ソフトウェア完全性検証技術

アイ・ビー・エム ビジネスコンサルティング サービス（株）
丸山　宏
MARUYAMA@jp.ibm.com
日本アイ・ビー・エム（株）東京基礎研究所
宗藤 誠治
MUNETOH@jp.ibm.com
ソニー（株）CE ソフトウェアプラットフォーム部門
横手 靖彦
Yasuhiko.Yokote@jp.sony.com
ソニー（株）CE ソフトウェアプラットフォーム部門
篠崎 郁生
Ikuo.Shinozaki@jp.sony.com

ソフトウェア完全性検証の必要性

Malware とは「悪意のこもった」ソフトウェアのことで，使用者の意図しない動作を行うコンピュータウイルス，ワーム，スパイウェアなどの不正ソフトウェアがこれに該当する．近年こうした Malware は，システムの脆弱性を巧妙につき短期間で大規模に感染，結果として大きな被害をもたらす傾向がある．また，システムの脆弱性に対する運用および管理体制も本質的な問題として問われている．たとえば，2003 年 8 月に流行した Blaster ワームでは，セキュリティ・ホールの存在を実証するプログラム，エクスプロイト・コードを元に Malware が作成され，大きな被害がもたらされた．さらに，個人情報や機密情報の漏洩も大きな社会問題になっている．不正な侵入が行われ，かつその痕跡が巧妙に消去されている場合にはその捜査は難航を極めることになる．

こうした背景には，現在のコンピュータ，およびその上で動作するソフトウェアがその完全性（Integrity）を確実に保証，検証できないという根本的な問題がある．ここでいうソフトウェアの完全性（Integrity）とは，ユーザが使用するソフトウェア自体やその設定，ログなどに改竄がなくオリジナルの状態であることを指す．たとえば，パスワードの取扱いを見てみると，オンラインバンキング等のサービス提供者はユーザの使用しているクライアント端末で Key Logger が動作しているとしても確かめることができない．また，一般のユーザが，パスワード入力の画面が正規のソフトウェアによるものなのか偽者なのかを確かめることは難しい．

ソフトウェアには脆弱性が存在し，改竄が起こり得るという前提に立ち，そうしたソフトウェアの管理，運用を考えた時，使用者が期待する正規のソフトウェアがインストールされて運用されていること，つまり不正な改竄が行われていないことを確実に検証する基盤技術が必要である．本稿では，コードやデータの改竄から情報システムを保護する既存の技術について説明した後，現在 TCG（Trusted Computing Group）[8] で提案されているセキュリティ・チップを使った Trusted Platform 実現のための基盤技術，新しい完全性検証技術について解説する．

既存技術

現在広く使用されているソフトウェアの完全性検証技

術について簡単に説明する．たとえば，ネットワークを使ってソフトウェアのダウンロードを行う場合，そのソフトウェア・パッケージのハッシュ値の確認や，パッケージに添付されたコード署名の検証が広く使われている．さらに，導入済みのソフトウェアの完全性については，インストールされたソフトウェアを定期的に検査するツールを利用することも可能である．以下，各々の技術についてその概要を説明する．

改竄防止技術

ハッシュ値

コードの検証ではよく CRC などのエラー訂正コードやハッシュ関数が用いられる．しかしエラー訂正のための符号化では，その値を整合させる改竄は非常に容易であるため，改竄防止の目的には不適切である．これに対し，ハッシュ関数はある一定の暗号強度を持った一方向性関数であり，同じハッシュ値を持つデータ系列を作為的に生成することは困難である．ハッシュ関数のこの性質を利用し，数十バイトのハッシュ値によりプログラムの完全性を確認することが可能となる．たとえば，ダウンロードサイトによっては，プログラムとそのハッシュ値が提示されており，その値からダウンロードしたデータの完全性を確認することが可能である．ただし，そのダウンロードサイトが正規のものであるかどうかはこのハッシュ値自体では確認することはできない．こうした，出所の正当性を検証するためには後で述べるコード署名がこのハッシュと組み合わせて用いられる．

ファイル整合性チェックソフト，システムスキャナ

システムの不正な改竄をチェックするソフトウェアとしては，Tripwire，AIDE，Osiris などがよく知られている．これらのソフトウェアは，重要なプログラムや設定ファイルのハッシュ値を計算し，事前に用意してあるデータベースの参照値と比較することにより改竄を検出する．ただし，参照用のデータベースは書き込み禁止メディアに保存するなどして，改竄から保護する必要がある．また，ソフトウェアの完全性のみならず，システム構成上のセキュリティの不備を発見して修正するツールも開発されている．

