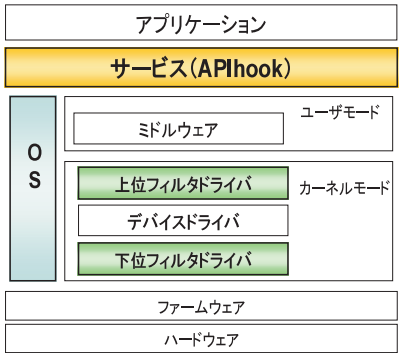

図 2　PC のソフトウェア階層
Fig. 2　Hierarchy of software.

(3)　APIHook を用いる：Windows の API をフックし，処理の変更を行う APIHook プログラムを OS 起動時に自動的に起動するサービスとして起動し，アプリケーションプログラムの起動命令（API）を監視する．

OS が不正なプログラムを監視する機能を持つことが望ましいと考えるが，その機能を現在の OS は持っていない．そこで，今回は (2)，(3) の検討を行い，著者らが現在実現することができる (3) の APIHook プログラムを研究対象として採用することにした．

3.5　APIHook プログラムの改竄の危険性と TPM を用いた対策案

3.2 節で説明した信頼性のあるプログラムが他のプログラムの起動制御を行うことで，不正なプログラムが起動しない環境の実現が可能になると考えた．

しかし今回，研究対象とした APIHook プログラム自体が改竄されるという危険性も存在する．なぜなら，APIHook プログラムはアプリケーション層で稼動するサービスプログラムである．よって HDD を外付け HDD として別の PC に接続すれば，その PC では HDD から読み込んでシステムとして起動することができないため，不正に APIHook プログラムを削除，または改竄することが可能となる．

この APIHook プログラムが改竄されるという攻撃に対処するために，TPM を用いることとした．

TPM はソフトウェアからの不正が困難な耐タンパ領域をハードウェア上に持つことができる IC チップである．PC ではマザーボード上に搭載されている．

TCG（Trusted Computing Group）が技術仕様の策定を行っており，RSA 鍵の生成・保

管，RSA 演算，ハッシュ演算，プラットフォーム状態情報（ソフトウェアの計測値）の保持，不揮発性メモリおよび揮発性メモリなどの機能が搭載されている[10]．

TPM を用いた主な理由は以下のとおりである．

TPM は，1 つの PC 固有のものとして搭載されている．すなわち TPM 内の暗号鍵はその PC 固有の暗号鍵であるといえる．PC 固有の暗号鍵で暗号化した HDD 内のデータは，別の PC で復号することはできない．

この TPM の鍵を用いた HDD 暗号化プログラムを利用することで HDD を別の PC に移行して，不正に監視プログラムの改竄，または削除する攻撃を防止することが可能となる．

また，下記にあげる点も TPM を用いた動機である．

- 特別なデバイスを用いる必要がない．
- USB デバイスなどに見られる，不注意でデバイスを引き抜くことにより，演算処理中のデータが消失するという問題を避けることができる．

3.2 節，3.4 節で述べた方式を用いることで，本提案方式は APIHook の機能を持つ信頼性のあるプログラムによって，正しいプログラムのみが稼働しているように制御することが可能になると考える．

4. Dig-Force の問題点を解決する提案方式

従来方式の問題点であった不正なプログラムの起動を防ぐために，3 章で検討を行った対策案を取り入れた提案方式 “Dig-Force2” の構成を記述する．

また，提案を行ってきた Dig-Force はセキュリティデバイスに eToken[11] を用いて評価を行っていたが，今回は 3.5 節で述べたように，セキュリティデバイスを実現するにあたって，TPM を用いる方式を採用した．

4.1 Dig-Force2 の構成

この TPM を用いた Dig-Force2 の構成要素として以下の 6 点をあげる．**図 3** は Dig-Force2 の構成要素間の流れを表した図である．

(1) ロガープログラム

- 操作者の証拠となる操作情報をデジタルデータ化し，オペレーションデータとしてログストレージプログラムに転送する．

(2) ログストレージプログラム

- ロガープログラムから転送されたオペレーションデータに時刻情報などを付加し，フォーマット化を行う．その後，フォーマット化したデジタルデータに対して，TPM と連携しヒステリシス署名を施したうえで，ログファイルに書き込み，最終署名履歴となる連鎖用データのみ TPM に保存する．
- ログストレージプログラムに TPM にアクセスするパスワードを保持させ，ログストレージプログラムのみ TPM にアクセス可能とする．

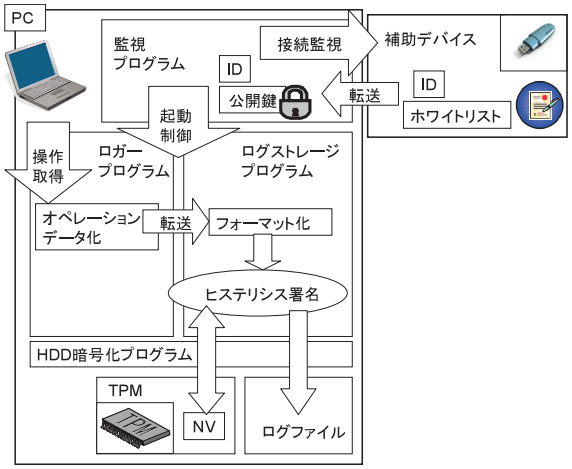

図 3　Dig-Force2 の構成
Fig. 3　Composition of Dig-Force2.

