(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4498735号
(P4498735)

(45)発行日　平成22年7月7日(2010.7.7)　(24)登録日　平成22年4月23日(2010.4.23)

(51)Int.Cl.　F I

G06F 12/14	(2006.01)	G06F	12/14	510A
G06F 21/24	(2006.01)	G06F	12/14	510D
G06F 21/22	(2006.01)	G06F	12/14	510E
		G06F	12/14	540A
		G06F	12/14	540P

請求項の数 38　(全 59 頁)　最終頁に続く

(21)出願番号　特願2003-504201 (P2003-504201)
(86)(22)出願日　平成14年6月7日(2002.6.7)
(65)公表番号　特表2004-537786 (P2004-537786A)
(43)公表日　平成16年12月16日(2004.12.16)
(86)国際出願番号　PCT/US2002/018180
(87)国際公開番号　WO2002/101504
(87)国際公開日　平成14年12月19日(2002.12.19)
審査請求日　平成17年6月6日(2005.6.6)
(31)優先権主張番号　60/296,958
(32)優先日　平成13年6月8日(2001.6.8)
(33)優先権主張国　米国(US)
(31)優先権主張番号　60/296,957
(32)優先日　平成13年6月8日(2001.6.8)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
ＨＥＷＬＥＴＴ－ＰＡＣＫＡＲＤ　ＣＯＭＰＡＮＹ
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　３０００
(74)代理人　110000039
特許業務法人アイ・ピー・エス
(72)発明者　ウイリアム・エス・ウォーリー・ジュニア
アメリカ合衆国コロラド州センテニアル　サウスアンデスサークル７１５７
(72)発明者　ジョン・エス・ウォーリー
アメリカ合衆国コロラド州フォートコリンズ　マリナーレーン３６１８

最終頁に続く

(54)【発明の名称】オペレーティングシステムおよびカスタマイズされた制御プログラムとインタフェースする安全なマシンプラットフォーム

(57)【特許請求の範囲】

【請求項１】

複数の特権レベルと、非特権命令と、非特権レジスタと、特権命令と、特権レジスタとを提供するハードウェアと、このハードウェアにおいて実行されるソフトウェアから構成されるプラットフォームインタフェースであって、

前記ハードウェアの命令セットのアーキテクチャによって提供され、あらゆる前記特権レベルで実行されるプロセスにアクセスされ得る非特権命令および非特権レジスタと、

オペレーティングシステムおよびカスタム制御プログラム（１５０６）のいずれかの制御プログラムにより起動されると、前記制御プログラムの特権レベルよりも高い特権を与えられる特権レベルで実行され、前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開することなく、特権命令および特権レジスタをシミュレーションすることなく、ハードウェアリソースの動作制御を提供する呼び出し可能なソフトウェアサービス（１５０４，１７０２）と、

前記複数の特権レベルごとに、前記プロセスの実行に使用されるメモリを複数の第１の領域に分割する第１の分割手段と、

並列に実行される複数のオペレーティングシステムごとに、前記複数の領域をさらに複数の第２の領域に分割する第２の分割手段と

を備え、

前記呼び出し可能なソフトウェアサービスは、

前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開せず、特

権命令および特権レジスタをシミュレーションしないハードウェアリソースの動作制御を、前記制御プログラムに提供するプラットフォーム管理サービス（１７１０）と、
内部で生成された秘密データを使用し、内部の秘密データを外部の計算エンティティに公開せずに、前記内部の秘密データを管理するセキュリティ管理サービス（１７１２）と
を含む
プラットフォームインタフェース。
【請求項２】
前記ハードウェアは、
前記ハードウェアを制御する１つ以上のファームウェアルーチンを提供するファームウェアインタフェースと、
少なくとも４つの特権レベルと、
Ｉ／Ｏ操作によってアクセスできる物理メモリアドレスを制限する手段と
をさらに備える
請求項１に記載のプラットフォームインタフェース。
【請求項３】
前記プラットフォーム管理サービス（１７１０）は、
前記制御プログラムにより起動されて、特権レベルの操作を使用する機能、プラットフォームのリソースを利用する機能、セキュリティの対象となる機能を実行するプラットフォームサービス（１７１６）と、
システムリソースの相互に隔離したパーティションを示すドメインの生成と、環境設定と、再初期化と、シャットダウンとを提供するドメイン制御サービス（１７１８）と、
論理ＣＰＵおよび物理ＣＰＵの環境設定とスケジューリングとを提供するプロセッサ制御サービス（１７２０）と、
システムリソースの割り当てを少なくとも示すプラットフォームポリシーを特定し、保存し、前記プラットフォームインタフェースに適用するプラットフォームポリシー制御サービス（１７２２）と、
ドメイン内の論理ＣＰＵ間のイベントおよびドメイン間のイベントの伝達と、ドメイン間でのリソースの共有と、ドメイン間での高速ネットワーク接続またはデータ交換のためのデータ転送を行うドメイン間およびドメイン内サービスと
を含む
請求項１に記載のプラットフォームインタフェース。
【請求項４】
前記セキュリティ管理サービス（１７１２）は、
前記制御プログラムまたはユーザアプリケーションプログラムによって起動され、システムの最も高い特権レベルで実行される所定のプロセスにのみ利用可能な秘密データを使用する機能を提供するリポジトリサービス（１７２６）と、
ドメインが、プラットフォームのみにアクセスし得る特権レベルを示すタグを、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、ディスパッチ可能オブジェクト、または、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、およびディスパッチ可能オブジェクトを含むシステムオブジェクトに関連付け、前記タグを使用して、プラットフォームサービスおよびリポジトリサービスへのアクセスを認証する手段を提供する呼び出し側認証サービス（１７２８）と
を含む
請求項１に記載のプラットフォームインタフェース。
【請求項５】
前記リポジトリサービスは、
一方向ハッシュ関数と、メッセージ認証コードと、デジタル署名と、乱数とを計算する機能を含む１つまたは２つ以上の暗号機能を使用したデータの暗号化および復号を提供する暗号サービスと、

データサービスであって、
前記制御プログラムとこのデータサービスとの間の特定のインタフェースのみを通じて、アクセスおよび変更されるログと、
前記リポジトリサービスが操作可能なセキュリティポリシーを保持するポリシーデータベースと、
システムコンポーネントのうち、このデータサービスのみからからアクセスされる前記秘密データを使用するサービスと
を含むデータサービスと
を含む請求項４に記載のプラットフォームインタフェース。
【請求項６】
コンピュータ読み取り可能媒体にコード化された
請求項１に記載のプラットフォームインタフェースを構成するソフトウェアをコンピュータにより実行させる
コンピュータ命令。
【請求項７】
少なくとも１つのユーザレベルプログラムと、
少なくとも１つのオペレーティングシステムと、
請求項１に記載のプラットフォームインタフェースと
を備えるコンピュータシステム。
【請求項８】
オペレーティングシステムおよびカスタム制御プログラム（１５０６）のいずれかの制御プログラムのホストコンピュータとして機能するコンピュータシステムであって、
複数の特権レベルと、非特権命令と、非特権レジスタと、特権命令と、特権レジスタとを提供するハードウェアプラットフォームと、
前記特権命令および前記特権レジスタの一方または双方を必要とする操作を実行する前記制御プログラムにより起動されると、前記制御プログラムが実行される特権レベルよりも高い特権を与えられる特権レベルで実行され、前記制御プログラムは、特権命令および特権レジスタをシミュレーションすることなく、特権命令および特権レジスタを前記制御プログラムに公開することなく、ハードウェアリソースの動作制御を提供する呼び出し可能なソフトウェアサービス（１５０４，１７０２）と、
最も高い特権レベルで実行されるプラットフォームカーネルと、
前記複数の特権レベルごとに、プロセスの実行に必要なメモリを複数の第１の領域に分割する第１の分割手段と、
並列に実行される複数のオペレーティングシステムごとに、前記複数の領域をさらに複数の第２の領域に分割する第２の分割手段と
を備え、
前記呼び出し可能なソフトウェアサービスは、
前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開せず、特権命令および特権レジスタをシミュレーションしないハードウェアリソースの動作制御を、前記制御プログラムに提供するプラットフォーム管理サービス（１７１０）と、
内部で生成された秘密データを使用し、内部の秘密データを外部の計算エンティティに公開せずに、前記内部の秘密データを管理するセキュリティ管理サービス（１７１２）と
を含む
コンピュータシステム。
【請求項９】
前記プラットフォームカーネルは、前記ソフトウェアサービスの実行の起動前に、前記ソフトウェアサービスに対して、ソフトウェアサービスが呼び出し可能かアクセス認証する
請求項８に記載のコンピュータシステム。
【請求項１０】

前記特権命令および前記特権レジスタを隠蔽し、前記ハードウェアの命令セットアーキテクチャによって提供される前記非特権命令および前記非特権レジスタと、前記呼び出し可能なソフトウェアサービスとにインタフェースとを提供するプラットフォームインタフェース
をさらに備える請求項８に記載のコンピュータシステム。
【請求項１１】
前記ハードウェアは、
前記ハードウェアを制御するファームウェアルーチンを提供するファームウェアインタフェースと、
少なくとも４つの特権レベルと、
Ｉ／Ｏ操作によってアクセスできる物理メモリアドレスを制限する手段と
をさらに備える
請求項８に記載のコンピュータシステム。
【請求項１２】
前記プラットフォーム管理サービスは、
前記制御プログラムにより起動されて、特権レベルの操作を利用する機能、プラットフォームのリソースを利用する機能、セキュリティの対象となる機能を実行するプラットフォームサービス（１７１６）と、
システムリソースの相互に隔離したパーティションを示すドメインの生成と、環境設定と、再初期化と、シャットダウンとを提供するドメイン制御サービス（１７１８）と、
論理ＣＰＵおよび物理ＣＰＵの環境設定とスケジューリングとを提供するプロセッサ制御サービス（１７２０）と、
システムリソースの割り当てを少なくとも示すプラットフォームポリシーを特定し、保存し、前記プラットフォームインタフェースに適用するプラットフォームポリシー制御サービス（１７２２）と、
ドメイン内の論理ＣＰＵ間のイベントおよびドメイン間のイベントの伝達と、ドメイン間でのリソースの共有と、ドメイン間での高速ネットワーク接続またはデータ交換のためのデータ転送を行うドメイン間およびドメイン内サービスと
を含む請求項８に記載のコンピュータシステム。
【請求項１３】
前記セキュリティ管理サービス（１７１２）は、
前記制御プログラムまたはユーザアプリケーションプログラムによって起動され、システムの最も高い特権レベルで実行される所定のプロセスにのみ利用可能な秘密データを使用する機能を提供するリポジトリサービス（１７２６）と、
ドメインが、プラットフォームのみにアクセスし得る特権レベルを示すタグを、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、ディスパッチ可能オブジェクト、または、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、およびディスパッチ可能オブジェクトを含むシステムオブジェクトに関連付け、前記タグを使用して、プラットフォームサービスおよびリポジトリサービスへのアクセスを認証する手段を提供する呼び出し側認証サービス（１７２８）と
を含む請求項８に記載のコンピュータシステム。
【請求項１４】
前記リポジトリサービスは、
一方向ハッシュ関数と、メッセージ認証コードと、デジタル署名と、乱数とを計算する機能を含む１つまたは２つ以上の暗号機能を使用したデータの暗号化および復号を提供する暗号サービスと、
データサービスであって、
前記制御プログラムとこのデータサービスとの間の特定のインタフェースのみを通じて、アクセスおよび変更されるログと、

前記リポジトリサービスが操作可能なセキュリティポリシーを保持するポリシーデータベースと、
システムコンポーネントのうち、このデータサービスのみからからアクセスされる前記秘密データを使用するサービスと
を含むデータサービスと
を含む請求項１３に記載のコンピュータシステム。
【請求項１５】
ＴＬＢ、ＶＨＰＴ、および他のページテーブルへの変換を挿入する特権動作を実行するメモリ管理メカニズムと、
ディスパッチ可能オブジェクトおよび論理ＣＰＵへの状態の保存、および、ディスパッチ可能オブジェクトおよび論理ＣＰＵからの状態の復元を処理するディスパッチメカニズムと、
フォルト、トラップ、特別なハードウェアイベント、およびＳＰ発生イベントを取り扱う例外メカニズムと、
インタラプトが発生すると、ＳＰは、予め構成された特別の実行環境で、関連付けられたハンドラルーチンを起動し、高速のインタラプト応答するインタラプトメカニズムと、
マシンレジスタの集まりを使用して、デバッグおよび性能監視を援助するデバッグおよび監視メカニズムと、
暗号、復号、鍵管理を行う暗号メカニズムおよび暗号記憶メカニズムと、
秘密データと、秘密データに関連付けられたアクセス方法を含むリポジトリメカニズムおよびリポジトリ記憶メカニズムと
を含むプラットフォームカーネルメカニズムを提供する内部インタフェースと
をさらに含む請求項８に記載のコンピュータシステム。
【請求項１６】
オペレーティングシステムを実行する前に前記オペレーティングシステムを初期化するように実行される前記ファームウェアルーチン、および、前記ソフトウェアサービスを認証し、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認するブートプロセス
をさらに含む請求項１１に記載のコンピュータシステム。
【請求項１７】
前記ブートプロセスは、オペレーティングシステムのルーチン、および、ユーザアプリケーションプログラムを含むファームウェアルーチン、または、ソフトウェアサービスの実行前に、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認する
請求項１６に記載のコンピュータシステム。
【請求項１８】
複数の特権レベルと、非特権命令と、非特権レジスタと、特権命令と、特権レジスタとを有するハードウェアプラットフォームと、プラットフォームインタフェースとを含むコンピュータシステムをセキュリティ保護する方法であって、
前記プラットフォームインタフェースは、
前記複数の特権レベルごとに、プロセスの実行に使用されるメモリを複数の第１の領域に分割する第１の分割を行うことと、
並列に実行される複数のオペレーティングシステムごとに、前記複数の領域をさらに複数の第２の領域に分割する第２の分割とを行うことと、
オペレーティングシステムおよびカスタム制御プログラム（１５０６）のいずれかの制御プログラムにより起動されると、前記制御プログラムの特権レベルよりも高い特権を与えられる特権レベルで実行され、前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開することなく、特権命令および特権レジスタをシミュレーションすることなく、ハードウェアリソースの動作制御するソフトウェアサービスを提供することと、

非特権命令および非特権レジスタをオペレーティングシステム、および、カスタム制御プログラムに公開し、前記ソフトウェアサービスにインタフェースを提供し、前記特権命令および前記特権レジスタを隠蔽することと、
前記制御プログラムを起動することと
をコンピュータに実行させ、
前記なソフトウェアサービスは、
前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開せず、特権命令および特権レジスタをシミュレーションしないハードウェアリソースの動作制御を、前記制御プログラムに提供するプラットフォーム管理サービス（１７１０）と、
内部で生成された秘密データを使用し、内部の秘密データを外部の計算エンティティに公開せずに、前記内部の秘密データを管理するセキュリティ管理サービス（１７１２）と
を含む
コンピュータシステムをセキュリティ保護する方法。
【請求項１９】
前記ハードウェアプラットフォームは、
前記ハードウェアを制御する１つ以上のファームウェアルーチンを提供するファームウェアインタフェースと、
少なくとも４つの特権レベルと、
Ｉ／Ｏ操作によってアクセスできる物理メモリアドレスを制限する手段と
をさらに備える
請求項１８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２０】
前記プラットフォーム管理サービス（１７１０）は、
前記制御プログラムにより起動されて、特権レベルで行う操作する機能、マシンリソースを利用する機能、セキュリティに依存する機能を実行するプラットフォームサービス（１７１６）と、
システムリソースの相互に隔離したパーティションを示すドメインの生成と、環境設定と、再初期化と、シャットダウンとを提供するドメイン制御サービス（１７１８）と、
論理ＣＰＵおよび物理ＣＰＵの環境設定とスケジューリングとを提供するプロセッサ制御サービス（１７２０）と、
システムリソースの割り当てを少なくとも示すプラットフォームポリシーを特定し、保存し、前記プラットフォームインタフェースに適用するプラットフォームポリシー制御サービス（１７２２）と、
ドメイン内の論理ＣＰＵ間のイベントおよびドメイン間のイベントの伝達と、ドメイン間でのリソースの共有と、ドメイン間での高速ネットワーク接続またはデータ交換のためのデータ転送を行うドメイン間およびドメイン内サービスと
を含む請求項１８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２１】
前記セキュリティ管理サービス（１７１２）は、
前記制御プログラムまたはユーザアプリケーションプログラムによって起動され、システムの最も高い特権レベルで実行される所定のプロセスにのみ利用可能な秘密データを使用する機能を提供するリポジトリサービス（１７２６）と、
ドメインが、プラットフォームのみにアクセスし得る特権レベルを示すタグを、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、ディスパッチ可能オブジェクト、または、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、および、ディスパッチ可能オブジェクトを含むシステムオブジェクトに関連付け、前記タグを使用して、プラットフォームサービスおよびリポジトリサービスへのアクセスを認証する手段を提供する呼び出し側認証サービス（１７２８）と
を含む請求項１８に記載のコンピュータシステムをセキュリティ保護する方法。

【請求項２２】
前記リポジトリサービスは、
一方向ハッシュ関数と、メッセージ認証コードと、デジタル署名と、乱数とを計算する機能を含む１つまたは２つ以上の暗号機能を使用したデータの暗号化および復号を提供する暗号サービスと、
データサービスであって、
前記制御プログラムとこのデータサービスとの間の特定のインタフェースのみを通じて、アクセスおよび変更されるログと、
前記リポジトリサービスが操作可能なセキュリティポリシーを保持するポリシーデータベースと、
システムコンポーネントのうち、このデータサービスのみからからアクセスされる前記秘密データを使用するサービスと
を含むデータサービスと
を含む請求項２１に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２３】
ＴＬＢ、ＶＨＰＴ、および他のページテーブルへの変換を挿入する特権動作を実行するメモリ管理メカニズムと、
ディスパッチ可能オブジェクトおよび論理ＣＰＵへの状態の保存、および、ディスパッチ可能オブジェクトおよび論理ＣＰＵからの状態の復元を処理するディスパッチメカニズムと、
フォルト、トラップ、特別なハードウェアイベント、およびＳＰ発生イベントを取り扱う例外メカニズムと、
インタラプトが発生すると、ＳＰは、予め構成された特別の実行環境で、関連付けられたハンドラルーチンを起動し、高速のインタラプト応答するインタラプトメカニズムと、
マシンレジスタの集まりを使用して、デバッグおよび性能監視を援助するデバッグおよび監視メカニズムと、
暗号、復号、鍵管理を行う暗号メカニズムおよび暗号記憶メカニズムと、
秘密データと、秘密データに関連付けられたアクセス方法を含むリポジトリメカニズムおよびリポジトリ記憶メカニズムと
を含むプラットフォームカーネルメカニズムを提供する内部インタフェースと
をコンピュータにさらに実行させる
請求項１８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２４】
オペレーティングシステムを実行する前に前記オペレーティングシステムを初期化するように実行される前記ファームウェアルーチン、および、前記ソフトウェアサービスを認証し、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認するブートプロセスを
コンピュータにさらに実行させる
請求項１９に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２５】
前記ブートプロセスは、オペレーティングシステムのルーチン、および、ユーザアプリケーションプログラムを含むファームウェアルーチン、または、ソフトウェアサービスの実行前に、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認することを
コンピュータに実行させる
請求項２４に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項２６】
請求項１８に記載のコンピュータシステムをセキュリティ保護する方法によって提供されるプラットフォームインタフェースを構成するソフトウェアをコンピュータにより実行させる

コンピュータ命令。
【請求項２７】
少なくとも１つのユーザレベルプログラムと、
少なくとも１つのオペレーティングシステムと、
請求項１８に記載のコンピュータシステムをセキュリティ保護する方法によって提供されるプラットフォームと
を備えるコンピュータシステム。
【請求項２８】
複数の特権レベルと、特権命令と、特権レジスタと、非特権命令と、非特権レジスタとを有するハードウェアプラットフォームと、プラットフォームインタフェースとを含むコンピュータシステムをセキュリティ保護する方法であって、
前記プラットフォームインタフェースは、
前記複数の特権レベルごとに、プロセスの実行に使用されるメモリを複数の第１の領域に分割する第１の分割を行うことと、
並列に実行される複数のオペレーティングシステムごとに、前記複数の領域をさらに複数の第２の領域に分割する第２の分割とを行うことと、
ソフトウェアサービスを提供することと、
前記非特権命令および非特権レジスタをオペレーティングシステムおよびカスタム制御プログラム（１５０６）に公開し、前記呼び出し可能なソフトウェアサービスにインタフェースを提供し、前記特権命令および前記特権レジスタを隠蔽することと
をコンピュータに実行させ、
前記ソフトウェアサービスは、
前記ハードウェアの特権命令および特権レジスタを前記制御プログラムに公開することなく、前記特権命令および特権レジスタをシミュレーションすることなく、ハードウェアリソースの動作制御を提供するプラットフォーム管理サービス（１７１０）と、
内部で生成された秘密データを使用し、内部の秘密データを外部の計算エンティティに公開せずに、前記内部の秘密データを管理するセキュリティ管理サービス（１７１２）と
を含む
コンピュータシステムをセキュリティ保護する方法。
【請求項２９】
前記ハードウェアプラットフォームは、
前記ハードウェアを制御する１つ以上のファームウェアルーチンを提供するファームウェアインタフェースと、
少なくとも４つの特権レベルと、
Ｉ／Ｏ操作によってアクセスできる物理メモリアドレスを制限する手段と
をさらに備える
請求項２８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３０】
前記呼び出し可能なソフトウェアサービスは、オペレーティングシステムおよびカスタム制御プログラム（１５０６）のいずれかの制御プログラムにより起動されると、前記制御プログラムの特権レベルより高い特権を与えられる少なくとも２つの特権レベルで実行する
請求項２８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３１】
前記プラットフォーム管理サービス（１７１０）は、
前記制御プログラムにより起動されて、特権レベルで行う操作する機能、マシンリソースを利用する機能、セキュリティに依存する機能を実行するプラットフォームサービス（１７１６）と、
システムリソースの相互に隔離したパーティションを示すドメインの生成と、環境設定と、再初期化と、シャットダウンとを提供するドメイン制御サービス（１７１８）と、