現在では各種のウイルスチェックソフトが市販されて普及しているが，こうしたソフトは未知のウイルスや脆弱性の対応には限界がある．また，こうしたシステムの検査ソフトウェアがシステムに与える性能上の問題も問題になっている．

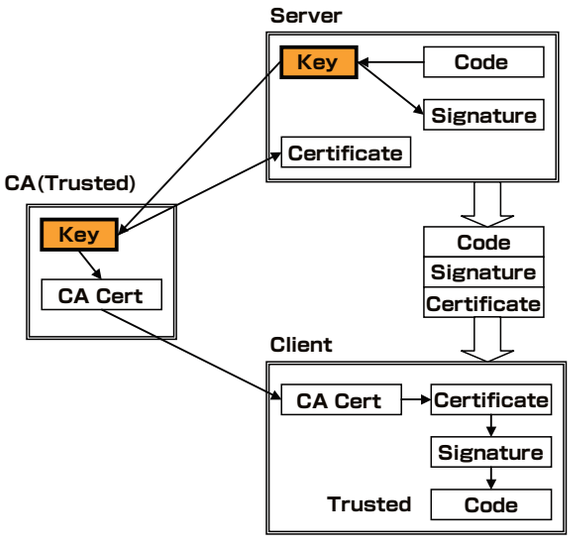

図 -1　コード署名の概略

出所を明らかにする技術―コード署名

ネットワークを通してソフトウェアを安全に配布する場合，その出所が検証可能である必要がある．そこで配布者が，配布するソフトウェアに電子署名を行うことによりその正当性をダウンロード先で検証可能にすることができるようになる．これがコード署名である．コード署名されたコードの身元を検証するには，署名に使用した暗号鍵が，双方で信頼することのできる第三者（一般に CA）によって署名されている必要がある．

図 -1 にサーバからクライアントにコードを配布する場合を例にコード署名の概略を示す．コード署名は RSA 等の公開鍵暗号を使って実現される．コードの配布者は公開鍵暗号の鍵ペアを生成し，プライベート鍵を安全に保持，そのパブリック鍵に対して認証機関（CA）からその鍵が配布者の物であることを示す証明書を入手する．コードを配布する際には，そのコードのハッシュ値を計算し，そのハッシュ値にプライベート鍵を用いて電子署名を行う．この電子署名と証明書を添付してコードの配布を行う．コードを受け取った者は，自らの持つ CA 証明書を元に送られてきたサーバの証明書が有効であることを確認し，さらに電子署名が正しいことを検証することでコードの身元を確認することができる．配布されたコードを信頼するための前提条件は，ここで使われている暗号が十分な強度を持ち，CA とサーバでプライベート鍵が安全に管理されていることである．ただ，注意が必要な点は，電子署名が正当な証明者によって行われ，かつ検証が正確になされていることを一般の使用者が確認するのは難しく，実際，証明者の「なりすまし」

や実装の脆弱性により不正なコードを誤って実行してしまう可能性があることである．

また，こうしたPKI（公開鍵インフラストラクチャ）に対応する法的基盤の整備として，国内では，電子署名および認証業務に関する法律が平成13年（2001年）4月1日に施行され，電子署名が手書きの署名や押印と同等に通用するようになっている．

Javaコード署名

Javaでは，クラスに対する電子署名が可能であり，さらにセキュリティ・ポリシーでこの電子署名を取り扱うことができるため，より安全な運用が可能である．Signed AppletはJDK1.1で導入されたコンセプトで，信頼できる第三者によってサインされたアプレットについてはサンドボックスの外でも実行可能にする．JARおよびマニフェスト仕様のコード署名ではより柔軟な署名が可能であったために，セキュリティ上の問題[3]が指摘された．Package-insertion Attackと呼ばれるこの問題を解決するためにSealed JarファイルがJDK1.2から導入されたが，まだクラスファイルの追加が可能であったため不十分であった．その後，1.2.2でSame-package Same-signerの原則が導入された．現在では，Jarファイルのパッケージに属する任意のクラスファイルを署名する場合，同じパッケージに属するすべてのクラスファイルが同じ署名者によって署名されていなければならない．

Microsoftプラットフォームのセキュリティ技術

AuthenticodeとはMicrosoft社のActiveX技術で使われているX509認証フォーマットと公開鍵暗号を用いたコード署名技術である．ブラウザでActiveXコントロールをダウンロードする時に，その出所の確認に使用されている．Authenticodeにはその実装に脆弱性[6]や，偽の証明書による「なりすまし」の危険[5]が発見され，コード署名技術の実装や運用上の問題が指摘されている．