以後，ロガープログラム，およびログストレージプログラムを DF システムプログラムとする．

(3) 補助デバイス

- 監視用プログラムと連携し，補助デバイスが PC に装着されているときのみ PC を動作可能とする．
- 補助デバイス内には，下記のデータを格納する．
  (a)　管理者が起動を許可した実行形式でのプログラムのハッシュ値リスト（以下，ホワイトリストとする）
  (b)　個人特定用の ID
  (c)　(a) に登録した各プログラム，そして (b) に対して管理者の秘密鍵によって署名した電子署名

(4) TPM
- 耐タンパ領域を持ち，署名鍵，そして連鎖用データを保護する．
- ヒステリシス署名の署名演算を行う．

(5) 監視プログラム

(a)　APIHook 機能を持ち，OS 起動時に自動的に起動するサービスとして動作する．

(b)　補助デバイス内の電子署名を検証する．

(c)　APIHook により，ユーザないしプログラムが他のプログラムを起動する前に，実行ファイルのハッシュ値を計算し，ホワイトリストと比較を行う．一致していた場合のみプログラムを起動する．

(d)　補助デバイス装着時で，補助デバイス内の ID の検証に成功したときのみ操作者に PC の操作を可能にさせる．

(e)　APIHook により，DF システムプログラムのプロセス監視を行い，不正に終了された場合は，再度プログラムを起動する処理を行う．

(c) の機能により，プログラムが起動する前に実行ファイルの正当性をチェックすることができる．よって DF システムプログラムの改竄を検知するだけでなく，許可しないプログラムの起動を制限することができる．また，(d) により，正当な補助デバイスを装着しているときのみしか PC を操作させないことにより，個人の特定を行うことができるようになる．

(6) HDD 暗号化プログラム
- 提案システムの導入時に，TPM 固有の鍵を用いて HDD 全体の暗号化を行う．
- 暗号化した HDD 内のデータの暗号化，復号は HDD へのアクセスのたびに自動的に行う．

この HDD 暗号化プログラムはフィルタドライバを用いる．フィルタドライバを用いると復号処理はドライバ層で行われるため，操作者は HDD へアクセスする際，暗号化を意識する必要はない．よって通常の HDD と同様に暗号化を意識することなく利用が可能となる．

4.2　Dig-Force2 の前提条件

提案方式では，2.5 節で示した Dig-Force の前提条件より，「(前提条件 1) PC 内の各プログラムは不正に変更されることはない」を外し，たとえ各プログラムを不正に変更したとしてもそれを検知できるようにする．ただし，その前提として，(1) BIOS，OS は正しく稼動していると仮定し，(2) カーネルモードで動作する RootKit は動いていないとする前提条件をおく．これは，「PC 内の各プログラムは不正に変更されることはない」という前提に比べ大幅に現実的なものとなっている．

以上より，Dig-Force2 の前提条件は以下のとおりである．

(前提条件 A) BIOS，OS は正しく稼動していると仮定する．

(前提条件 B) カーネルモードで動作する RootKit は動いていないとする．

(前提条件 C) 操作者は不正を行う可能性はあるが，管理者，検証者は不正を行わない．

(前提条件 D) 操作者は補助デバイスを他人に渡さない．

これらの前提条件によりこの提案システムが有効な範囲は BIOS，OS が正しく稼動している環境とする．

4.3　Dig-Force2 の処理フロー

処理フローを，Dig-Force と同様に導入フェーズ，運用フェーズ，検証フェーズに分けて順に説明を行う．システムに関わる登場人物も同様に管理者，PC の操作者，検証者の 3 者が存在する．

管理者と検証者には OS の管理者権限を持たせ，操作者は管理者権限を持たせないという制限のもとで作業を行わせる．管理者権限を持たせないことでサービスとして稼動しているプログラムの停止やプログラムのインストールならびにアンインストールが不可能となる．

A) 導入フェーズ

管理者は，以下の処理を行い操作者に TPM 搭載 PC，補助デバイスを渡す．

(1)　TPM 内部にルートとなる鍵であり，他の TPM 鍵の保護に使われる Storage Root Key（以下，SRK）を生成する．

(2)　ヒステリシス署名に用いる秘密鍵 S，署名検証に用いる公開鍵 P を TPM 内部で生成し，SRK で暗号化し HDD に SRK(S)，SRK(P) として保存する．