論理ＣＰＵおよび物理ＣＰＵの環境設定とスケジューリングとを提供するプロセッサ制御サービス（１７２０）と、
システムリソースの割り当てを少なくとも示すプラットフォームポリシーを特定し、保存し、前記プラットフォームインタフェースに適用するプラットフォームポリシー制御サービス（１７２２）と、
ドメイン内の論理ＣＰＵ間のイベントおよびドメイン間のイベントの伝達と、ドメイン間でのリソースの共有と、ドメイン間での高速ネットワーク接続またはデータ交換のためのデータ転送を行うドメイン間およびドメイン内サービスと
を含む請求項２８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３２】
前記セキュリティ管理サービス（１７１２）は、
前記制御プログラムまたはユーザアプリケーションプログラムによって起動され、システムの最も高い特権レベルで実行される所定のプロセスにのみ利用可能な秘密データを使用する機能を提供するリポジトリサービス（１７２６）と、
ドメインが、プラットフォームのみにアクセスし得る特権レベルを示すタグを、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、ディスパッチ可能オブジェクト、または、特定のユーザ、アプリケーションプログラム、オペレーティングシステムのコンポーネント、ディレクトリ、ファイル、およびディスパッチ可能オブジェクトを含むシステムオブジェクトに関連付け、前記タグを使用して、プラットフォームサービスおよびリポジトリサービスへのアクセスを認証する手段を提供する呼び出し側認証サービス（１７２８）と
を含む請求項２８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３３】
前記リポジトリサービスは、
一方向ハッシュ関数と、メッセージ認証コードと、デジタル署名と、乱数とを計算する機能を含む１つまたは２つ以上の暗号機能を使用したデータの暗号化および復号を提供する暗号サービスと、
データサービスであって、
前記制御プログラムとこのデータサービスとの間の特定のインタフェースのみを通じて、アクセスおよび変更されるログと、
前記リポジトリサービスが操作可能なセキュリティポリシーを保持するポリシーデータベースと、
システムコンポーネントのうち、このデータサービスのみからからアクセスされる前記秘密データを使用するサービスと
を含むデータサービスと
を含む請求項３２に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３４】
ＴＬＢ、ＶＨＰＴ、および他のページテーブルへの変換を挿入する特権動作を実行するメモリ管理メカニズムと、
ディスパッチ可能オブジェクトおよび論理ＣＰＵへの状態の保存、および、ディスパッチ可能オブジェクトおよび論理ＣＰＵからの状態の復元を処理するディスパッチメカニズムと、
フォルト、トラップ、特別なハードウェアイベント、およびＳＰ発生イベントを取り扱う例外メカニズムと、
インタラプトが発生すると、ＳＰは、予め構成された特別の実行環境で、関連付けられたハンドラルーチンを起動し、高速のインタラプト応答するインタラプトメカニズムと、
マシンレジスタの集まりを使用して、デバッグおよび性能監視を援助するデバッグおよび監視メカニズムと、
暗号、復号、鍵管理を行う暗号メカニズムおよび暗号記憶メカニズムと、
秘密データと、秘密データに関連付けられたアクセス方法を含むリポジトリメカニズム

およびリポジトリ記憶メカニズムと
を含むプラットフォームカーネルメカニズムを提供する内部インタフェースと
をさらに含む請求項２８に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３５】
オペレーティングシステムを実行する前に前記オペレーティングシステムを初期化するように実行される前記ファームウェアルーチン、および、前記ソフトウェアサービスを認証し、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認するブートプロセスを
コンピュータにさらに実行させる
請求項２９に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３６】
前記ブートプロセスは、オペレーティングシステムのルーチン、および、ユーザアプリケーションプログラムを含む前記ファームウェアルーチン、または、前記ソフトウェアサービスの実行前に、前記ファームウェアルーチン、および、前記ソフトウェアサービスのデジタル署名の正当性を確認することを
コンピュータに実行させる
請求項３５に記載のコンピュータシステムをセキュリティ保護する方法。
【請求項３７】
請求項２８に記載のコンピュータシステムをセキュリティ保護する方法によって提供されるプラットフォームインタフェースを構成するソフトウェアをコンピュータにより実行させる
コンピュータ命令。
【請求項３８】
少なくとも１つのユーザレベルプログラムと、
少なくとも１つのオペレーティングシステムと、
請求項２８に記載のコンピュータシステムをセキュリティ保護する方法によって提供されるプラットフォームと
を備えるコンピュータシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、コンピュータアーキテクチャ、オペレーティングシステム、コンピュータシステムセキュリティに関し、特に、安全なメモリ管理と、ディスパッチ可能オブジェクトのディスパッチと、例外処理と、インタラプト処理と、デバッグおよび性能監視と、暗号サービスと、安全なリポジトリサービスとを、オペレーティングシステムおよびカスタマイズされた制御プログラムに提供する安全なマシンおよび安全なマシンインタフェースに関する。
【０００２】
［相互参照］
この出願は、いずれも２００１年６月８日に出願された係属中の米国仮出願第６０／２９６，９５８号、第６０／２９６，９５７号、および第６０／２９７，１７５号の出願日の利益を主張する。
【背景技術】
【０００３】
［発明の背景］
コンピュータセキュリティは、大学のコンピュータ機関、政府のコンピュータ機関、および民間のコンピュータ機関で熱心に研究される極めて重要な研究分野となっている。
コンピュータシステムの製造者およびユーザは、長年、コンピュータシステム内の記憶データ、処理リソース、および他の計算リソースへのアクセスを制御する安全なコンピュータシステムを提供するように試みてきたが、コンピュータ製造者およびユーザのこの相

当な努力は、まだ、安全なコンピュータシステムの作成には成功していない。
多くの悪名高いコンピュータシステムセキュリティ侵害が、過去数年に広く公表されている。
これらのコンピュータシステムセキュリティ侵害には、ＡＴＭシステムおよび金融機関内のコンピュータシステムからの現金の窃盗、政府のコンピュータシステム、商用のコンピュータシステム、および個人のコンピュータシステムからの極秘文書および情報ならびに秘密文書および秘密情報の取得、電子メールおよび他のインターネット関連サービスを通じた多くの極めて犠牲が大きくかつ損害を与えるコンピュータウィルスの蔓延、ならびに他の多くの深刻なセキュリティ侵害が含まれる。

【０００４】

今日まで、商用のコンピュータシステムのコンピュータセキュリティの問題は、大部分は受身の体制で対処されてきた。
一般に、セキュリティの問題は、潜在的なセキュリティ侵害を含む新しい市販のコンピュータシステムの製造後に認識され、幾分場当たり的な設計後の方法論を通じて対処される。
セキュリティ技法は、一般に、基礎を成すコンピュータ技術およびリソースの発展の結果として発展する。
既存のシステムにおいて、さまざまなセキュリティの問題を認識して、パッチすることができるが、潜在的な多くのさらなるセキュリティの問題は、多くの場合認識されないままであり、不注意のエラーおよび偶然のエラーを通じて、またはより一般的には、悪意のあるユーザおよびコンピュータエンティティによるコンピュータシステムセキュリティ機能の体系的な究明を通じて、後に発見されるために供されている。

【０００５】

後述する本発明は、オペレーティングシステムと、オペレーティングシステムおよびコンピュータシステムのセキュリティとに関する。
この議論を円滑にし、さらに背景の議論を円滑にするために、まず、コンピュータハードウェアアーキテクチャおよびオペレーティングシステムの簡潔な概要を以下に提供する。

【０００６】

図１は、汎用コンピュータシステム内のハードウェアレイヤ、オペレーティングシステムレイヤ、およびアプリケーションプログラムレイヤを示すブロック図である。
コンピュータシステム１００は、ハードウェアレイヤ１０２と、オペレーティングシステムレイヤ１０４と、アプリケーションプログラミングレイヤ１０６とを備えると考えることができる。
コンピュータシステムは、多くの付加的なコンポーネント、サブレイヤ、および論理エンティティと相互関係を有し、かなり複雑であるが、図１に示す３つのレイヤの階層は、コンピュータソフトウェア業界およびコンピュータハードウェア業界内で共通に使用されるコンピュータシステムの論理ビューを表す。

【０００７】

ハードウェアレイヤ１０２は、コンピュータシステムの物理コンポーネントを備える。
これらの物理コンポーネントは、多くのコンピュータシステムでは、プロセッサ１０８、メモリストレージコンポーネント１１０、１１２、および１１４、内部バスおよび信号線１１６～１１９、バス相互接続デバイス１２０および１２２、ならびにさまざまなマイクロプロセッサに基づく周辺インタフェースカード１２４～１２９を含む。
プロセッサ１０８は、命令実行デバイスであり、内部メモリコンポーネント１１０、１１２、および１１４から当該プロセッサが取得した命令のストリームを実行する。
プロセッサは、レジスタ１３０と呼ばれる高速アクセスが可能な少数のメモリストレージコンポーネントを含む。
その上、現代のプロセッサは、比較的少ない量の記憶された命令も含むことができる。
一般に、データおよび命令は、内部バス１１６および１１７とバス相互接続デバイス１

２０とを介して、読み出し専用メモリ（「ＲＯＭ」）コンポーネント１１０から読み出され、また、メモリコンポーネント１１２および１１４から読み出され、当該メモリコンポーネント１１２および１１４に書き込まれる。

さらに大きなデータ記憶容量は、例えば、ディスクドライブ、ＣＤ－ＲＯＭドライブ、ＤＶＤドライブなどの周辺データ記憶デバイス、および他の、内部バス１１６、１１８、および１１９と、相互接続デバイス１２０および１２２と、周辺機器相互接続カード１２４～１２９の１つまたは２つ以上のものとを介して、プロセッサによりアクセスされるコンポーネントに存在する。

例えば、大きなプログラムの記憶された命令は、プログラムの実行中に、必要に応じて検索され、内部メモリコンポーネント１１２および１１４に記憶されるようにディスクドライブ上に存在することがある。

より高機能なコンピュータは、それに相応して、より複雑な内部バス相互接続および追加されたコンポーネントと共に複数のプロセッサを含むことがある。

【０００８】

オペレーティングシステムレイヤ１０４は、さまざまなソフトウェアルーチンを備えた論理レイヤである。

このさまざまなソフトウェアルーチンは、プロセッサ１０８または一組のプロセッサのうちの１つまたは２つ以上のプロセッサ上で実行され、コンピュータシステムの物理コンポーネントを管理する。

コンピュータオペレーティングシステムは、一般に、コンピュータシステムの最下位レベルのソフトウェアレイヤと考えられており、結合されたハードウェア／ソフトウェア環境を作成するように機能する。

この環境では、アプリケーションプログラムを含むユーザプログラムが実行でき、ユーザが、さまざまな入出力（Ｉ／Ｏ）デバイスを通じてコンピュータシステムのサービスをインタラクティブに使用することがサポートされる。

【０００９】

従来のコンピュータシステムのオペレーティングシステムのルーチンは、ユーザレベルのアプリケーションプログラムよりも高い優先度で実行される。

そして、これらのルーチンは、多くのアプリケーションプログラムの同時並行した実行を調整し、各アプリケーションプログラムに実行時環境を提供する。

このランタイム環境は、プロセッサ時間と、アプリケーションプログラムに提供されるアドレス空間によってアドレス指定されるメモリの少なくとも１つの領域と、さまざまなデータ入力サービスおよびデータ出力サービスを含む。

これらのデータ入力サービスおよびデータ出力サービスには、メモリコンポーネントへのアクセスと、周辺機器へのアクセスと、通信媒体へのアクセスと、他の内部デバイスおよび外部デバイスへのアクセスとが含まれる。

【００１０】

現在実行中の各プログラムは、プロセスと、さまざまな状態変数によって定義される論理エンティティと、オペレーティングシステムによって管理されるデータ構造体との関連で実行される。

オペレーティングシステムによって管理される１つの重要な内部データ構造体は、１つまたは２つ以上のプロセスキュー１３２である。

このプロセスキュー１３２は、現在アクティブな各プロセスに対して、プロセス制御ブロックまたは同様のデータ構造体を含む。

このプロセス制御ブロックまたは同様のデータ構造体は、オペレーティングシステムによって管理される現在アクティブなプロセスの状態を定義するデータを記憶するものである。

オペレーティングシステム１０４は、各同時並行プログラムに対して、そのプログラムが、コンピュータハードウェア１０２を連続的に使用しているという錯覚を与える。

しかし、実際には、オペレーティングシステム１０４は、いかなる瞬間においても、１

つのプロセッサにつき、せいぜい１つのプログラムしか実行しておらず、現在アクティブなさまざまなプログラム間の実行を高速に切り換え、プログラムの実行の再起動に先立ってそのプログラムのコンテクストを復元し、かつ、プログラムがアイドル状態になり、別のプログラムが実行可能になると、プログラムのコンテクストをメモリに記憶する。

また、オペレーティングシステム１０４は、実行中のプログラムに、幾分一般化した論理インタフェースも提供する。

この論理インタフェースを通じて、実行中のプログラムは、多くのタイプのリモートコンピュータリソースおよびローカルコンピュータリソースにアクセスしてこれらを使用することができる。

これらのコンピュータリソースには、物理メモリ、マスストレージデバイス、通信ネットワーク、Ｉ／Ｏデバイス、および他のリソースが含まれる。

【００１１】

アプリケーションプログラミングおよびユーザインタフェースレイヤ１０６は、ユーザが見ることができるコンピュータシステムのレイヤである。

アプリケーションプログラムは、メモリ領域内に記憶された一組の命令およびデータ１３４と、オペレーティングシステムのインタフェースを通じてアプリケーションプログラムによりアクセスされるリソース１３６～１４２とを備える。

メモリ領域内に記憶された一組の命令およびデータは、アプリケーションプログラムを実行するプロセスに対してオペレーティングシステムにより提供されるアドレス空間内でアドレス指定されている。

リソース１３６～１４２は、アプリケーションプログラムが、外部デバイスおよびデータベース管理システムにデータを記憶すること、外部デバイスおよびデータベース管理システムからデータを検索すること、内部クロックの値およびシステム環境設定情報などのシステム情報を取得すること、および付加的なサービスにアクセスすることを可能にする。

このメモリ領域およびリソースは、ハードウェアメモリ、データ記憶装置、通信手段、および他のコンポーネント内に実装され、図１では、論理的な意味で、アプリケーションプログラミングおよびユーザインタフェースレイヤ１０６内に示される。

【００１２】

初期のコンピュータシステムには、オペレーティングシステムは存在しなかった。

図２は、初期のコンピュータシステムを示すブロック図である。

単一のアプリケーションプログラム２０２は、簡単なハードウェアインタフェース２０６を介してコンピュータシステムハードウェア２０４とインタフェースされて、コンピュータシステムハードウェア２０４上で直接実行される。

ハードウェアインタフェース２０６は、コンピュータシステムハードウェアによって提供される機械語命令およびレジスタを含む。

アプリケーションプログラムは、紙テープまたは穿孔された一組のカードとしてマシンにロードされ、実行時にアプリケーションプログラムによって必要とされるルーチンを提供するさまざまな計算ライブラリおよびＩ／Ｏライブラリをコード化した一組の紙テープまたは穿孔されたカードもともマシンにロードされる。

初期のマシンでは、１つのアプリケーションプログラムは、ロードされ、完了するまで実行される。

アプリケーションプログラムの実行のスケジューリングは、人間のシステム管理者が、アプリケーションプログラムをコード化した紙テープまたは穿孔されたカードを棚またはトレイに物理的に並べて、アプリケーションプログラムに割り当てられた優先度に従って、先入れ先実行の順序により手動で再配列することによって実行される。

【００１３】

オペレーティングシステムは、アプリケーションプログラムのロードおよび実行に関連したシステム管理を自動化したいという願望と、多くのユーザにコンピュータシステムの同時並行使用およびインタラクティブな使用を可能にする必要性とから生まれた。

オペレーティングシステムは、例えばＩＢＭ　Ｓｙｓｔｅｍ／３６０シリーズなどのメインフレームコンピュータシステム用に開発された。

図３は、初期のメインフレームコンピュータ内のオペレーティングシステムの論理的配置を示すブロック図である。

オペレーティングシステムの最も特権を有する部分は、通常、カーネル３０２と呼ばれ、ハードウェアインタフェースを通じてコンピュータハードウェア３０４とインタフェースする。

カーネル３０２には、非特権命令および特権命令と、非特権レジスタおよび特権レジスタと、ハードウェア割り込みメカニズムとが含まれる。

レジスタおよび命令は、オペレーティングシステムの所定のマシン機能を予約して高レベルのアプリケーションプログラムがその機能にアクセスすることを防止するために、特権の組と非特権の組とに分割される。

例えば、実行中のプログラムの実行優先度の設定に関与するレジスタは、一般に特権を有し、これにより、アプリケーションプログラムが、オペレーティングシステムのスケジューリングおよびアプリケーションプログラムの優先順位を妨げたり破損したりすることが防止される。

オペレーティングシステムは、特権命令および非特権命令の双方と、特権レジスタおよび非特権レジスタの双方と、ハードウェア割り込みメカニズムへのフルアクセスを有するが、非特権命令および非特権レジスタのみを、オペレーティングシステムインタフェース３０８を介してアプリケーションプログラムに公開する（expose）。

非特権命令に加えて、オペレーティングシステムインタフェースは、多くの種々のオペレーティングシステムサービスルーチンを提供する。

例えば、アプリケーションプログラム３００がオペレーティングシステムインタフェース上で実行される場合に、アプリケーションプログラム３００は、オペレーティングシステムがなかった初期のコンピュータシステムの場合のように、Ｉ／Ｏライブラリと共にパッケージ化される必要はないが、オペレーティングシステム３０２によって提供されるＩ／Ｏサービスルーチンを介してＩ／Ｏデバイスにアクセスすることができる。

【００１４】

現代のパーソナルコンピュータ（「ＰＣ」）システムでは、オペレーティングシステムは、システム内のほぼ等しい論理的配置に存在する。

図４は、現代のＰＣのアプリケーションレイヤ、オペレーティングシステムレイヤ、およびハードウェアレイヤのブロック図である。

しかし、ＰＣハードウェア４０４とオペレーティングシステム４０６との間のインタフェース４０２は、新しいファームウェアサービスインタフェースを含む。

このファームウェアサービスインタフェースは、オプションとして、部分的に、アプリケーションプログラムに公開されることがある。

ファームウェアインタフェースは、表示機能を含むハードウェア機能の制御を提供するさまざまなファームウェアルーチンを含む。

【００１５】

現代のオペレーティングシステムの内部では、ハードウェア特権レベルが使用されて、オペレーティングシステムと、他のルーチンおよびプログラムとの間でリソースが分割される。

図５Ａ～図５Ｄは、汎用コンピュータハードウェアシステムの基本的な特権レベルのメカニズムおよび機能を示している。

いかなる時間においても、実行されるプログラムは、現在の特権レベルと関連付けられる。

この特権レベルは、プロセッサステータスレジスタ５０４内のビットフィールド５０２に保持される。

このプロセッサステータスレジスタは、プロセッサの基本的状態を制御し、かつ反映する。

現在の多くのコンピュータシステムは、特権および非特権の２つの特権レベルのみを使用する。

したがって、特権レベルのビットフィールドには、現在の特権レベルを制御するのに、１ビットのみが必要とされる。

現代のコンピュータシステムは、数年前のシステムも同様であるが、３つ以上の特権レベルを使用する。

一般に、特権レベルのビットフィールド内のｎビットにより、通常、特権レベルの中で最も高い特権を有する特権レベル０から特権レベル２ｎ－１までの２ｎ個の別々の特権レベルが可能になる。

コンピュータシステム内のリソースは、一般に、１つまたは２つ以上の特権レベルで実行されるプロセスにのみアクセス可能なリソースの組に分割される。

例えば、図５Ｂに示すように、多くの共通のマシンアーキテクチャは、命令セット５０６を、非特権命令５０８の組と特権命令５１０の組とに分割する。

非特権命令は、あらゆる特権レベルで実行されるプロセスにアクセス可能であるのに対して、特権命令は、特権レベル０で実行されるプロセスにのみアクセス可能である。

一般に、オペレーティングシステムのカーネルルーチンは、特権レベル０で実行され、正しく構成されている場合には、特権命令の組を独占的に使用することができ、これにより、アプリケーションレベルのプロセス生成の実行と、そのプロセスの実行と、そのプロセスのスケジューリングとを管理する。

さらに、図５Ｃに示すように、ハードウェアレジスタセット５１２の全体は、非特権レジスタのセット５１４と特権レジスタセット５１６とに分割される。

非特権レジスタは、あらゆる特権レベルで実行されるプロセスにアクセス可能であるのに対して、特権レジスタは、特権レベル０で実行されるプロセスにのみアクセス可能である。

例えば、プロセッサステータスレジスタ５０４は特権レジスタである。

これにより、カーネルでないコードが、その現在の特権レベルを高くして、カーネルでないコードにアクセスできないように意図されたリソースにアクセスすることが防止される。

【００１６】

また、図５Ｄに示すように、メモリも、特権レベルに関して分割することができる。

通常、メモリ５１８は、ページ（例えば、メモリページ５２０）に分割される。

メモリページは、Ｉ／Ｏ操作、仮想メモリと物理メモリの間のマッピング、および他のタイプのオペレーティングシステムの活動に関するメモリの基本単位である。

多くのシステムでは、各メモリページは、特権レベル表示５２２は、ページ５２０と関連付けられるなど、１つまたは２つ以上の特権レベルの表示と関連付けられることがある。

これらの表示は、メモリページ内にタグとして直接組み込まれもよいし、メモリページを記述または参照する補助データ構造体に存在してもよい。

命令セットおよびレジスタは、多くの場合、上述したように、特権レベルに基づく異なる２つのパーティションにのみ分割されるのに対して、特権レベルに関するメモリページの分割は、より複雑となり得る。

例えば、あるページは、特権レベル０～２で実行されるプロセスにアクセス可能にされることがあり、別のページは、２ｎ個のすべての特権レベルで実行されるプロセスにアクセス可能とされることがあり、それ以外のページは、特権レベル０で実行されるプロセスにのみアクセス可能とされることがある。

【００１７】

リソースの分割は、信頼性のある安全なオペレーティングシステムの設計に使用される基本的な技法であるが、それだけでは、十分ではない。

ハードウェアレベルの特権に基づく分割を通じてだけでは対処できないセキュリティおよび信頼性の多くの態様が存在する。

図６は、多くの現代のコンピュータシステムに存在するセキュリティおよび信頼性の侵害の一例を示している。

コンピュータシステムの簡略化したブロック図の表現６０２が、図６に示されている。

通常、オペレーティングシステム供給業者は、ハードウェア供給業者によって提供されるハードウェアレイヤ６０６上で実行されるオペレーティングシステムレイヤ６０４を供給する。

多くの現代のコンピュータシステムでは、第三者の供給業者が、周辺ハードウェアＩ／Ｏデバイスを提供することができ、この周辺ハードウェアＩ／Ｏデバイスが、ハードウェア供給業者またはコンピュータシステムの所有者によってコンピュータシステムハードウェアに組み込まれる。

これらの周辺ハードウェアデバイスは、Ｉ／Ｏドライバとして知られているインタフェースソフトウェアルーチンを必要とする。

Ｉ／Ｏドライバは、インストールされて、オペレーティングシステム内で実行されるか、または、オペレーティングシステムと共に実行される。

例えば、図６では、Ｉ／Ｏドライバ６０８は、オペレーティングシステムレイヤ６０４内に存在する。

一般に、Ｉ／Ｏドライバ６０８は、特権レジスタ、および、アプリケーションレベルのプログラムによってアクセスできることを意図していないメモリリソースとインタフェースするために、最も高い特権レベルで実行される必要がある。

例えば、図６では、メモリリソースのマシン用の部分６１０が示されている。

メモリのある領域６１２は、アプリケーションレベルのプログラムによるアクセスを対象としている。

メモリの別の領域６１４は、オペレーティングシステムルーチンのみによるアクセスを対象としている。

この領域は、例えば、記憶された暗号鍵情報６１６を含むことがある。

メモリの別の領域６１８は、Ｉ／Ｏドライバ６０８およびオペレーティングシステムによる使用を対象とし、Ｉ／Ｏドライバおよびオペレーティングシステムがデータの交換に使用するＩ／Ｏバッファを含む。

【００１８】

Ｉ／Ｏドライバが、正しく記述され、悪意のコードを含んでいない場合には、Ｉ／Ｏドライバルーチンは、メモリのＩ／Ｏバッファ領域６１８からの読み出し、および、当該領域への書き込みを行うだけであり、メモリ操作を通じて、オペレーティングシステムの妨害もできないし、アプリケーションルーチンの妨害もできない。