Windowsで用いられているWFP（Windows File Protection）[10]は重要なWindowsシステムファイルが不正に変更や改竄が行われた場合，キャッシュに保存されているオリジナルのファイルで復旧を行う．WFPはファイルの復元の際にそのファイルのディジタル署名を確認することにより，万が一，キャッシュに置かれているファイルが悪意のあるファイルに入れ替えられていたとしても誤って復元されることはない．ただし，ファイルのバージョンによる脆弱性の確認は行わないため，その証明書が有効な場合は注意が必要である．またWFP機構は無効化が可能である．またWFPについてはルート証明書の取扱いの脆弱性が報告[1]された．

Linuxプラットフォームのセキュリティ技術

Linuxでソフトウェアの配布に使われている，RPMパッケージはGnuPGPで電子署名を付加することができる．また，その他の配布パッケージ形式についても，各ディストリビュータにより電子署名が採用されつつある．

既存技術の問題点

コード署名を中心に既存技術について解説を行ったが，これらの技術はクライアントに新しいソフトウェアを導入する際，その身元をPKIを用いて確認することにより悪意のある外部からの攻撃を防いでいるといえる．ここでの仮定はクライアントが正しく運用されていることである．逆にいえばこの仮定が成立しなければ，これらの既存技術によってソフトウェアの完全性を保証することは難しいといえる．実際，現在広く使われている各種プラットフォームには多くのセキュリティ・ホールが発見されており，今後も脆弱性が存在すると考えるのが妥当である．本質的な問題としてソフトウェアでソフトウェアを検証する従来の手法には限界があるともいえる．

過去にセキュリティ機能を強化したプラットフォームは存在したが，一般市場で受け入れられることはなかった．ユーザはクライアント・プラットフォームに対してさまざまな機能拡張を望んでおり，こうした需要とセキュリティ確保の両立が現在求められている．次章で解説する，TCGのTrusted Platformの基本的な考え方はプラットフォーム固有のセキュリティ・チップを用いてソフトウェアの完全性をハードウェア･レベルで保証し，クライアントが正しく運用されていることを安全確実に証明することである．

TCGによる完全性検証

TCG（Trusted Computing Group）[8]は2003年春に結成された業界標準団体であり，その前身であるTCPAの技術を引き継いでいる．TCGではTPM（Trusted Platform Module）と呼ばれるセキュリティ・チップをプラットフォームに搭載することにより，信頼できるプラットフォーム（Trusted Platform）を実現しようとしている．ここでいうTrusted Platformとはプラットフォームの完全性（Integrity），つまりプラットフォームが使用者の期待通りのものであること，が証明できるプラットフォームのことを示す．

従来のセキュリティ・チップは，秘密鍵の安全な保持と使用，つまり Identity 情報の物理的な保護が目的であった．このようなセキュリティ・チップをプラットフォーム固有のデバイスとして定義したのが TPM である．TPM では従来のセキュリティ・チップが持つ Identity 情報の保護のほかに，チップ内部に PCR（Platform Configuration Register）と呼ばれる 20 バイトのレジスタ（現在の仕様では 16 本）を持つことによってプラットフォームの完全性情報の物理的な保護を実現している．それに付随して次節以降で解説する Attestation や Seal などのプラットフォーム固有のセキュリティ機能を持つ．

ハードウェアによって保護された情報に対しても物理的な攻撃が可能であるが，チップ内部で保持している情報は最低限ソフトウェア・レベルからの攻撃からは守ることが可能である．つまり，Malware などによってチップで保持している情報が不正に改竄される可能性は非常に低いものになる．また，こうしたハードウェアは各種のセキュリティ評価によってある一定のセキュリティレベルが確保されており，このことを上手く利用することによって，システム全体，ソフトウェアも含んだプラットフォーム全体，としてのセキュリティレベルを高めることが可能となる．

PCR（Platform Configuration Register）

TPM が提供する機能はたくさんあるが，従来のセキュリティ・チップと違うユニークな機能が，チップ内に用意された PCR と呼ばれるレジスタ群で，次節で述べる Trusted bootstrap によりプラットフォーム上に起動したコードの痕跡を安全に記録するレジスタである．この機能がハードウェアにより保護されていることにより，TCG プラットフォームではソフトウェアの完全性をこのハードウェアの持つセキュリティ・レベルで保証することが可能になる．