(3)　連鎖用データの初期値 R1 に任意の値を指定し，TPM 内部の不揮発性メモリ（以下，NV）に保存する．

(4)　検証時に必要な初期値を，検証者に送付する．

(5)　補助デバイスにホワイトリスト，個人特定用の ID ならびにホワイトリスト，ID の電子署名を格納する．

(6)　監視プログラムにホワイトリスト検証用の公開鍵，個人特定用の ID をセットし，TPM にアクセスするパスワードを設定した DF システムプログラムとともに PC にインストールする．その後，管理者権限で監視プログラムをサービスとして登録する．

(7)　HDD 暗号化プログラムで HDD 全体の暗号化を行う．

(8)　操作者に初期設定が終了した PC と補助デバイスを渡す．

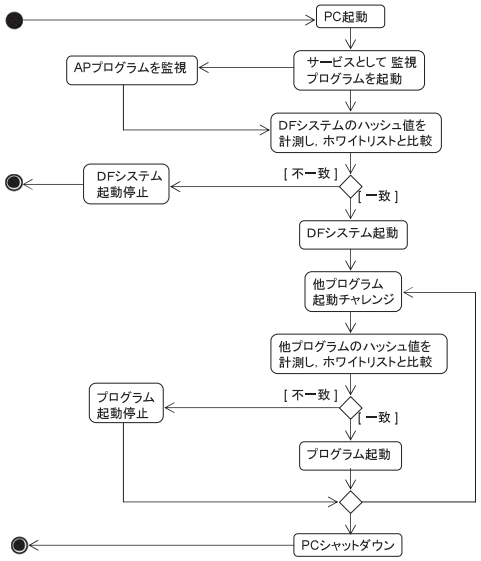

図 4　監視フロー
Fig. 4　Flow of monitoring.

B) 運用フェーズ

運用フェーズでは，監視プログラムが起動している監視下で補助デバイスを PC に装着し，操作者の操作情報をログとして取得する．取得した操作情報に TPM と通信を行い，ヒステリシス署名を施していく．**図 4** は監視プログラムのフローを表す．

(1)　HDD 暗号化プログラムは起動時に TPM の暗号鍵を用いて暗号化された HDD 内の OS を復号し，起動を行う．

(2)　HDD 暗号化プログラムによる復号処理の後，監視プログラムは OS が起動した直後に，サービスとして起動される．監視プログラムは管理者権限で起動しているため，管理者権限を持たない操作者は不正にプログラムを停止することができない．

(3)　監視プログラムは補助デバイスの装着を監視しており，装着されていない場合は PC をロックする．

(4)　操作者は PC を操作するために，補助デバイスを PC に装着する．

(5)　監視プログラムは補助デバイスから個人特定用の ID とその ID の電子署名を読み込み，検証を行う．検証が成功した場合のみ PC が動作可能となる．

(6)　監視プログラムは装着された補助デバイスからホワイトリスト，およびホワイトリストの電子署名を読み込む．

(7)　監視プログラムは，まずホワイトリストに登録されている DF システムプログラムの電子署名の検証を行う．検証が成功した場合，DF システムプログラムのハッシュ値を計算し，ホワイトリストと比較を行う．一致した場合は，DF システムプログラムの起動を行い，ログデータにヒステリシス署名を施していく（**図 5**）．処理フローは従来の方式と TPM の鍵（SRK(S)）を用いて TPM 内部で署名演算を行い，TPM の NV に連鎖用データを格納するフロー以外は同一である．

(8)　監視プログラムはヒステリシス署名の署名データを書き換えるなどの攻撃を行う不正なプログラムが起動しないように他のプログラム（実行ファイル）の監視も行っている．そして，操作者が他のプログラムを起動しようとした際に，監視プログラムが起動の API をフックする．その後，(7) と同様にホワイトリストの電子署名の検証を行ってからプログラムのハッシュ値を計算し，ホワイトリストと比較を行い，一致した場合のみ起動を許可する．

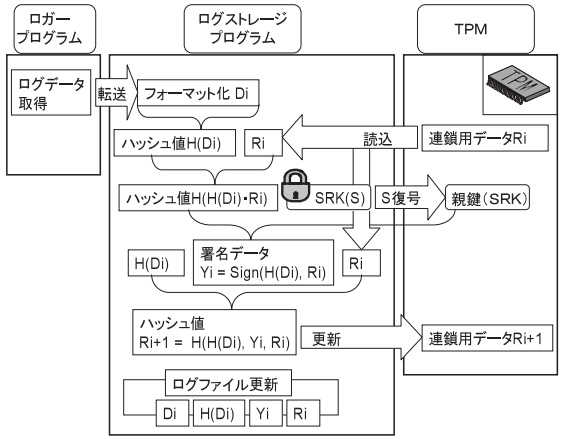

図 5　TPM を用いたヒステリシス署名
Fig. 5　Hysteresis signatures using the TPM.

