修士論文

情報漏洩防止のためのプラットフォーム認証

指導教員　　坂井　修一　教授

東京大学大学院　情報理工学系研究科
電子情報学専攻

文　栄光

February 2010

概要

近年，情報社会の発展とともにインタネットを通じたデータの配信が増加している．しかし，そのデータの中には機密データや著作物など扱いに制限がかかっているものもあり，それらのデータに対しては，クライアント側でデータの不正な解析や改ざんが行われないことをサーバー側に対して保証する必要性がある．そのため近年プラットフォーム認証に関する関心が高まって来ている．

TCG ではセキュリティモジュールである TPM を用いた遠隔認証を提案している．しかし，TPM を用いた遠隔認証は，認証に用いるプラットフォームの情報の完全性を確立するための制約の固さなどの問題点があり，実際には利用されていない．

本論文では，ソフトウェアによるタンパを防ぐ機能を持つ耐ソフトウェアパンパ・プロセッサを用い，さらにその機能を拡張させることで遠隔認証を可能とする手法について提案する．耐ソフトウェアタンパ・プロセッサにプロセッサと Secure DMA 専用の非発揮性メモリを追加し，さらに Secure DMA にハッシュ演算機構を追加することで，Secure DMA が転送を行いながらハッシュ値を取得できるようにする．また，耐ソフトウェアタンパ・プロセッサのアクセス保護機能や拡張 Secure DMA のメモリへのアクセスを保護することで計測したプラットフォームの情報の完全性を確立する．また，サーバーは認証の対象となるコンポーネントの計測をサポートする検証プログラムを保護データを配布する前に配布することで遠隔認証をサポートする．耐ソフトウェアタンパ・プロセッサは認証に用いる情報を保存しておき，認証後の保護データの実行時に再認証を行うことで認証後のプラットフォームの改ざんやなりすましを防ぐことができる．

目 次

図 目 次

第1章　はじめに

近年，情報機器の目覚ましい発達により，情報社会が実現されつつある．企業や公共機関では多くの情報が電子化された形で管理され，エンターテインメント部分では電子コンテンツが主流となっている [13]．また，インターネットインフラストラクチャーの充実はさらに情報社会を広げている．PC を含む情報機器のモバイル化と高速ネットワーク環境の普及により，人はいつでもどこでもネットワークに接続し，情報にアクセスすることができるようになった．しかし，これらのデータの中には機関や会社などの機密データや個人情報，著作物なども含まれている．これらの情報が漏れた場合，ネットワークを通じて急速に，情報の劣化なしで広がってしまう．公共サービスの情報化やデジタルコンテンツビジネスの拡大などにより保護されるべき情報はますます増加されると考えられ，それらが漏えいされた場合の被害は今後深刻なものになると見られている．

情報漏えいの原因には人の不注意やミスもあるが，マルウェアを用いてデータが盗まれるケースも後を絶たない．情報漏えい防止策やコンテンツの著作権保護策としてよくデータの暗号化技術が用いられる．暗号化された状態でデータを管理し，復号鍵を持っている者のみが復号化処理を行って再生できる．しかし，復号処理を行うソフトウェアや，再生するアプリケーションが情報を漏らすように改ざんされている場合までの対策にはならない．それ以外のソフトウェアの保護技術が提案されているがソフトウェアレベルだけでは完全でないという報告もある [1].

そのため，今後よりその重要性が大きくなるコンテンツの著作権保護やソフトウェアの無断な改ざん・解析の防止，機密情報漏えい防止対策として，それらを開発・配布する側での対策の施しだけでなく，それらを利用するクライアント側の環境 (プラットフォーム) の検証に対しる技術的要求が高まっている．その例としてディジタルコンテンツビジネスにおけるクライアント側のプラットフォーム認証の必要性について述べる．ある会社が製品としてデジタルコンテンツをインタネットを通じて配布するとする．その会社はコンテンツの著作権を守るために暗号化された形で，正当にコンテンツを買ったクライアントにのみ復号化鍵と一緒に配布する．しかし，クライアント側のプレイヤが改ざんされていて復号化されたコンテンツのデータやその復号鍵を洩らすと会社は損

害を受けることとなる．もし，その会社がコンテンツを再生するプレイヤまで特定のもの (改ざんが難しいものとする) に限ってそのプレイヤを配布のときに認証をするとする．そのような場合でも，OS が改ざんされているとOS の特権などを用い，メモリからデータを盗むことができる．

このように完全な情報漏えい防止対策をとるためには，その情報が扱われる側のプラットフォームが安全なものであるかの検証が必要となる．現在，プラットフォーム認証を支援する技術として多く知られている方法は TPM [8, 9] を用いた遠隔認証というものである．しかし，TPM を用いた遠隔認証法では認証を行うためのデータの信頼性を確立するため TPM を除いた全てのコンポーネントを信頼せず，その全てのコンポーネントを TPM を用いて計測する．そのため，BIOS をはじめ，OS までの全てのコンポーネントの有り得る全ての状態を予測できないと認証を行うことができない．しかし，それは現実的に困難であり，その理由で TPM が広く普及はされているもののその遠隔認証での活用はされていない．

本論文では，本研究室で提案した耐ソフトウェアタンパ・プロセッサのメカニズムを用い，遠隔認証を行う方法について提案する．

以下，本論文は次のように構成される．2 章で，遠隔認証について定義し，TPM を用いた遠隔認証法について説明する．またその問題点について述べる．3 章で，耐ソフトウェアタンパ・プロセッサについて解説を行い，4 章でそのメカニズムを用いた遠隔認証法について提案する．5 章でまとめと今後の課題について述べる．

第2章　遠隔認証とTPM

本節では遠隔認証について説明し，Trusted Coputing Group(TCG) が提案している TPM を用いた遠隔認証の手法について述べる．

2.1　遠隔認証

2.1.1　プラットフォーム認証

まず，あるコンテンツが信頼できる環境で実行されることを保証するためのプラットフォーム認証について説明する．

本論文で指す**「信頼」**とは，サーバー側やベンダ側のように情報を提供する側が期待する通りに行動することを意味し，信頼できるプラットフォームとは，提供される情報 (コンテンツ) が提供する側が期待する通りに扱われるようなプラットフォームのこと意味する．プラットフォーム認証とは，プラットフォームが信頼できるものかを検証することを意味し，プラットフォームがある機能を備えているか判断することである．その手段としてプラットフォームの構成要素がある特定のものであることを検証する．

2.1.2　Local 認証と遠隔認証

プラットフォームの認証には認証を行う側と対象プラットフォームの位置関係より”Local 認証”と”遠隔認証”に分かれる．Local 認証と遠隔認証の概要を図 2.1 に示す．

Local 認証とは図 2.1 の左側のように認証対象のプラットフォームと認証を行うエージェントが同一のマシン上にある環境で行われる認証を指す．このような環境での認証は一般にそのマシンを使用するユーザーが主体になって認証を要求することが多い．

遠隔認証とは図 2.1 の右側のように認証対象のプラットフォームが認証を行うエージェントと分離され，離れている環境で行われる認証を指す．このような環境での認証は認証対象のプラットフォームの情報をそのプラットフォーム

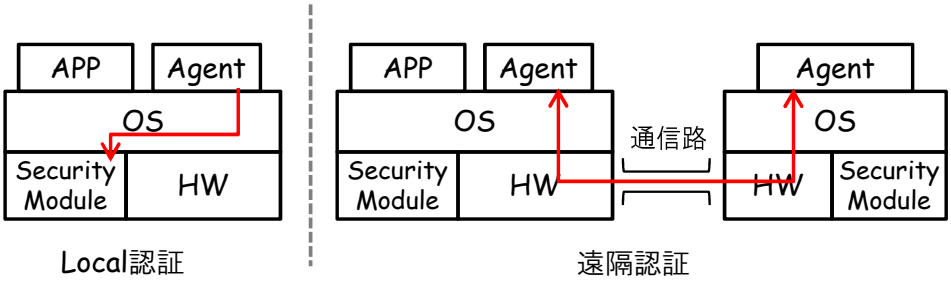

図 2.1: Local 認証と遠隔認証の位置関係

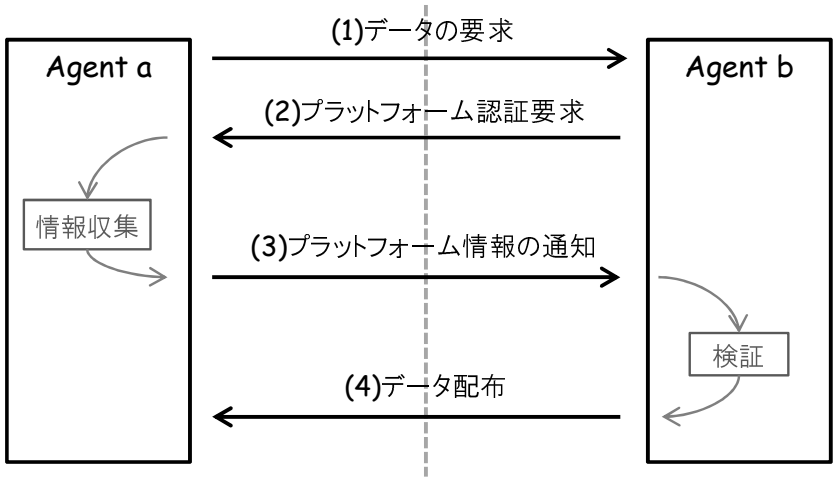

図 2.2: 一般的な遠隔認証の概要

に乗っているエージェントが集め，認証を行うエージェントに通知し，その情報に対する検証を行うことで行われる．遠隔認証が行われるときの一般的な流れを図 2.2 に示す．

エージェント a が機密情報などをエージェント b から得るため，エージェント b にデータを要求する．機密データの配布の要求を受け取ったエージェント b はその機密データの保護のため，エージェント a にプラットフォーム認証を要求する．プラットフォーム認証の要求をもらったエージェント a は認証対象のプラットフォームの情報を集め，エージェント b に通知する．エージェント b は予め持っていた信頼できるとみなされるプラットフォームの情報リストと通知された情報を比較し，検証を行う．一致する情報があり，検証に成功するとエージェント a のプラットフォームが信頼できるものと判断し，機密データ

を配布する．

遠隔認証の環境では認証を要求するエージェント (サーバやベンダ側) がその認証対象であるプラットフォームの持ち主を信用できない．つまり OS の機能も改ざんされていることを考慮しなければならない．

本稿では，遠隔認証でのプラットフォーム認証についてのみ考慮する．遠隔認証の最も重要な検証すべきことは以下の 2 つである．

- 集められたプラットフォームの情報が対象のプラットフォームのものであること
- 認証されたプラットフォームの情報通りの環境で動いていること

1 つ目は集められた情報が他のプラットフォームからのなりすまし情報であったり，改ざんされたりすることを防ぐことを指す．2 つ目は集められた情報から変更されることなく情報通りの環境が維持されていることを指す．つまり集められた情報と実行されるプラットフォームが同一であることを意味する．遠隔環境ではユーザーが信用できない枠に入っているため対象のプラットフォームの OS を信用することはできない．つまり OS の機能に依存しない情報の収集手法を考慮する必要がある．

2.2 Trusted Platform Module

2.2.1 Trusted Computing Group

Trusted Computing Group(TCG) は，信頼できるコンピュータプラットフォームを実現するためのハードウェア及びソフトウェア API の標準仕様を開発し，普及することを目的とする業界標準化団体である [7, 8]．ここで述べるプラットフォームとは，ソフトウェアが実行されるにあたり必要な基盤全体を指す．この仕様はオープンなものとし，その上で安全な環境を目指している [8].