PCR に対するアクセス方法は次の 4 通りが TPM の Version1.1b で定義されている．まず，1）電源投入時のリセット信号によってのみでしか PCR はゼロクリアされない．次に，2）書き込みは Extend と呼ばれる操作のみが提供されており，具体的には，新しい PCR の値は，現在の PCR の値と，Extend で入力された新しい値のハッシュになる．PCR の値を任意に設定するには，このハッシュの逆変換が必要であり，困難である．また Extend した値はすべてログに記録される．PCR の値をそのままプラットフォームの完全性を示す値として使用することも可能であるが，組合せにより管理しきれない場合が考えられる．そうした場合，Extend 操作のログにより，完全性を確認，PCR の値はログの検証に用いるといった使い方も可能である．残り 2 つの方法は 3）Read と 4）Quote であり，Read は PCR の値をそのまま読み出すだけで，この操作では PCR の値が直接 TPM から得られたものなのかは検証できない．これに対し Quote はチップ内部の RSA プライベート鍵を用い PCR の値に対し署名を行うことにより，PCR の値が TPM チップ由来のものであることを検証可能にする．これは Attestation と呼ばれ，あとで詳しく解説する．

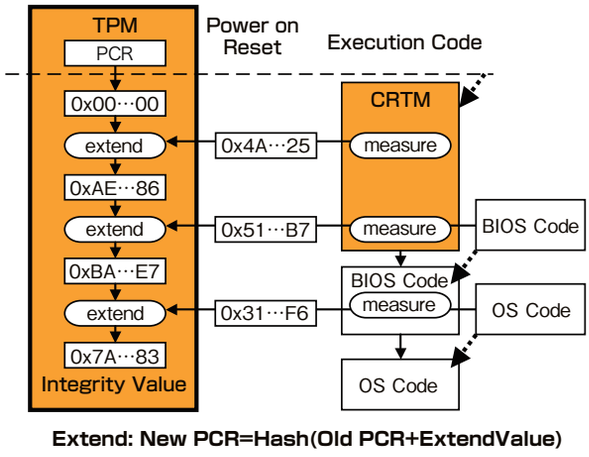

図 -2　Trusted bootstrap の概要

TPM は昨年秋に Version 1.2 の仕様 [9] が発表され，いくつかの機能が追加および拡張された．PCR に関しては Locality が導入され Trusted なソフトウェアによる PCR のリセットが可能になった．

Platform Root of Trust および Trusted bootstrap

TCG で定義する Platform の Root Of Trust とはソフトウェアから改竄が不可能な領域であり，TPM と CRTM（Core Root Of Trust Measurement）と呼ばれる書き込み禁止のブートコードで構成される．図 -2 に Trusted bootstrap の簡単な例を示す．TPM 自体ではプラットフォームの検査を行うことは不可能なため，電源投入時，最初に実行されるコードを CRTM とし，このコードを書き込み禁止とすることにより，実装の柔軟性と安全性を確保している．PCR は電源投入時にのみリセットされる．最初に起動する CRTM はまず自分自身の測定（コードイメージのハッシュ値の計算）を行い PCR に記録，次に，CRTM の次に起動するコード（図 -2 では BIOS コード）のハッシュ値の計算を行い，PCR に記録した後，この新しいコードを実行する．TPM チッ

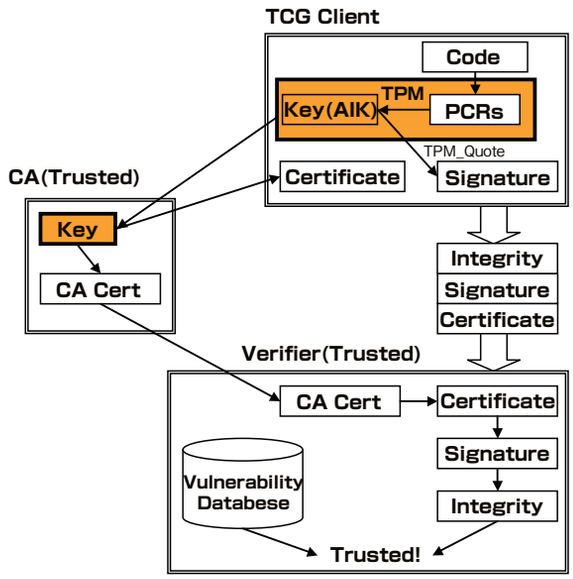

図 -3　TCG Attestation の概要

プにソフトウェアの完全性を記録するためには，ソフトウェアコンポーネントはこの Trusted bootstrap に対応しなければならない．CRTM から始まる一連のブート・シーケンスは，次に実行するコードの測定と PCR への記録（Extend）を行った後，そのコードを実行する．つまり，上のレイヤーは必ず下のレイヤーにより検査され PCR に記録されている．もしブート・シーケンスの途中に悪意のあるコードや改竄されたコードが潜んでいたとしても，そのコードはその起動前に PCR に記録されており，この PCR の記録を改竄することはできない．図 -2 では CRTM から OS までの Trusted bootstrap の形跡が PCR に記録されていることになる．