C) 検証フェーズ

検証者は，操作者から TPM 搭載 PC，補助デバイスを受け取りログファイルの検証を行う．

検証フローは従来の方式と TPM の鍵（SRK(P)）を用いて TPM 内部で署名データの検証を行う，および検証時に TPM の NV に格納されている連鎖用データを用いること以外は同一である．

この提案方式を用いることで，正しいプログラムのみが PC 上で稼働していることの確認が可能になると考えられる．

5. 提案方式の安全性の検討

下記に Dig-Force2 を運用していくうえで，考えられる攻撃を示し，その攻撃方法に対する Dig-Force2 の安全性の検討を行う．

まず脅威と対策を明確にするため，この提案方式での要件とその実現方法を示す．その後，この実現方法に対する攻撃方法，そして攻撃方法に対する対策案の順に記述していく．

まずこの提案方式の要件，その実現方法は下記のように整理できる．要件は 3.2 節で述べた要件と同一である．

（要件 A）証拠性のあるデータを残したい，またそのデータに改竄が加えられても検知したい．

（実現方法 A）ヒステリシス署名を施し，署名履歴を安全に保管．

（要件 B）確実にログデータを収集したい．

（実現方法 B）監視プログラムが DF システムを起動し，そしてプロセスを監視している．

（要件 C）不正なプログラムが稼動していない PC 環境で作業をしたい．

（実現方法 C）監視プログラムを稼動させ他プログラムの起動制御を行う．

ここから，この実現方法に対する攻撃，そして攻撃に対する対策案を示していく．

まず（実現方法 A）に対する攻撃は，次のような攻撃方法があると考えられる．

(1)　DF システムプログラムの停止

(2)　改竄された DF システムプログラムや，DF システムの機能を妨害する不正なプログラムが起動

(3)　不正な署名データの作成

攻撃 (1)，(2) が起こってしまえば，ログと署名を取得することができなくなり，操作情報に関する証拠となるデータを残すことができなくなる．また攻撃 (3) についても攻撃が起こってしまえば，不正な操作を正しい操作として偽ることが可能となり作業の正当性を示すことができなくなってしまう．よってこれら 3 つの攻撃が起こってしまえばシステムとしての要件を満たすことができなくなる．

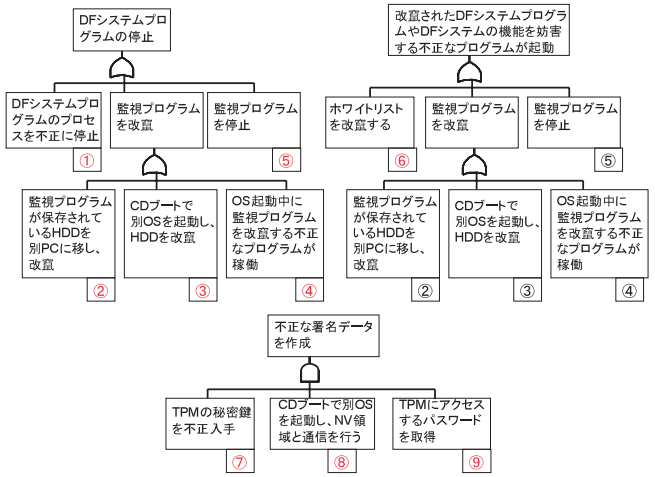

図 6　アタックツリー
Fig. 6　Attack trees.