TCG において初めから信頼できるハードウェアはプロセッサではなく，*Trusted Platform Module*(TPM) と呼ばれるセキュリティモジュールである [9, 10]．TPM をプラットフォームに載せ，プラットフォームの信頼性を測るためのエンジンとする．

2.2.2 TPM の概要

TPM は，乱数生成，公開鍵暗号の処理，ハッシュ値の計算，秘密鍵の保存，耐タンパー性など，暗号モジュールとしての基本機能を持つ．TPM が IC カー

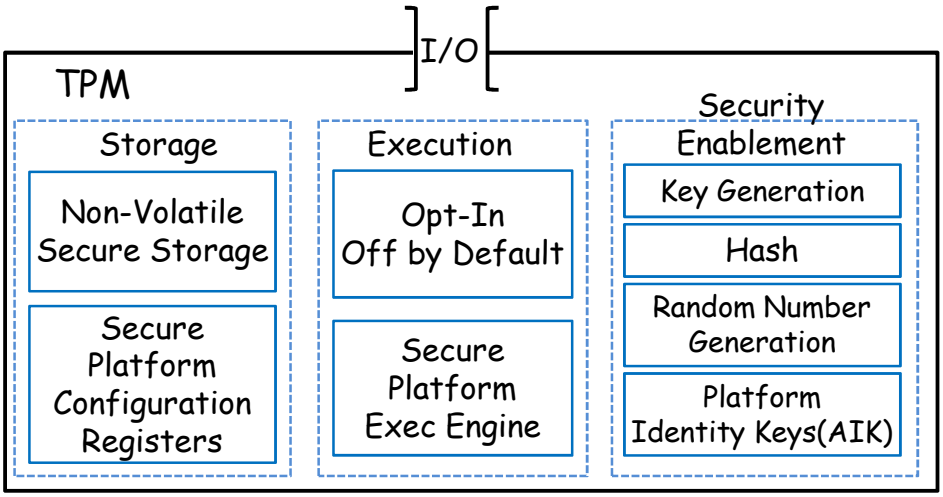

図 2.3: TPM 基本構成

ドなど既存の暗号モジュールと大きく違う点は，TPM 内の PCR(Platform Configuration Register) と呼ばれる非発揮性記憶領域が，クライアント PC 内で稼働する各種ソフトウェアモジュールのハッシュ値を記録可能な点である．

図 2.3 に TPM の基本構成を示す．

2.3 TPM を用いた遠隔認証

TCG で提案する TPM を用いた遠隔認証の目標は次のとおりである [5] [3].

> TPM を搭載している local platform(クライアント側) のコンポーネントを離れた側 (サーバー側など) で認証が行えるようにする．

この目標を達成するための手法について以下で説明する．

2.3.1 基本的考え方

TCG で提案している TPM を用いた遠隔認証は大きく 3 つのメカニズムに分けられている [4]．クライアント側が自分自身のプラットフォームを計測し，計測した値を保存する部分に当たる「Trusted Boot」，遠隔側から来た認証要求に応じて保存したプラットフォームの計測値を遠隔側に送る「通知」，予め認証側が持っていた期待値との比較を行う「認証」がそれである．ここで言う計測とはプラットフォームにインストールされているコンポーネントらのコードや

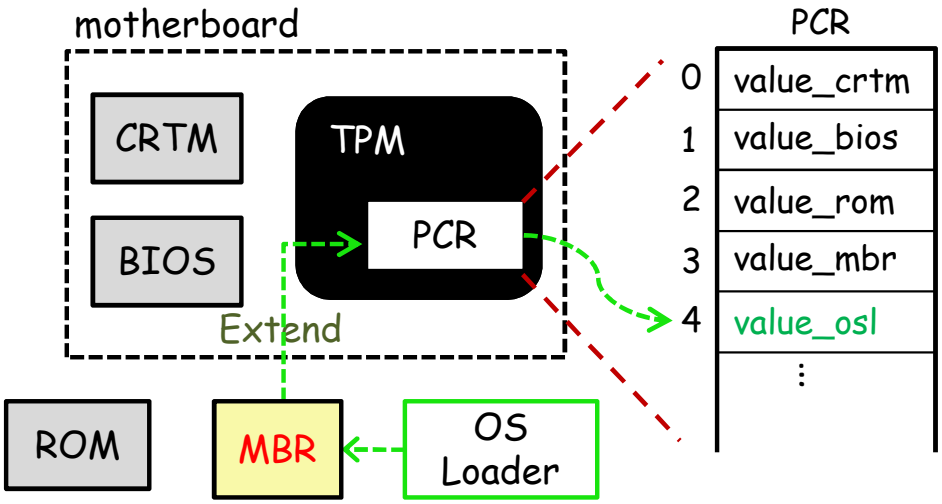

図 2.4: Chaining Trust

データを取ってきてハッシュ値を求めることを指す．また計測した値は TPM の PCR に格納される．

2.3.2 必要な技術

上記した手法を実現するための技術や制約について説明する．

Trusted Boot

遠隔認証を行うためには前節で述べたように，まず，クライアント側がクライアント側のプラットフォームの構成要素を正しく計測し，それを安全に保存できる必要がある．ここで言う正しく計測とはプラットフォームに搭載されている構成要素のコードやデータそのものを改ざんされることなく取り，それらのハッシュ値を求めることを意味する．また，安全に保存するとは求められたハッシュ値が計算後に改ざんされずに保存され，それが維持できることを意味する．これらを実現するため，TPM では「Chaining Trust」メカニズムを用いて計測と計測をされたデータのハッシュ演算を TPM 内で行い，その結果を TPM 内にあるレジスタ (PCR) に格納する．これによって計算したハッシュ値が TPM 外に漏れることがなくなり「Trusted Boot」[14] が正しく安全に行われる．

Chaining Trust の概要について説明する (図 2.4)．PC 起動時に安全にコンポーネントの計測を行う技術である Trusted Boot において Chaining Trust メカニズムは最初の信頼できる領域である TPM からその信頼できる領域を連鎖的に増

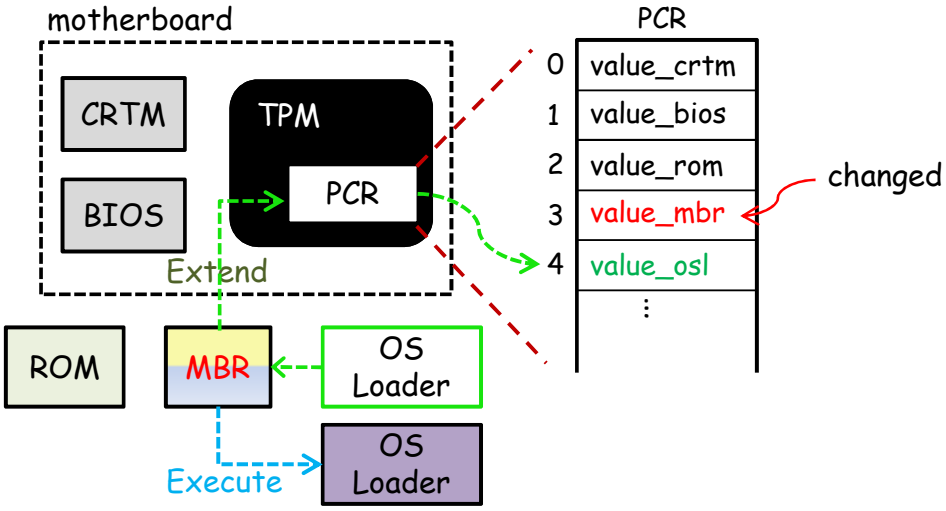

図 2.5: Chaining Trust における攻撃

やしていき，計測したハッシュ値の完全性を確立するものである．TPM を搭載しているプラットフォームは TPM と一緒に CRTM (Core Foot of Trust for Measurement) という起動時最初に実行される TPM 初期化コードを物理的に保護された形でマザーボードに埋め込んでいる．Trusted Boot では CRTM が PC 機動と同時に自分自身を計測する．CRTM の計測及び実行が終わったら BIOS に制御を渡す前に BIOS の計測を行う．このときまだ制御権限は CRTM にある．BIOS の計測が終わると BIOS を起動し BIOS に制御を渡す．BIOS は ROM と MBR(Master Boot Record) の計測を行いそれらを起動させ，制御を渡す．同じ処理を OS への制御権限が渡される時まで繰り返す．図 2.4 で灰色のコンポーネントは計測及び実行が終了したものを意味し，黄色の赤い文字のコンポーネントが現在実行中であることを意味する (MBR)．緑のコンポーネントは現在実行はされておらず，その下のレイヤのコンポーネントにより計測が行われているものを意味する (OS Loader)．PCR に格納されているハッシュ値は次に実行されるコンポーネントのものである．

つまり，あるコンポーネントが 1 つ上のレイヤのコンポーネントを起動される前にそのコンポーネントのコードやデータを計測し，自分自身の計測には関与できない．そのため，仮にあるコンポーネントを悪意を持つものが改ざんし，その計測時にはただしコンポーネントを計測しながら実際の起動は改ざんされたものにし，計測値を騙すことで行える攻撃を完全に防ぐことができる．その例を図 2.5 に示す．

改ざんした OS を起動させるため OS Loader を改ざんしたとする (図 2.5 の紫

の OS Loader) このとき改ざんした紫の OS Loader を MBR でそのまま計測を行うとその計測値がサーバーが期待する予測値と異なることになり，OS Loader の改ざんは知られてしまう．OS Loader の改ざんが分からないように OS Loader を計測する MBR まで改ざんし，この MBR に正しいと期待される OS Loader (図 2.5 の緑四角の OS Loader) を計測させながら実際には先ほどの改ざんした OS Loader を起動させるようにしたとする．この場合確実に OS Loader に対する計測値は正しくなり，実際には改ざんした OS Loader が動くため攻撃に成功したかのように見える．しかし，MBR の計測を行うのはその下のレイヤの ROM であり，今度は MBR の計測値が予測値と異なり改ざんが知られてしまう．また，その下のレイヤを改ざんすることを繰り返すとしても CRTM が物理的に保護されているため CRTM の改ざんはできない．そのため，あるコンポーネントを改ざんした場合 PCR のあるハッシュ値が必ず予測値と異なることとなり，改ざんされたことを分かるようになる．