しかしながら、ハードウェア供給業者もオペレーティングシステム供給業者も、Ｉ／Ｏドライバの内容を管理しない。

彼らは、認証方法を使用して、Ｉ／Ｏドライバが、特定の第三者の供給業者から入手されたものであると判定することがある。

彼らは、Ｉ／Ｏドライバのインストールに先立ってＩ／Ｏドライバをテストすることさえある。

第三者の供給業者の中には、自身のＩ／Ｏドライバのソースコードを検査に利用できるようにせず、これにより、オペレーティングシステム供給業者およびハードウェア供給業者がＩ／Ｏドライバを独立に検証できる範囲を制限する供給業者がいる。

Ｉ／Ｏドライバは、悪意の目的で挿入されたトロイの木馬コードを含んだり、あるいは、バグを含んだりすることが可能である。

いずれの場合も、Ｉ／Ｏドライバは、Ｉ／Ｏバッファ領域６１８以外の、Ｉ／Ｏドライバのアクセスが禁止されるように意図されたメモリの部分にアクセスを試みることができる。

現在の多くのコンピュータシステムでは、Ｉ／Ｏドライバが、システム内の任意のメモリ領域の読み出しおよび書き込みを行うことを防止するものは存在しない。

その理由は、Ｉ／Ｏドライバが、一般に、カーネルレベルの特権で実行されるからであ

る。
誤ったプログラムまたは悪意のプログラムが、一旦、カーネルレベルの特権で実行されると、そのプログラムは、オペレーティングシステムに利用可能なリソースのフルセットを使用することができる。
この場合に、例えば、Ｉ／Ｏドライバ内のトロイの木馬ルーチンは、Ｉ／Ｏドライバの動作中に実行を開始し、オペレーティングシステム６１４に割り当てられたメモリに記憶された暗号鍵６１６にアクセスし、他のオペレーティングシステムのルーチンおよびハードウェアリソースを使用して、その暗号鍵を遠隔のエンティティにエクスポートすることができる。
【特許文献１】米国特許第５０４３８７８号
【特許文献２】米国特許第６１３１１６５号
【特許文献３】米国特許第５７７４６５２号
【特許文献４】米国特許第５９４８０６４号
【発明の開示】
【発明が解決しようとする課題】
【００１９】
図６に示す問題は、現在のコンピュータシステムに起こり得るほとんど無限個のセキュリティ問題のほんの１つにすぎない。
確認されている多くの問題に対して、その問題に関連したシステムの脆弱性を解消または縮小する試みとして、さまざまな対策を設計して、オペレーティングシステムに追加設置することができる。
しかしながら、このような対策は、一般に、たった１つの問題または少数の問題に特有のものであり、非常に多くの別の脆弱性は、検出されない状態にあり、今後のセキュリティ問題の潜在的な原因のままである。
時々、試みた処置が、不完全であるか、または、誤っていることがある。
ある特定の場合には、試みた処置が、それ自体、別の問題の原因となる。
安全なシステムを製造することが望ましいことであることは認識されているにもかかわらず、今日まで、十分に安全で汎用的な商用のコンピュータシステムは、作成されてこなかった。
実際には、コンピュータシステムのセキュリティの一般的な概念は、長年の間、設計努力および開発努力に動機を与えてきたが、コンピュータセキュリティと、コンピュータセキュリティに関連した問題を解決する合理的なアプローチとの明確で包括的な理解は、得難い状態のままであった。
したがって、コンピュータシステムの設計者、製造者、開発者、およびユーザは、コンピュータセキュリティと、安全なコンピュータシステムと、コンピュータセキュリティの方法論との包括的な理解の必要性を認識してきた。
【課題を解決するための手段】
【００２０】
［発明の概要］
本発明の一実施の形態は、結合されたハードウェアおよびソフトウェアの安全なプラットフォームと、コンピュータシステム内でオペレーティングシステムおよびカスタマイズされた制御プログラムがインタフェースする、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースとを提供する。
本発明のこの実施の形態は、少なくとも４つの特権レベル、すなわち非特権命令、非特権レジスタ、特権命令、特権レジスタと、ファームウェアインタフェースとを提供するハードウェアプラットフォームを使用する。
結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースは、特権命令と、特権レジスタと、ファームウェアインタフェースとを、オペレーティングシステムおよびカスタマイズされた制御プログラムから隠蔽し、一組の呼び出し可能なソフトウェアサービスだけでなく、ハードウェアプラットフォームによって提供される非

特権命令および非特権レジスタをオペレーティングシステムおよびカスタマイズされた制御プログラムに提供する。

呼び出し可能なソフトウェアサービスは、ハードウェアの特権命令および特権レジスタの公開も行わず、特権命令および特権レジスタのシミュレーションも行わないハードウェアリソースの動作制御用の一組の安全なプラットフォーム管理サービスを提供する。

また、この呼び出し可能なサービスは、安全なリポジトリとも呼ばれる一組のセキュリティ管理サービスも提供する。

この一組のセキュリティ管理サービスは、内部で生成された秘密データを使用し、各セキュリティ管理サービスは、内部の秘密データをこのセキュリティ管理サービス自体以外のどの計算エンティティにも公開せずに、この内部の秘密データを管理する。

【発明を実施するための最良の形態】

【００２１】

［発明の詳細な説明］

本発明は、コンピュータシステムおよびコンピュータオペレーティングシステムに関する。

より詳細には、本発明の一実施の形態は、結合されたハードウェアおよびソフトウェアの安全なプラットフォーム（combined-hardware-and-software secure-platform）、および、安全な基盤を提供する、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェース（combined-hardware-and-software secure-platform interface）に関する。

この安全な基盤は、安全なコンピュータシステムを作成するために、当該基盤上で、１つまたは２つ以上のオペレーティングシステムをレイヤ化することができるものである。

結合されたハードウェアおよびソフトウェアのプラットフォーム、ならびに、結合されたハードウェアおよびソフトウェアのプラットフォームインタフェースは、マシン／オペレーティングシステムインタフェースの発展した段階を表す。

図２～図４を参照して上述したように、マシン／オペレーティングシステムインタフェースは、図２に示したような一組の命令およびレジスタから、図５に示したような一組の非特権命令および非特権レジスタと、一組の特権命令および特権レジスタと、ハードウェア割り込みメカニズムと、一組のファームウェア実施サービスとに発展した。

本発明の一実施の形態は、安全なプラットフォームおよび安全なプラットフォーム／オペレーティングシステムインタフェースである。

この安全なプラットフォーム／オペレーティングシステムインタフェースは、一組の非特権命令および非特権レジスタと一組の安全なプラットフォームサービスとを提供する。

この一組の安全なプラットフォームサービスは、メモリ管理と、ディスパッチ可能オブジェクトのディスパッチと、例外処理と、割り込み処理と、デバッグおよび性能監視と、暗号サービスと、安全なデータリポジトリサービスと、ドメイン制御サービスと、論理的なプロセッサ制御サービスおよび物理的なプロセッサ制御サービスと、ドメイン間通信サービスおよびドメイン内通信サービスと、プラットフォームポリシー管理サービスと、呼び出し側認証サービスと、他のサービスとにサポートを提供する呼び出し可能なルーチンを含む。

本発明のこの実施の形態の別のビューは、ちょうど図４に示すシステムのファームウェアおよびハードウェアが、図２および図３に示す初期のシステムに対して新しい異なるタイプのマシンを表すように、ハードウェアおよびその上に位置する安全なプラットフォームソフトウェアが、共に、新しいタイプのマシンを表すことである。

この新しいマシンは、安全で基本的な機能を提供し、それらの機能の上に、安全なオペレーティングシステムおよび安全なシステムを構築することができる。

オペレーティングシステムは、単にハードウェア特権レベルに頼るのではなく、当該オペレーティングシステムよりも高い特権レベルで実行され、かつ、当該オペレーティングシステムに対して安全な状態に維持される、基本的な機能の一式全てを使用することができる。

【００２２】
結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースを説明するために、セキュリティの議論と、結合されたハードウェアおよびソフトウェアの安全なプラットフォームでの使用に適したハードウェアプラットフォームの一例と、暗号化の一般的な議論とが、以下の３つの小節に提示される。
その後、４番目の小節、ならびにそれに続く付録Ａおよび付録Ｂでは、本発明の一実施の形態を表す、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースの詳細な説明が詳述される。
【００２３】
＜コンピュータセキュリティ＞
コンピュータセキュリティを定義する多くの異なる方法がある。
コンピュータシステムに発生するセキュリティ問題およびセキュリティ欠陥は、いずれの部類も、当初はシステム設計者によってセキュリティの問題として認識されなかった問題を表すことがかなり頻繁にある。
「コンピュータセキュリティ」という用語を適切かつ包括的に定義することが、安全なコンピュータシステムの設計において、重要かつ不可欠なステップを構成し得る。
本発明の一実施の形態を表す、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースの設計のステップは、安全なコンピュータシステムが、少なくとも以下の９つの特性およびプロパティを包含することを認識することである。
【００２４】
［可用性］
可用性のあるシステムは、エンドユーザが、システムサービスを起動する時は常に、正常に機能して応答するものと信用でき、かつ、頼ることができるシステムである。
可用性は、信頼できるハードウェアコンポーネントの冗長構成をシステム内に組み込み、かつ、ハードウェアコンポーネントまたはソフトウェアコンポーネントのいずれかの誤った動作の検出および／または補償を行うことにより、部分的に対処することができる。
しかしながら、可用性は、検出可能なエラーの部類を拡大し、かつ、エラーの発生により生じる損害を制限することにより、大幅に促進することもできる。
【００２５】
［秘密性］
コンピュータシステム内に記憶された情報、ネットワーク全体にわたって通信される情報、または、アーカイブ記憶装置に存在する情報は、認可された関係者のみが理解できる必要がある。
秘密性は、認可された関係者のみが、データおよびシステム情報を理解できるように制限できることを確実にするのに使用可能な暗号数字（cryptographic cipher）および暗号プロトコルを指定することによって、大幅に促進することができる。
このような制限は、ファイルデータの暗号化、ページングデータの暗号化、ローカルなシステムコンポーネント間の通信の暗号化、およびインターネット上の通信の暗号化を介して行われる。
【００２６】
［認証］
人の同一性、サーバの同一性、クライアントの同一性、機器の同一性、印刷サブシステムの同一性、記憶サブシステムの同一性、ならびにデータの生成およびアクセスを行う他のハードウェアおよびソフトウェアコンポーネントの同一性は、サービスがこれらのエンティティに提供される前に、正確に確認する必要がある。
認証は、このようなすべての同一性を認証できる暗号手段および暗号プロトコルを指定することよって、大幅に促進することができる。
【００２７】
［アクセス制御］
コンピュータシステム内に記憶された情報またはアーカイブ記憶装置に存在する情報へ

のアクセスは、適切に認可された場合にのみ許可される必要がある。
アクセス制御は、ある範囲のアクセス制御方式を提供することによって促進することができる。
【００２８】
［否認防止性（nonrepudiation）］
電子商取引による取引では、メッセージの送信関係者も、メッセージの受信関係者も、取引を指定するメッセージおよび取引を行うメッセージの存在ならびに送信を否認できないようにすべきである。
例えば、「機械装置」の発送を注文するメッセージを送信した関係者は、後に、「機械装置」を注文しなかったと主張できないことを保証する手段を提供する必要がある。
否認防止は、取引の関係者に責任があることを確実にするために使用できる暗号メカニズムおよび暗号プロトコル、例えばデジタル署名などを特定することによって促進することができる。
【００２９】
［ポリシー制御］
所有者および管理者は、システムのセキュリティエレメントの挙動を統率する彼らの制御ポリシー下で、それらのシステムを定義する能力を身に付ける必要がある。
【００３０】
［デジタル権利保護］
安全なシステムは、デジタルコンテンツを保護する必要があり、特に、デジタル画像を、認可されていない複製から保護する必要がある。
例として、貨幣、ＭＰＥＧ家庭用娯楽コンテンツ、美術の高品質画像、および限定配布用の印刷された資料が含まれる。
【００３１】
［プライバシー］
プライバシーのプロパティは、上述したプロパティによって決まるが、人およびグループの素性に関連した情報の管理および制御を行う付加機能を必要とする。
個人のプライベートな情報は、認可されていないアクセスから保護する必要がある。
場合によっては、アクセスが認可されている場合であっても、個人の素性は、開示されるべきではない。
プライバシーを保護するシステムの機能としては、次のものが含まれる。
すなわち、
（１）収集の目的に見合った、当該目的の範囲を超えない合法的な方法による個人情報の合法的収集、
（２）正確で最新の個人情報の保持、
（３）情報の保持、使用、または第三者への転送について個人の同意を得て、個人情報を公正で合法的に処理すること、
（４）個人は、自身の情報を修正または削除する権利を含めて、自身の情報にアクセスする権利を与えられるべきであること、
（５）個人は、どの個人情報が収集されるのかと、その使用目的と、責任を有するエンティティと、個人がその情報の使用に関して有する選択肢とを告げられる必要があること、
（６）個人は、自身の個人情報を取り扱う選択肢の選択および／または更新を行う能力を与えられる必要があること、ならびに
（７）プライベート情報が正確かつ合法的に取り扱われたことを監視機関または実施機関に実証する手段が提供される必要があること
が含まれる。
【００３２】
［データ完全性（data integrity）］
コンピュータシステムに保管されているデータは、高価で重要な資産である。
安全なシステムは、データ資産が失われることもなく、損傷も受けることなく、かつ、

認可されていない改竄、削除、偽造、置換、または他の形式の不正変更を受けないことを確実にする必要がある。

うまく設計されたファイルとデータベースシステムとの結合、体系的なバックアップおよび復元手順、ならびにデータ自体の完全性および秘密性のための暗号による保護は、このプロパティを保証するために必要である。

【００３３】

＜現代のコンピュータアーキテクチャ＞

Ｉｎｔｅｌ（登録商標）のＩＡ－６４コンピュータアーキテクチャに従って構築されたプロセッサは、安全なプラットフォームソフトウェアレイヤとの結合に適した現代の一連のコンピュータハードウェアプラットフォームの１つの基本的なハードウェアインタフェースを表す。

この基本的なハードウェアインタフェースにより、本発明の一実施の形態を表す、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースが作成される。

図７は、現代の一プロセッサ内のレジスタを示すブロック図である。

これらのレジスタは、プロセッサの実行状態を定義する値を保持し、この値は、メモリに保存されると、実行中のプロセスの実行停止前の状態を記録する。

メモリに保存されたある種のレジスタを回復させることにより、割り込まれたプロセスの実行を再開することができる。

図７に示すレジスタセットは、かなり複雑であり、重要度の高いレジスタのいくつかのみを、以下に説明する。

【００３４】

１つの重要なレジスタは、プロセスステータスレジスタ（「ＰＳＲ（process status register）」）７０２である。

ＰＳＲは、６４ビットレジスタであり、現在実行中のプロセスの制御情報を含む。

ＰＳＲは、多くのビットフィールドを備える。

これらのビットフィールドには、現在実行中のプロセスが実行されている時の現在の特権レベル（「ＣＰＬ（current privilege level）」）を含む２ビットフィールドが含まれる。

０、１、２、および３の４つの特権レベルが存在する。

最も高い特権レベルは、特権レベル０である。

最も低い特権レベルは、特権レベル３である。

特権レベル０で実行されているプロセスのみが、「システムレジスタセット」として知られているレジスタのサブセットを含む所定のマシンリソースにアクセスして、操作することが許可される。

このシステムレジスタセットは、図７では、下部の長方形７０４内に示されている。

これらのシステムレジスタは、一組の領域レジスタ７０６と、一組の保護キーレジスタ７０８と、データおよび命令の変換索引バッファレジスタ７１０と、一組のデバッグブレークポイントレジスタ７１２と、一組の性能監視環境設定レジスタ７１４と、一組の制御レジスタ７１６とを含む。

１つの制御レジスタ、すなわち、割り込みプロセッサステータスレジスタ（「ＩＰＳＲ（interruption processor status register）」）７１８は、最も近時の割り込まれたプロセスのＰＳＲの値を記憶する。

割り込みステータスレジスタ（「ＩＳＲ（interruption status register）」）７２０は、最も近時に発生した割り込みの性質を示す多数のフィールドを含む。

他の制御レジスタは、他のイベントに関連した情報を記憶する。

他のイベントに関連した情報としては、仮想アドレス変換フォルトに関連した仮想メモリアドレス変換情報、最後に実行に成功した命令バンドルへのポインタ、および他のこのような情報などである。

外部制御インタラプトレジスタ７２２の組は、インタラプトベクタを設定するために、

部分的に使用される。

【００３５】

図７の上部の長方形領域７２４に示すレジスタは、「アプリケーションレジスタセット」として知られている。

これらのレジスタは、一組の汎用レジスタ７２６を含む。

そのうちの１６個のレジスタ７２８は、割り込み処理コードの即値レジスタを提供するためにバンクされる。

少なくとも９６個の汎用レジスタ７３０が、汎用レジスタスタックを形成する。

その部分部分は、バンクメモリから自動的に記憶および検索を行うことができ、これにより、呼び出した側のソフトウェアルーチンと呼び出された側のソフトウェアルーチンとの間の関連付けが容易になる。

また、アプリケーションレジスタセットは、浮動小数点レジスタ７３２、述語レジスタ（predicate register）７３４、分岐レジスタ７３６、命令ポインタ７３８、現フレームマーカ７４０、ユーザマスク７４２、性能監視データレジスタ７４４、プロセッサ識別子７４６、アドバンストロードアドレステーブル７４８、および一組の特定のアプリケーションレジスタ７５０も含む。

【００３６】

ＩＡ－６４アーキテクチャは、プロセッサステータスレジスタに２ビットの特権レベルフィールドを設け、これにより、ＰＬ０、ＰＬ１、ＰＬ２、およびＰＬ３の４つの特権レベルを提供する。

ＰＬ０は、最も高い特権を有するカーネル特権レベルである。

ＰＬ１は、その次に高い特権を有する特権レベルである。

ＰＬ２は、その次に高い特権を有するレベルである。

ＰＬ３は、アプリケーションレベルの特権レベルである。

【００３７】

以下に、図８～図１１を参照して、ＩＡ－６４コンピュータアーキテクチャのメモリおよび仮想アドレス変換アーキテクチャを説明する。

ＩｎｔｅｌのＩＡ－６４コンピュータアーキテクチャにおいて定義される仮想アドレス空間は、例えば、図８に示す領域８０２～８０７などの２２４個の領域を含む。

各領域は、連続した仮想メモリアドレスによって連続的にアドレス指定される２６１バイトを含む。

したがって、仮想メモリアドレス空間は、全体で、２８５バイトのメモリからなるアドレス空間に及ぶものとみなすことができる。

次に、８５バイトの仮想メモリアドレス８０８は、２４ビットの領域フィールド８１０および６１ビットのアドレスフィールド８１２を備えるものとみなすことができる。

【００３８】

一方で、一般に、仮想メモリアドレスは、６４ビットの量としてコード化される。

図９は、領域レジスタと、保護キーレジスタと、変換索引バッファ（「ＴＬＢ（translation look-aside buffer）」）とに記憶された情報を介した６４ビットの仮想メモリアドレスの物理メモリアドレスへの変換を示している。

Ｉｎｔｅｌ（登録商標）のＩＡ－６４アーキテクチャでは、仮想アドレスは、６４ビットのコンピュータワードであり、図９では、３つのフィールド９０４～９０６に分割された６４ビット量９０２によって表されている。

最初の２つのフィールド９０４および９０５は、メモリページのサイズに応じたサイズを有する。

メモリページのサイズは、ある範囲のメモリページサイズ内で調整することができる。

１番目のフィールド９０４は、「オフセット」と呼ばれる。

オフセットは、メモリページ内のバイトを指定する整数である。

例えば、メモリページが、４０９６バイトを含む場合、オフセットは、０～４０９５の値を２進法で表現するために１２ビットを含む必要がある。

２番目のフィールド９０５は、仮想ページアドレスを含む。
仮想ページアドレスは、物理メモリにマッピングされる、仮想アドレス空間内のメモリページを指定する。
このメモリページは、さらに、ディスクなどのマスストレージデバイスに記憶されたメモリページによってバックアップされる。
３番目のフィールド９０６は、３ビットのフィールドであり、仮想メモリの領域の識別子を含んだ領域レジスタを指定し、仮想ページアドレス９０５によって指定される仮想メモリページが含まれる。

【００３９】

仮想メモリアドレス９０２の物理メモリアドレス９０８への変換は、時にカーネルおよびオペレーティングシステムのルーチンと組み合わせて、プロセッサにより実行される。
この物理メモリアドレス９０８には、コンピュータシステムの物理メモリコンポーネントのページを参照する物理ページ番号９１２に加えて、仮想メモリアドレスのオフセット９０４と同じオフセット９１０が含まれる。
仮想メモリアドレスから物理メモリアドレスへの変換が、ＴＬＢ９１４に含まれる場合には、仮想メモリアドレスから物理メモリアドレスへの変換は、オペレーティングシステムが介入することなく、もっぱらプロセッサによって実行することができる。
プロセッサは、領域レジスタセレクタフィールド９０６を使用して、一組の領域レジスタ９１８内のレジスタ９１６を選択する。
選択された領域レジスタ９１６は、２４ビットの領域識別子を含む。
プロセッサは、選択された領域レジスタに含まれる領域識別子と、ハードウェア機能の仮想ページアドレス９０５とを共に使用して、領域識別子および仮想メモリアドレスを含むＴＬＢエントリ９２０を選択する。
この領域識別子および仮想メモリアドレスは、選択された領域レジスタ９１６に含まれる領域識別子および仮想ページアドレス９０５と一致するものである。
ＴＬＢエントリ９２２などの各ＴＬＢエントリは、領域識別子９２４と、ＴＬＢエントリ９２６によって記述されたメモリページと関連付けられた保護キーと、仮想ページアドレス９２８と、アクセス権フィールド９３０を共に構成する特権フィールドおよびアクセスモードフィールドと、物理メモリページアドレス９３２とを含むフィールドを収容する。

【００４０】

ＴＬＢのエントリであって、仮想メモリアドレスの領域レジスタセレクタフィールドによって指定された領域レジスタ内に含まれる領域識別子を含み、かつ、仮想メモリアドレス内で指定された仮想ページアドレスを含むエントリが、発見できる場合には、プロセッサは、仮想メモリアドレスによって記述された仮想メモリページが、現在実行中のプロセスによってアクセス可能かどうかを判断する。
ＴＬＢエントリ内のアクセス権が、現在実行中のプロセスがそのメモリページにアクセスすることを許可しており、かつ、ＴＬＢエントリ内の保護キーが、現在実行中のプロセスがそのメモリページにアクセスすることを許可するアクセスモードと関連した保護キーレジスタ９３４内に発見できる場合には、現在実行中のプロセスは、そのメモリページにアクセスすることができる。
ＴＬＢエントリ内に含まれるアクセス権は、３ビットのアクセスモードフィールドおよび２ビットの特権レベルフィールドを含む。
３ビットのアクセスモードフィールドは、特権の読み出し、書き込み、および実行のうちの１つまたはそれらの組み合わせを示す。
２ビットの特権レベルフィールドは、アクセスを行うプロセスによって必要とされる特権レベルを指定する。
各保護キーレジスタは、アクセスモードフィールドと関連付けられた２４ビットの保護キーと、保護キーレジスタが現在有効であるかどうかを示す有効ビットとを含む。
アクセスモードフィールドは、アクセスモードの読み出し、書き込み、および実行が許

