(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-128537
(P2007-128537A)
(43)公開日　平成19年5月24日(2007.5.24)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 21/22	(2006.01)	G06F	9/06	660J	5B017
G06F 21/24	(2006.01)	G06F	12/14	560C	5B276

審査請求 有　請求項の数 14　ＯＬ　（全 13 頁）

(21)出願番号　特願2006-339906 (P2006-339906)
(22)出願日　平成18年12月18日 (2006.12.18)
(62)分割の表示　特願2002-514509 (P2002-514509)の分割
原出願日　平成12年7月25日 (2000.7.25)

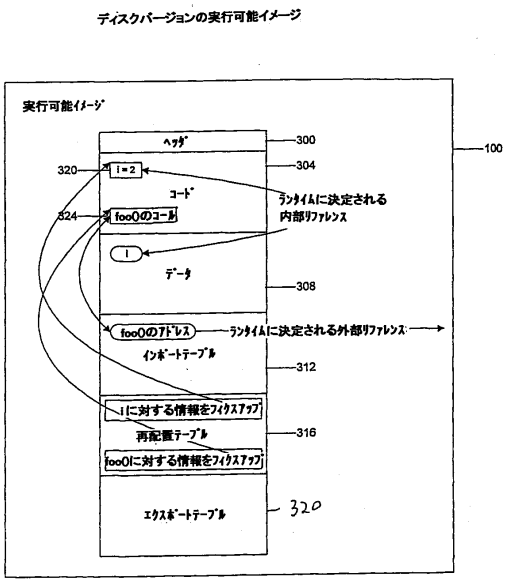
（特許庁注：以下のものは登録商標）
１．Ｌｉｎｕｘ

(71)出願人　597095197
マクロビジョン・コーポレーション
アメリカ合衆国　カリフォルニア州　９５０５０　サンタクララ　デ・ラ・クルーズ・ブールバード　２８３０
(74)代理人　100070150
弁理士　伊東　忠彦
(72)発明者　ボドロフ，ドミトリー
アメリカ合衆国　カリフォルニア州　９２１３０　サン・ディエゴ　デル・マー・ハイツ・ロード　３５２５　ピー・エム・ビー　１３７

Ｆターム(参考)　5B017 AA08　CA15
5B276 FB02

(54)【発明の名称】動的に接続可能な実行イメージの真正性検証システム及び方法

(57)【要約】
【課題】　動的にコネクト可能な実行可能イメージの真正性を検証するシステム及び方法を提供することである。
【解決手段】　実行可能イメージの真正性を判定するシステムであって、一つ以上のポインタを有する実行可能イメージが設けられ、実行可能イメージは、各ポインタが実行可能イメージ内のロケーションを参照するかどうかを指定する情報を含み、各ポインタが実行可能イメージ内の対応したロケーションを参照するかどうかを判定する機能を備えた検証部が更に設けられている、システム。
【選択図】　図３



【特許請求の範囲】
【請求項１】
実行可能イメージの真正性を判定するシステムであって、
一つ以上のポインタを有する実行可能イメージが設けられ、
実行可能イメージは、各ポインタが実行可能イメージ内のロケーションを参照するかどうかを指定する情報を含み、
各ポインタが実行可能イメージ内の対応したロケーションを参照するかどうかを判定する機能を備えた検証部が更に設けられている、
システム。
【請求項２】
ポインタは実行可能イメージ内のインポートテーブルを参照する、請求項１記載のシステム。
【請求項３】
検証部は、ポインタが実行可能イメージ内のロケーションを参照しないことを判定したときに警報を生成する、請求項１記載のシステム。
【請求項４】
検証部は、ポインタが実行可能イメージ内のロケーションを参照しないことを判定したときに実行可能イメージの供給元へ警報を生成する、請求項１記載のシステム。
【請求項５】
実行可能イメージの真正性を判定するシステムであって、
第１の実行可能イメージと、
第１の実行可能イメージ内のロケーションを参照するポインタを含む第２の実行可能イメージと、
ポインタが第１の実行可能イメージ内のロケーションを参照するかどうかを判定する機能を備えた検証部と、
を具備するシステム。
【請求項６】
検証部は、ポインタが第１の実行可能イメージ内のロケーションを参照しないことを判定したときに警報を生成する、請求項５記載のシステム。
【請求項７】
検証部は、ポインタが第１の実行可能イメージ内のロケーションを参照しないことを判定したときに実行可能イメージの供給元へ警報を生成する、請求項５記載のシステム。
【請求項８】
検証部は、
オペレーティングシステムから第１の実行可能イメージのベースアドレスを要求し、
第１の実行可能イメージの先頭に置かれ、第１の実行可能イメージの開始アドレス及び終了アドレスを指定する予め定義されたヘッダを読み、
ポインタが予め定義されたヘッダによって指定されるような第１の実行可能イメージ内のロケーションを参照したかどうかを判定することにより、
ポインタが第１の実行可能イメージ内のロケーションを参照したかどうかを判定する、
請求項５記載のシステム。
【請求項９】
実行可能イメージの真正性を判定する方法であって、
メモリにロードされた実行可能イメージ内で、プログラムローダーによってメモリ内の選択されたアドレスに結合されたポインタをもつ一つ以上のロケーションを特定する手順と、
特定されたロケーションの各ポインタが実行可能イメージによって指定された宛先を参照するかどうかを判定する手順と、
を有する方法。
【請求項１０】