このメカニズムにより，計測したコンポーネントと実際に動作するコンポーネントが同じであることを保証する．

通知

PCR に保存しておいたプラットフォームのコンポーネント等の計測値を遠隔側からの認証要求に応じて安全に通知することが必要である．ここで言う安全に通知するとは PCR に保存しておいた値が通知される間に改ざんされないことを保証することを意味する．Trusted Boot メカニズムは CRTM を信頼することを前提でブート時にプラットフォームのコンポーネントを計測してその結果を保存しただけであるため，OS が起動した時点ではまだそのプラットフォームが正しいものか否か判断できない．このような環境で，遠隔側からの認証要求に対し PCR に保存されていた計測値を OS を通じて通知すると OS による改ざんが行われないとは保証できない．OS が安全であることが判断できるのは，CTRM から始まった Chaining Trust メカニズムで計測したハッシュ値がすべて遠隔側に通知され，期待値との比較で認証された後からである．

TPM では安全な通知を実現するため，認証の要求が来たら TPM 内部で持っている秘密鍵を用いて PCR の値に対し電子署名を行い，署名に使用された鍵の証明書と一緒に通知する．この際，用いる鍵は *Attestation Identity Key*(AIK) と呼ばれる．この鍵に対し，認証局から証明証を発行することで中間者攻撃を防ぐことができる．TPM で使用される証明書については [8, 15] を参照されたい．

TPM に組み込まれている鍵を用いることで電子署名のための鍵を安全に配布することができる．また，TPM 内部で暗号化を行うので安全に暗号化を行

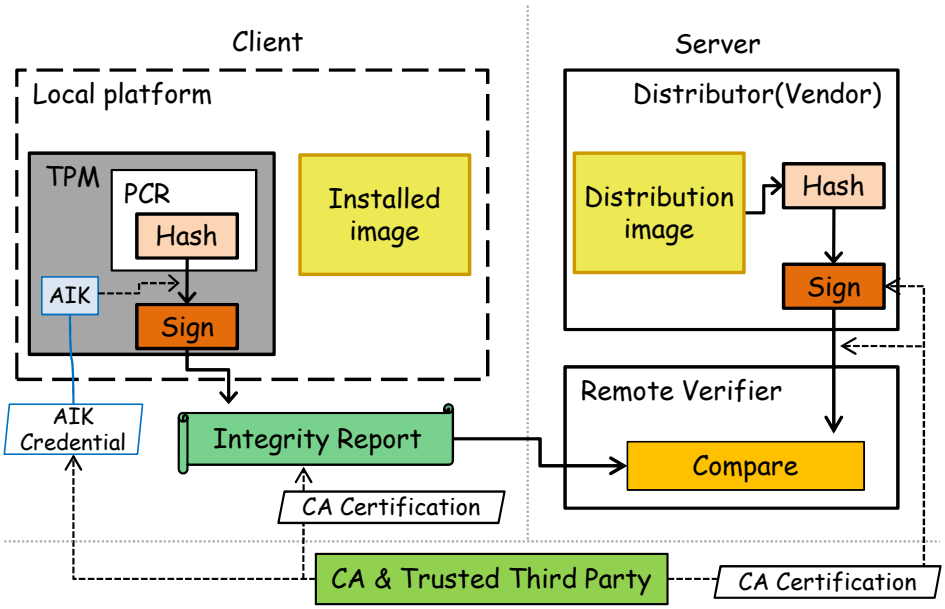

図 2.6: TPM を用いた遠隔認証の基本構造

うこともできる．

2.3.3 認証の流れ

Trusted Boot メカニズムにより計測・保存されたクライアント側のプラットフォームのコンポーネント情報はサーバーやベンダ側の認証要求により通知され，サーバーやベンダ側が持っている期待値との比較を行うことで認証が行われる．TCG では通知を行うためのフォーマットを TPM とベンダ側において確立しており，各レポートは CA による証明書を用いて真偽性を判断する．TPM を用いた遠隔認証の基本構造を図 2.6 に示す．

2.4 TPM を用いた遠隔認証のまとめと問題点

TCG では，信頼の起点として Trusted Platform Module と初期化コードである CRTM(Core Root for Trust for Measurement) を利用し Trust Boot を行うことでプラットフォーム全体を計測し，それを用いて認証を行う方法を提案している．Trust Boot では Chaining Trust メカニズムを用いることで改ざんやすり替えを防ぎ，正しい計測が可能となる．また，最初の認証が行われるまでは認証の手順を OS に依存しない方法で行うことで不正な認証を防ぐことができる．

しかし，先述したようにTPMを用いた認証を行うためには全てのプラットフォームのコンポーネント等の計測値と期待値の比較を行う必要があり，コンポーネント等のあり得る全ての期待値を予め用意する必要がある．それはパッチやアップデートなどが頻繁に行われる近年の状況を考慮すると現実的に困難である．また，オープンソースベースのOSなどのコンポーネントに対する期待値は用意することがさらに難しく，近年その領域を広げているオープンソースベースのコンポーネントへの対応に限界が見られる．

また，Trusted Bootにより計測されるタイミングと認証の要求が来るタイミングには時間差が生じるため，その間のプラットフォームのコンポーネントの変更に対して通知する機構も必要となる．最後に，TPMを用いて認証か行われたとしてもそれはプラットフォームの構成の完全性であり，潜在的バーグなどによる危険性がないという安全性を保証するわけではない．

このような問題点があり，現在多くのコンピュータにTPMが搭載されているにも関わらず，その遠隔認証への実利用はまだ行われていない．

第3章　耐ソフトウェアタンパ・プロセッサ

本論文では TPM のみを信頼する TCG の問題点を解決し，より実用的な遠隔認証機構を実現するため当研究室で提案したセキュア・プロセッサである耐ソフトウェアタンパ・プロセッサ [16] を用いた遠隔認証の手法について考察する．本章では，その基本コンポーネントとなる耐ソフトウェアタンパ・プロセッサについて説明する．

セキュア・プロセッサ [2, 6, 12] はハードタンパを考慮してプロセッサ/メモリ間で暗号化を施し，メモリにあるデータは暗号化された状態になっており，ソフトウェアタンパ及びハードタンパを防ぐことができる．しかし，暗号化によるメモリアクセス速度が低下する問題点が生じる．清水らは容易に行うことが難しいハードタンパの脅威は少ないと見込み，ソフトウェアタンパのみに着目し，メモリ上のデータに対し暗号化を行わないことでメモリアクセスのオーバヘッドが生じない高速実行の可能な耐ソフトウェアタンパ・プロセッサを提案している．

3.1　Secure DMA と Secure TLB

本論文では 2.1 節で述べたように遠隔認証を対象としている．そのため OS を信頼できないものとしている．信頼していない OS 上で耐タンパ性を確立するためには，機密データへのアクセスを制限する必要がある．具体的なアクセス制限は下の 2 つである．

- アクセス制限によって OS からのアクセスを防ぐ
- アクセス制限の外に出す物には，暗号化と認証をする

耐ソフトウェアタンパ・プロセッサではこれを実現するためのプロセッサに *Secure TLB* と *Secure DMA* の機能を追加し，さらに OS に変更を加えることで耐タンパ性を確立する．

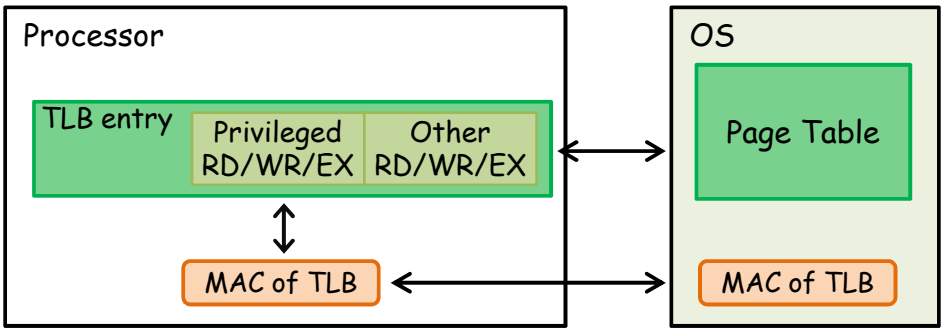

図 3.1: Secure TLB

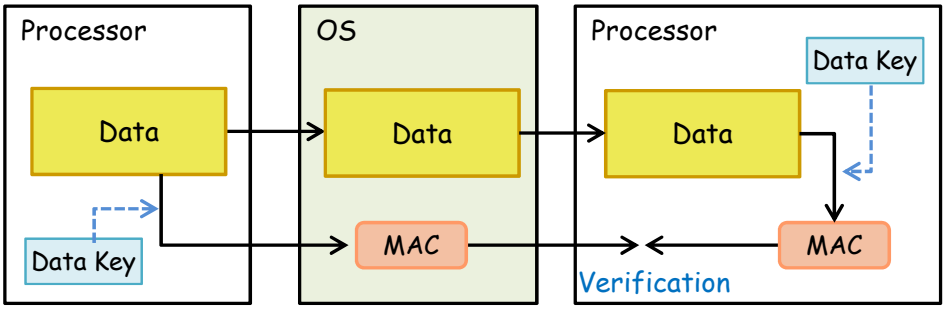

図 3.2: MAC による改ざん確認

3.1.1 Secure TLB

Secure TLB は，拡張されたアクセス権限と，MAC による認証機能を持った TLB である．アクセス権限の拡張によってキャッシュ/メモリを秘匿し，*Message Authentication Code*(MAC) による認証でページテーブルの改ざんを防止する (図 3.1).

キャッシュやメモリを他のプロセスから秘匿するために，ページのアクセス権限を拡張し，特権プロセス用のアクセス権限と，他のプロセス用のアクセス権限を追加する．Secure TLB でこれをチェックし，権限がなければアクセス違反となる．また，ページテーブルそのものが書き換えられないように，ページテーブルエントリの MAC を生成し，ページテーブルと共に OS に管理させる．

MAC は，メッセージにパスワードを付加したデータのハッシュを取ったもので，一般に，メッセージとともに通信相手に送り，相手側でメッセージが改ざんされていないことを確認できるようにする技術である．図 3.2 に耐ソフトウェアタンパ・プロセッサのプロセッサと OS 間の MAC の利用例を示す．

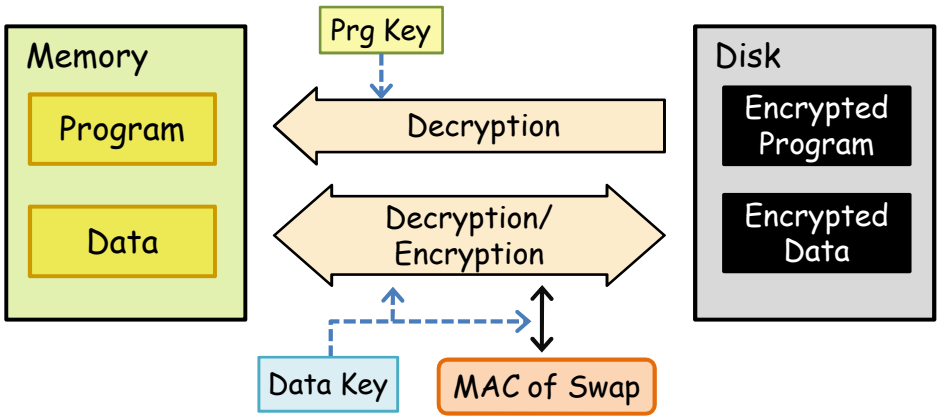

図 3.3: Secure DMA

3.1.2 Secure DMA

Secure DMA は，暗号処理機能と MAC による認証機能を持った DMA 転送で，二次記憶とメモリの間で暗号化や復号化を行いながら転送をすることができる (図 3.3).