可されたことを指定する。
したがって、ＴＬＢエントリによって記述されたメモリページにアクセスするには、アクセスを行うプロセスは、保護キーレジスタ内の有効な保護キーと、ＴＬＢエントリのメモリページとに関連付けられたアクセスモードと適合した方法でページにアクセスする必要があり、かつ、ＴＬＢエントリ内のメモリページと関連付けられた特権レベルと適合した特権レベルで実行されている必要がある。
【００４１】
仮想メモリアドレス内の仮想ページアドレスと、仮想メモリアドレスの領域レジスタ選択フィールドによって選択された領域識別子とに等しい領域識別子と、仮想ページアドレスとを有するエントリが、ＴＬＢ内で発見できない場合には、ＴＬＢミスが発生し、ハードウェアは、カーネルメモリに位置するＶＨＰＴと呼ばれる構成されたマッピング制御テーブルから正しいＴＬＢエントリを突き止めるように試みる。
ハードウェアが、マッピング制御テーブルから正しいＴＬＢエントリを突き止めることができない場合には、ＴＬＢフォルトが発生し、指定されたメモリページを物理メモリ内で発見するために、カーネルまたはオペレーティングシステムのルーチンが起動される。
あるいは、必要に応じて、指定されたメモリページを外部デバイスから物理メモリにロードするために、カーネルまたはオペレーティングシステムのルーチンが起動される。
その後、適切に変換したものが、ＶＨＰＴおよびＴＬＢにエントリとして挿入される。
仮想メモリアドレスの物理メモリアドレスへの変換を試みた時に、プロセスが、有効な保護キーを保護キーレジスタ９３４内に発見しなかった場合、または、現在実行中のプロセスが試みたアクセスが、ＴＬＢエントリのアクセスモードと適合しない場合、または、保護キーレジスタの保護キー内の読み出し／書き込み／実行ビットと適合しない場合、もしくは、現在実行中のプロセスの特権レベルが、ＴＬＢエントリによって必要とされる特権レベルより低い場合には、カーネルルーチンによって操作されるフォルトが発生し、カーネルルーチンは、オペレーティングシステムのルーチンへ実行をディスパッチする。
【００４２】
図１０は、オペレーティングシステムのルーチンが、仮想メモリアドレスに対応する物理メモリのメモリページを発見するのに使用するデータ構造体の一形式を示している。
仮想メモリアドレス９０２は、図１０において、図９と同じフィールドおよび符号で示されている。
オペレーティングシステムのルーチンは、領域セレクタフィールド９０６および仮想ページアドレス９０５を使用して、仮想ページテーブル１００４内のエントリ１００２を選択する。
仮想ページテーブルエントリ１００２は、物理メモリのページ１００８を参照する物理ページアドレス１００６を含む。
仮想メモリアドレスのオフセット９０４は、仮想メモリページ１００８の適切なバイト位置１０１０を選択するために使用される。
仮想ページテーブル１００２は、物理アドレスが有効であるかどうかを示すビットフィールド１０１２を含む。
物理アドレスが有効でない場合には、オペレーティングシステムは、物理メモリ内のメモリページを選択してそのメモリページを含め、そのメモリページの内容を、例えばディスクドライブ１０１４などの外部記憶デバイスから検索する。
仮想ページテーブルのエントリ１００２は、付加フィールドを含み、この付加フィールドから、ＴＬＢエントリに必要とされる情報を検索することができる。
オペレーティングシステムが、仮想メモリアドレスの物理メモリアドレスへのマッピングに成功するとすぐに、そのマッピングは、仮想ページテーブルのエントリに入力され、また、ＴＬＢエントリとしてフォーマットされた後、ＴＬＢに挿入される。
【００４３】
図１１は、ＴＬＢエントリに使用されるアクセス権のコード化を示している。
アクセス権は、３ビットのＴＬＢ．ａｒモードフィールド１１０２および２ビットのＴ

ＴＬＢ．ｐｌ特権レベルフィールド１１０４を備える。
ＴＬＢ．ａｒモードフィールド１１０２は、読み出し権、書き込み権、実行権、および組み合わせアクセス権を指定する。
ＴＬＢ．ｐｌ特権レベルフィールド１１０４は、メモリページと関連付けられた特権レベルを指定する。
図１１では、ＴＬＢ．ａｒフィールドおよびＴＬＢ．ｐｌフィールドにそれぞれ含まれる可能な値のアクセス権が示されている。
アクセス権は、現在のプロセスが実行される特権レベルによって決まることに留意されたい。
したがって、例えば、ＴＬＢ．ａｒが０に等しく、かつ、ＴＬＢ．ｐｌが３に等しいＴＬＢエントリによって指定されるメモリページには、あらゆる特権レベルで実行されるルーチンが、読み出しのためにアクセスすることができる。
このことは、図１１では、各特権レベル１１０６～１１０９に対応する列の文字「Ｒ」によって示される。
これに対して、ＴＬＢ．ａｒが０に等しく、かつ、ＴＬＢ．ｐｌが０に等しいＴＬＢエントリによって記述されるメモリページには、特権レベル０で実行されるプロセスのみが、読み出しによりアクセスすることができる。
このことは、図１１では、特権レベル０に対応する列の下のある文字「Ｒ」１１１０によって示される。
図１１に記述したアクセス権は、図４を参照して行った上記議論による特権レベルによってネストする。
一般に、特定の特権レベルで実行されるプロセスは、その特権レベルおよびそれより低いすべての特権レベルに関連付けられたメモリページにアクセスすることができる。
ＴＬＢエントリに含まれるアクセス権のみを使用すると、レベル３で実行されるプロセスおよびレベル０で実行されるカーネルルーチンにはアクセス可能であるが、特権レベル２で実行されるオペレーティングシステムのルーチンにはアクセス可能でないメモリ領域を作成することができない。
また、特権レベル３で実行されるルーチンにアクセス可能なあらゆるメモリページは、特権レベル２で実行されるオペレーティングシステムのルーチンにもアクセス可能である。

【００４４】
コンピュータリソースが、特権レベルに従って分割されるので、プロセスは、一般に、その実行中のプロセスの特権レベルまたはそれより低い特権レベルに関連付けられたメモリページからの命令およびデータにアクセスする。
なお、特権レベル０は、最も高い特権レベルとみなされる。
実行中のプロセスには、ハードウェア生成割り込みによって割り込みが行われることがある。
この場合、割り込み処理ルーチンが、ハードウェアによって自動的に起動され、特権レベル０で実行される。
割り込みが、部分的に、または、完全に処理されるとすぐに、割り込み処理ルーチンは、インタラプトからの復帰（「ｒｆｉ（return from interrupt）」）命令を実行して、前に実行中のルーチンにおけるインタラプト発生ポイントに実行を戻し、そのＣＰＬを、前に実行中のルーチンが割り込み前に実行されていたＣＰＬに再び下げることができる。
【００４５】
割り込みプロセスを図１２に示す。
図１２では、アプリケーションプログラムを実行しているプロセスは、アプリケーションプログラムのプロセスの低い特権レベルと関連付けられたメモリページ１２０２からの命令を実行する。
このメモリページからの命令の実行は、濃い曲線の矢印、すなわち塗りつぶされた曲線の矢印１２０４～１２０６によって表されている。

特権レベル０の割り込み処理ルーチンの実行は、薄い曲線の矢印、すなわちアウトライン型の曲線の矢印１２０８によって図１２に示されている。
最初、アプリケーションプログラムのプロセスは、メモリ位置１２１０－１２１５に記憶された命令を逐次的に実行する。
メモリ位置１２１５に記憶された命令の実行中、マシン割り込みが発生し、特権レベル０の割り込みハンドラが起動される。
起動後、割り込みハンドラは、特権レベル０に関連付けられたメモリページ１２１８内のメモリ位置１２１６に記憶された命令の実行を開始する。
割り込みハンドラは、メモリ位置１２２０に記憶されたｒｆｉ命令を最後に実行する前に、多数の命令を実行する。
ｒｆｉ命令の実行により、ＣＰＬは、アプリケーションプログラムのプロセスが割り込み発生前に実行されていた低い特権レベルに再び降格され、制御は、濃い矢印、すなわち塗りつぶされた矢印１２０５によって示されるように、アプリケーションプログラムに戻される。
その後、アプリケーションプログラムは、濃い矢印、すなわち塗りつぶされた矢印１２０６によって示されるように、実行を再開する。
所定のタイプの割り込みでは、特権レベル０のインタラプトハンドラは、完全に割り込みを受けたプロセスの状況の中で実行することができる。
より複雑な割り込み処理を行うために、オペレーティングシステムは、アプリケーションプログラムのプロセスに関連付けられた状況情報を、そのプロセスと関連付けられたＰＣＢに記憶して、割り込みハンドラが実行できる新しい状況を準備することができる。
そして、割り込みの処理に続いて、アプリケーションプログラムのプロセスが復元される時、アプリケーションプログラムの状況が復元される。

【００４６】

アプリケーションプログラムのプロセスが、そのアプリケーションプログラムのプロセスの特権レベルよりも高い特権レベルで実行されるオペレーティングシステムのサブルーチンまたは機能を呼び出す必要がある場合が存在する。
ＰＳＲは、特権レベル０のルーチンによってのみアクセスおよび変更を行うことができるので、それより低い特権レベルで実行されるアプリケーションプログラムは、ＣＰＬを直接変更することができない。
このような呼び出しを実行する１つの方法は、コード化されたパラメータを含む「ブレーク」命令を実行することである。
この命令は、特権レベル０の割り込みを引き起こし、ブレーク命令のコード化されたパラメータはデコードされて、所望の特権レベル０の機能に制御を送ることができる。
しかしながら、ＩＡ－６４コンピュータアーキテクチャは、より高速なメカニズムを提供し、このメカニズムによって、アプリケーションプログラムのプロセスは、特権レベル０のプロセスを起動することができる。
図１３は、アプリケーションレベルのルーチンに、より高い特権レベルのルーチンの呼び出しを許可するＣＰＬ昇格メカニズムを示している。
図１３は、図１２に使用されるのと同じ表示規則を使用する。
塗りつぶされた矢印１３０２によって表される実行中のアプリケーションプログラムは、より高い特権レベルのルーチン「Ａ」を呼び出すために、メモリ位置１３０４に記憶された呼び出し命令を実行する。
この高い特権レベルのルーチンは、その高い特権レベルに関連付けられたメモリページ１３０６に記憶されている。
この高い特権レベルのルーチンは、メモリ位置１３０８に記憶されたｅｐｃ命令を最初に実行する。
このｅｐｃ命令は、ＣＰＬを、当該命令を含むメモリページと関連付けられたＣＰＬに昇格させる。
その後、呼び出されたルーチンは、その高い特権レベルで実行可能であり、ｂｒ．ｒｅ

ｔ命令１３１０で実行を終了する。
ｂｒ．ｒｅｔ命令は、アプリケーションレベルのプログラムに実行を戻し、自動的に、ＣＰＬを、そのアプリケーションレベルのプログラムがサブルーチン呼び出し前に実行されていたＣＰＬに再び降格させる。
このメカニズムは、アプリケーションプログラムがＣＰＬを昇進させて、保護されたコンピュータリソースへの認可されていないアクセスを実行する手段を提供するものではないことに留意されたい。
特権レベル０で実行される命令だけが、唯一、メモリページを特権レベルに関連付けることができる。
したがって、ｅｐｃのＣＰＬ昇格命令を使用するルーチンは、最初に、特権レベル０のルーチンによって設定される必要がある。
このように、オペレーティングシステムは、ｅｐｃメカニズムを使用して、低い特権レベルのルーチンから呼び出し可能なオペレーティングシステムエントリポイントを安全に作成することができる。
【００４７】
＜暗号化技術＞
６つの基本タイプの暗号技法が存在する。
すなわち、
（１）対称（秘密）鍵暗号、
（２）非対称（公開／秘密）鍵暗号、
（３）一方向ハッシュ関数、
（４）メッセージ認証コード、
（５）デジタル署名、および
（６）乱数発生器
である。
【００４８】
［暗号化／復号］
対称鍵暗号および非対称鍵暗号は、ともに、データを非常に徹底的にスクランブルして、正しい鍵を持たない攻撃者が、スクランブルされていないオリジナルのデータの内容も意味も判断できないようにするために使用される。
データ自体は、どの形式のデジタル情報であってもよく、あらゆる言語の文字や数字、言葉や句読点や制御の特別なシンボル、またはビデオ画像およびオーディオ画像を含む。
【００４９】
データをスクランブルするプロセスは、データを「暗号化する（encrypting）」または「暗号文に変える（enciphering）」と呼ばれる。
スクランブルを解除してデータをそのオリジナルの形式に再び戻す逆操作は、データを「復号する（decrypting）」または「解読する（deciphering）」と呼ばれる。
スクランブルされていないオリジナルのデータは、「平文」と呼ばれる。
スクランブルされたデータは、「暗号文」と呼ばれる。
暗号化および復号を行う多くの異なるアルゴリズムが存在する。
暗号化および復号のアルゴリズムは、一般に、２つの入力引数を取り込み、出力として、暗号化されたデータまたは復号されたデータをそれぞれ生成する。
第１の入力引数は、暗号化または復号の対象となるデータである。
第２の入力引数は、「鍵」と呼ばれる。
最良な実践では、暗号化および復号の方式の秘密性が、鍵にのみ備わっているとみなされる。
暗号化および復号は、次のように簡潔に記述することができる。
暗号文＝暗号化（平文，暗号鍵）
平文＝復号（暗号文，復号鍵）
【００５０】

［対称鍵暗号］
対称鍵暗号では、暗号化されたメッセージの送信者および受信者の双方が、同じ秘密鍵を使用する。
例えば、秘密鍵を単語「ａｐｐｌｅｓａｕｃｅ（アップルソース）」とすると、上記方程式は次のようになる。
暗号文＝暗号化（平文，「ａｐｐｌｅｓａｕｃｅ」）
平文＝復号（暗号文，「ａｐｐｌｅｓａｕｃｅ」）
【００５１】
標準的に使用される鍵は、長くランダムなビット列である。
鍵の長さは、具体的な対称暗号によって決定される。
鍵のバイナリ値は、それらの値が十分予測不可能であることを保証するように構成されるプロセスによって発生される。
【００５２】
対称暗号は、（１）ブロック暗号、および（２）ストリーム暗号の２つの基本タイプを有する。
ブロック暗号は、単一の固定サイズのブロックを一度に暗号化するか、または、復号する。
ブロックのサイズは、具体的なブロック暗号によって定められる。
ストリーム暗号は、一連のバイナリ値を発生し、この一連のバイナリ値は、暗号化を行うために平文とＸＯＲ演算されるか、または、復号を行うために暗号文とＸＯＲ演算される。
ストリーム暗号によって発生される各バイナリ値のサイズは、具体的なストリーム暗号によって定められる。
【００５３】
対称鍵暗号は、高速に実行される。
セキュリティおよび性能の理由から、対称暗号は、非常に高く位置付けられる。
あらゆる大容量の機密データの交換にとって、対称鍵暗号は、その高い性能により好ましい選択である。
対称暗号を実施するにあたっての難問は、秘密鍵の配布および管理であり、これは、「鍵配布問題（Key Distribution Problem）」とうまく名付けられている。
【００５４】
鍵配布問題、すなわち、秘密に通信を行う必要のある人の各ペアまたはプログラムの各ペアに、ネットワークを通じてユニークな秘密鍵を安全に配布するプロセスの定義は、非常に複雑である。
システム全体が、この機能を実行するためだけに構築されている。
開発されたあるシステムは、単純に、全体のサーバを、秘密鍵配布を行う信用される第三者であるタスクに専用化している。
【００５５】
［非対称鍵暗号］
非対称鍵暗号では、データの暗号化に使用される鍵は、データの復号に使用される鍵とは異なる鍵である。
暗号化および復号の双方に同じ秘密鍵を使用する対称鍵暗号と異なり、非対称鍵暗号は、２つの異なる鍵を使用する。
非対称鍵暗号の利点は、それらの鍵の一方だけをプライベート（すなわち秘密）に維持する必要があるということである。
他方の鍵は、広く知られてもよい、すなわち公開されてもよい。
この理由から、非対称鍵暗号は、「公開／秘密（public/private）」鍵暗号と広く呼ばれる。
非対称暗号は、鍵配布の問題を大幅に改善する。
非対称鍵暗号を記号で表すと、次のようになる。

暗号文＝暗号化（平文，公開鍵）
平文＝復号（暗号文，プライベート鍵）
【００５６】
最も広く使用される非対称暗号は、「ＲＳＡ」と呼ばれ、これは、その発明者達の姓の頭文字から構成される頭字語である。
ＲＳＡは、２つの秘密の素数の積を法とする非常に大きな整数で動作する。
ＲＳＡの公開鍵およびプライベート鍵は、それぞれ、このような大きな整数のペアである。
今日の良好なセキュリティでは、１０２４ビットから２０４８ビットの整数が使用されるが、アプリケーションの中には、５１２ビットの整数を使用し続けるものもある。
ＲＳＡの暗号強度は、積を２つの秘密の素数に因数分解する困難さに由来する。
【００５７】
より最近の研究では、「楕円曲線」上に位置する点からなる有限個の群が、非対称暗号についてＲＳＡよりもおそらく高い強度の基数を提供することが示された。
楕円曲線に基づく方式のセキュリティは、楕円曲線上の離散対数問題を解く困難さに由来する（詳細については、Blake、Seroussi、Smart著のElliptic Curves in Cryptography, Cambridge Univ. Press, 1999を参照されたい）。
計算に必要とされる鍵は、ＲＳＡに必要とされる鍵よりも短い。
例えば、１０２４ビットのＲＳＡの鍵と強度が同等の楕円曲線の鍵は、２００ビットよりも短い長さになる。
しかしながら、計算は、ＲＳＡの計算よりも複雑である。
非対称暗号の第１の不利な点は、はるかに多くの計算を必要とすることである。
換言すると、非対称暗号は、対照暗号よりも極めて遅い。
対称暗号および非対称暗号の相対的な強度および弱点のために、通常の実践では、鍵の配布には非対称鍵暗号が使用され、転送されるデータの大部分には、対称鍵暗号が使用される。
【００５８】
［一方向ハッシュ関数］
一方向ハッシュ関数は、任意の入力ビットストリームを固定サイズの結果に変換するアルゴリズムである。
一方向ハッシュ関数が「一方向」と呼ばれる理由は、ハッシュ関数を実行することが直接的であるからであるが、ハッシュ関数の逆を計算することはおそらく不可能であること、すなわち、固定サイズのハッシュ関数の出力のみを与えて、入力ビットストリームを計算することはおそらく不可能であることによる。
【００５９】
さらに、安全なハッシュ関数は、可能な範囲内で、関数の出力のすべてのビットの値が、入力のすべてのビットの値によって影響を受けるように設計され、かつ、所与のハッシュ出力値に対して、同一のハッシュ関数出力を生成する別の入力ビットストリームを発見することが、非常に困難となるように設計される。
安全なハッシュ関数のこれらのプロパティにより、入力ビットストリームの識別「署名」または識別「ダイジェスト（digest）」としてハッシュ出力値を使用することが合理的に可能となる。
ハッシュ関数出力は、入力ビットストリームの電子指紋（electric fingerprints）と同様である。
【００６０】
安全なハッシュ関数は、通常、２つのフェーズで動作する。
第１に、任意の長さの入力ビットストリームが、特定の方法で、複数の固定ブロックサイズにパッド（pad）される。
通常、入力ストリーム全体のビット長は、最終のパッドブロックの最後の６４ビットのダブルワードに、最終の６４ビット値として現れる。

第２に、パッドされた入力は、各ブロックから次のブロックの処理への出力値を初期入力値として使用して、ハッシュ関数により一度に１ブロックずつ処理される。
最終ブロックを処理した後の結合された結果が、ハッシュ関数の出力となる。
これを記号で表すと次のようになる。
メッセージダイジェスト＝ハッシュ関数（パッド関数（メッセージ入力ビット））
【００６１】
ハッシュ関数は、最も多目的で、かつ、広く使用される暗号ツールの１つである。
ハッシュ関数は、ほとんどすべてのインターネットプロトコルに使用される。
一般の実践では、多くのエレメントを論理的に関連付けることが必要となるときは常に、それらエレメントを表すバイト列が連結されて、全体の結果がハッシュされる。
これは、全体の順序付けられた関連だけでなく、すべてのエレメントを反映する電子指紋を生成する。
文書、データ構造体、およびコード画像のハッシュ値は、デジタル署名にとって非常に重要である。
【００６２】
［メッセージ認証コード］
メッセージ認証コード（「ＭＡＣ（Message Authentication Code）」）は、メッセージおよびデータの秘密性を保護するのではなく、それらの認証および完全性を保護するように設計される。
ＭＡＣは、メッセージまたはデータの最後に追加される値であり、この値により、メッセージまたはデータの内容が、いかなる方法によっても変更されていないことを保証することできる。
【００６３】
ＭＡＣは、対称鍵暗号と同様に、秘密鍵を使用する。
メッセージの内容を保護することは重要ではないが、その完全性を保証することだけが重要な場合には、秘密鍵が使用されて、ＭＡＣが計算され、その後、ＭＡＣはメッセージの最後に追加される。
秘密鍵を知るいかなる関係者も、受信したメッセージからＭＡＣを再計算することができ、新しく計算された値を、受信したメッセージの最後に追加されたＭＡＣと比較することができる。
新しく計算された値と、送信された値とが一致した場合には、そのメッセージは、正しく送信されていることになる。
【００６４】
ＭＡＣは、安全なＩＰ（「ＩＰｓｅｃ」）に使用されて、パケットの内容が送信中に変更されていないことを保証する。
また、ＭＡＣは、銀行間のメッセージの認証を行う転送プロトコルにも使用される。
また、ＭＡＣは、ファイルまたはデータベースに記憶されるデータに取り付けることもできる。
すべての場合において、秘密鍵を知るいかなる者も、ＭＡＣが計算されたデータが変更されていないことを検証することができる。
ＭＡＣの計算は、データに対称鍵暗号を利用するか、または、時に、データの安全なハッシュに対称鍵暗号を利用することもある。
【００６５】
［デジタル署名］
デジタル署名は、記憶されたデータまたは送信されたデータの完全性の保証を提供する点においてＭＡＣと類似している。
しかし、デジタル署名は、使用される秘密事項およびそれらの秘密事項の配布に関してＭＡＣと異なる。
【００６６】
ＭＡＣでは、ＭＡＣの計算に使用される秘密鍵を知る人からなる集合は、空集合ではな

い。
その集合の人々のいずれもが、秘密鍵により計算されたＭＡＣを検証することができる。
また、同時に、これらの人々のいずれもが、異なるメッセージまたはデータのＭＡＣを計算したり、あるいは、変更されたメッセージまたはデータのＭＡＣを偽造したりすることができる。
秘密鍵を知るすべての人は、互いに信用し合って、秘密鍵を秘密に維持すること、および、秘密鍵を適切に使用することの双方を行う必要がある。
【００６７】
ＭＡＣは、秘密事項を保持する者が一個人しかいない場合には、よく機能する。
この人は、秘密事項を使用して、自身のデータを保護することでき、その完全性を保証することができる。
また、ＭＡＣは、２人からなる組に対してもよく機能する。
これら２人は、秘密事項を明かさないように互いを信用し合い、各メッセージの完全性が危険にさらされていないことを知りつつ、メッセージを往復して送信することができる。
これらのモデルは、多くの重要な状況に適合する。
【００６８】
デジタル署名は、署名を作成できる者が一人の関係者しかいないが、署名の作成後は、誰もがその署名を検証できる場合の完全性の保護メカニズムである。
このモデルでは、一人の指定された関係者、すなわち有効な秘密事項の唯一の保持者が、署名を計算して、データの最後に追加することを担当することになる。
このことは、データに「署名する」と呼ばれる。
データに署名が行われた後、誰もが、その指定された関係者、より正確には、指定された秘密鍵が、実際に署名を計算したことを検証することができる。
このモデルは、多くの重要な取引の状況および実用的な状況に適合する。
【００６９】
デジタル署名は、非対称鍵暗号の注目すべきプロパティを使用することによって計算される。
すなわち、デジタル署名は、逆にも機能する。
非対称鍵暗号に関する前節において、我々は、記号として、以下のものを記載した。
暗号文＝暗号化（平文，公開鍵）
平文＝復号（暗号文，プライベート鍵）
しかし、これらの鍵を逆の順序で使用できることが分かる。
すなわち、次のようなる。
暗号文＝暗号化（平文，プライベート鍵）
平文＝復号（暗号文，公開鍵）
【００７０】
前者の場合には、秘密鍵の所有者が復号を行い、誰もが暗号化を行うことができる。
後者の場合には、秘密鍵の所有者が暗号化を行い、誰もが復号を行うことができる。
後者の場合が、デジタル署名に使用される。
【００７１】
デジタル署名は、単独で使用することもできるし、あるいは、秘密性を保護するために設計された暗号化と組み合わせて使用することもできる。
多くの場合、データの内容が秘密であることは重要ではなく、データが正確であることが極めて重要である。
デジタル署名は、データのすべてをプライベート鍵で暗号化することにより計算されるとは限らない。
むしろ、デジタル署名は、プライベート鍵を使用して、データの安全なハッシュを暗号化することにより計算される。