この検査結果によってブートシーケンスの継続の可否を決定するのが，Secure bootstrap である．これに対し，測定と PCR への記録は行うがブートの可否は行わないのが Trusted bootstrap である．Trusted bootstrap では改竄されたソフトウェアが起動する場合も考えられるが，TPM に記録した完全性の結果を元にして次節で述べる Attestation によりソフトウェアの改竄の有無の検出が可能で，より柔軟な運用が可能となる．

こうした Trusted bootstrap を従来のプラットフォームで実現する方法はいくつか報告[4), 7)]されており．プラットフォームのブートシーケンスが明確であれば，その実装自体は比較的容易である．PC プラットフォームについては BIOS 部分までの実装仕様が TCG により標準化されている．課題としては，プラットフォームのより上位の大規模で複雑化したソフトウェア・コンポーネントの完全性をどのように効率的に管理し運用するかであり，今後の研究が必要とされている．

リモートからの完全性の検証 — Attestation

TPM に記録されたソフトウェアの完全性をリモートからアクセスし安全に検証することを TCG では Attestation と呼ぶ．ここでは，PCR の値に対する署名を TPM チップ内で生成することにより，PCR の値が，TPM チップ由来であることを証明する．TPM には Endorsement 鍵（EK）と呼ばれる，TPM 固有の RSA 鍵が保持されており，これにより TPM の Identity を検証することが可能である．EK は TPM チップの製造者により生成が行われ，通常，製造者が Credential を発行する．プラットフォームのオーナーは TPM チップを使用可能にするために Take Ownership 命令を発行し，TPM を使用するための Pass phrase を設定するとともに，SRK（Storage Root Key）と呼ばれる鍵を生成する．実際は TPM 内部で保持されている鍵は，EK とこの SRK のみであり，その他の鍵については，SRK で暗号化してチップ外で保持し，必要に応じてチップにロードされる．このため，TPM は数に限りなく鍵を取り扱うことが可能である．Attestation で PCR に対して署名を行う鍵は Attestation Identity 鍵（AIK）と呼ばれ，TPM に固有の鍵であることを保証するためにプラットフォームのオーナーにしか生成できない．Attestation 専用の鍵を使う理由は，Attestation により，TPM の Identity（つまりユーザが使用しているマシン）が特定されることを防ぐためである．TCG クライアント上の TPM 内に保持される AIK による署名が TPM チップ由来であることを証明するためには，信頼できる第三者が AIK が TPM の中に存在することを検証し，証明書を発行する必要がある．信頼できる第三者は AIK に対する証明書を発行するにあたり，EK の Credential を含むプラットフォームの正当性を含みさまざまな Credential を元に証明書発行請求を検証し，この証明書を発行する．図 -3 に Attestation の概略を示す．TCG クライアントの検証者は Attestation で対象となる TCG クライアントの PCR の値，AIK による署名，AIK の証明書を受け取り，この信頼できる第三者による証明書から AIK の正当性（TPM 由来であること）を検証し，次に AIK により署名された PCR の値を検証することにより，得られた PCR の値を信用することができる．このように，Attestation を行うには信頼できる第三者，つまり PKI が必要になる（注：昨年秋にリリースされた TPM の Version1.2 のスペックでは新たに，新しいゼロ知識

証明を応用した，よりプライバシーに配慮した Direct Anonymous Attestation と呼ばれる Attestation 方法が追加された[9]）．

ただ，TCG で実現されるソフトウェアの完全性の検証技術，Attestation により，プラットフォームのソフトウェア・コンポーネントの完全性に関する情報を安全に入手できたとしても，実際にはその完全性の値が示すソフトウェアに脆弱性がないのか，またプラットフォームが適切に管理されているのかをマネージできるシステムやフレームワークがなければ，現実問題として意味がない．この点に関しては現在，各方面で研究が進められており，今後の進展が期待される．