暗号処理機能は，プログラムコードの復号化と，スワップの暗号化/復号化に分けられる．二次記憶からプログラムコードを読み込む際，プログラム鍵を利用して転送と同時に復号化を行う．また，プログラムが利用したメモリがスワップ処理によって二次記憶に書き出される際，そのデータは Secure DMA によってデータ鍵を使って暗号化され，再びメモリに戻される時に復号化される．

Secure DMA においては，メモリの内容にパスワードとしてデータ鍵を用いて MAC を生成する．この MAC を用いてスワップインの際に検証を行うことで，スワップの改ざんを防ぐことができる．

以下で，OS の変更点ごとに各機能を説明する．

3.2 OS の変更

耐ソフトウェアタンパ・プロセッサでは，OS に手を加えることでファイル，プロセス，メモリの管理を OS で行いながらも OS の信頼性には依存せずに耐タンパ性を確立する．

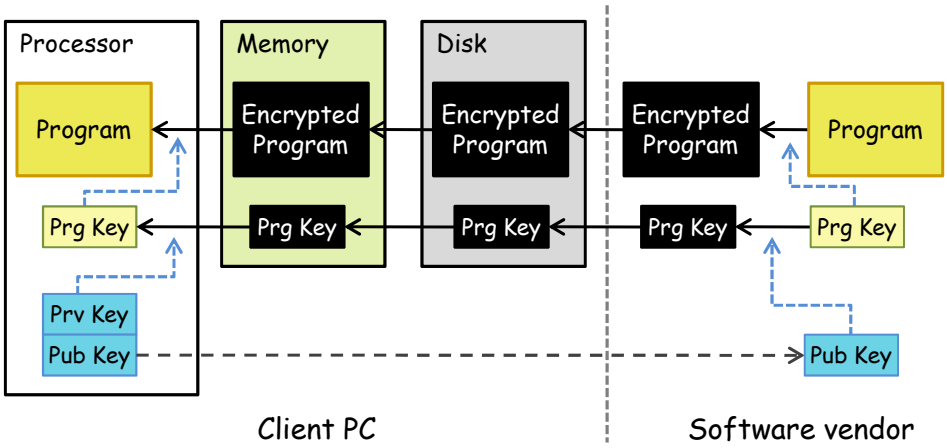

図 3.4: 暗号化されたプログラムの配布及び実行

3.2.1 ファイル管理

耐ソフトウェアタンパ・プロセッサでは，暗号化されたプログラムファイルを読み込み時に復号化し，メモリに展開し実行する．プログラムの復号化に用いる鍵 (プログラム鍵) はあらかじめプロセッサ内部に用意する．この鍵はセキュア・プロセッサの場合と同様にして配送される．一般にセキュア・プロセッサで暗号化されたデータとその復号鍵の配布には，図 3.4 に示したようにプロセッサが予め持っている公開鍵暗号方式の鍵ペアを用いる．ソフトウェアベンダはまずプログラムを共通鍵暗号方式で暗号化し，その鍵 (暗号化・復号化同一) をプロセッサの公開鍵を用いて暗号化しプロセッサに配布する．プロセッサは受け取った鍵をプロセッサが持っている秘密鍵で復号化し，プログラムを実行する．全ての鍵の管理はプロセッサが行う．

プログラムの読み込み，データの読み/書き込みには，先に述べた Secure DMA を用いる．ここでは，Secure DMA を用いたプログラムの読み込みについて説明する．

プログラムの実行のためプログラムの読み込みを行うとき，耐ソフトウェアタンパ・プロセッサでは，図 3.5 のように Secure DMA でメモリ上に展開する．DMA コントローラは転送のコマンドとともにプログラム鍵を受け取り，これを使って復号化しながらメモリ上に展開する．

また，このときの転送先アドレスは OS が決定するが，このアドレスを OS が偽ると，プログラムが正常に動作しない．これを防ぐため，ページテーブルが改ざんされていないことを確認する必要があるが，この方法については 3.2.3 で述べる．

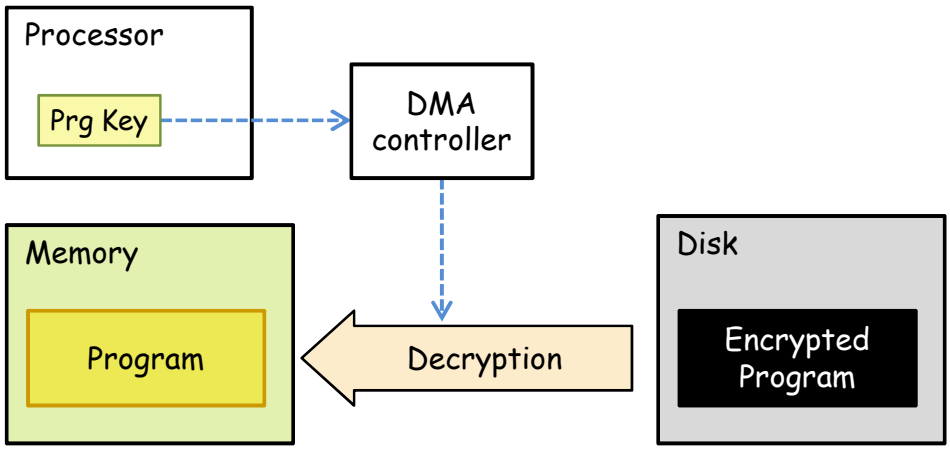

図 3.5: プログラムファイルの読み込み

3.2.2 プロセス管理

耐ソフトウェアタンパ・プロセッサではプロセスの切り替え時の OS による解析や改ざんを防ぐため，プロセスの切り替えを行う際に，プロセッサは全てのレジスタについて暗号化と MAC の生成を行う．このため，耐ソフトタンパ実行をするプロセスを起動するとき，プロセスごとにランダムなデータ鍵を用意する．プロセスごとに用意することで，同一プログラムを複数起動した場合にも区別できる．

先に述べたプログラム鍵やデータ鍵はプロセッサが現在実行中のコンテキスト番号と関連づけ，プロセッサ内部のテーブルに記憶する．他のプロセスの鍵を利用することはできず，またプロセッサの外部からこれにアクセスすることもできない．

プロセスの切り替えが発生した場合，実行中のプロセスのレジスタなどの状態を退避させる．このプロセスの状態は OS が管理し，そのプロセスを再開する際にプロセッサに読み込まれ，プロセスの実行が続けられる．しかし，プロセスの状態を OS が管理するため，OS による解析や改ざんを防ぐ必要がある．プロセッサはプロセス切り替えのための割り込みが発生すると，割り込み直前に実行されていたプロセスのレジスタなどの状態に関する全ての情報に対して暗号化と MAC の生成を行う．この際使う鍵はそのプロセスのコンテキスト番号に対応するデータ鍵を用いる．図 3.6 にプロセス切り替えが行われる際のレジスタの退避について示す．

暗号化されたプロセスの状態情報と MAC は，プロセッサから OS が読み込んで管理し，再びこのプロセスに実行が切り替わるとき，OS からプロセッサ

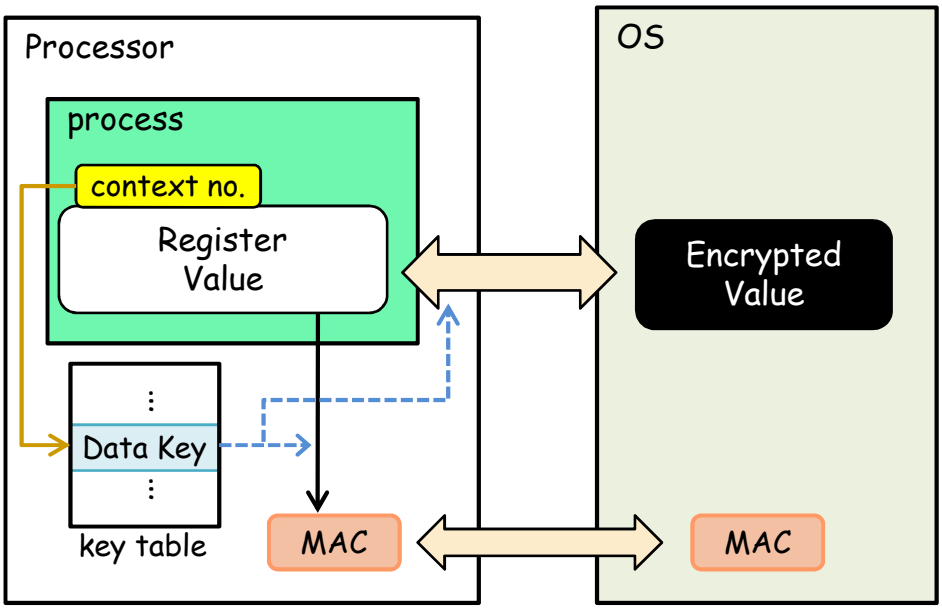

図 3.6: プロセス切り替え際のレジスタ退避

に戻される．このとき，復号化を行ってハッシュを再計算し，MAC の一致を確認することで改ざんの検出を行う．

プロセスの終了時に，そのプロセスに関連づけられた鍵をテーブルから削除する．

3.2.3 メモリ管理

耐ソフトウェアタンパ・プロセッサでは，Secure DMA によりメモリ上に展開されるプログラムやデータが復号化された状態になっている．そのため，そのままでは OS や他のプロセスからタンパが可能である．これを防ぐため，メモリ管理の点において以下の 3 つを変更している．

メモリ保護の強化

一般のメモリ保護機能では，ページやセグメントの所有者を設定することでそれ以外のプロセスからの不正なアクセスを防いでいる．しかし，OS などの特権を持つプロセスは，すべてのメモリにアクセスすることができてしまう．そこで，通常のメモリ保護のためページテーブルエントリに含まれる所有者によるアクセス権限の読み込み (RD)/書き込み (WR)/実行 (EX) に，他のプロセス