この 3 つの攻撃方法に関するアタックツリーを用いて分析を行い，それぞれの攻撃の難易度を検討した（**図 6**）[12]．

まず分析の結果，下記に示す 8 つの攻撃方法が根源の攻撃になっているという結果が得られた．

攻撃 ①：DF システムプログラムのプロセスを不正に停止する．

攻撃 ②：監視プログラムが保存されている HDD を別 PC に移し，監視プログラムを改竄する．

攻撃 ③：CD ブートで別 OS を起動し，監視プログラムが保存されている HDD を改竄する．

攻撃 ④：OS 起動中に監視プログラムを改竄する不正なプログラムが稼働する．

攻撃 ⑤：監視プログラムを停止させる．

攻撃 ⑥：ホワイトリストを改竄する．

攻撃 ⑦：TPM の秘密鍵を不正入手する．

攻撃 ⑧：CD ブートで別 OS を起動し，NV 領域と通信を行う．

攻撃 ⑨：TPM にアクセスするパスワードを取得する．

上記の攻撃方法を図に対応させると**図 7** のとおりである．

次に作成したアタックツリーを用いて攻撃の難易度を検討した．(1) の DF システムプロ

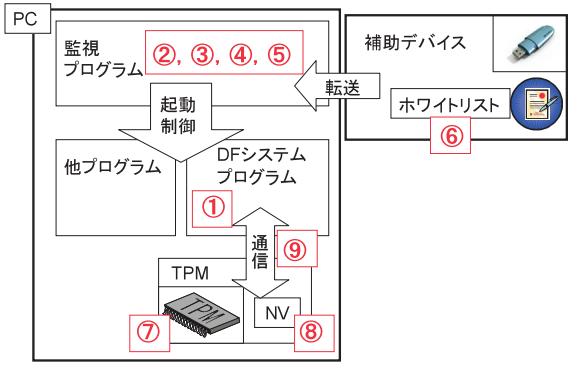

図 7　Dig-Force2 に対する脅威
Fig. 7　Threat against the Dig-Force2.

グラムが停止されるという攻撃については ①，②，③，④，⑤ のうちの 1 つでも攻撃が起こってしまえば，防ぐことができないという結果が得られた．また (2) 改竄された DF システムプログラムや，DF システムの機能を妨害する不正なプログラムが起動という攻撃についても同様に，②，③，④，⑤，⑥ のどれか 1 つでも起こってしまえば安全性を保つことができなくなる．よって Dig-Force2 はこれらの (1)，(2) の攻撃が発生する可能性が高いといえ，それぞれ対策をしておく必要があるといえる．反対に (3) の不正な署名データの作成という脅威については ⑦，⑧，⑨ の 3 つの攻撃が可能にならなければ起こらない脅威といえるので，上記の 2 つの大きな攻撃方法よりも攻撃の実現は困難であり，難易度が高いといえる．

上記の検討から（要件 A）を実現するためには，根源の攻撃である ①，②，③，④，⑤，⑥ の攻撃を防ぎつつ，さらに ⑦，⑧，⑨ のどれか 1 つの攻撃を防ぐ必要があると考える．そして，これらの根源の攻撃にはそれぞれ以下に示すように対処する．

対策 ①：監視プログラムによって DF システムプログラムのプロセス状態の監視を行い，停止された場合は，自動的に再度起動する．

対策 ②：TPM 固有の暗号鍵で HDD 内の監視プログラムに対し暗号化を行う．

対策 ③：BIOS の設定でローカル HDD 以外の起動を禁止する．操作者は管理者権限を持たせないので設定の変更は不可能となる．

対策 ④：監視プログラムがサービスとして起動しているため，その監視プログラムを改竄する不正なプログラムの起動は不可能となる．

対策 ⑤：管理者権限（Administrator）でサービスを稼動させる．管理者より下の権限（User）ではプログラムを停止することができない（OS のセキュリティ機能）．

対策 ⑥：ホワイトリストの各プログラムに管理者の電子署名を施し，監視プログラムが署名検証を行う．

対策 ⑦：TPM の署名鍵は TPM の親鍵で暗号化されており，署名鍵を入手するためには耐タンパ領域で保護されている TPM の親鍵を入手しなければならないため困難である．

対策 ⑧：BIOS の設定でローカル HDD 以外の起動を禁止する．操作者は管理者権限を持たせないので設定の変更は不可能となる．

対策 ⑨：TPM にアクセスする DF システムプログラムにパスワードを保持させる．DF システムプログラムのみ TPM にアクセス可能とする ．

（実現方法 B）（実現方法 C）についても同様に攻撃方法，対策案を記述していく．（実現方法 B）に対する攻撃は，次のような攻撃方法があると考えられる．

(1)　DF システムプログラムの停止

(2)　改竄された DF システムプログラムや，DF システムの機能を妨害する不正なプログラムが起動

（実現方法 A）と同様にアタックツリー分析を行い，①，②，③，④，⑤，⑥ の攻撃をすべて防ぐことで安全性を保つことができるという結論が得られた．

また，（実現方法 C）に対する攻撃については，次のような攻撃方法があると考えられる．

(1)　改竄された DF システムプログラムや，DF システムの機能を妨害する不正なプログラムが起動

この（実現方法 C）に対する攻撃についてもアタックツリー分析を行い，②，③，④，⑤，⑥ の攻撃をすべて防ぐことで安全性を保つことができるという結論が得られた．

6. 提案方式の機能実験

提案方式の実用性を検証するために，監視プログラムに関する機能実験と，TPM に関する機能実験の 2 つを行った．

6.1 監視プログラムに関する機能実験

監視プログラムに用いている APIHook でプログラムの起動を制御する機能についての実験を行った．APIHook は，オリジナルの API を，新たに作成した別の API に置き換える技術である．この技術を応用し，オリジナル API の処理を変更することで，プログラムの動作を変更させることができる[13]．