Seal と Unseal

TPM を使った暗号，復号化機能でユニークなのがこの Seal と Unseal である．Seal とはあるデータを暗号化する際，復号時（Unseal）の PCR の値を指定でき，もし PCR の値が暗号時と異なると復号できない．その結果，Seal で暗号化されたデータはある特定のプラットフォーム構成でしか使えない．言い換えると，Seal で暗号化されたデータは自分自身の実行環境を検証することができる．前節の Attestation はリモートからの完全性検証技術であるが，この Seal/Unseal を用いることでより柔軟なシステム構成やデータの保護が実現できると期待される．

今後の展望

コード署名を正しく用いることにより，ソフトウェアの出所を検証することができる．しかしながら，現在一般に広く使われているオープンプラットフォームは依然 Malware やソフトウェアの脆弱性による脅威から開放されていはいない．

従来のソフトウェアによるセキュリティ機能の実装では，そのセキュリティ・ソフトウェア自体に脆弱性はないのか，改竄はないのか，の疑問が常につきまとう．理想は Trusted Computing（もしくは Code）Base（TCB）と呼ばれる十分に検証されたコードを使用することである．Microsoft 社の次期 OS のセキュリティ機構，NGSCB では Virtual Machine Monitor（VMM）を TCB として導入する予定である．また同時に TPM を用いたさまざまなセキュリティ機能も提案されている[2]．

従来，クライアント・デバイスで一定の安全性を確保するためには，携帯電話のような，クローズなプラットフォームを採用することが一般的であった．近年，機器に要求される機能の複雑化や開発期間の短縮に対応するために，オープンスタンダード技術の採用による，安価で短期間な開発手法が注目されている．しかしながら，ここで問題とされるのは端末の安全性の確保であり，そうした背景の下，TCG では安価なセキュリティ・チップとソフトウェアの組合せによる現実的な手法を提案，プラットフォームのオープン化と安全性との両立を目指している．TPM と呼ばれるセキュリティ・チップによってソフトウェアの完全性にかかわる情報を不正な改竄から保護するとともに，その信頼をベースにした新しい情報処理の基盤の構築をこの標準化技術によって実現しようとしている．現在 TCG では TPM の仕様とそれを使うためのソフトウェア・スタックの API を定義しているに過ぎず，こうした技術を今後どのように活用し Trusted Platform を実現するか，今後の研究が期待されている．

また，こうしたセキュリティ技術はユーザのプライバシー保護の問題とも深くかかわっている．たとえば，TCG では端末の特定からユーザを守るために Attestation 専用の鍵を使用している．また新しい仕様[9]では，匿名性を高めた Direct Anonymous Attestation が採用された．プライバシー保護の問題は社会的な要求であり，新しいセキュリティ技術の研究開発は絶えずこのことを念頭において取り組まなければならない．

参考文献

1）Coombs, J.: Windows File Protection Arbitrary Certificate Chain Vulnerability, http://www.science.org/secalert/WFP_Arbitrary_Certificate_Chain_Vulnerability.txt
2）England, P., Lampson, B., Manferdelli, J., Peinado, M. and Willman, B.: A Trusted Open Platform, Computer, pp.55-62（July 2003）.
3）児島　尚，丸山　宏：Java のコード署名モデルに関する議論，第 1 回インターネットテクノロジーワークショップ論文集，日本ソフトウェア科学会（Aug. 16, 1998），http://spa.jssst.or.jp/WIT/wit98/proceedings/kojima/html/
4）MacDonald, R., Smith, S. W., Marchesini, J. and Wild, O.: Bear: An Open-Source Virtual Secure Coprocessor based on TCPA Technical Report TR2003-471, Department of Computer Science, Dartmouth College（Aug. 2003）.
5）MS01-017, http://www.microsoft.com/japan/technet/treeview/default.asp?url=/japan/technet/security/bulletin/ms01-017.asp
6）MS03-041, http://www.microsoft.com/japan/technet/treeview/default.asp?url=/japan/technet/security/bulletin/MS03-041.asp
7）Munetoh, S., Nakamura, T., Maruyama, H.: Implementing Trusted Platform SCIS2004, Proceedings, Vol.2, p.1477（Jan. 2004）.
8）Trusted Computing Group; http://www.trustedcomputinggroup.org
9）TCG TPM Specification Version 1.2; http://www.trustedcomputinggroup.org
10）WFP; http://support.microsoft.com/default.aspx?scid=kb;ja;222193

（平成 16 年 3 月 16 日受付）