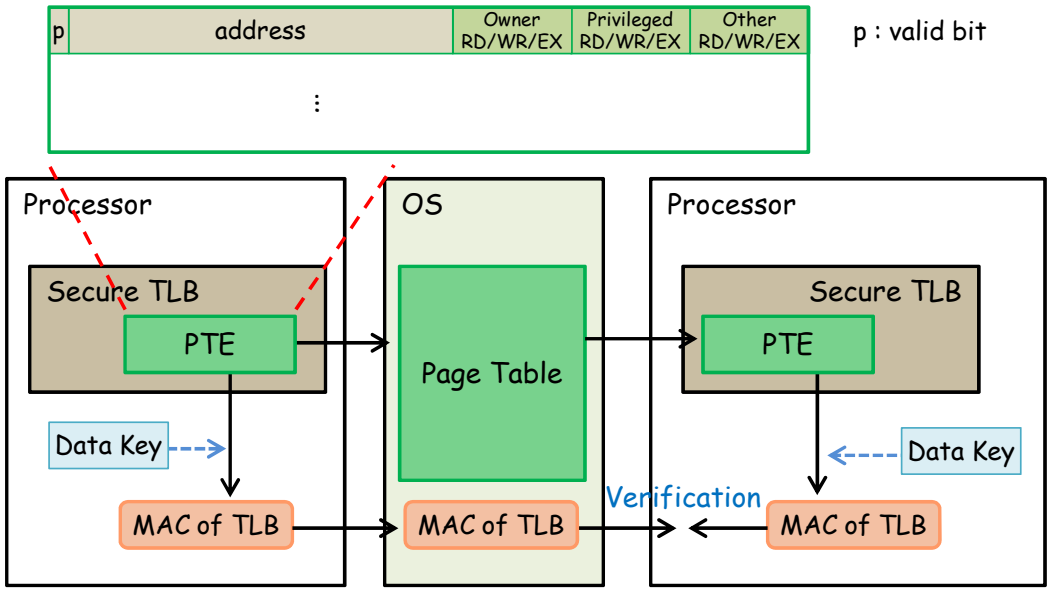

図 3.7: OS によるページテーブルの改ざん防止

によるアクセス権限の RD/WR/EX ビットと，特権プロセスによるアクセス権限の RD/WR/EX ビットの拡張アクセス権限ビットを設けることで，特権を用いたタンパを防ぐ．

しかし，普通これらのアクセス権限は OS が新しくページデーブルを作成するとき各ページに対して OS が書き込むため，このビットがプログラムの指定通りになっているかどうかは信頼できない．そのため，先述の Secure TLB にてプログラムが指定するビットを直に書き込む．Secure TLB でも同じくアクセス権限のビットを拡張し，メモリアクセス時にアクセス権限をチェックし，権限がなければアクセス違反とする．キャッシュのアクセスに関しても，同様に Secure TLB で権限をチェックすることでアクセス違反を検出できる．

ページテーブルの改ざん防止

通常，ページテーブル管理は OS がする．OS が不正にページの内容を書き換えられてしまうとアクセス権限が無効化されたり，異なるページを読ませたりすることができてしまう．OS による不正なページテーブルエントリの改ざんを防ぐため，図 3.7 で示すようにページテーブルエントリに対しても MAC を生成し，改ざん検証を行う．MAC は，Secure TLB からページテーブルエントリが追い出されるときページテーブルエントリとデータ鍵を用いて生成する．生成された MAC は OS が読み込んで管理し，再びこのページテーブルエ

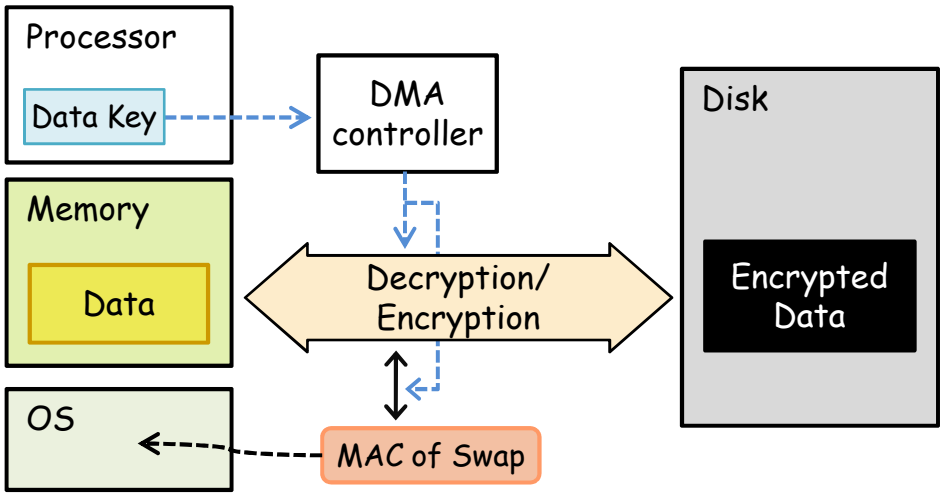

図 3.8: スワップ処理

ントリが必要になったときに，ページテーブルエントリと共に Secure TLB に渡される．Secure TLB ではこの MAC を確認し，改ざんされていなければ有効とする．

スワップの保護

二次記憶上にあるデータに対し上記の機能ではデータが保護されない．そのため，スワップによって二次記憶上に書き出されるメモリの情報は上記の機能では保護されない．メモリからスワップによって追い出される情報を保護するために，Secure DMA による暗号化と MAC による改ざん検証を行う．図 3.8 にスワップ処理の概要を示す．スワップアウト時，ページのデータは Secure DMA で先述したプロセスのデータ鍵を用いて転送時に暗号化され，二次記憶に書き込まれる．また，データにデータ鍵を付加した上でハッシュを計算し，MAC とする．この MAC は OS が読み込み，スワップインまで保存しておき，スワップインするとき OS が Secure DMA に対して MAC を渡し，プロセッサ内でハッシュを再計算して MAC との一致を確認する．

3.3 Secure TLB と Secure DMA の協調

上述の方法によって，アクセス制限や暗号化，MAC 検証などが行われ，生成されたプロセスの状態やデータが保護される．しかし，以下の 2 つの DMA に関する問題が残っている．

- アクセス権限のないページの DMA 転送
- DMA 転送に関する PTE の改ざん

その対策として，Secure TLB と Secure DMA の協調による方法を述べる．

3.3.1　アクセス権限を持たない DMA 転送の防止

メモリ上のデータは，先述したメモリ保護の拡張によって特権を持つプロセスからのアクセスであっても防ぐことが可能になる．しかし，DMA 転送によるアクセスには先述のアクセス保護が及ばない．そのため，DMA 転送によってアクセス権限を持っていないページでも転送できてしまう．これができると，メモリ上で保護すべきページのアクセスを制限しても，それらのページを DMA 転送を用いてディスクに転送してからアクセスすることが可能になる．

これを防ぐために，Secure TLB と Secure DMA を連携させ，Secure TLB が持つアクセス権限情報を Secure DMA と共有し，転送するページのページテーブルエントリを参照して DMA を行うプロセスが正しく権限を持っているか確認する．この確認によって，権限を持たない DMA 転送は防げる．

3.3.2　DMA に関するページテーブルの改ざん防止

ページテーブルの改ざん防止については 3.2.3 で説明したが，DMA 転送が行われるとページテーブルを変更しなければならない．この DMA 転送に関するページテーブルの変更を OS に任せると，改ざんが行われる可能性がある．例えば，DMA 転送していないのにしたことにすると，その領域は空き領域扱いになり，全てのプロセスからアクセス可能になってしまう．

これを防ぐため，Secure DMA が行った転送の情報を，Secure TLB 上の PTE に反映させることを行う．DMA に関するページテーブルの変更は Secure DMA からの情報を受けた Secure TLB が行い，OS に任せることはしない．

この方法をとることで，DMA 転送におけるページテーブルの改ざんを防ぐことができる．

3.4　耐ソフトウェアタンパ・プロセッサのまとめ

耐ソフトウェアタンパ・プロセッサでは，以下に挙げた機能を用いることでプログラムやデータを保護し，ソフトタンパを防ぐ．全体の流れを図 3.9 に示す．

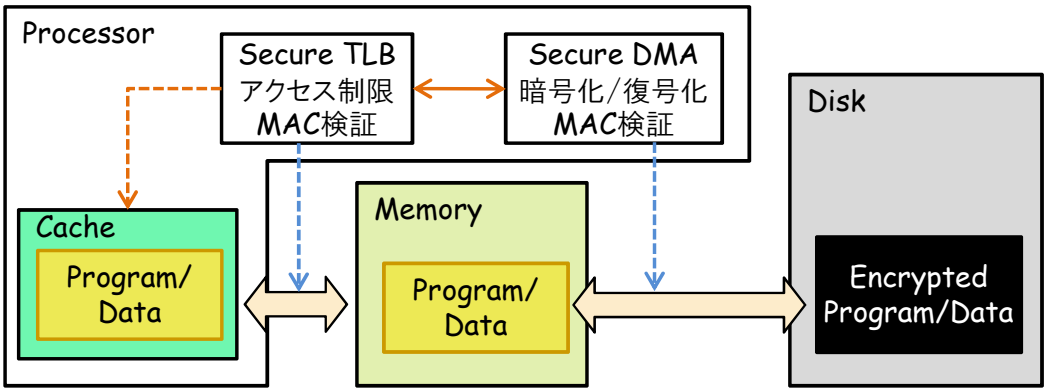

図 3.9: 耐ソフトウェアタンパ・プロセッサの全体図

- プログラム鍵/データ鍵の管理
- メモリ保護の拡張
- ディスク上/OS 管理下での暗号化
- ディスク上/OS 管理下にあったデータの MAC による検証
- Secure TLB と Secure DMA の協調

第4章　耐ソフトウェアタンパ・プロセッサを用いた遠隔認証

4.1　耐ソフトウェアタンパ･プロセッサの改良及び拡張

4.1.1　ページ保護の拡張

前章で説明した耐ソフトウェアタンパ・プロセッサの機能及びそれらの機能に互換性を持たせるための変更されたOSを用いることで，起動したプロセスを他のプロセスやOSによるソフトウェアタンパから保護することができる．しかし，先述したハードウェア的機能の追加やOSの変更だけでは信頼できないOS環境での耐ソフトウェアタンパ性が及ばない部分が存在する．プロセス毎に用意されるページテーブルは作成された後，Secure TLBによるアクセス権限チェックやMACを用いたページテーブルの改ざん検出を行うことで，OSに管理されながらもOSによる攻撃から保護できる．しかし，ページテーブルの生成からアドレスマッピングまでもOSが行うが，そのタイミングでのOSの攻撃に対しては保護対策がされていない．そのため，OSがマッピングされるアドレスを偽ることで攻撃を行うことができる．そのような攻撃は大きく2つに分けられる．