実際上、我々は、まず、データの電子指紋を取り、次に、この電子指紋をプライベート鍵で暗号化する。
記号で表すと、次のようになる。
デジタル署名＝暗号化（ハッシュ関数（メッセージ入力ビット），プライベート鍵）
電子指紋＝復号（デジタル署名，公開鍵）
【００７２】
また、デジタル署名は、２つのステップで検証も行われる。
まず、データの内容の電子指紋が、ハッシュ関数を使用して再計算される。
次に、追加されたデジタル署名が、公開鍵を使用して復号される。
再計算された電子指紋と復号されたデジタル署名とが一致すると、その署名は、検証されたことになる。
【００７３】
［乱数発生器］
乱数は、暗号アルゴリズムおよび暗号プロトコルの全体にわたって使用される。
乱数は、鍵、チャレンジ値（challenge value）、ノンス（nonce）、パスワード用のプリハッシュの追加値などに使用される。
ある形式の物理的なランダム性に基づくハードウェアデバイスが、登場し始めている。
もちろん、このようなハードウェアデバイスに関連する問題は、当該ハードウェアデバイスが正しく動作することを保証するために検査中である。
十分予測不可能な数値を計算する計算手段は存在し、それらの数値は、真の乱数の代わりに使用することができる。
このような数値は、「擬似乱数」（「ＰＲＮ（Pseudo-Random Number）」）と呼ばれる。
擬似乱数発生方法のあるものは、コンピュータシステム内のランダムなイベントの物理測定値、例えばタイプ速度、任意のマウスの動作、ディスクＩ／Ｏインタラプトの到着時間などに基づく値を使用する。
他のものは、対称暗号、または、例えば因数分解問題などの数学の難問の困難さに基づいている。
十分予測不可能な値を生成する擬似乱数発生器（「ＰＲＮＧ（Pseudo random number generator）」）は、「暗号的に強い擬似乱数発生器（「ＣＳＰＲＮＧ（Cryptographically Strong Pseudo Random Number Generator）」）と呼ばれる。
【００７４】
＜本発明の安全なプラットフォームカーネルの実施の形態＞
図１４は、ＩＡ－６４プロセッサに基づくマシン上にオペレーティングシステムを直接レイヤ化したものを示している。
この直接的なレイヤ化のアプローチでは、オペレーティングシステム１４０２は、ＩＡ－６４アーキテクチャ特有のインタフェース１４０４を使用する。
このインタフェース１４０４は、オペレーティングシステムのサービスを介してアプリケーションプログラムの管理および分配を行うコンピュータリソースとして、ＩＡ－６４プロセッサに基づくマシン１４０６によって提供される。
ＩＡ－６４アーキテクチャ特有のインタフェース１４０４は、非特権命令および特権命令と、非特権レジスタおよび特権レジスタと、ハードウェア割り込みメカニズムと、ファームウェアインタフェースとを備える。
しかしながら、ＩＡ－６４ハードウェアインタフェース上へのオペレーティングシステムの直接的なレイヤ化を使用することにより、現在のオペレーティングシステムに本来備わっている多くのセキュリティ問題およびセキュリティ欠陥を、ＩＡ－６４アーキテクチャの機能を使用して容易に対処することはできない。
一例として、Ｉ／Ｏドライバが、継続してオペレーティングシステム内に組み込まれ、ＰＬ０で実行されている場合に、図６に示した問題は、このコンピュータシステムの潜在的な脆弱性のまま残る。

オペレーティングシステムは、比較的大きなプログラムであり、正確さおよび安全な動作のために検証することは実際には不可能である。

【００７５】

本発明の一実施の形態は、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースである。

この結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースは、１つまたは２つ以上のコンピュータオペレーティングシステムおよびカスタマイズされた制御プログラムとインタフェースする安全なレイヤを提供するものである。

結合されたハードウェアおよびソフトウェアの安全なプラットフォーム（「ＳＰ」）は、オペレーティングシステムまたはカスタマイズされた制御プログラムが、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェース（「ＳＰＩ」）を介して、非特権機械語命令および非特権レジスタに直接アクセスすることを可能にすると共に、特権機械語命令および特権レジスタを公開することなく、オペレーティングシステムまたはカスタマイズされた制御プログラムにハードウェアの制御を提供するソフトウェアルーチンの起動を可能にする。

その上、ＳＰおよびＳＰＩは、一組の安全なリポジトリサービスを提供する。

このリポジトリサービスには、特に、暗号化サービスおよび復号サービスと、セキュリティポリシー管理と、例えば平文による暗号鍵などの内部で生成された秘密データを使用する他のセキュリティ関連サービスとが含まれる。

この秘密データは、オペレーティングシステムおよびカスタマイズされた制御プログラムに公開することもできないし、結合されたハードウェアおよびソフトウェアの安全なプラットフォームのソフトウェア部分の他のルーチンに公開することもできない。

ＳＰＩは、ハードウェアだけでなく、ファームウェアインタフェースおよびソフトウェアルーチンの拡張可能セットをも含むマシンに対するマシンインタフェースと考えることができる。

このハードウェアには、１つまたは２つ以上のプロセッサ、バス、メモリコンポーネント、および図１に示す他のハードウェアコンポーネントが含まれる。

このように、ＳＰＩは、ＳＰに対するインタフェースであり、ハードウェア、ファームウェア、およびソフトウェアの複合マシンに対するインタフェースである。

また、ＳＰＩは、オペレーティングシステムおよびカスタマイズされた制御プログラムの下に位置する新しいタイプのソフトウェアレイヤによって提供されるインタフェースと考えることもできる。

【００７６】

図１５は、コンピュータシステム内のＳＰを示すブロック図である。

このコンピュータシステムは、ＳＰならびに１つまたは２つ以上のオペレーティングシステムおよびカスタマイズされた制御プログラムを備える。

ＳＰＩ１５０２は、ＳＰのソフトウェアレイヤ１５０４と、１つまたは２つ以上のオペレーティングシステムおよび／またはカスタマイズされた制御プログラム１５０６との間に位置する。

ＳＰは、ソフトウェアレイヤ１５０４と、ハードウェアおよびファームウェアのプラットフォームインタフェース１５０８と、ハードウェアおよびファームウェアのプラットフォーム１５１０とを備える。

上述したように、ハードウェアインタフェース１５０８に設けられた非特権命令および非特権レジスタは、図１５の縦の矢印１５１２によって示すように、ＳＰＩ１５０２によって直接公開される。

縦の破線１５１４の右にある、ハードウェアインタフェースの他のコンポーネントは、公開されない。

その代わりに、これらのハードウェアインタフェースコンポーネントを通じて提供される機能は、ＳＰＩの安全なプラットフォームサービスコンポーネント１５１６および安全なリポジトリサービスコンポーネント１５１８を介して、オペレーティングシステムおよ

びカスタマイズされた制御プログラムに利用可能にされる。

【００７７】

ＳＰが提供する機能に加えて、ＳＰが、オペレーティングシステムまたは従来の仮想マシンと全く異なること、および、ＳＰＩが、オペレーティングシステムインタフェースまたは仮想マシンインタフェースと異なることを強調することは重要である。

仮想マシンは、特権命令および特権レジスタと、下に位置するハードウェアプラットフォームのファームウェアインタフェースとを直接公開するか、または、シミュレーションする、下に位置するハードウェアプラットフォームのソフトウェアの抽象化である。

仮想マシンは、例えばオペレーティングシステムの開発などの多くの状況で役立つ。

しかし、ＳＰは、仮想マシンインタフェースとは異なる。

ＳＰＩは、特権命令および特権レジスタならびにファームウェアインタフェースへの、上に位置するオペレーティングシステムによるアクセスを故意に妨げる。

さらに、ＳＰメカニズムは、拡張可能な良質な組の安全なサービス、例えば暗号サービスなどを提供する。

これらの安全なサービスは、仮想マシンによって提供される機能の範囲をかなり越えて拡張されているものである。

ＳＰの安全なサービスは、それらのコードイメージ画像およびコードデータを、上に位置するオペレーティングシステムおよびカスタム制御プログラムから完全に区画化されて隔離することにより、安全なものとされている。

この区画化および隔離は、オペレーティングシステムおよびカスタム制御プログラムが、特権命令および特権レジスタならびにファームウェアインタフェースへのいかなる直接アクセスも持たないとの理由によってのみ可能である。

したがって、例えば、ＳＰ暗号サービスは、暗号鍵の実際の値、および、暗号鍵から導出される鍵とする材料（keying material）を、すべてのオペレーティングシステム、カスタム制御プログラム、およびアプリケーションプログラムから完全に隠蔽することができる。

一方、それにもかかわらず、それらのオペレーティングシステム、カスタム制御プログラム、およびアプリケーションプログラムは、その暗号サービスを自由に使用することが許可されている。

【００７８】

ＳＰは、オペレーティングシステムの安全なプラットフォームではあるが、ＳＰは、単独では、ＳＰ上に構築されるコンピュータシステムのセキュリティを保証しない。

オペレーティングシステムなどのシステム制御プログラムと異なり、ＳＰは、非特権命令および非特権レジスタに加えて、一組の呼び出し可能なメカニズムのみを提供する。

これらのＳＰメカニズムは、基本的なビルディングブロックを提供し、このビルディングブロックから、後述するように、安全なシステムを構築することができる。

ＳＰメカニズムにより、オペレーティングシステムは、下に位置するハードウェアの十分な制御を実行することが可能になり、これによって、全てのオペレーティングシステムのサービスをアプリケーションプログラムおよびユーザに提供することが可能になる。

しかし、ＳＰＩは、いかなる特権命令も特権レジスタもファームウェアインタフェースも、上に位置するオペレーティングシステムやカスタム制御プログラムに公開することなく、この機能を提供する。

本発明のＩＡ－６４の実施の形態では、内部的に、ＳＰメカニズムは、ウィルス攻撃および侵入攻撃に対する脆弱性を縮小するように設計された方法で、ＩＡ－６４アーキテクチャの保護機能を一貫して利用する。

使用される手法は、次のものを含む。

すなわち、すべての実行可能コードイメージは、設定されて実行可能になる前に、まず、それらのデジタル署名をチェックすることにより、正当であることが確認されることと、実行可能コードイメージは、実行可能および書き込み可能の双方に同時に許可されることは決してないことと、スタックおよびデータ領域は、読み出しまたは書き込みは可能で

あるが、決して実行することができないことと、ＳＰメカニズムは、特権レベルおよび保護キーの使用によって区画化されて、これにより、各メカニズムのコードおよびデータは、上に位置するすべてのオペレーティングシステムおよびカスタム制御プログラムから隔離され、かつ、他のすべてのＳＰメカニズムから隔離されることとを含む。

ＳＰの上に位置するオペレーティングシステムおよびカスタム制御プログラムも、同様のメモリ保護手法を採用することができ、これにより、自身のセキュリティを改善することができる。

しかし、実際には、あらゆるオペレーティングシステムまたはカスタム制御プログラムは、ＳＰインタフェースによって保護を行うツールを提供されているにもかかわらず、自身を適切に保護できないことがある。

【００７９】

ＳＰは、システムのリソースを、１つまたは２つ以上の重複しない、相互に隔離されたパーティションに編成する。

このパーティションは、「ドメイン」と呼ばれる。

各ドメインは、オペレーティングシステムまたはカスタム制御プログラムの制御下で動作する。

ＳＰは、ある範囲の仮想メモリアドレスと、ある範囲の物理メモリアドレスと、多数の論理ＣＰＵおよび物理ＣＰＵと、一組のＩ／Ｏデバイスと、一組の識別値とを各ドメインに割り当てる。

ＳＰサービスを提供するために、ＳＰソフトウェアレイヤは、ハードウェアレイヤによって一般に提供される非特権命令、非特権レジスタ、および他のプロセッサ機能に加えて、ＳＰハードウェアレイヤからのある特定の機能を必要とする。

第１の重要な追加機能は、さまざまなプロセスを実行できる少なくとも４つの特権レベルを設けることである。

ＳＰのソフトウェアレイヤは、ソフトウェアレイヤの独占的な使用のために、最も高い２つの特権レベルＰＬ０およびＰＬ１を確保する。

少なくとも３番目の特権レベルＰＬ２は、オペレーティングシステムのルーチンおよびカスタム制御プログラム用に残され、４番目の特権レベルＰＬ３は、アプリケーションプログラム用に残される。

５つ以上の特権レベルを提供するハードウェアも、使用することができ、この場合、付加的な特権レベルは、必要とされないか、または、オペレーティングシステムが使用するためにオペレーティングシステムレイヤに割り当てられる。

本発明の一実施の形態では、ＰＬ０で実行されるＳＰメカニズムは、安全なプラットフォームカーネル（「ＳＰＫ（Secure Platform Kernel）」）と呼ばれ、ＰＬ１で実行されるＳＰコードは、安全なプラットフォームグローバルサービス（「ＳＰＧＳ（Secure Platform Global Service）」）と呼ばれる。

ドメイン内で実行されるオペレーティングシステムまたはカスタム制御プログラムは、安全なプラットフォームサービスを起動することによって、そのドメインに割り当てられた論理リソースおよび物理リソースを管理する。

このドメインのオペレーティングシステムまたはカスタム制御プログラムは、ＳＰインタフェースより上の最も高い特権レベルで動作し、ＳＰインタフェースは、ＳＰＧＳおよびＳＰＫの動作特権レベルよりも低い特権レベルである。

【００８０】

第２の重要な追加機能は、少なくとも２次元のメモリ分割制御が、オペレーティングシステムに基づくドメイン間でのメモリの分割および各ドメイン内でのメモリの分割と、ＳＰ、オペレーティングシステム、およびユーザレベルのアプリケーションプログラムのプロセス間でのメモリの分割とを行うＳＰに利用可能となることである。

図１６は、この第２のメモリ分割機能を示している。

第２のメモリ分割機能はＳＰによって必要とされ、Intel（登録商標）のＩＡ－６４プロセッサアーキテクチャなどの現代のプロセッサアーキテクチャによって提供される。

図１６では、全仮想メモリ空間が、大きな長方形１６０２によって表されている。
全仮想メモリ空間の一部１６０４は、大きな長方形１６０２の横方向の全範囲にわたる横方向の長方形として示され、ＳＰのソフトウェアレイヤのみにアクセス可能である。
全仮想メモリ空間の第２の部分は、大きな長方形１６０２の横方向の全範囲にわたる横方向の長方形１６０６として示され、各ドメイン内において、そのドメインを管理するオペレーティングシステムによって使用される。
全仮想メモリ空間の第３の部分は、大きな長方形１６０２の横方向の全範囲にわたる横方向の長方形１６０８として示され、各ドメイン内の個々のアプリケーションプログラムおよび他のユーザレベルのプロセスによって使用される。
図９を参照して上述したように、ＴＬＢエントリのメモリページと関連付けられた特権レベルは、異なる特権レベルで実行されるプロセス間でメモリを分割することを可能にする。
しかし、この特権レベルは、ドメインおよび各ドメイン内で動作するアプリケーションの横方向の分割には十分ではない。
このように、ＳＰのプロセス、オペレーティングシステムのプロセス、およびユーザレベルのプロセスが、異なる特権レベルで実行されているという条件において、横方向のＳＰのパーティション１６０４内、横方向のオペレーティングシステムのパーティション１６０６内、および横方向のユーザレベルのパーティション１６０８内のメモリの分割を、特権レベルに基づくメモリの分割を使用して部分的に達成することができる。
しかしながら、特権レベルに基づくメモリの分割は、例えば、ユーザレベルのプロセスに割り当てられたメモリの領域を、オペレーティングシステムのプロセスによるアクセスから保護することができない。
【００８１】
ＳＰは、多数の同時に実行されるオペレーティングシステムにサービスを提供する。
ここで、各オペレーティングシステムは、多数の種々のユーザレベルのプロセスを管理する。
あるオペレーティングシステムに割り当てられたメモリを、別のオペレーティングシステムによるアクセスから保護するために、オペレーティングシステムのメモリ空間１６０６およびユーザレベルのメモリ空間１６０８は、さらにドメインに分割される。
単一のオペレーティングシステムと、そのオペレーティングシステムによって管理されるすべてのユーザレベルのプロセスとが、共にドメインを備え、各ドメインは、仮想メモリの別々の部分を割り当てられる。
したがって、図１６において、第１のオペレーティングシステムは、オペレーティングシステムのメモリ空間１６０６の第１の部分１６１０を使用し、第２のオペレーティングシステムは、オペレーティングシステムの全メモリ空間１６０６の第２の部分１６１２を使用し、第３のオペレーティングシステムは、オペレーティングシステムの全メモリ空間１６０６の第３の部分１６１４を使用する。
また、ユーザレベルのメモリ空間１６０８もドメインに分割される。
その結果、図１６において、仮想メモリ空間の小部分１６１６～１６１９、およびオペレーティングシステムのメモリ空間の第１の部分１６１０は、共に、第１のドメインであるドメイン０に割り当てられたメモリ空間を構成する。
ユーザレベルのメモリ部分１６２０～１６２３およびオペレーティングシステムのメモリ部分１６１２は、共に、第２のドメインであるドメイン１に割り当てられたメモリを構成する。
ユーザレベルのメモリ部分１６２４～１６２７およびオペレーティングシステムのメモリ部分１６１４は、共に、第３のドメインであるドメイン２に割り当てられたメモリ空間を構成する。
もちろん、実際のコンピュータシステムでは、特定のドメインに割り当てられたメモリ量は、他のドメインに割り当てられたメモリ量と異なることがあり、異なる個数のユーザプロセスが、異なるドメイン内で実行されることがある。

【００８２】
本発明の一実施の形態では、ＳＰは、図９を参照して上述した領域レジスタおよび領域識別子を使用して、メモリをドメインについて分割する。
したがって、共に１つのドメインを構成する各オペレーティングシステムおよびそのオペレーティングシステムによって管理されるプロセスは、特定の組の領域識別子を割り当てられ、それらを独占的に使用する。
図１６に示すように、領域識別子に基づくメモリ分割は、２次元のメモリ分割メカニズムのうちの１次元１６２８と考えることができる。
【００８３】
ドメイン間の領域識別子に基づくメモリ分割に加えて、ＳＰは第２の分割メカニズムを必要とする。
この第２の分割メカニズムは、オペレーティングシステムおよびユーザレベルのプロセスに割り当てられたメモリからＳＰに割り当てられたメモリを分割するだけでなく、オペレーティングシステムとユーザレベルのプロセスとの間で、ドメイン内のメモリを分割するものである。
図１６では、この第２の分割メカニズムは、メモリ分割の第２の次元１６３０として表されている。
説明されたＳＰの実施の形態では、この第２のメモリ分割次元は、ＳＰソフトウェアレイヤおよびドメインオペレーティングシステム用の領域識別子を含む一定の個数の領域レジスタを割り当てることによって提供される。
残りの領域レジスタは、各ユーザレベルのプロセス用の領域識別子を含む。
ＩＡ－６４の保護キーは、メモリ領域１６０４、１６０６、および１６０８のすべてにわたって、さらに細かな粒度の区画化を提供する。
例えば、保護キーは、ＳＰＫレイヤ内のメカニズムを区画化するのに使用される。
ＳＰは、すべてのレジスタ、物理メモリ、および保護キーを制御するので、ＳＰは、物理メモリと１組の識別子とを、各ドメインの独占的使用のために割り当てることができる。
保護キーに基づくメモリ分割を使用することにより、原則として、メモリの特定の領域を任意のプロセスに割り当てることができ、その領域は、そのプロセスによって独占的にアクセスされる。
【００８４】
このように、領域識別子によるメモリの横方向の分割、および、領域レジスタによるメモリの横方向の分割を使用すると、Intel（登録商標）のＩＡ－６４アーキテクチャによって提供されるメモリ保護メカニズムにより、ＳＰは、メモリを分割して、ＳＰのみによって独占的にアクセス可能なメモリの領域と、ＳＰおよびある他のオペレーティングシステムまたはユーザレベルのエンティティのみによって独占的にアクセス可能なメモリの領域と、一組のオペレーティングシステムおよびユーザレベルのエンティティならびにＳＰによってアクセス可能なメモリの領域とを提供することができる。
Intel（登録商標）のＩＡ－６４アーキテクチャによって提供される保護キーメカニズムは、さらに細かな粒度を提供する。
現在のコンピュータシステムでは、特権レベル０で実行されるオペレーティングシステム、または、特権レベル０で実行される必要があるＩ／Ｏドライバもしくは他の居住者のいずれも、状態を操作して、コンピュータシステム内のすべてのメモリへアクセスすることができる。
これと対照的に、図１６に示す２次元メモリ分割方式を使用するＳＰに基づくシステムでは、ＳＰは、システム内の多くの同時並行実行中のオペレーティングシステムの１つにのみアクセス可能なメモリ領域と、特定のユーザレベルのプロセスにのみアクセス可能なメモリ領域とを提供することができる。
例えば、図１６では、ＳＰは、保護キーの割り当てを制御することにより、特定のＳＰメカニズムのみがアクセス可能なメモリ領域、例えばメモリ領域１６３２を自ら割り当て