プログラム起動時に，どの API が呼ばれているかの調査を，呼ばれた API のログを取得する APIMonitor[14] を用いて行った．実際に APIMonitor を起動させた状態で他のプログラムを起動したところ，数十種類の API が APIMonitor によって出力された．この出力された API を 1 つ 1 つフックし，プログラムを起動させないために，オリジナルの API の呼び出しを停止する処理を記述し，挙動を調べる実験を行った．その結果，他のプロセスに影響を及ぼさず，特定のプログラムの起動を停止することができたのは ntdll.dll に定義されている API である RtlCreateProcessParameters のみであった．NtCreateProcess も同様にプログラム起動に関する API であるが，この API を用いても Windows XP 環境では，プログラムの起動を停止することができなかった．また特定のプログラムの起動を防止するためにはプログラムのパスが必要であり，この値に関しても NtCreateProcess では取得できなかった．

結果，Windows XP 環境で，起動しようとしているプログラムのパスを取得でき，プログラムの起動を制御できることから RtlCreateProcessParameters を監視プログラムの起動制御 API として採用することとした．

この API をフックし，起動制御を行う処理フローを以下に示す（**図 8**）．

この処理フローのプロトタイププログラムを作成し，起動制御実験を行った．実験内容はプロトタイププログラムが稼動している環境で，アプリケーションの起動制御が行えるのかを調べるというものである．この起動実験には OS に標準で搭載されている notepad.exe から自作の Windows アプリケーションまで様々なプログラムを用いた．

実験の結果，ホワイトリストに登録されているプログラムは起動を許可し，登録されていないものは起動を停止することができるという結果を得ることができた．よって許可したプログラム以外は起動ができない状況を実現することが可能となった．なお開発にあたっては，kenji 氏のウェブサイト[15] を参考にさせていただいた．

Windows アプリケーションでない DOS プログラムの動作は APIHook により監視することができないが，DOS プログラムの起動は監視することができる．よって今回は DOS プログラムの起動を許可しないことで対処した．しかし，利便性の点で問題があるため，DOS プログラムの起動制御は，今後の課題としたい．

6.2　TPM に関する機能実験

TPM を用いる方式の有用性を確認するために，下記の 2 点で実験を行った．

(1)　TPM の署名演算パフォーマンス

(2)　TPM の NV 利用について

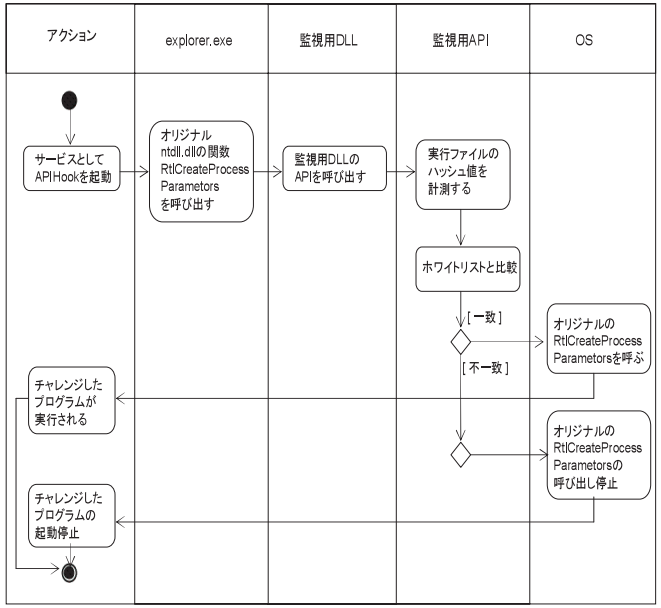

図 8　起動制御フロー
Fig. 8　Flow of boot control.

まず (1) についてはヒステリシス署名に必要な署名演算処理，署名検証処理を実装し，要した時間を計測した．評価は**表 1** で示す環境で行った．

処理時間の計測を 10 回繰り返して行い，その平均値を計測結果とした．計測結果は，署名演算時間が 0.076 秒であり，署名検証処理時間が 0.029 秒であった．本稿で提案するシステムでは，従来方式と同様に 1 秒ごとに TPM にデータを署名演算のために転送することを考えている．よってこの結果は実用的な値であると判断できる．

(2) の TPM の NV 利用については，ヒステリシス署名の連鎖用データを格納するために必要な機能である．上記の表 1 の環境で，NV を利用できるか実験を行ったところ，TPM のユーザ用に 20 BYTE 確保されていることが確認できた．ヒステリシス署名の連鎖用データは 20 BYTE であり，NV は格納する領域として十分である．

表 1　署名演算の測定環境
Table 1　Measuring environment for signature operations.