- 異なるプロセス間で同一物理アドレスをマッピング
- 論理アドレスに偽の物理アドレスをマッピング

図4.1に概要を示す．

1つ目は，図の上で示すように異なるプロセスのページの物理アドレスの割り当てに，同じ物理アドレスを割り当てることで行えるものである．この場合，もし，プロセスaが保護データを使用していてプロセスaのページのアクセス権限を特権を持ったプロセスや他プロセスがアクセスできないようにしたとしても，OSがプロセスbにプロセスaと同じ物理アドレスをマッピングするとプロセスbはプロセスaのアクセス権限に関係なくプロセスaのメモリ領域のデータを読みこみ/書き込みすることができてしまう．これではメモリ上にあるデータが保護されないこととなる．これを防ぐためにプロセッサにOSのア

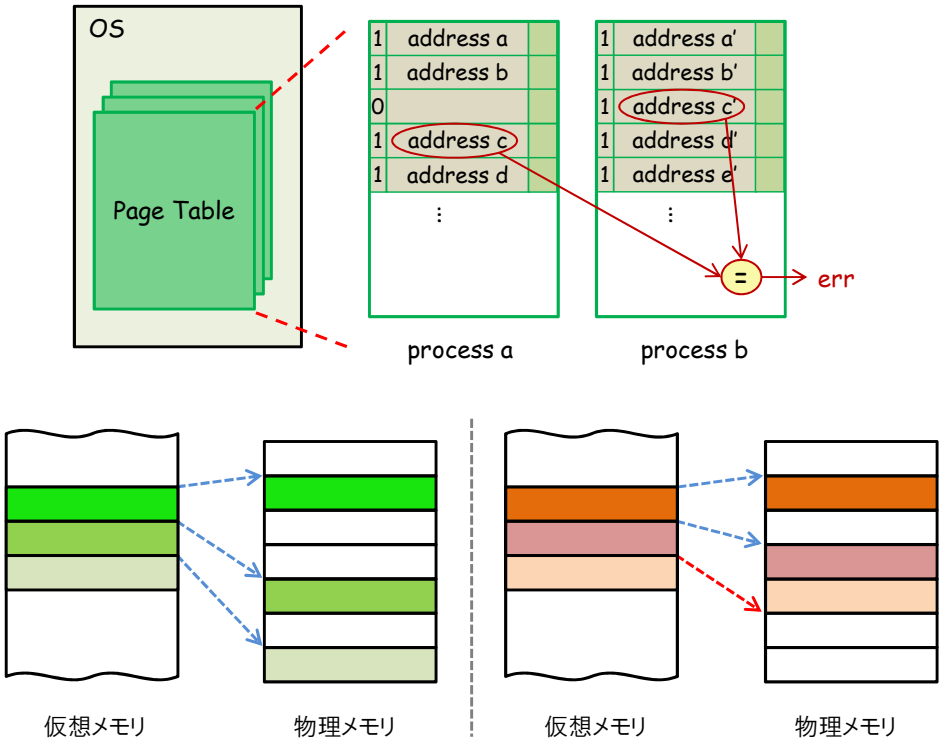

図 4.1: 物理アドレスのマッピング処理

ドレスマッピング時の不正を監視できる逆引きページテーブルを設け，マッピングをチェックさせる．

2 つ目は，図の下で示すように，あるプロセスの論理ページを物理ページにマッピングするとき左側のように各ページに正確に対応するページをマッピングしなければならないが，OS が悪意を持って右側のように (赤い点線のマッピング) 間違ったマッピングを行うとプログラムが正常に動かない可能性が生じる．これを防ぐために，OS がページの物理アドレスのマッピングを終了したらプロセスが実行の前に直接マッピングの内容を確認する機構を追加する．

4.1.2 計測機構の追加

耐ソフトウェアタンパ・プロセッサを遠隔認証に用いるため既存の機能に認証用データの計測機構を追加する必要がある．追加される機構は以下の 3 つである．

- Secure DMA にハッシュ演算機構を追加

- 起動時のカーネルのコア部の計測メカニズム
- 認証用データを集めるための検証モード追加

この 3 つの機構以外にプロセッサと Secure DMA に各々の専用非発揮性メモリを設け，計測データの保存などに用いる．

拡張 Secure DMA

まず，Secure DMA を拡張し Secure DMA にハッシュ演算機構を追加し，DMA が独立的にハッシュ値を取得できるようにする．また Secure DMA 内部に DMA コントローラ専用のメモリを設け，Secure DMA が取得したハッシュ値を Secure DMA 内部で保存できるようにする．このメモリには Secure DMA 外部から直接アクセスすることはできず，Secure DMA に参照を要求してハッシュ値を通知してもらう．参照要求にはアクセス鍵が必要で，この鍵は Secure DMA が取得したハッシュ値を構造化する際プロセッサから指定する．TPM 同様，Secrue DMA には製造時に公開鍵暗号方式の鍵ペアが組み込まれ，公開鍵には認証局から証明書発行しておく．通知の際には秘密鍵でディジタル署名を行う．

ハッシュ値の構造化は DMA 転送が終わった時点で行う．アクセス鍵は通常，耐ソフトウェアタンパ・プロセッサがプロセス毎に割り当てたプロセスのデータ鍵を用いる．プロセッサが検証モードの場合は検証プログラムのプログラム鍵を用いる．検証用モードついては後で詳しく述べる．アクセス鍵は Secure DMA 内部の変換テーブルを用いてメモリのアドレスと変換される．

図 4.2 にハッシュ値の構造化や通知に概要を示す．

以下このような機能の拡張された Secure DMA のことを拡張 Secure DMA と呼ぶ．

図 4.3 に拡張 Secure DMA が暗号化されたプログラムをメモリ上に展開するときの概要を示す．拡張 Secure DMA は Secure DMA と同様，プロセッサからプログラムの復号化鍵を DMA 転送の命令と一緒に受け取ると復号化を行いながらメモリ上に OS か指定する場所に展開する．しかし，拡張 Secure DMA は展開するデータのハッシュ値も同時に計算し，DMA コントローラ専用のメモリに保存しておく．転送が終わるとハッシュ値は対象と関連付けられアクセス鍵を用いて構造化された形で管理される．

起動時のカーネルのコア部の計測メカニズム

耐ソフトウェアタンパ・プロセッサは通常のプロセスに対し，Secure TLB を用いたページのアクセス保護やページデーブルの改ざんを防止することでメモ

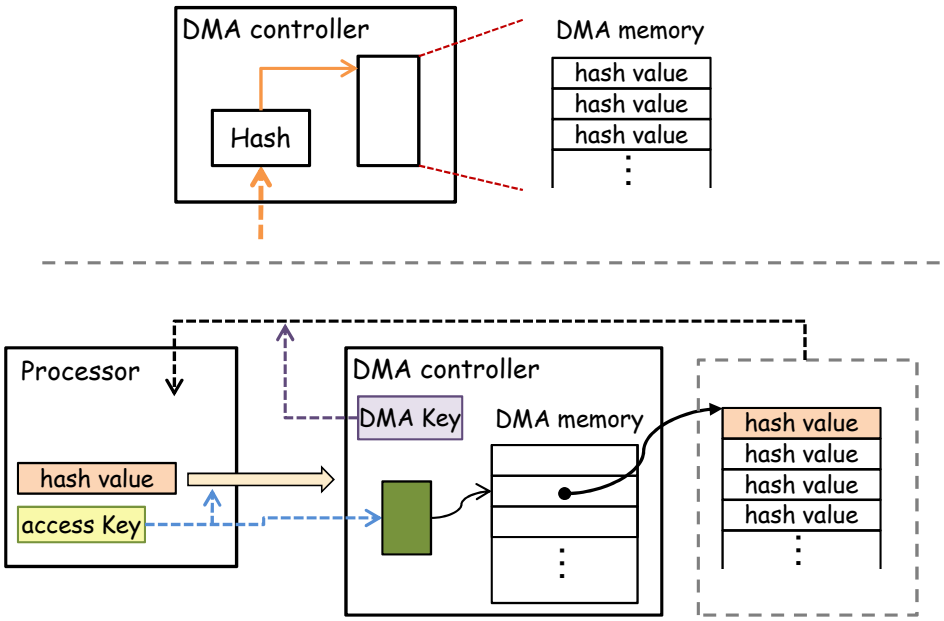

図 4.2: ハッシュ値の構造化及び通知

リ上のデータを保護し，さらに Secure DMA を用いることで耐ソフトウェアタンパ・プロセッサの保護機能が及ばない二次記憶などへのスワップ時の改ざんや解析を防ぐことができる．つまり，あるプロセスが保護するデータはデータの所有プロセス以外の OS を含む全てのプロセスが触ることができない．この特長を用いることで OS を信頼度が証明されていない状態で計測し，計測値の改ざんを防ぐことができる．しかし，信頼できない OS 上で OS を計測するためには計測する際に OS の関与による改ざんが行われないことを保証する必要がある．

本論文では，耐ソフトウェアタンパ・プロセッサを用いて OS を計測するメカニズムを提案する上，OS の構造を理想化する．理想化された OS は以下のような特長を持つ．

- OS はコアなカーネル部と様々なサービスを提供するためのモジュールで構成
- コアなカーネル部は PC が完全に起動した時点で全てメモリ上に展開され，その状態が維持される
- モジュールはサービスの種類や機能ごとに分割されている

コアなカーネル部とは，OS の駆動に必要な最低限な部分を指し，スケジュー

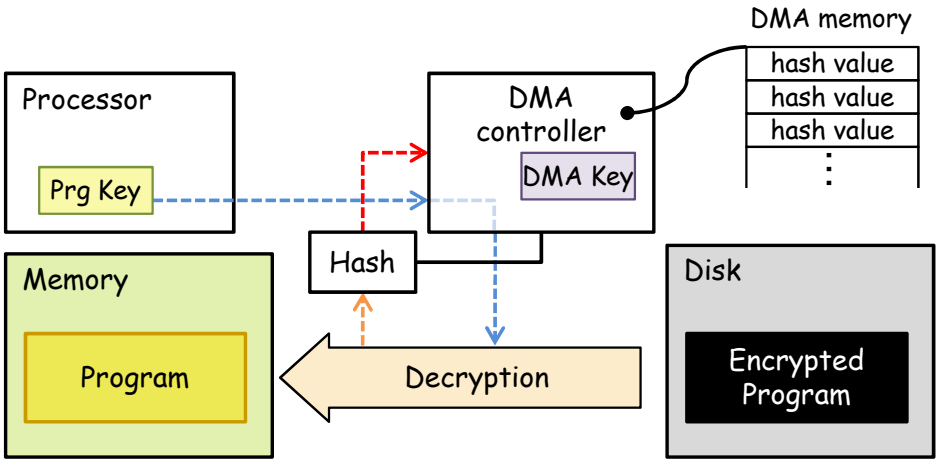

図 4.3: 拡張 Secure DMA

ラや割り込み処理などを含む．モジュールは各ドライブなどを指し，カーネルがサービスを提供する際にメモリに展開される．

図 4.4 に起動時の OS 計測の概要を示す．

通常 PC のブートプロセスはプロセッサがメモリを直接管理するリアルモードからカーネルがメモリの管理を行うプロテクトモードの順に起動される．リアルモードでは必要なプログラムやデータをプロセッサが直接メモリを指定して展開し，実行する．この際には Secure DMA による転送が行われないため，プロセッサがリアルモードで展開するデータのハッシュ値を直接取得する必要がある．取得したハッシュ値はプロセッサが管理する．プロテクトモードに入ると Secure DMA による転送が行われるので拡張 Secure DMA がハッシュ値を取得する．