ることができる。
このようなＳＰメカニズム独占領域は、暗号鍵の実際の値を記憶するためにＳＰによって使用され、これにより、いかなるオペレーティングシステムもユーザレベルのプロセスも、秘密鍵の材料にアクセスすることができない。
もちろん、ＳＰＫは、特権レベル０で動作するので、悪意のＳＰＫコードはメモリ保護を回避できることになる。
ＳＰＫコードを信用する基本は、
（１）限られた個数のＳＰＫメカニズムが存在すること、
（２）各ＳＰＫメカニズムは、比較的小さく、分かりやすく、独立しており、かつ、自己充足していること、
（３）ＳＰＫソースコードは、独立した専門家によって公開され、詳しく調べられること、
（４）ＳＰＫコードの開発プロセスは、複数の人達による詳細なコード検査を含むこと、および
（５）ＳＰＫコードイメージは、実行前にデジタル署名され、検証されること
を含む。
【００８５】
多くの現在利用可能コンピュータシステムでは、Ｉ／Ｏデバイスのコントローラは、物理メモリに直接アクセスすることができる。
ＳＰは、ＳＰ以外のどのルーチンによる物理メモリへの直接アクセスも、外部ハードウェアによる物理メモリへの直接アクセスも許可しない。
その代わり、ＳＰのハードウェアコンポーネントは、Ｉ／Ｏハードウェアにアクセス可能な物理アドレスをハードウェアバスアダプタおよびルーティングコンポーネント内に制限するための仮想アドレスから物理アドレスへの変換を提供するメカニズム、または他の手段を含む必要がある。
これにより、Ｉ／Ｏコントローラは、その目的でＳＰにより割り当てられたメモリ領域のみとデータを交換することができ、Ｉ／Ｏコントローラに割り当てられたメモリ領域以外のメモリにアクセスすることはできない。
【００８６】
ＳＰのソフトウェアレイヤ（図１６の１６０４）は、ハードウェアファームウェアインタフェース（図１５の１５０８）とインタフェースし、また、１つまたは２つ以上のオペレーティングシステムとＳＰＩ（図１５の１５０２）を介してインタフェースする。
ＳＰのソフトウェアレイヤは、それ自体少なくとも２つのレイヤに分割され、その少なくとも１つは内部インタフェースである。
図１７は、ＳＰのソフトウェアレイヤのブロック図である。
ＳＰのソフトウェアレイヤ１７０２は、破線１７０４に沿って、安全なプラットフォームグローバルサービス（「ＳＰＧＳ」）レイヤ１７０６と、安全なプラットフォームカーネル（「ＳＰＫ」）レイヤ１７０８とに分割される。
本発明の一実施の形態では、ＳＰＧＳ１７０６のルーチンは、特権レベルＰＬ１で実行されるのに対して、ＳＰＫ１７０８のルーチンはＰＬ０で実行される。
この内部分割によって、比較的小さなＳＰＫを構成することができる。
この比較的小さなＳＰＫは、形式的検証方法であるコードウォークスルーによって検証することができ、最終的には、セキュリティおよびコンピュータオペレーティングシステムの団体内の専門家への公開により検証することができる。
このように、ＳＰＫの最も高い特権コードを、可能な限りバグのない強固なものとすることができ、システムの他のすべての部分は、トロイの木馬や、他のプロセスに割り当てられた保護メモリ領域に有害にアクセスして当該領域を変更できるバグ生成型ＰＬ０コードから十分に保護される。
【００８７】
ＳＰＧＳインタフェースは、論理的に、次の３つの主なサブインタフェースを備える。

（１）ＳＰプラットフォーム管理サービス１７１０。ＳＰプラットフォーム管理サービス１７１０は、システムの論理リソースおよび物理リソースを環境設定する手段と、ドメインオペレーティングシステムまたはカスタム制御プログラムがマシンの特権状態に作用する手段とを提供する。
（２）ＳＰセキュリティ管理サービス１７１２。ＳＰセキュリティ管理サービス１７１２は、ＳＰの暗号機能および安全な記憶機能を提供する。これらの機能には、例えば、さまざまなシステムコンポーネントを隔離する個別の保護キーによって、それぞれ保護可能な個別の機能コンポーネントが含まれる。
（３）ＳＰ拡張部１７１４。ＳＰ拡張部１７１４は、ＳＰの機能要求を次第に拡張することを可能にし、例えば暗号アルゴリズムの追加といったＳＰへの機能の追加を可能にする。図１７では、これらのサブインタフェースの分割は、関連した議論を明確にするためにＳＰＫに対して実行される。ＳＰインタフェースは、全体的に、ＳＰＫメカニズムに加えて、ＳＰＫメカニズムを順に起動するＳＰＧＳサービスからも構成することができるし、ＳＰＫメカニズムを必要に応じて起動するＳＰＧＳサービスからも構成することができる。

【００８８】
ＳＰプラットフォーム管理サービスのサブインタフェース１７１０は、
（１）ＳＰプラットフォームサービス１７１６、
（２）ドメイン制御サービス１７１８、
（３）プロセッサ制御サービス１７２０、
（４）プラットフォームポリシー制御サービス１７２２、ならびに
（５）ドメイン間サービスおよびドメイン内サービス１７２４
を含む。
ＳＰプラットフォームサービス１７１６は、特権動作を必要とする機能、ＳＰリソースを伴う機能、またはセキュリティ依存の機能を実行するオペレーティングシステムまたはカスタム制御プログラムによって起動される。
ドメイン制御サービス１７１８は、ドメインの生成、環境設定、再初期化、およびシャットダウンを提供する。
プロセッサ制御サービス１７２０は、論理ＣＰＵおよび物理ＣＰＵの環境設定およびスケジューリングを提供する。
プラットフォームポリシー制御サービス１７２２は、例えばシステムリソースの割り当てなどのプラットフォームポリシーの仕様、保存、および適用を提供する。
ドメイン間サービスおよびドメイン内サービス１７２４は、ドメイン内の論理ＣＰＵ間およびドメイン間の論理ＣＰＵ間でのイベントの伝達、ドメイン間でのリソースの共有、ならびに高速ネットワーク接続またはデータ交換のためのデータ転送の提供を提供する。
ＳＰプラットフォーム管理サービス１７１０は、後述する１つまたは２つ以上のＳＰメカニズムを順に起動することができる。
ＳＰプラットフォーム管理サービスは、ＳＰメカニズムを呼び出す前にパラメータが正当であることを確認し、かつ呼び出しを認証する。
いくつかの具体的なＳＰプラットフォームサービスルーチンインタフェースのさらに詳細な内容は、付録Ａに提供される。

【００８９】
ＳＰセキュリティ管理サービスのサブインタフェース１７１２は、
（１）安全なリポジトリサービス１７２６、および
（２）呼び出し側認証サービス１７２８
を含む。
安全なリポジトリサービス１７２６は、オペレーティングシステム、カスタム制御プログラム、またはユーザアプリケーションプログラムによって起動され、システムの最も高い特権レベルＰＬ０で実行される特定のメカニズムにのみ利用可能な秘密事項を使用する機能を提供する。

これらの特定のメカニズムには、
（ａ）１つまたは２つ以上の暗号アルゴリズムを使用してデータの暗号化および復号を提供する暗号サービス、および
（ｂ）安全なデータサービス
が含まれる。
呼び出し側認証サービス１７２８は、ドメインが、ＳＰのみにアクセス可能な状態に維持される安全なタグをシステムオブジェクトと関連付ける手段と、ドメインが、そのタグを使用して、プラットフォームおよび安全なリポジトリサービスへのアクセスを認証する手段とを提供する。
上記システムオブジェクトには、特定のユーザ、アプリケーションプログラム、オペレーティングシステムコンポーネント、ディレクトリ、ファイルおよびディスパッチ可能オブジェクトが含まれる。
これらの機能によって使用される秘密事項には、暗号鍵の材料が含まれる。
鍵を隠すことにより、ドメインオペレーティングシステム、カスタム制御プログラム、またはユーザアプリケーションプログラムが、たとえ攻撃者によって危険にさらされても、鍵を使用するドメインからでさえも、鍵の材料を安全な状態に維持することができる。
安全なシステムは、特定のインタフェースを通じて行う場合を除いて、アクセスも変更もできないリポジトリ、例えば安全なログを確立する必要がある。
ＳＰは、ドメインオペレーティングシステムまたはカスタム制御プログラムが、安全なリポジトリに貢献でき、かつ照会できる基本的な機能を提供する。
具体的なＳＰリポジトリサービスのルーチンインタフェースのさらに詳細な内容は、付録Ｂに提供される。

【００９０】
ＳＰプラットフォームサービス１７１６は、メモリ管理サービス、プロセス管理サービス、および割り込み管理サービスを含む。
メモリ管理サービスは、仮想アドレスと物理アドレスとのマッピングの生成、操作、および削除の制御と、プロセッサＴＬＢ（図９の９１４）および仮想ページテーブル（図１０の１００４）の制御とを提供する。
ＩＡ－６４に基づくＳＰの実施は、領域識別子の重複しない組を各ドメインに割り当てる。
これにより、拡張された仮想アドレスは、すべてのドメインにわたってユニークなものとなる。
また、各ドメインには、重複しない範囲の物理メモリも割り当てられる。
これにより、あるドメインが、別のドメインの物理メモリにアクセスできないことが保証される。
また、メモリのすべてのページは保護キーとも関連付けられる。
これにより、メモリアクセスの区画化または他のさらに細かな粒度の制御が可能になる。

【００９１】
プロセス管理サービスは、物理プロセッサ上でのプロセスの実行の制御を提供する。
ＳＰは、ディスパッチ可能オブジェクトおよび論理ＣＰＵという２つのプロセッサに関連した抽象化を実施する。
ディスパッチ可能オブジェクトは保存された制御のスレッドであり、ドメイン内の関連付けられたアドレス指定コンテキストである。
ディスパッチ可能オブジェクトは、制御のスレッドが以前に割り込みを受けたポイントで実行を再開するのに必要な状態を含む。
アドレス指定コンテキストは、関連付けられた領域ＩＤおよび保護キーの組であり、ドメインの組から引き出される。
これらの値は、ディスパッチ可能オブジェクトの合法的な拡張仮想アドレス指定コンテキストを決定する。

オペレーティングシステムまたはカスタム制御プログラムは、現在の実行状態をあるディスパッチ可能オブジェクトに保存し、別のディスパッチ可能オブジェクトから実行状態を復元することにより、ディスパッチ可能オブジェクトをスケジューリングする。

【００９２】

論理ＣＰＵは、１つまたは２つ以上の物理プロセッサのドメインのビューである。

ドメインは、１つまたは２つ以上の論理プロセッサを持つことができる。

あるドメインの論理プロセッサの個数、および、すべてのドメインの論理プロセッサの総数は、システムの物理プロセッサよりも少なくてもよいし、等しくてもよいし、あるいは多くてもよい。

論理プロセッサは、ＳＰによってスケジューリングされて、物理プロセッサにディスパッチされる。

オペレーティングシステムまたはカスタム制御プログラムは、一般にこのスケジューリングに気付かない。

これにより、ＯＳが関与することなく、ＣＰＵの環境設定に対する変更が可能となる。

【００９３】

ＳＰ割り込み管理サービスは、直接割り込みの処理、および、その後のオペレーティングシステムまたはカスタム制御プログラムへの割り込みの通知のディスパッチを提供する。

割り込みは、特定の処理ルーチンへの制御の転送を引き起こす任意のイベントである。

ＩＡ－６４アーキテクチャによって定義されるように、割り込みは次の４つのカテゴリーに分割される。

（１）アボート：アボートは、プロセッサのリセットまたは内部マシンの故障のいずれかで発生する。

（２）フォルト：フォルトは、実行できないか、または実行されるべきでない動作を指定する命令により発生する同期インタラプト、あるいはその命令が実行される前に、システムの介入が必要とされる同期インタラプトである。

（３）トラップ：トラップは、完了した命令がシステムの介入を必要とする場合に発生する同期インタラプトである。

（４）非同期インタラプト：非同期インタラプトは、外部エンティティまたはプロセッサとは独立したエンティティが発生する非同期のイベントまたは信号を表す。

このイベントには、Ｉ／Ｏデバイスに関連したイベントが含まれる。

非同期インタラプトは、優先クラス内における非同期インタラプトの発生時間順でハンドラに配布される必要がある。

図１７では、アボート、フォルト、およびトラップが、「例外」と呼ばれ、非同期インタラプトが「インタラプト」と呼ばれる。

【００９４】

さまざまなＰＳＫメカニズムが、図１７の破線１７０４と論理的に一致した内部ＳＰＫ／ＳＰＧＳインタフェースを通じてＳＰＧＳに提供される。

ＳＰＫメカニズムは、呼び出しを介してＳＰＧＳによってアクセスされる。

ＳＰＫは、呼び出し側プロセスの状況の中で一般に実行される一組のメカニズムを備える。

ＳＰＫメカニズムは、

（１）メモリ管理メカニズム１７３０、

（２）プロセッサディスパッチメカニズム１７３２、

（３）例外処理メカニズム１７３４、

（４）インタラプト処理メカニズム１７３６、

（５）デバッグおよび監視メカニズム１７３８、

（６）暗号メカニズムおよび記憶装置１７４０、

（７）安全なリポジトリメカニズムおよび記憶装置１７４２

を含む。

【００９５】
メモリ管理メカニズムは、ＴＬＢ、ＶＨＰＴ、および他のページテーブルへの変換を挿入する特権動作を実行する。
プロセッサディスパッチメカニズムは、ディスパッチ可能オブジェクトおよび論理ＣＰＵへの状態の保存、ならびに、ディスパッチ可能オブジェクトおよび論理ＣＰＵからの状態の復元を処理する。
これらのメカニズムは、例えばアドレス指定コンテキストなどの状態の多くが特権を有することから必要とされる。
例外処理メカニズムは、フォルト、トラップ、特別なハードウェアイベント、およびＳＰ発生イベントを取り扱う。
特別なハードウェアイベントは、例えば、マシンチェックやハードウェア環境設定の変更などである。
オペレーティングシステムまたはカスタム制御プログラムは、ＳＰＩを介して例外ハンドラルーチンを登録し、例外が発生した時に例外ハンドラルーチンが起動されるようにする。
例外が発生すると、ＳＰは最小量のプロセッサの状態を保存し、次に、登録された例外ハンドラに制御を転送する。
フォルトおよびトラップの中には、性能のため、または、ドメインオペレーティングシステムもしくはカスタム制御プログラムに対するインタフェースを簡略にするために、ＳＰによって直接処理されるものがある。
オペレーティングシステムまたはカスタム制御プログラムが、インタラプトイベントの通知を受信する必要がある場合には、オペレーティングシステムまたはカスタム制御プログラムは、対応するドメインにおいて発生が予想される各インタラプトに対して、ＳＰＩを介してインタラプトハンドラルーチンを登録する。
インタラプトが発生すると、ＳＰは、予め構成された特別の実行環境で、関連付けられたハンドラルーチンを起動し、高速のインタラプト応答を可能にする。
インタラプトは、時間順を維持した方法で各オペレーティングシステムまたは各カスタム制御プログラムへ配信するために、ＳＰによってキューに入れられる。
ＳＰによって提供される１つの特別なインタラプトは、タイマチック（timer tic）であり、これは一定時間間隔で発生するインタラプトである。
各オペレーティングシステムまたは各カスタム制御プログラムは、この一定時間間隔の任意の整数倍の時間間隔で通知を受けることを指定することができる。
【００９６】
デバッグおよび監視メカニズムは、マシンレジスタの集まりを使用して、デバッグおよび性能監視を援助する。
このＳＰメカニズムは、オペレーティングシステムまたはカスタム制御プログラムに必要な特権アクセスを実行して、デバッグおよび性能監視のマシンリソースを利用する。
暗号メカニズムおよび安全な記憶メカニズムは、暗号機能、復号機能、および鍵管理機能を提供する。
暗号アルゴリズムには、標準的な対称鍵ブロック暗号および対称鍵ストリーム暗号、公開鍵暗号、ならびに安全なハッシュ関数が含まれる。
鍵管理機能には、鍵生成、鍵インポート、鍵エクスポート、および鍵の材料準備が含まれる。
安全なリポジトリメカニズムおよび記憶メカニズムは、秘密データを、関連付けられたアクセス方法と共に含むソフトウェアオブジェクトであって、システムの最も高い特権レベルに存在するソフトウェアオブジェクトとみなすことができる。
このメカニズムの組は、拡張可能である。
【００９７】
＜安全なブートプロセス＞
適切な構造化および実施により、オペレーティングシステムが、安全な基盤をユーザレ

ベルのアプリケーションおよびプロセスに提供できるように、ＳＰは、システムのコンポーネントを安全にできるメカニズムを提供する。
ＳＰの重要なコンポーネントは、安全なブート手順である。
この安全ブート手順は、最初の電源投入から、１つまたは２つ以上のオペレーティングシステムのプロセスの開始までの一連の既知の正しいシステム状態を保証するものである。
図１８は、安全なブートプロセスに関連したシステムのハードウェアコンポーネントのブロック図である。
関連するハードウェアコンポーネントは、
（１）プロセッサ１８０２、
（２）読み出し専用メモリ（「ＲＯＭ」）１８０６、
（３）メインメモリ１８１６、
（４）マスストレージデバイス１８１８、ならびに
（５）さまざまな内部バス１８２４～１８２９およびバスブリッジ／ルータ／コントローラ１８３０～１８３２
を含む。
プロセッサ１８０２は、初期ファームウェア正当性確認メカニズム１８０４を含む。
この初期ファームウェア正当性確認メカニズム１８０４は、アクティブロジックおよび記憶されたファームウェア命令を備える。
ＲＯＭ１８０６は、デジタル署名されたファームウェア命令の組１８０８～１８１１を含む（デジタル署名は、小さなフィールド１８１２～１８１５として示される）。
これらのファームウェア命令の組は、ｎ個の異なるファームウェア命令の組を表し、各組は、安全なブートプロセスの異なるフェーズに関連する。
マスストレージデバイス１８１８は、ＯＳローダプログラムおよび少なくとも１つのＯＳ１８２０の命令と共に、当該ＯＳローダのデジタル署名１８２２および当該ＯＳのデジタル署名１８２３を記憶する。
さまざまな内部バス１８２４～１８２９およびバスブリッジ／ルータ／コントローラ１８３０～１８３２は、上記コンポーネント間のデータ通信を容易にする。
【００９８】
図１９は、ＳＰの安全なブートプロセスコンポーネントの一実施の形態を表す安全なブートプロセス「ｓｅｃｕｒｅ＿ｂｏｏｔ（安全なブート）」の制御流れ図である。
図１９では、ｓｅｃｕｒｅ＿ｂｏｏｔは、シングルプロセッサシステム用に説明されるが、マルチプロセッサシステムをブートする代わりとなるｓｅｃｕｒｅ＿ｂｏｏｔルーチンは、図１９を参照して説明されるｓｅｃｕｒｅ＿ｂｏｏｔルーチンを直接拡張したものとなる。
ステップ１９０２では、ｓｅｃｕｒｅ＿ｂｏｏｔは、プロセッサに電源を投入し、プロセッサを制御して、プロセッサの一部として含まれるあらゆる自己テストルーチンを実行する。
ステップ１９０４では、ｓｅｃｕｒｅ＿ｂｏｏｔは、フェーズ１のファームウェア命令の組（図１８の１８０８）のデジタル署名（図１８の１８１２）の正当性を確認する。
ステップ１９０６では、ｓｅｃｕｒｅ＿ｂｏｏｔは、プロセッサを制御して、フェーズ１のファームウェア命令の組を実行する。
次に、ステップ１９０８～１９１２を備えたｗｈｉｌｅループにおいて、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９０８で、次のファームウェア命令の組をチェックする。
ステップ１９０８で、これから実行される別の組のファームウェア命令が残っていると判断されると、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９０９で、そのファームウェア命令の組を実行のためにメインメモリに移動させるかどうかを判断する。
この次のファームウェア命令の組が、メインメモリに移動される場合には、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９１０で、この次のファームウェア命令の組をＲＯＭからメインメモリにコピーする。

次に、ステップ１９１１で、ｓｅｃｕｒｅ＿ｂｏｏｔは、このコードを実行可能にすることができ、コードの変更を防止すると共に、この次の組の命令に関連付けられたデジタル署名の正当性を確認する。

正当性の確認に続いて、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９１２で、プロセッサを制御して、この次の命令の組を実行する。

「正当性を確認する」という用語は、以下では、デジタル署名の正当性の確認、および、実行可能ではあるが書き込み可能ではないコードアクセスの許可の設定を意味するために使用される。

もちろん、例えばステップ１９０４および１９１１といったすべてのデジタル署名の正当性の確認ステップにおいて、デジタル署名が正当であると確認できない場合には、ｓｅｃｕｒｅ＿ｂｏｏｔは、エラーを返し、機能しなくなる。

正当であると確認できないファームウェアまたはソフトウェアは、ｓｅｃｕｒｅ＿ｂｏｏｔが再実行される前に、訂正される必要がある。

【００９９】

ＲＯＭのファームウェア命令の組のすべてが、ｓｅｃｕｒｅ＿ｂｏｏｔによって正当であると確認され、かつ、実行されるとすぐに、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９１４で、ＳＰローダをマスストレージデバイスからメモリにロードする。

そして、ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９１６で、ＳＰローダに含まれるデジタル署名の正当性を確認する。

ｓｅｃｕｒｅ＿ｂｏｏｔは、ステップ１９１８で、ＳＰローダを実行してＳＰソフトウェアをロードし、ＳＰソフトウェアに関連付けられたデジタル署名の正当性を確認し、その後、１つまたは２つ以上のＳＰＧＳルーチンを実行する。

ファームウェア命令およびＳＰコードイメージのさまざまなフェーズは、異なるプライベート鍵によって署名されている可能性があり、デジタル署名の正当性の確認を行うために、対応する公開鍵を必要とする可能性があることに留意されたい。

実行可能なコードイメージの正当性が確認されるとすぐに、そのコードイメージに含まれるあらゆる公開鍵を信用することができる。

安全なブートプロセスは、安全なブートプロセスの開始時にハードウェアによって実行された正当性の確認に基づいて、信用チェーン（chain of trust）を構築し、検証された最後のコードイメージの実行を通じて信用チェーンを延長するプロセスとみなすことができる。

すべてのＯＳローダおよびＯＳは、この信用チェーンをＯＳレイヤおよびアプリケーションレイヤに延長することが可能であり、かつ、そのことが推奨される。

【０１００】

ステップ１９２０では、ＳＰＧＳは、ルート（root）暗号鍵を取得する必要がある。

これは、外部からその鍵を供給することにより行うことができるか、または、プラットフォームに実装されたセキュリティチップ、例えばＴＣＰＡチップなどによって行うことができる。

これらのルート鍵を使用し、ＯＳがＳＰを越えて信用チェーンを延長するように設計されていたと仮定すると、ＳＰＧＳは、ＯＳローダをロードし、正当性を確認し、ＯＳローダを実行して少なくとも１つのＯＳをロードし、ＯＳの正当性を確認し、そして、ステップ１９２４で、少なくとも１つのＯＳを実行する。

ＯＳが実行を開始する時点では、十分に正当性が確認され認証された安全なＳＰが、ＯＳをサポートするためにすでに初期化されている。

その後、システム動作中は、ＳＰによって提供される良質な組のサービスを使用してシステムセキュリティを維持することが、ステップ１９２４で起動された１つまたは２つ以上のＯＳの責務である。