暗号アルゴリズム	RSA
鍵長	1024bit
CPU	Genuine Intel(R) CPU U1300 @1.06GHz
RAM	1024MB
OS	Windows XP Professional
開発言語	C++
開発ツール	Microsoft Visual Studio.NET 2003 Infineon TPM Integration SDK

7. 提案システムの性能実験・評価

提案システムの実用性を検証するために，実装を行った監視プログラム，DF システムプログラムに関する実験・評価を行った．実験・評価は以下の項目について行った．

7.1 性能実験内容

(1) プログラム正当性チェック時間
プログラム起動に関する RtlCreateProcessParameters が呼ばれてから，ホワイトリストのハッシュ値と照合し，電子署名の検証が終了するまでの時間

(2) ホワイトリスト作成時間
C:¥Program Files と C:¥Windows¥System32 に格納されているプログラムについてホワイトリストを作成

(3) CPU 使用率，メモリ使用量

なお実験・評価は表 1 の環境で行った．

7.2 実 験 評 価

まず (1) は，notepad や Microsoft Office などをはじめとして数十種類のプログラムを対象に実験を行った．各プログラムの処理時間は 10 回繰り返して計測を行い，その平均値とした．実験の結果，チェック時間は約 0.01 秒から 0.9 秒ほどで実用性に耐えうる時間であるという結果が得られた．

次に (2) の実験についてはプログラムファイル数 2,739 個に対して，計測時間 535 秒という結果が得られた．平均では 1 ファイルにつき 0.195 秒である．ホワイトリストの作成は初期設定のときのみ行う作業であるので，上記の結果は実用性があると考える．

最後に (3) の実験については提案方式で述べたプログラムを実際に稼動させパフォーマンスを計測した．

計測結果は監視プログラムの CPU 使用率が 0%でメモリ使用量が 3660 KB，また DF システムプログラムについては CPU 使用率が 0〜11%でメモリ使用量が 4048〜4112 KB という結果が得られた．また同時に稼動させても他の作業に対する支障は見られなかった．

8. 今後の課題

APIHook の機能を持つ監視プログラムと補助デバイスを用いることで高い信頼性を確保できたと考えられる．しかし，さらに信頼性を上げるためには，解決しなければならない課題が存在する．その課題を下記に示す．

8.1 ホワイトリスト関連

(1) 運用に必要な実行ファイル収集

提案方式では，ホワイトリスト方式を採用しているが，ホワイトリスト方式では，通常の PC 作業に必要な正しいプログラムを過不足なく登録しておく必要がある．このホワイトリストに登録していなければならない正しいプログラムに抜けがあっては，通常の PC 作業に支障をきたすといった運用上の問題が生じる．そのため，運用に必要だと考えられる実行ファイルを収集する必要がある．

(2) 監視プログラムの計測対象

今回は，ホワイトリストの内容を実行形式でのプログラムのハッシュ値のみとしていた．しかし実際には，プログラムが起動する際に，必要に応じてロードする DLL や追加機能を提供する plug-in なども改竄されてしまえば，正しいプログラムであったとしても，不正な動作をしてしまう危険性がある．そのため，ホワイトリストの内容をプログラムのハッシュ値だけでなく，DLL や plug-in を含めて，正当性のチェックを行う方式の検討が必要である．

現在，米国などではホワイトリストを提供するサービスが行われており[16]，上記の対応の可能性は高いと考えているが，具体的に確認していく必要がある．

8.2 HDD 暗号化プログラムの実現方法

3.5 節で述べた HDD 暗号化プログラムについてであるが，監視プログラムの安全性を保持するためには必ず実現しなければならない機能である．この HDD 暗号化プログラムの実現方法として Windows Vista には BitLocker というフィルタドライバを用いたボリュームの暗号化機能があるが，今回は Windows XP 環境で検討を行っているため，利用することができない[17]．そのため，セクタレベルでハードディスク全体の暗号化を行い，カーネルレベルで暗号や復号処理を実現できる McAfee Endpoint Encryption を用いて実現する方向で検討を行う[18]．

9. おわりに

本稿では，監視プログラムと TPM を用いて，不正なプログラムの起動制御機能を持つ DF システムの提案を行った．また，提案システムの性能実験・評価を行うことで方式の有効性を示すことができた．今後は，8 章で述べた提案方式の実現に必要である課題について検討を行っていきたい．

謝辞　TCG の技術に関して親切にご教授いただいた日本アイ・ビー・エム株式会社，ならびにインフィニオンテクノロジーズジャパン株式会社の方々にこの場をお借りして深い感謝の意を表する．