最初の OS のブートプロセスが終わった時点でカーネルのコアな部分ははメモリに全て展開されており，またそのハッシュ値も取得された状態である．ブートが終わると拡張 Secure DMA はプロセッサからリアルモードの際にプロセッサが計測しておいたハッシュ値を受け取り，拡張 Secure DMA が持っているカーネルのコア部のハッシュ値と一緒に管理する．カーネルのコア部以外のモジュール部は OS がそれらのサービスを提供するときメモリに展開される．その転送を行うのは拡張 Secure DMA であるため，拡張 Secure DMA がハッシュ値を取得できる．

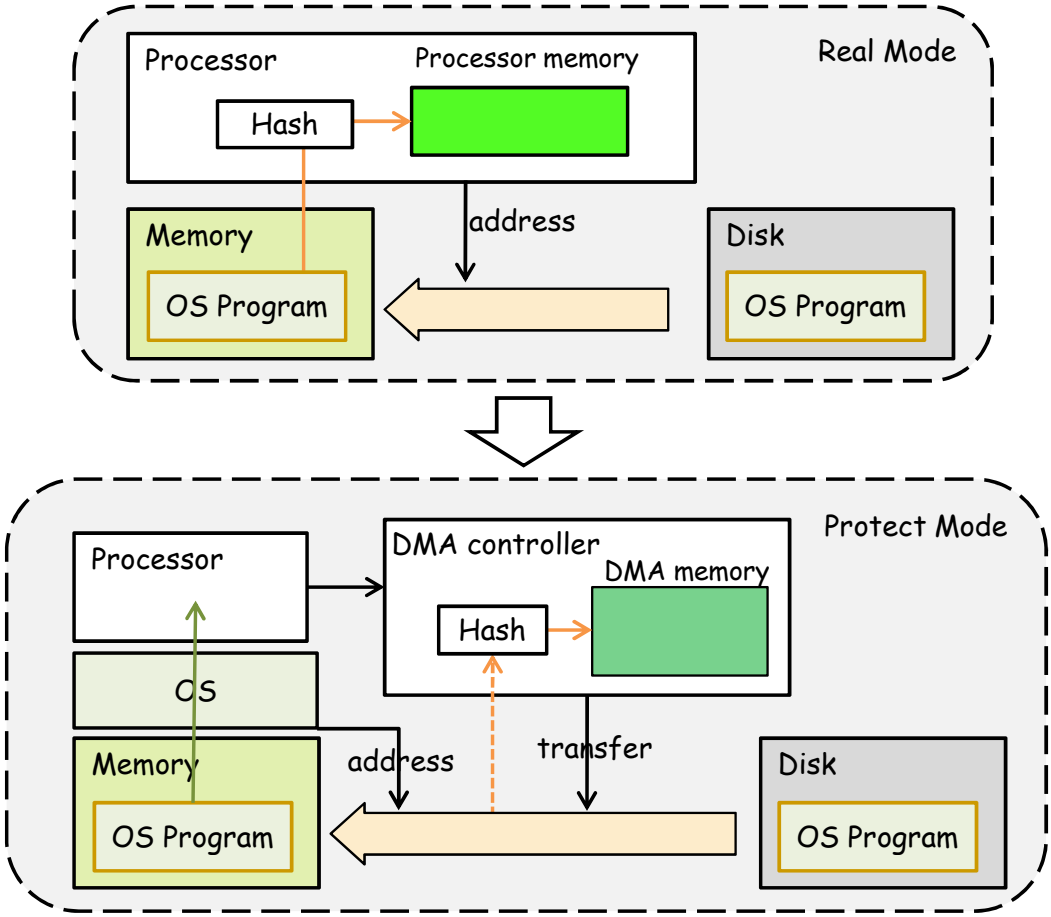

図 4.4: 起動時の OS 計測

検証モード

遠隔認証を行うためには 2.1.2 で述べたようにサーバー側にクライアント側のプラットフォームの情報を伝える必要がある．プラットフォーム情報を集めることをサポートするため，検証モードという特殊なモードを定義する．プロセッサは検証モードの状態ではカーネルが様々サービスを提供するため読み込むモジュールを，もし，そのモジュールがすでにメモリに展開されている状態であるとしても，拡張 Secure DMA で転送を再度行う．これによりモジュールの最も最新の状態のハッシュ値が取得でき，前もって展開されていたモジュールの改ざんから保護できる．

検証モードに入るのはサーバーが配布した検証用プログラムの指定によって行う．

4.2 耐ソフトウェアタンパ・プロセッサを用いた遠隔認証の流れ

4.2.1 検証プログラム

サーバー側はクライアント側のプラットフォームのコンポーネントの内，認証対象であるものを予め定めて，それらの計測をクライアント側上で行う検証用プログラムを作成する．認証対象をサーバーが指定してそれらの情報のみを用いることでプラットフォーム認証のため行われる情報収集のコストを減らすことができる．

サーバー側は認証対象を定めるための検証プログラムを作成し，クライアントがそのプログラムを実行することでサーバーが要求する情報を集めることが可能になるようにする．検証プログラムには，認証を行う対象のリストが入っており，それらをクライアント側で実行させることでクライアント側の拡張 Secure DMA にてハッシュ値を取得する．

検証プログラムにはクライアントのプラットフォーム情報をサーバーに通知するためのレポート様式が指定されており，サーバーその様式以外の通知は不正なものとみなす．また，検証プログラムにはランダムなプログラム番号か指定されており，クライアント側が検証プログラムを区別するときに使う．

検証プログラムの配布は暗号化を用い 3.2.1 節で述べたような方法で配布する．このとき用いられるプログラム鍵は，認証成功後に配布される保護すべきプログラムやデータの暗号化に用いられる鍵とは無関係である．

4.2.2 遠隔認証の流れ

ここではあるコンテンツを配布するときの認証の流れについて説明する．配布するコンテンツは著作権を持つもので保護すべきものであるとする．

図 4.5 に遠隔認証の流れの概要を示す．

サーバー側はコンテンツを配布する前にクライアント側のプラットフォームの認証を行い，コンテンツがクライアントにより不正に扱われないことを保証してもらう．認証の対象となるものを検証プログラムで指定し，コンテンツの配布の前にクライアントに送る．検証プログラムは実行されると，まず，プロセッサを検証モードにする．その後，検証プログラムのプログラム鍵をアクセス鍵としてハッシュ値を保存する領域を拡張 Secure DMA のメモリに確報する．プログラムが持っている認証対象のリスト順に拡張 Secure DMA にメモリへ転送を行い，それらのハッシュ値を拡張 Secure DMA が取得する．それらのハッシュ値は，拡張 Secure DMA がもっているブートの際計測しておいたカー

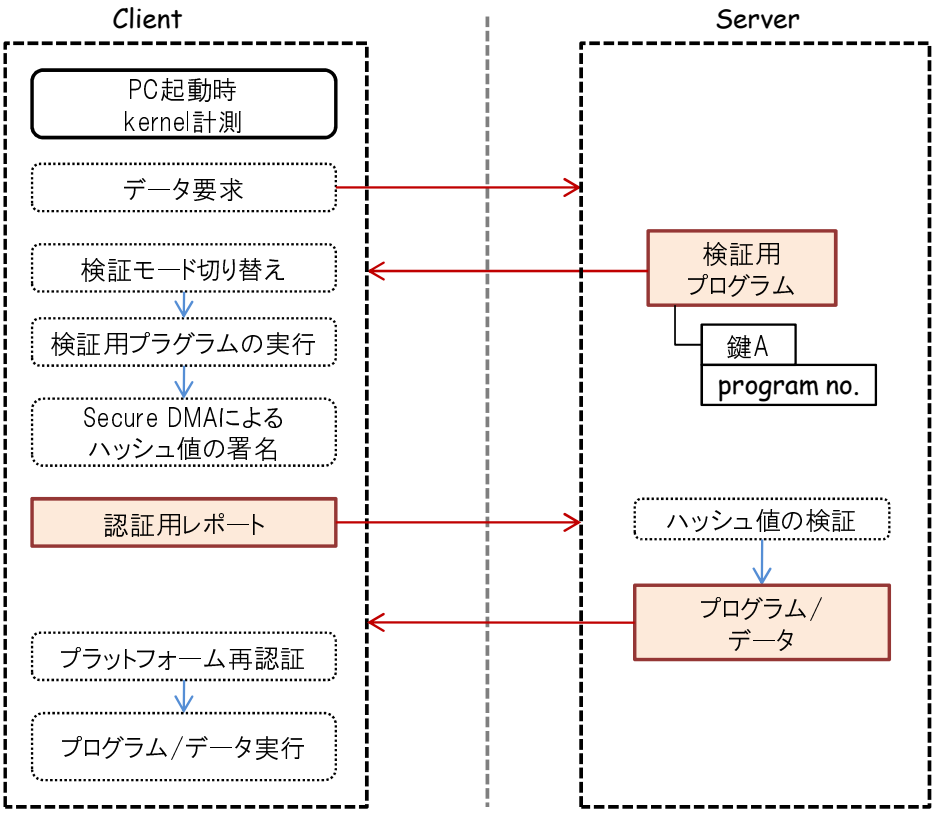

図 4.5: 遠隔認証の流れの概要

ネルのコア部のハッシュ値とともに，拡張 Secure DMA にてディジタル署名を行い，検証プログラムが指定する通知様式でレポートを作成してプログラム鍵で暗号化してサーバーに通知する．

通知する際，クライアント側は拡張 Secure DMA がディジタル署名を行ったハッシュ値をプログラム鍵で暗号化と MAC の生成を行って保存しておく．また，プログラム鍵は検証プログラムのプログラム番号と関連付け，検証プログラムのプロセスが終了した後でもプロセッサ内部に保存しておく．保存されるメモリはプロセッサ専用のもので非発揮性なものをプロセッサ内部に搭載し，それを用いる．クライアント側で行われる計測から通知までの流れを図 4.6 に示す．

クライアント側で計測されたプラットフォームの情報の完全性は耐ソフトウェアタンパ・プロセッサのアクセス制限による検証プログラムのデータ保護と拡張 Secure DMA のメモリへのアクセス制限で確立できる．