ＯＳが、たとえ、成功した攻撃によって侵入されたとしても、現在のＯＳの脆弱性とは反対に、攻撃者は、特権レベル０を取得しない。

【０１０１】

本発明を、特定の実施の形態の観点で記載してきたが、本発明をこの実施の形態に限定することを意図するものではない。
本発明の精神内における変更は、当業者に明らかであろう。
例えば、本発明の一実施の形態を表す結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースは、多くの異なるプログラミング言語およびプログラム仕様言語によるほとんど無限の個数の異なるルーチンインタフェースにより実施することができる。
記載された実施の形態は、ハードウェアレイヤの主要なコンポーネントとして、Intel（登録商標）のＩＡ－６４アーキテクチャを特徴として備えているが、他の異なる現代のコンピュータプロセッサアーキテクチャを、安全なプラットフォームのハードウェアレイヤに使用することもできる。
本発明の一実施の形態を表す、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースは、上述したサービスおよび機能に加えて、さらに別のサービスおよび機能を提供するように拡張することができ、多くの別の、結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェースを考案することができる。
【０１０２】
上記記載は、説明の目的で、具体的な専門用語を使用して本発明の十分な理解を提供した。
しかしながら、本発明を実施に移すのに、その具体的な詳細は必要とされないことが、当業者には明らかであろう。
上記本発明の具体的な実施の形態の記載は、例証および記載の目的で提供されている。
それらの記載は、網羅することを意図するものではなく、また、開示された正確な形式に本発明を限定することを意図するものでもない。
多くの変更および変形が、上記教示を考慮して可能なことは明らかである。
実施の形態は、本発明の原理およびその実際の適用を最も良く説明するために示され、かつ、記載されており、これにより、他の当業者は、本発明、および、予想される特定の使用に適したさまざまな変更を有するさまざまな実施の形態を最も良く利用することが可能になる。
本発明の範囲は、特許請求の範囲の請求項およびそれらの均等物によって定義されることが意図されている。
【０１０３】
［付録Ａ］
本発明の実施の形態では、安全なプラットフォームサービスは、すべてのオペレーティングシステムおよびカスタム制御プログラムに、下に位置するハードウェア特権レジスタおよび命令によって実施される機能を提供する。
これは、ＳＰＧＳおよびＳＰＫによって提供されたプラットフォームサービスに対する、構成されたソフトウェアインタフェースによって行われる。
【０１０４】
いくつかのプラットフォームサービスは、ある特権レベルレジスタおよび特権レベル命令によって提供される機能を抽象化する。
他のプラットフォームサービスは、特定の特権レジスタおよび／または特権命令に密接に対応する。
安全なプラットフォームアーキテクチャの目的は、各オペレーティングシステムおよび各カスタム制御プログラムに、下に位置するハードウェアの全機能を提供することである。
ただし、この全機能は、その制御下にあるそれらのシステムリソースのみに提供される。
【０１０５】
安全なプラットフォームサービスは、いくつかのサービスのカテゴリーを備える。

これらのサービスは、オペレーティングシステムまたはカスタム制御プログラムに次のことを可能にするサービスを含む。

・物理メモリアドレス指定および仮想メモリアドレス指定の管理

・Ｉ／Ｏ操作およびＤＭＡ　Ｉ／Ｏ操作のアドレス指定の管理

・ディスパッチ可能なオブジェクト（「ＤＯ（dispatchable object）」、さまざまなオペレーティングシステムによって「プロセス」や「タスク」や「スレッド」と呼ばれるエレメントの状態保存領域）の管理

・同期割り込みおよび非同期割り込みの管理

【０１０６】

本発明の一実施の形態では、これらのカテゴリーの２～３の代表的なサービスインタフェースは、以下のものを含む。

【０１０７】

＜アドレス指定＞

ｓｐＳｅｔＰａｇｅＳｉｚｅ（ページサイズ、仮想アドレス、サイズ）

仮想アドレスの範囲のページサイズを指定する。

【０１０８】

Ｏｌｄ＿Ａｔｔｒｉｂｕｔｅｓ＝ｓｐＭａｐＭｅｍｏｒｙ（仮想アドレス、物理アドレス、サイズ、プロパティ）；

仮想アドレスの範囲をマッピングする。

【０１０９】

ｓｐＳｙｎｃＭａｐｐｉｎｇｓ（仮想アドレス、サイズ）；

複数のプロセッサ間のマッピングを同期させる。

【０１１０】

Ｉ／Ｏ　Ａｄｄｒｅｓｓ＝ｓｐＥｎａｂｌｅＤＭＡｍａｐ（仮想アドレス、サイズ、読み出し｜書き込み）；

ＤＭＡ　Ｉ／Ｏの読み出し操作または書き込み操作用の物理アドレスのマッピングを確立する。

【０１１１】

ｓｐＤｉｓａｂｌｅＤＭＡｍａｐ（仮想アドレス、サイズ）；

ＤＭＡ　Ｉ／Ｏ操作用の物理アドレスのマッピングを無効にする。

【０１１２】

＜プロパティ＞

プロパティ値は、マッピングの設定可能パラメータをドメインＯＳとＳＰとの間で通信するために使用される。

例として、以下の表に記載のものが含まれる。

【０１１３】

【表１】

Properties	Meaning
SP_MAPPING_VALID	This mapping is valid and should be used.
SP_MAPPING_PRIVILEGED	This mapping is only available to privileged code
SP_MAPPING_PRIVILEGED_READ	The page mapped must be readable by privileged code
SP_MAPPING_PRIVILEGED_WRITE	The page mapped must be writable by privileged code
SP_MAPPING_PRIVILEGED_EXECUTE	The page mapped must be executable by privileged code
SP_MAPPING_USER_READ	The page mapped must be readable by unprivileged code
SP_MAPPING_USER_WRITE	The page mapped must be writable by unprivileged code
SP_MAPPING_USER_EXECUTE	The page mapped must be executable by unprivileged code
SP_MAPPING_ACCESSED	The page mapped should be considered accessed
SP_MAPPING_DIRTY	The page mapped should be considered dirty
SP_MAPPING_EXCEPTION_DEFERRAL	Exceptions generated speculatively on this page should be deferred
SP_MAPPING_PAGESIZE_4K, etc	Should these be exported?
SP_MAPPING_KEY(key)	SP_MAPPING_KEY is a macro which is used to specify the protection key associated with this page
SP_MAPPING_PROMOTE	The page mapped must be executable, and must allow privilege promotion

【０１１４】
＜ディスパッチ可能オブジェクト（注１）＞
（注１）このインタフェースは、ＩＡ－６４特有である。
他のアーキテクチャについて類似のインタフェースを定義することができる。
【０１１５】
［初期化および起動］
ｖｏｉｄ（＊ｏｓ＿ｅｎｔｒｙ）（ｓｐＨａｎｄｌｅ　ｄｅｆａｕｌｔＤＯ，ｓｐＨａｎｄｌｅ　ＣＰＵ）
これは、ＯＳブートイメージからのエントリアドレスである。
実行は、（ＭＰシステムの）ブートプロセッサのここで開始する。
実行環境は、このルーチンをＣ言語で記述できるように十分初期化されている。
ＳＰサービスが利用可能である。
【０１１６】
ｖｏｉｄ（＊ｏｓ＿ｍｐ＿ｅｎｔｒｙ）（ｓｐＨａｎｄｌｅ　ｄｅｆａｕｌｔＤｏ，ｓｐＨａｎｄｌｅ　ＣＰＵ）
これは、マルチプロセッサの動作が直ちに開始可能な場合に、ＯＳによって登録されるエントリアドレスである。
実行環境は、カーネルの追加機能が初期化されて有効にされる点を除いて、ＯＳエントリポイントと同じである。
【０１１７】
ｓｐＥｒｒ　ｓｐＳｔａｒｔＭＰ（ｖｏｉｄ（＊ｍｐＥｎｔｒｙ）（ｓｐＨａｎｄｌｅ，ｓｐＨａｎｄｌｅ））
マルチプロセッサ動作に移行した時にＯＳから呼び出されると、他のＣＰＵは、この読み出しが復帰する前に実行を開始することができる。
【０１１８】
［プロセス管理］
ｓｐＥｒｒ　ｓｐＣｒｅａｔｅＤＯ（ｓｐＨａｎｄｌｅ　＊ｎｅｗＤＯ，ｖｏｉｄ　＊ｓ

ｔａｃｋ，　ｖｏｉｄ　＊ｒｓｅ，　ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｆｌａｇｓ）
　指定されたｓｔａｃｋ（スタック）／ＲＳＥ領域を有する新しいディスパッチ可能なオブジェクトを生成する。
　ｆｌａｇｓ（フラグ）は、さまざまな生成モード、例えば、親のＤＯ仮想アドレス指定をコピー／共有／書き込み時コピー（copy-on-write）のいずれにするかを指定する。
　ｓｔａｃｋおよびＲＳＥ領域は、親のＤＯによってアクセス可能でなければならない。
【０１１９】
ｓｐＥｒｒ　ｓｐＳｗｉｔｈｃＤｏ（ｓｐＨａｎｄｌｅ　ｎｅｗＤＯ，　ｉｎｔ　ｄｉｅ）
　現在のＣＰＵの実行を、指定されたＤＯに切り換える。
　ｄｉｅ（消滅）フラグが設定されている場合には、現在のＤＯは、この呼び出しが復帰する前に削除される。
　ｎｅｗＤＯ（新しいＤＯ）が、合法的なＤＯハンドルでもなく、すでにディスパッチされているわけでもない場合には、この呼び出しは、失敗に終わることがある。
【０１２０】
ｓｐＥｒｒ　ｓｐＲｅｓｕｍｅ（ｓｐＲｅｓｕｍｅＳｔａｔｅ　＊ｒｅｓｕｍｅＳｔａｔｅ）
　構成されたプロセス状態、または、予め保存されたプロセス状態を再開する。
　この呼び出しは、特権状態の変更、例えばドメイン制御されるＰＳＲビットの変更が必要な場合に使用される。
【０１２１】
ｓｐＥｒｒ　ｓｐＤｅｌｅｔｅＤＯ（ｓｐＨａｎｄｌｅ　ｖｉｃｔｉｍ）
　指定されたＤＯを削除する。
　指定されたＤＯは、この呼び出しが復帰する前に削除される。
　ｎｅｗＤＯが、合法的なＤＯハンドルでもないか、あるいは現在ディスパッチされている場合には、この呼び出しは、失敗に終わることがある。
【０１２２】
　デバッグおよび性能監視
ｓｐＥｒｒ　ｓｐＧｅｔＤｅｂｕｇＩｎｆｏ（ｓｔｒｕｃｔ　ｓｐＤｅｂｕｇＩｎｆｏ　＊ｄｅｂｕｇＣｏｎｆｉｇ）
　次の情報、すなわち、データブレークレジスタの個数、命令ブレークレジスタの個数、有効な最大データアドレスマスク、および有効な最大命令マスクを記憶する。
　ブレークレジスタが利用可能でない場合には、値は、ゼロに設定される。
　ＳＰは、１つまたは２つ以上の高い特権レベルのレジスタを自己が使用するために予約することができるので、戻り値は、実際のハードウェアより少なくなることがある。
【０１２３】
ｓｐＥｒｒ　ｓｐＳｅｔＤＢＲ（ｉｎｔ　ｒｅｇ，ｖｏｉｄ　＊ａｄｄｒ，ｕｎｓｉｇｎｅｄ　ｌｏｎｇ　ｍａｓｋ，ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｍｏｄｅ）
ｓｐＥｒｒ　ｓｐＳｅｔＩＢＲ（ｉｎｔ　ｒｅｇ，ｖｏｉｄ　＊ａｄｄｒ，ｕｎｓｉｇｎｅｄ　ｌｏｎｇ　ｍａｓｋ，ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｍｏｄｅ）
　（ａｃｃｅｓｓ　＆　ｍａｓｋ）＝＝（ａｄｄｒ　＆　ｍａｓｋ）である場合に、データ（命令）デバッグフォルトを発生させるために、指定されたデータ（命令）ブレークレジスタを設定する。
　ｍｏｄｅ（モード）パラメータは、データブレークポイントの読み出しアクセスおよび／または書き込みアクセス、命令ブレークポイントの実行、およびアクセス特権レベル（ドメインＯＳの特権レベル、アプリケーションの非特権レベル）を指定する。
　ｍｏｄｅが０である場合に、指定されたデバッグレジスタは、無効にされる。
【０１２４】
　ｒｅｇ（レジスタ）が有効なデバッグレジスタでなく、ａｄｄｒ（アドレス）が無効である場合、または、ｍａｓｋ（マスク）またはｍｏｄｅが、ハードウェアによって受け付

けられない場合に、この呼び出しは失敗に終わる。
【０１２５】
ｓｐＥｒｒ　ｓｐＳｅｔＰＭＣ（ｉｎｔ　ｒｅｇ，ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｅｖｅｎｔ，ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｆｌａｇｓ）
指定された汎用モニタの制御を設定する。
４≦ｒｅｇ≦７である。
ｅｖｅｎｔ（イベント）は、実施に固有のものである。
ｆｌａｇｓパラメータは、どの特権レベルが監視されるか、関連付けられたカウンタをリセットするかどうか、および有効なモニタが特権を有するか、または、アプリケーション制御下にあるかを指定する。
【０１２６】
他の呼び出しと異なり、これは、アプリケーション（特権レベル３）から呼び出すことができる。
ｒｅｇが無効である場合、または、アプリケーションが特権を有するモニタを設定しようとした場合に、この呼び出しは失敗に終わる。
【０１２７】
ｓｐＥｒｒ　ｓｐＲｅａｄＰＭＤ（ｕｎｓｉｇｎｅｄ　ｌｏｎｇ　ｌｏｎｇ　＊ｖａｌ，ｉｎｔ　ｒｅｇ）
指定された汎用モニタのイベントカウントを読み出す。
４≦ｒｅｇ≦７である。
【０１２８】
他の呼び出しと異なり、これは、アプリケーション（特権レベル３）から呼び出すことができる。
ｒｅｇが無効である場合に、この呼び出しは失敗に終わる。
モニタが特権を有し、かつ、呼び出し側が特権レベル３である場合に、ＩＡ－６４ハードウェアの挙動との一貫性を保つために、値０が戻される。
【０１２９】
ｓｐＥｒｒ　ｓｐＳｅｔＰＭＣＸ（ｉｎｔ　ｒｅｇ，ｕｎｓｉｇｎｅｄ　ｌｏｎｇ　ｖａｌ，ｕｎｓｉｇｎｅｄ　ｉｎｔ　ｆｌａｇｓ）
指定された汎用モニタの制御を設定する。
ｒｅｇ＜４またはｒｅｇ＞７である。
ｖａｌ（変数）は、実施に固有のものである。
【０１３０】
他の呼び出しと異なり、これは、アプリケーション（特権レベル３）から呼び出すことができる。
ｒｅｇが実施されておらず、ｖａｌが違法な場合、または、アプリケーションが、特権を有するモニタを設定しようとした場合に、この呼び出しは失敗に終わる。
【０１３１】
ｓｐＥｒｒ　ｓｐＲｅａｄＰＭＤＸ（ｕｎｓｉｇｎｅｄ　ｌｏｎｇ　ｌｏｎｇ　＊ｖａｌ，ｉｎｔ　ｒｅｇ）
指定された性能監視データのイベントカウントを読み出す。
０≦ｒｅｇ≦２５５である。
【０１３２】
他の呼び出しと異なり、これは、アプリケーション（特権レベル３）から呼び出すことができる。
ｒｅｇが無効である場合に、この呼び出しは失敗に終わる。
モニタが実施されていないか、または、特権を有し、かつ、呼び出し側が特権レベル３である場合に、ＩＡ－６４ハードウェアの挙動との一貫性を保つために、値０が戻される。
【０１３３】

［スケジューリングおよびシャットダウン］
ｓｐＥｒｒ　ｓｐＤｉｓｐａｔｃｈＡｆｆｉｎｉｔｙ（ｉｎｔ　ｖａｌｕｅ）
現在のＣＰＵのディスパッチとの親和性（affinity）を設定する。
ｖａｌｕｅは、最後のＣＰＵへの再ディスパッチの相対的な重要度である。
０≦ｖａｌｕｅ≦１００である。
【０１３４】
ｓｐＥｒｒ　ｓｐＰａｕｓｅ（ｖｏｉｄ）
現在のＣＰＵを、準備状態にあるスケジューリングプールに入れる。
【０１３５】
ｓｐＥｒｒ　ｓｐＨａｌｔ（ｖｏｉｄ）
外部インタラプトがドメインに配信されるまで、現在のＣＰＵのスケジューリングを一時的に停止する。
プロセッサ間通信は、外部インタラプトとしてモデル化されることに留意されたい。
【０１３６】
ｓｐＥｒｒ　ｓｐＳｈｕｔＤｏｗｎ（ｖｏｉｄ）
現在のＣＰＵをスケジューリングから永久に取り除き、関連付けられたすべての状態を削除する。
【０１３７】
＜割り込み＞
ＩＡ－６４アーキテクチャは、割り込みを、特定の処理ルーチンへの制御の引渡しを引き起こすイベントとして定義する。
４つの割り込みのクラスがある。
【０１３８】
１．アボート：プロセッサがマシンチェック（内部故障）またはプロセッサのリセットを検出した場合の割り込みである。
アボートは、ＰＡＬに基づく割り込みである。
アボートは、ドメインＯＳに反映される場合もあるし、反映されない場合もある。
２．インタラプト：プロセッサがサービスの実行要求を受信した場合の割り込みである。
インタラプトは、外部Ｉ／Ｏによる場合もあるし、他のプロセッサによる場合もあるし、ＣＰＵの内部イベント（例えばタイマチック）による場合もある。
３．フォルト：現在のＩｔａｎｉｕｍまたは ＩＡ－３２の命令であって、実行できない動作もしくは実行されるべきでない動作、または、その命令が実行される前にシステムの介入が必要とされる動作を要求する命令に対して発生する割り込みである。
フォルトは、命令ストリームに対して非同期である。
４．トラップ：ちょうど実行されたＩＡ－３２またはＩｔａｎｉｕｍの命令が、システムの介入を必要とする時に発生する割り込みである。
トラップは、命令ストリームに対して同期している。
【０１３９】
アボート、フォルト、およびトラップは、例外（exceptions）と総称される。
【０１４０】
ｓｐＥｒｒ　ｓｐＲｅｇｉｓｔｅｒＩＳＲ（ｉｎｔ　ｖｅｃｔｏｒ，ｖｏｉｄ（＊ｉｓｒ）（．．．），ｖｏｉｄ　＊ｐａｒａｍ）
外部インタラプトがドメインに配信される時に呼び出されるルーチンを登録する。
【０１４１】
ｓｐＥｒｒ　ｓｐＲｅｇｉｓｔｅｒＨａｎｄｌｅｒ（ｖｏｉｄ（＊ｅＨａｎｄｌｅｒ）（．．．），ｖｏｉｄ　＊ｐａｒａｍ）
例外がドメイン内に発生した時に呼び出されるルーチンを登録する。
【０１４２】
ｓｐＥｒｒ　ｓｐＳＰＬ（ｉｎｔ　ｌｅｖｅｌ）
指定されたｌｅｖｅｌ（レベル）以下のベクタ上のすべてのインタラプトの配信を防止

する。
ｌｅｖｅｌ＝＝ＳＰＤＩＳＡＢＬＥＩＮＴＲの場合には、すべてのインタラプトは無効にされる。
ｌｅｖｅｌ＝＝ＳＰＥＮＡＢＬＥＩＮＴＲの場合には、すべてのインタラプトは有効にされる。
インタプラトレベルが低い場合には、任意の有効にされた保留中のインタラプトを、呼び出しが復帰する前に配信することができる。
【０１４３】
ｓｐＥｒｒ　ｓｐＳｉｇｎａｌ（ｓｐＨａｎｄｌｅ　ｌｏｇｉｃａｌＣＰＵ，ｖｏｉｄ＊ｍｅｓｓａｇｅ）
指定されたＣＰＵにプロセッサ間の「インタラプト」を配信する。
６４ビットのｍｅｓｓａｇｅ（メッセージ）の値および送信側ＣＰＵ用のハンドルは、インタラプト値として配信される。
【０１４４】
［付録Ｂ］
本発明の実施の形態では、安全なリポジトリサービスが、オペレーティングシステム、カスタム制御プログラム、およびユーザアプリケーションに、当該サービス自体以外では決してアクセス可能であってはならない重要なデータの処理と、場合によっては当該重要なデータの保存とを必要とする機能を提供する。
ＳＰアーキテクチャの構造によって、このような重要なデータを操作する機能のバイナリコードイメージに加えてこのような重要なデータも、オペレーティングシステムのコードによっても、カスタム制御プログラムのコードによっても、アプリケーションのコードによっても直接読み出しできず、また、直接変更できないことが保証される。
【０１４５】
安全なリポジトリサービスの第１の例は、暗号サービスである。
暗号サービスは、平文の暗号鍵の材料などのデータによる処理、および、当該データの保管を必要とする。
暗号サービスの基本的な組は、本発明のあらゆる実施の形態において必要不可欠である。
安全なリポジトリサービスの追加例としては、操作可能なセキュリティポリシーおよび状態データを利用するサービスや、重要な会計監査情報をログに記録するサービスや、暗号サービスの代わりの組もしくは高級な組を提供するサービスが含まれる。
上記操作可能なセキュリティポリシーおよび状態データは、例えば、ＩＰＳｅｃおよびＳＥＬｉｎｕｘに必要とされるものである。
暗号サービスの基本的な組のソースコードだけでなく、ＳＰＫのすべてのソースコードも、公開され、公共のセキュリティに開放される。
付加物を有する安全なリポジトリサービスのプロバイダは、自身の付加物の詳細を自由に公開することもできるし、秘密にすることもできる。
【０１４６】
ＳＰアーキテクチャは、組み込まれ得る安全なリポジトリサービスの組について制限はない。
このアーキテクチャは、慎重に構造化されて、安全なリポジトリサービスの各組を区画化されている。
これにより、各組は、他のすべての安全なリポジトリサービスと、ＳＰの安全なプラットフォームサービスとの双方から隔離される。
これは、重要なデータおよびコードイメージを、オペレーティングシステムおよびアプリケーションだけでなく、他のすべてのＳＰＧＳコンポーネントおよびＳＰＫコンポーネントからも保護することによって、それら重要なデータおよびコードイメージへの直接アクセスをさらに制限する。
これは、従来のオペレーティングシステムアーキテクチャと明らかに大きく異なる。

従来のオペレーティングシステムアーキテクチャでは、限られた量の重要なデータを、成功した攻撃者から保護することにさえも、専用の特別な目的のハードウェアが必要とされる。

特別な目的のハードウェアを備えていても、このような攻撃者は、メインメモリの他のすべてのデータもしくは外部記憶デバイスに記憶された他のすべてのデータに自由にアクセスすることができるか、または、当該特別な目的のハードウェアの可用性もしくは内部状態とインタフェースすることができる。

大きなサーバでは、コスト、重要なデータの量、および性能のボトルネックを回避する必要性から、十分な量の専用の特別な目的のハードウェアモジュールが十分な解決策となることが妨げられている。

これに対して、ＳＰアーキテクチャは、安全に保護されなければならないデータを保持および処理するすべてのメインプロセッサの利用、および、必要に応じて大量のメインメモリの利用が可能となる。

【０１４７】

本発明の一実施の形態では、ＩＡ－６４の保護キーレジスタを使用して、区画が実施される。

ＳＰへのエントリにおいて、ハードウェアの状態は、指定された区画にのみアクセスを制限するように設定され、制御は、指定されたサービスに渡される。

この関連付けを統率するすべてのＳＰのソースコードは、公開され、公共のセキュリティに開放されてもよい。

【０１４８】

暗号サービスの基本的な組の正確なレパートリは、ＳＳＬ／ＴＬＳ（注２）、ＴＣＰＡ（Trusted Computing Platform Alliance）の主仕様のバージョン１．０（注３）、またはＭｉｃｒｏｓｏｆｔ　ＣＡＰＩ（Crypto API）仕様（注４）などの必要とされる暗号の標準規格にサポートを提供するために選択される。

ハードウェアセキュリティチップが、現在、開発され、ハードウェアに組み込まれて、いくつかの基本的な暗号機能にサポートを提供している。

ＳＰの暗号サービスの基本的な組の実施は、構造化され、これらのセキュリティチップが存在する場合に、その使用を可能にする。

【０１４９】

広範に見ると、基本的な暗号サービスは、対称鍵および非対称鍵の生成手段、エクスポート手段、インポート手段、取り替え手段、保存手段、および破壊手段を含む。

また、内部の使用またはサポートされた標準規格に必要な暗号のレパートリの暗号機能および復号機能の一式が含まれる。

同様に、乱数発生器およびユニークのＩＤの発生器だけでなく、暗号のハッシュ関数、メッセージ認証機能、デジタル署名機能の必要な一式も提供される。

【０１５０】

（注２）詳細は、http://www.ietf.org/ID.htmlに見ることができる。

（注３）この仕様の変更が現在予定されている。

詳細は、http://www.trustedpc.org/home/pdf/Mainv1_0.pdfに見ることができる。

（注４）詳細は、http://www.microsoft.com/technet/security/prodtech.cryptech.aspに見ることができる。