参考文献

1) 佐々木良一，芦野佑樹，増渕孝延：デジタル・フォレンジックの体系化の試みと必要技術の提案，JSSM 学会誌，Vol.20, No.2, pp.49–61 (2006).
2) AccessData — Forensic Toolkit. http://www.accessdata.com/forensictoolkit.html
3) Schneier, B. and Kelsey, J.: Cryptographic Support for Secure Logs on Untrusted Machine, *Proc. 7th USENIX Security Symposium*, pp.53–62, USENIX Press (Jan. 1998).
4) 小畑直裕，川口信隆，東　雄介，重野　寛，岡田謙一：フォレンジックコンピューティングのための効率的なログ署名手法の提案，研究報告「マルチメディア通信と分散処理」，情報処理学会第 123 回 DPS 研究会，No.58, pp.31–36 (2005).
5) 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).
6) McCune, J.M., Parno, B., Perrig, A., Reiter, M.K. and Isozaki, H.: Flicker: An execution infrastructure for TCB minimization, *Proc. 3rd ACM SIGOPS/EuroSys European Conference*, pp.315–328 (Mar./Apr. 2008).
7) Munetoh, S., Nakamura, M., Yoshihama, S. and Kudo, M.: Integrity Management Infrastructure for Trusted Computing, *IEICE Trans. Information and Systems*, pp.1242–1251 (2008).
8) Reid, J.F. and Caelli, W.J.: DRM, trusted computing and operating system architecture, *3rd Australasian Information Security Workshop* (*AISW 2005*), pp.127–136 (2005).
9) 岩村　充，宮崎邦彦，松本　勉，佐々木良一，松木　武：電子署名におけるアリバイ証明問題と経時証明問題—ヒステリシス署名とデジタル古文書の概念，コンピュータサイエンス誌 bit，Vol.32, No.11, 共立出版 (2000).
10) Trusted Computing Group. https://www.trustedcomputinggroup.org/home/
11) Aladdin Knowledge Systems, Ltd. http://www.aladdin.com/
12) Schneier.com: Attack Trees. http://www.schneier.com/paper-attacktrees-ddj-ft.html
13) Richter, J.（著），長尾隆弘（翻訳）：Advanced Windows 改訂版第 4 版，株式会社アスキー (2001).
14) rohitab.com: API Monitor: Spy on API Calls. http://www.rohitab.com/apimonitor/
15) KENJI'S HOMEPAGE. http://ruffnex.oc.to/kenji/
16) SignaCert, Inc. http://japan.signacert.com/
17) Windows Vista: Features Explained: Windows BitLocker Drive Encryption. http://www.microsoft.com/Windows/products/Windowsvista/features/details/bitlocker.mspx
18) McAfee Co., Ltd.: McAfee Total Protection for Data. http://www.mcafee.com/japan/products/total_protection_for_data.asp

(平成 21 年 11 月 1 日受付)
(平成 22 年 6 月 3 日採録)

藤田　圭祐

平成 19 年東京電機大学工学部情報メディア学科卒業．平成 21 年東京電機大学大学院工学研究科情報メディア学専攻修了．在学中，デジタルフォレンジックに関する研究を実施．同年日本電気株式会社入社．セキュリティに関するソフトウェアの製品開発に従事．

芦野　佑樹（正会員）

平成 14 年北海道東海大学工学部電子情報工学科卒業．卒業後，会社員としてシステム開発に従事しながら，平成 16 年東京電機大学大学院工学研究科情報メディア学専攻修士課程に入学．平成 21 年 3 月東京電機大学大学院先端科学技術研究科情報通信メディア工学専攻博士後期課程修了．博士（工学）．平成 21 年 4 月より，日本電気株式会社サービスプラットフォーム研究所システムセキュリティTG にてコンピュータセキュリティに関する研究に従事．

上原哲太郎（正会員）

1990 年京都大学工学部情報工学科卒業．1995 年京都大学大学院博士課程研究指導認定退学．同年京都大学大学院工学研究科助手，1996 年和歌山大学システム工学部講師，2003 年京都大学大学院工学研究科附属情報センター助教授，2006 年京都大学学術情報メディアセンター助教授，2007 年同准教授．システムソフトウェア，システム管理，情報セキュリティ関係の研究に従事．京都大学博士（工学）．IEEE，電気学会，電子情報通信学会，日本ソフトウェア科学会，システム制御情報学会，情報ネットワーク法学会，CIEC 各会員．

佐々木良一（フェロー）

昭和 46 年 3 月東京大学卒業．同年 4 月日立製作所入社．システム開発研究所にてシステム高信頼化技術，セキュリティ技術，ネットワーク管理システム等の研究開発に従事．平成 13 年 4 月より東京電機大学工学部教授，平成 19 年 4 月より未来科学部教授．工学博士（東京大学）．平成 10 年電気学会著作賞，平成 14 年情報処理学会論文賞，平成 19 年総務大臣表彰，平成 20 年情報処理学会功績賞等を受賞．著書に，『IT リスクの考え方』岩波新書，2008 年等．情報処理学会コンピュータセキュリティ研究会顧問．日本セキュリティ・マネージメント学会会長，情報ネットワーク法学会理事長，日本ネットワークセキュリティ協会会長．