サーバーは通知されたレポートの検証を行い，クライアント側のプラットフォームの情報が期待値と一致したら，クライアント側のプラットフォームを

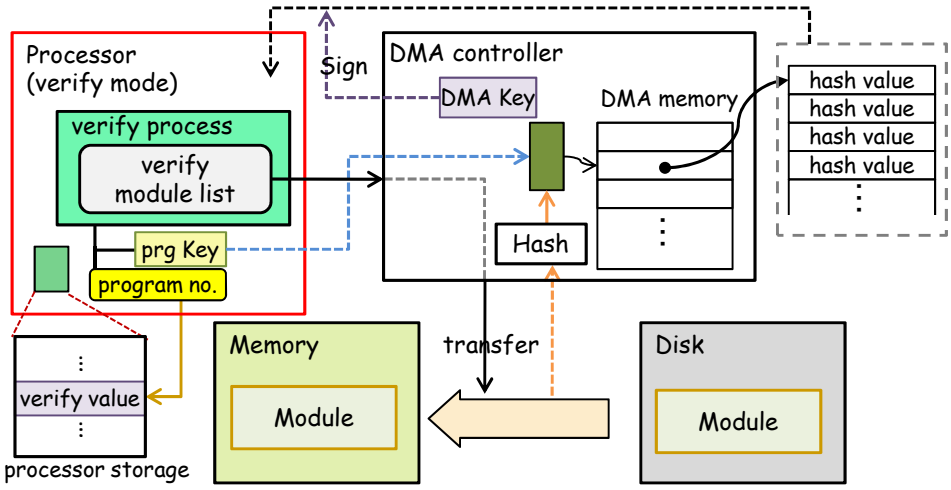

図 4.6: クライアント側のプラットフォーム計測及び通知

安全なものとみなし，認証成功とする．認証が終了しプラットフォームの安全性が確認できたら，サーバーはクライアントにコンテンツを配布する．このコンテンツは検証プログラムに用いたプログラム番号の情報を付けて配布する．また，コンテンツの暗号化には検証プログラムの暗号化に用いられた鍵とは別の鍵で暗号化を行う．

4.3 情報漏洩防止の仕組み

先述した遠隔認証を行うことで，サーバー側はクライアント側のプラットフォームの情報を検証することができ，その安全性を確認してから機密データなどの保護データを配布することができる．しかし，先述の遠隔認証だけでは 2.1.2 節で述べた

- 認証されたプラットフォームの情報通りの環境で動いていること

この条件に対する確認までは及ばない．この環境の同一性を保証するため，耐ソフトウェアタンパ・プロセッサでは，保護データを実行する前に前節で述べたプロセッサ内部に保存しておく認証に用いた拡張 Secure DMA がディジタル署名を行ったハッシュ値を用いて，クライアント内部で再認証を行う．

プロセッサは，まず，保護データに付けられている検証プログラムのプログラム番号を用いて検証プログラムのプログラム鍵と認証データを取り出し，MAC の検証によるデータの改ざんをチェックする．次に保護データを実行するための OS のサービスなどを用意する．この際，拡張 Secure DMA によるハッ

シュ値の計測が行われる．実行環境が整えたタイミングで拡張 Secure DMA には認証を行うため集めた際の対象と同じ対象のハッシュ値が全て揃っていることになる．プロセッサな拡張 Secure DMA に署名してもらい，その値を読み込み，保存しておいた値と比較を行う．一致したら保護データの実行を開始する．不一致したら保護データの実行を拒否し，再遠隔認証を要求する．

4.3.1 情報漏洩防止プラットフォームとの協調

先述した拡張された耐ソフトウェアタンパ・プロセッサを用いた遠隔認証を行うことでクライアント側のプラットフォームが保護データを不正に扱わないものであるかの検証を行うことができる．プラットフォームの認証に用いる安全性の根拠はサーバーが予め安全であろうと期待するプラットフォームの情報との比較である．しかし，この手法ではプラットフォームやアプリケーションの潜在的なバグなどによる情報漏えいの場合までの対策にはならない．そのような潜在的なバグなどによる情報漏えいまで完全に防ぐため，本研究室で提案した情報漏洩防止プラットフォーム [11] との協調について述べる．情報漏洩防止プラットフォームとは

4.4 耐ソフトウェアタンパ・プロセッサを用いた遠隔認証のまとめ

耐ソフトウェアタンパ・プロセッサを用いた遠隔認証では，既存の耐ソフトウェアタンパ・プロセッサの機能を拡張することで遠隔認証に用いている．プロセッサと Secure DMA に各々の非発揮性専用メモリを設け，さらに Secure DMA には転送を行いながら，ハッシュ値を取得する機構を追加することでプラットフォームの情報の取得する上で OS の改ざんを防ぐ．また，サーバーは自ら検証の対象を指定する検証プログラムを作成し，保護データの配布に前もってプラットフォームの認証を行うことをサポートする．耐ソフトウェアタンパ・プロセッサのアクセス制限により認証用のデータの改ざんを防ぎ，保護データの実行時の再認証を行うことで，なりすましを防ぐことができる．

第5章　おわりに

5.1　まとめ

本論文では，商業用コンテンツなど保護すべきデータをインターネットで配布する際，それらのデータがクライアント側で不正なことが行われないことを検証するための遠隔認証の必要性について述べた．また，遠隔認証の概要について説明し，従来手法である TPM を用いた遠隔認証について説明し，その問題点について述べた．そこて，本研究室で提案したセキュア・プロセッサである耐ソフトウェアタンパ・プロセッサを拡張し，遠隔認証を行う手法を提案した．拡張した機能は

- プロセッサ及び Secure DMA 内部に各々の専用非発揮性メモリの追加
- Secure DMA にハッシュ演算機構の追加

であり，サーバー側は遠隔認証をサポートするため，検証プログラムを保護データの配布の前に配布する．耐ソフトウェアタンパ・プロセッサは検証プルグラムを実行することで認証用の情報を集め，サーバーに通知する．また，その際，プロセッサは認証用の情報を保存しておき，保護データの実行時に再認証を行うことで遠隔認証後のプラットフォームの改ざんやなりすましなどをチェックすることができる．

5.2　今後の課題

本論文で提案した耐ソフトウェアタンパ・プロセッサを用いた遠隔認証を行うことで保護データの不正な扱いを防ぐことができる．しかし，各々の機能について詳細な部分までの検討や評価はまだ行っていない．さらにより効率な手法についての検討を行う余地がある．たとえば，拡張 Secure DMA のハッシュ値の扱いやプロセッサとのハッシュ値の通信などに関する詳細などがある．

関連図書

[1] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (Im)possibility of Obfuscating Programs. In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp. 1–18. Springer-Verlag, 2001.

[2] David Lie, Chandramohan A. Thekkath, and Mark Horowitz. Implementing an untrusted operating system on trusted hardware. In *Proceedings of ACM Symposium on Operating Systems Principles*, 2003.

[3] Seiji Munetoh. Practical integrity measurement and remote verification for linux platform. IBM Tokyo Research Laboratory.

[4] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pp. 16–16, Berkeley, CA, USA, 2004. USENIX Association.

[5] Dries Schellekens, Brecht Wyseur, and Bart Preneel. Remote attestation on legacy operating systems with trusted platform modules. *Sci. Comput. Program.*, Vol. 74, No. 1-2, pp. 13–22, 2008.

[6] G. Edward Suh, Dwaine Clarke, Blaise Gassend, Marten van Dijk, and Srinivas Devadas. AEGIS: Architecture for tamper-evident and tamper-resistant processing. In *International Conference on Supercomputing*, 2003.

[7] Trusted Computing Group. *https://www.trustedcomputinggroup.org/*.

[8] Trusted Computing Group. *TCG Specification Architecture Overview*.

[9] Trusted Computing Group. *TPM Specification Version 1.2 Revision 103*.

[10] Trusted Computing Group. *Trusted Platfrom Module(TPM) Summary*.

[11] 横田侑樹, 塩谷亮太, 五島正裕, 坂井修一. 情報漏洩防止プラットフォーム. 電子情報通信学, 信学技報, vol. 109, no. 237, pp. 7–12, 2009.

[12] 橋本幹生, 春木洋美. 敵対的な OS からソフトウェアを保護するプロセッサアーキテクチャ. 情報処理学会論文誌 コンピューティングシステム, Vol. 45, No. 3, March 2004.

[13] 経済産業省, ECOM, NTT データ経営研究所. 電子商取引に関する市場規模・実態調査.

[14] 上杉忠興, 圷毅, 宗藤誠治, 吉濱佐知子. Trusted network connect - tpm の利用管理技術の動向. 情報処理学会, Vol. 48, No. 11, pp. 1232–1241, 2007.

[15] 情報処理相互運用技術協会. TCG(Trusted Computing Group) 動向調査, 2005.

[16] 清水一人, 入江英嗣, 五島正裕, 坂井修一. レジスタ間接分岐ターゲット・フォワーディング. 情報処理学会研究報告 2007 no.17, pp. 239–244, 2007.

発表文献

1. 情報漏洩防止のためのプラットフォーム認証
   文 栄光，塩谷 亮太，五島 正裕，坂井 修一
   電子情報通信学会　CPSY2009-29，vol.109, no 237, pp. 13-18, 2009

2. 情報漏洩防止のためのプラットフォーム認証
   文 栄光，塩谷 亮太，五島 正裕，坂井 修一
   情報処理学会創立 50 周年記念全国大会
   （投稿中）

謝辞

非常に多くの方々から多大なご指導，ご協力，励ましを頂き，本論文を完成させることができました．この場を借りて，感謝の意を表したいと思います．

指導教官である坂井修一教授には修士の 2 年間に渡って多くのご指導，ご助言を頂きました．いろいろとご心配をおかけしたこともありましたが，非常に細かく気を遣ってくださり，様々な面で助けられました．本当にありがとうございました．

また，五島正裕助教授からも，大変多くのご指導を頂きました．いろいろと至らない点の多い私に根気強く指導してくださいました．常にネガティブな思考の私に対し，ポジティブな考えで研究を直接引っ張っていただきました．本研究がこの形になったのは，五島助教授のおかげです．ありがとうございました．

塩谷亮太博士，金 大雄氏，横田侑樹氏をはじめ，セキュリティ・グループのメンバーの皆さまには，ミーティングにおける議論を通して，貴重なご意見を頂きました．特に，横田侑樹の尽力なくしては，この研究は完成しなかったと思います．また，金 大雄氏には，同じ韓国からの留学生先輩として様々な面でお世話になりました．

八木原晴水さん，長谷部環さん，伊世知代さんには，研究室における設備の導入や各種事務手続きなど，研究室で過ごすための様々なご支援を頂きました．特に，八木原晴水さんには様々な面での相談まで頂き本当にお世話になりました．

また，ここでは紹介しきれなかった研究室のメンバーの皆様にも様々な形でお世話になりました．本当にありがとうございます．