【図面の簡単な説明】

【０１５１】

【図１】汎用コンピュータシステム内のハードウェアレイヤ、オペレーティングシステムレイヤ、およびアプリケーションプログラムレイヤを示すブロック図である。

【図２】初期のコンピュータシステムを示すブロック図である。

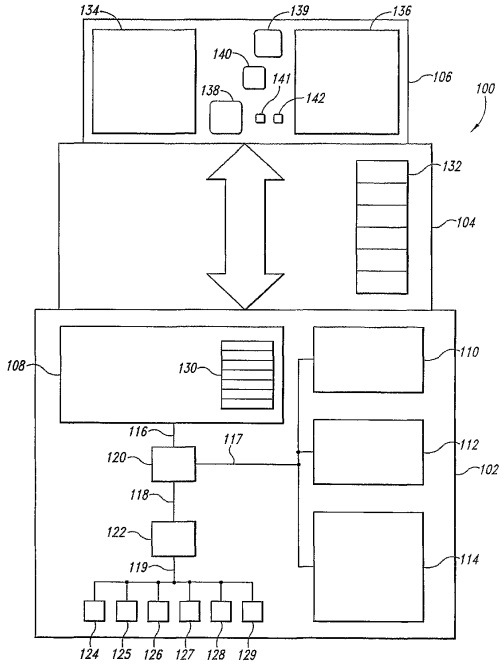
【図３】初期のメインフレームコンピュータ内のオペレーティングシステムの論理的配置を示すブロック図である。

【図４】現代のＰＣのアプリケーションレイヤ、オペレーティングシステムレイヤ、およ

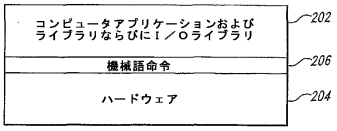
びハードウェアレイヤのブロック図である。
【図５Ａ】汎用コンピュータハードウェアシステムの基本的な特権レベルのメカニズムおよび機能である。
【図５Ｂ】汎用コンピュータハードウェアシステムの基本的な特権レベルのメカニズムおよび機能である。
【図５Ｃ】汎用コンピュータハードウェアシステムの基本的な特権レベルのメカニズムおよび機能である。
【図５Ｄ】汎用コンピュータハードウェアシステムの基本的な特権レベルのメカニズムおよび機能である。
【図６】多くの現代のコンピュータシステムに存在するセキュリティおよび信頼性の侵害の一例を示す図である。
【図７】一タイプの現代のプロセッサ内のレジスタを示すブロック図である。
【図８】現代の一コンピュータアーキテクチャによって提供される仮想アドレス空間を示す図である。
【図９】領域レジスタと、保護キーレジスタと、変換索引バッファとに記憶された情報を介した仮想メモリアドレスの物理メモリアドレスへの変換を示す図である。
【図１０】オペレーティングシステムのルーチンが、仮想メモリアドレスに対応する物理メモリのメモリページを発見するのに使用するデータ構造体を示す図である。
【図１１】ＴＬＢエントリに使用されるアクセス権のコード化を示す図である。
【図１２】割り込みメカニズムを示す図である。
【図１３】アプリケーションレベルのルーチンに、より高い特権レベルのルーチンの呼び出しを許可するＣＰＬ昇格メカニズムを示す図である。
【図１４】ＩＡ－６４プロセッサに基づくマシン上におけるオペレーティングシステムの直接的なレイヤ化を示す図である。
【図１５】結合されたハードウェアおよびソフトウェアの安全なプラットフォームと、１つまたは２つ以上のオペレーティングシステムおよびカスタマイズされた制御プログラムとを備えたコンピュータシステム内のＳＰインタフェースを示すブロック図である。
【図１６】ＳＰによって必要とされ、Intel（登録商標）のＩＡ－６４プロセッサアーキテクチャなどの現代のプロセッサアーキテクチャによって提供される第２のメモリ分割機能を示す図である。
【図１７】ＳＰのソフトウェアレイヤによって提供される機能のブロック図である。
【図１８】安全なブートプロセスに関連したシステムのハードウェアコンポーネントのブロック図である。
【図１９】ＳＰの安全なブートプロセスコンポーネントの一実施の形態を表す安全なブートプロセス「ｓｅｃｕｒｅ＿ｂｏｏｔ」の制御流れ図である。
【符号の説明】
【０１５２】
１５０２・・・結合されたハードウェアおよびソフトウェアの安全なプラットフォームインタフェース（ＳＰＩ）、
１５０４・・・結合されたハードウェアおよびソフトウェアの安全なプラットフォーム（ＳＰ）のソフトウェアレイヤ、
１５０６・・・オペレーティングシステム（ＯＳ）および／またはカスタマイズされた制御プログラム、
１５０８・・・ハードウェアおよびファームウェアのプラットフォームインタフェース、
１５１０・・・ハードウェアおよびファームウェアのプラットフォーム、
１６０２・・・全仮想メモリ空間、
１６０６・・・オペレーティングシステムのメモリ空間、
１６０８・・・ユーザレベルのメモリ空間、
１７１０・・・ＳＰプラットフォーム管理サービス、
１７１６・・・プラットフォームサービス、

１７１８・・・ドメイン制御サービス、
１７２０・・・プロセッサ制御サービス、
１７２２・・・プラットフォームポリシー制御サービス、
１７２４・・・ドメイン間およびドメイン内サービス、
１７１２・・・ＳＰセキュリティ管理サービス、
１７２６・・・安全なリポジトリサービス、
１７２８・・・呼び出し側認証サービス、
１７１４・・・ＳＰ拡張部、
１７３０・・・メモリ管理メカニズム、
１７３２・・・プロセッサディスパッチメカニズム、
１７３４・・・例外処理メカニズム、
１７３６・・・インタラプト処理メカニズム、
１７３８・・・デバッグおよび監視メカニズム、
１７４０・・・暗号メカニズムおよび記憶装置、
１７４２・・・安全なリポジトリメカニズムおよび記憶装置、
１８０４・・・初期ファームウェア正当性確認メカニズム、
１８０８，１８０９，１８１０，１８１１・・・デジタル署名されたファームウェア命令の組、
１８１２，１８１３，１８１４，１８１５・・・デジタル署名、
１８２０・・・ＯＳ、
１８２２・・・ＯＳローダのデジタル署名、
１８２３・・・ＯＳのデジタル署名、

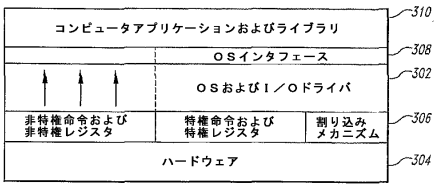
【図１】



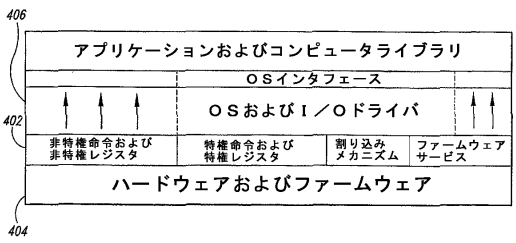
【図２】



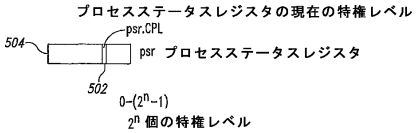
【図３】



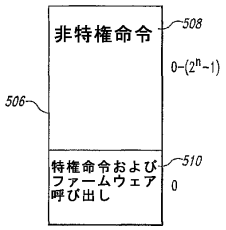
【図４】



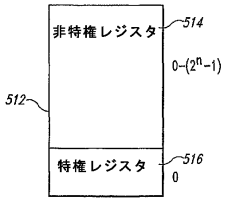
【図５Ａ】



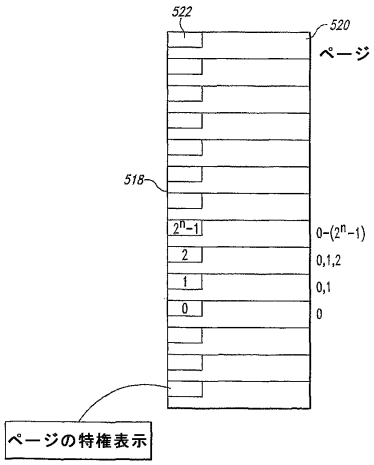
【図５Ｂ】



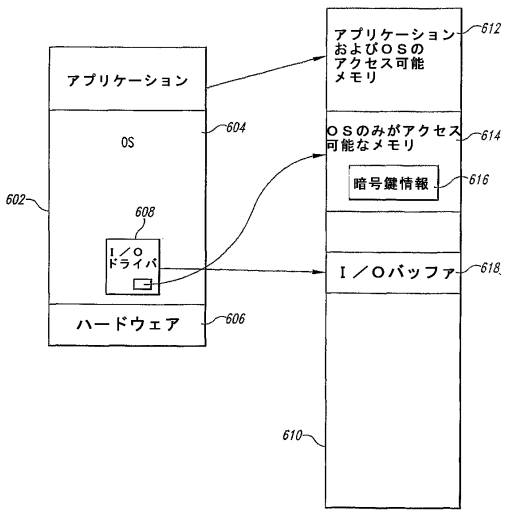
【図５Ｃ】



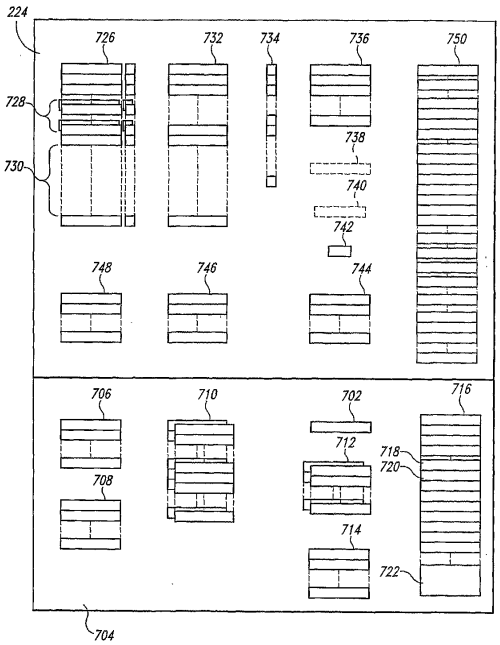
【図５Ｄ】



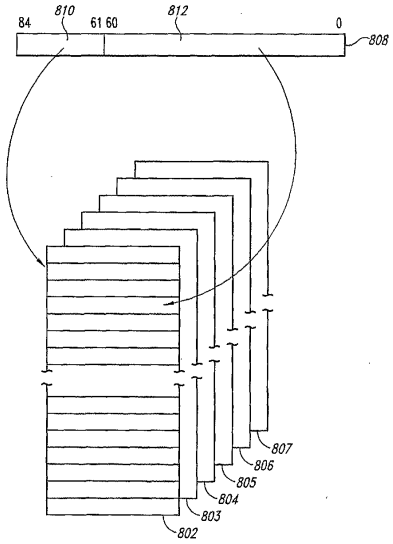
【図６】



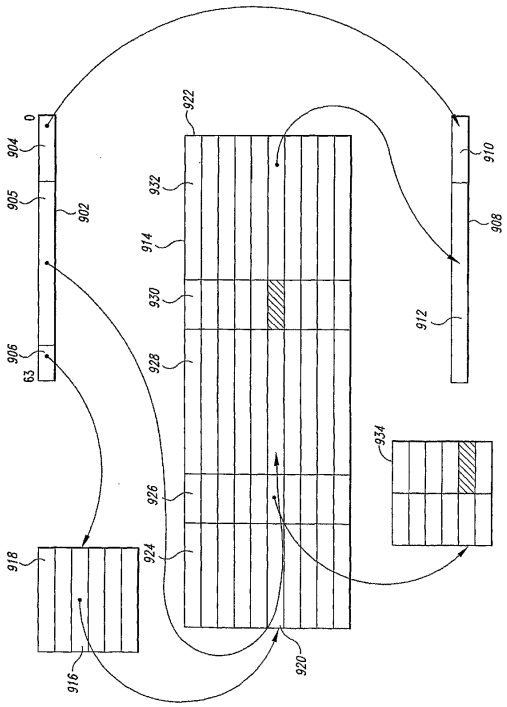
【図７】



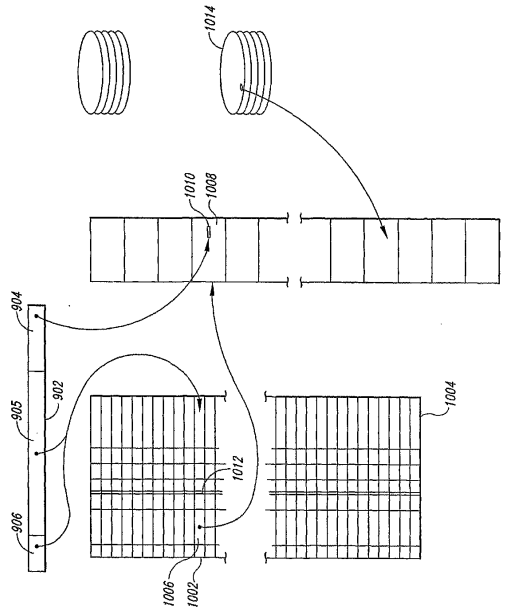
【図８】



【図９】



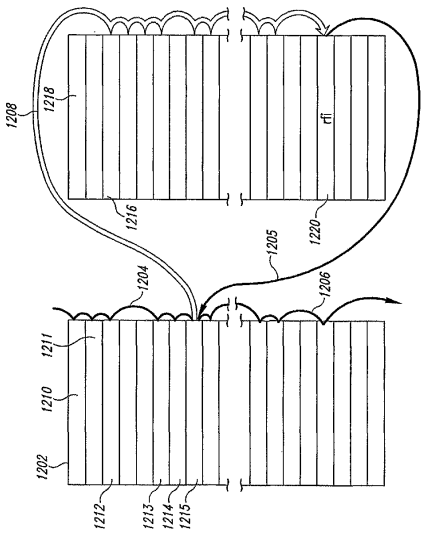
【図１０】



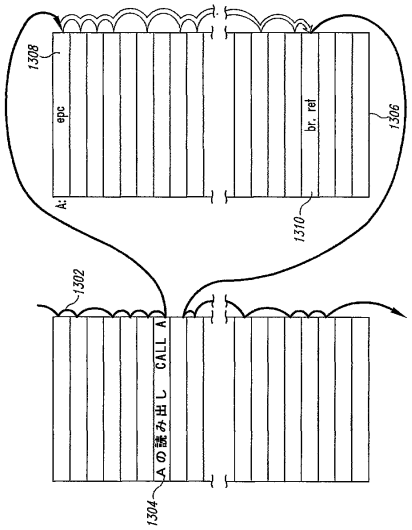
【図１１】

TLB.AR	TLB.PL	特権レベル 3	2	1	0	説明
0	3	R	R	R	R	READ ONLY 読み出し専用
	2		R	R	R	
	1			R	R	1110
	0				R	
1	3	RX	RX	RX	RX	READ, EXECUTE
	2		RX	RX	RX	
	1			RX	RX	読み出し，実行
	0				RX	
2	3	RW	RW	RW	RW	READ, WRITE
	2		RW	RW	RW	
	1			RW	RW	読み出し，書き込み
	0				RW	
3	3	RWX	RWX	RWX	RWX	READ, WRITE, EXECUTE
	2		RWX	RWX	RWX	
	1			RWX	RWX	読み出し，書き込み，実行
	0				RWX	
4	3	R	RW	RW	RW	READ ONLY/READ, WRITE
	2		R	RW	RW	
	1			R	RW	読み出し専用／読み出し，書き込み
	0				RW	
5	3	RX	RX	RX	RWX	READ, EXECUTE/READ, WRITE, EXEC
	2		RX	RX	RWX	
	1			RX	RWX	読み出し，実行／読み出し，書き込み，実行
	0				RWX	
6	3	RWX	RW	RW	RW	READ, WRITE, EXECUTE/READ, WRITE
	2		RWX	RW	RW	
	1			RWX	RW	読み出し，書き込み，実行／読み出し，書き込み
	0				RW	
7	3	X	X	X	RX	EXEC, PROMOTE/READ, EXECUTE
	2	XP2	X	X	RX	
	1	XP1	XP1	X	RX	実行，昇格／読み出し，実行
	0	XP0	XP0	XP0	RX	

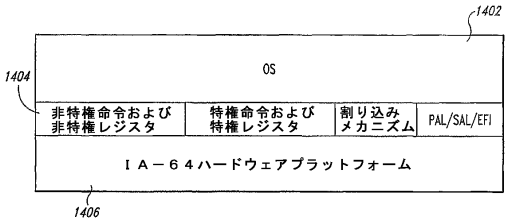
【図１２】



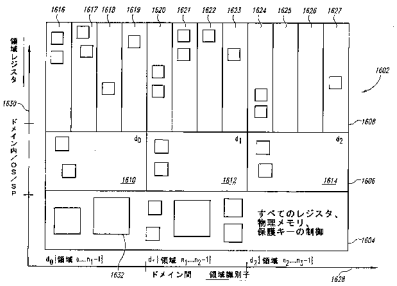
【図１３】



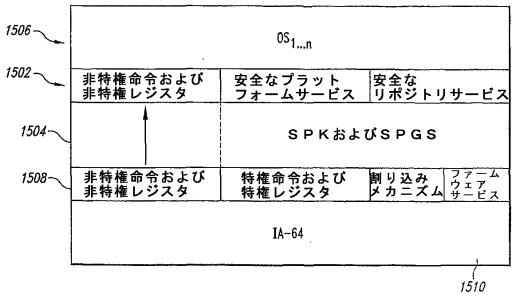
【図１４】



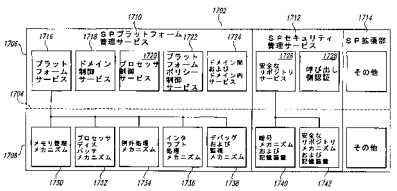
【図１６】



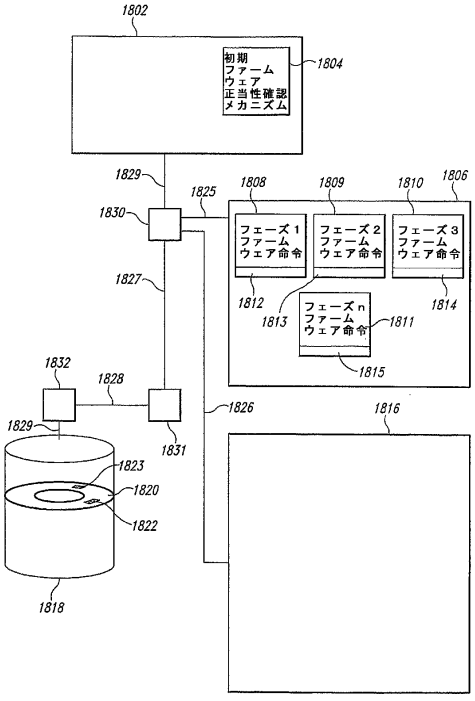
【図１５】



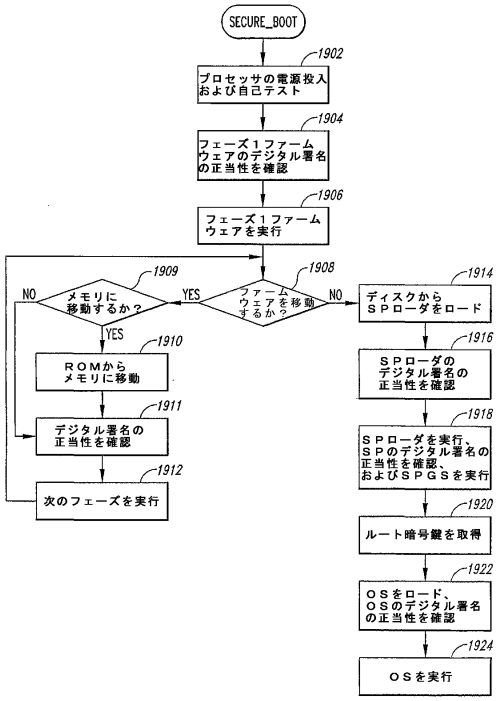
【図１７】



【図１８】


1802
初期
ファーム
ウェア
正当性確認
メカニズム
1804
1829
1825
1808
1809
1810
1806
1830
フェーズ１
ファーム
ウェア命令
フェーズ２
ファーム
ウェア命令
フェーズ３
ファーム
ウェア命令
1827
1812
1813
フェーズｎ
ファーム
ウェア命令
1811
1814
1815
1832
1828
1826
1816
1829
1831
1823
1820
1822
1818

【図１９】


SECURE_BOOT
1902
プロセッサの電源投入
および自己テスト
1904
フェーズ１ファーム
ウェアのデジタル署名
の正当性を確認
1906
フェーズ１ファーム
ウェアを実行
1909
1908
NO
メモリに
移動するか？
YES
ファーム
ウェアを移動
するか？
NO
1914
ディスクから
ＳＰローダをロード
YES
1910
ＲＯＭから
メモリに移動
1916
ＳＰローダの
デジタル署名の
正当性を確認
1911
デジタル署名の
正当性を確認
1918
ＳＰローダを実行、
ＳＰのデジタル署名の
正当性を確認、
およびＳＰＧＳを実行
1912
次のフェーズを実行
1920
ルート暗号鍵を取得
1922
ＯＳをロード、
ＯＳのデジタル署名
の正当性を確認
1924
ＯＳを実行

フロントページの続き

(51)Int.Cl. FI
G06F 12/14 560B
G06F 9/06 660N

(31)優先権主張番号 60/297,175
(32)優先日 平成13年6月8日(2001.6.8)
(33)優先権主張国 米国(US)
(31)優先権主張番号 10/118,646
(32)優先日 平成14年4月8日(2002.4.8)
(33)優先権主張国 米国(US)

(72)発明者 ダニエル・ジェイ・マーゲンハイマー
アメリカ合衆国コロラド州フォートコリンズ アシュフォードレーン８３９
(72)発明者 クリス・ディ・ハイサー
アメリカ合衆国コロラド州フォートコリンズ スキマーホーンストリート２９４８
(72)発明者 トム・クリスチャン
アメリカ合衆国コロラド州フォートコリンズ ケンウッドコート１４１２
(72)発明者 ブレット・マキー
アメリカ合衆国コロラド州フォートコリンズ ローリングゲートロード４１２０
(72)発明者 ロバート・ガードナー
アメリカ合衆国コロラド州フォートコリンズ シダーウッドドライブ２５０１

審査官 児玉 崇晶

(56)参考文献 特開２００１－０５６７８３（ＪＰ，Ａ）
特開昭６０－０７３７６２（ＪＰ，Ａ）
特開２０００－０７６０８７（ＪＰ，Ａ）
特開昭６２－００５４４１（ＪＰ，Ａ）
特開２０００－０５７０４５（ＪＰ，Ａ）
米国特許第０５７７４６５２（ＵＳ，Ａ）
米国特許第０５９４８０６４（ＵＳ，Ａ）
特開平０６－３３２８０３（ＪＰ，Ａ）
特開平０５－２０４７６０（ＪＰ，Ａ）
特開昭６２－２７６６３４（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 12/14
G06F 21/22
G06F 21/24