宛先は、実行可能イメージにあるインポートテーブル内のアドレスである、請求項９記載の方法。
【請求項１１】
宛先は、別の実行可能イメージにおけるアドレスである、請求項９記載の方法。
【請求項１２】
実行可能イメージの真正性を判定するシステムであって、
メモリにロードされた実行可能イメージ内で、プログラムローダーによってメモリ内の選択されたアドレスに結合されたポインタをもつ一つ以上のロケーションを特定する手段と、
特定されたロケーションの各ポインタが実行可能イメージによって指定された宛先を参照するかどうかを判定する手段と、
を有するシステム。
【請求項１３】
宛先は、実行可能イメージにあるインポートテーブル内のアドレスである、請求項１２記載のシステム。
【請求項１４】
宛先は、別の実行可能イメージにおけるアドレスである、請求項１２記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はコンピュータシステムに係る。特に、本発明は、動的にコネクト可能な実行可能イメージの真正性を検証するシステム及び方法に関する。
【背景技術】
【０００２】
新しいオブジェクトモデルは、ランタイム時にソフトウェアアプリケーションを動的に統合する。例えば、マイクロソフト社によりライセンスが供与されるウィンドウズ（登録商標）は、ソフトウェアアプリケーションの実行中にソフトウェアアプリケーションを動的リンクライブラリと動的に統合することができる。ユーザがソフトウェアアプリケーションの実行を要求した後、プログラムローダーは、プロセスイメージを作成するため、アプリケーションのディスクイメージをディスク記憶装置からメインメモリへコピーする。ディスクイメージは、ロードされる前の実行可能イメージを表し、プロセスイメージはメモリにロードされた後の実行可能イメージを表す。ディスクイメージとプロセスイメージの両方は、典型的に、ソフトウェアの中でランタイム時に動的リンクライブラリを参照するため用意されるべき部分を識別するフィクスアップ・セクションを含む。
【０００３】
注意すべきことは、ローディング後に、プロセスイメージはディスクイメージと異なる点である。そのため、ディスクイメージに関して準備されているチェックサムは、たとえプロセスイメージが不正に改ざんされていなくても、プロセスイメージのチェックサムと一致しなくなる。
【０００４】
したがって、動的ローディング環境においてソフトウェアアプリケーションの正体を検証し得るシステムが要望される。特に、このシステムは、別のデータオブジェクトへ動的にコネクトされたソフトウェアアプリケーションがソフトウェアアプリケーションの実行に続いて改ざんされたかどうかを判定できなければならない。
【発明の開示】
【発明が解決しようとする課題】
【０００５】
本発明の課題は、動的にコネクト可能な実行可能イメージの真正性を検証するシステム及び方法を提供することである。
【課題を解決するための手段】

【０００６】
本発明の一実施例は、実行可能イメージの真正性を判定するシステムである。このシステムは、一つ以上のポインタを有する実行可能イメージと、時間的に第１の点（第１の時点）で各ポインタ以外の実行可能イメージの選択された内容に基づいて対照デジタル署名を生成する機能を備えた検証部と、を具備し、検証部は、各ポインタ以外の実行可能イメージの選択された内容に基づいて第２の時点で真正デジタル署名を生成し、検証部は、対照デジタル署名が真正デジタル署名と一致するかどうかを判定する。
【０００７】
本発明の他の実施例は、実行可能イメージの真正性を判定するシステムであり、このシステムは、一つ以上のポインタを有する実行可能イメージを具備し、実行可能イメージは、各ポインタが実行可能イメージ内のロケーションを参照するかどうかを指定する情報を含み、このシステムは、各ポインタが実行可能イメージ内の対応したロケーションを参照するかどうかを判定する機能を備えた検証部を具備する。
【０００８】
本発明の更に別の実施例は、実行可能イメージの真正性を判定するシステムであり、このシステムは、第１の実行可能イメージと、第１の実行可能イメージ内のロケーションを参照するポインタを含む第２の実行可能イメージと、ポインタが第１の実行可能イメージ内のロケーションを参照するかどうかを判定する機能を備えた検証部と、を具備する。
【０００９】
本発明の更に別の実施例は、実行可能イメージの真正性を判定する機能を備えたシステムであり、このシステムは、第１の実行可能イメージ、第２の実行可能イメージ、及び、検証部を具備し、第２の実行可能イメージは、第１の実行可能イメージの識別子及び一つ以上の外部ポインタを収容するインポートテーブルと、機械コード及び一つ以上のインポートポインタを収容するコードセクションとを含み、各外部ポインタは第１の実行可能イメージ内のロケーションを参照し、各インポートポインタはインポートテーブル内のロケーションを参照し、検証部は、実行可能イメージの選択された内容に基づいて対照デジタル署名を第１の時点で生成する機能を備え、各インポートポインタ及び各外部ポインタは選択された内容に含まれず、検証部は、一つ以上のポインタの各々を含まない実行可能イメージの選択された内容に基づいて第２の時点で真正デジタル署名を生成し、検証部は、対照デジタル署名が真正デジタル署名と一致するかどうかを判定し、検証部は、各インポートポインタが第１の実行可能イメージ内のロケーションを参照するかどうかを判定し、検証部は、インポートポインタが第１の実行可能イメージ内のロケーションを参照するかどうかを判定する。
【発明の効果】
【００１０】
本発明によると、動的にコネクト可能な実行可能イメージの真正性を検証するシステム及び方法を提供することが可能となる。
【発明を実施するための最良の形態】
【００１１】
以下の説明は、本発明のある特定の実施例を対象にしている。しかし、本発明は、特許請求の範囲に記載され、対象とされているように様々な異なる態様で実施され得る。
【００１２】
・システム概要
図１は、コンピュータ９０を示す上位レベルのブロック図である。コンピュータ９０は、コンピュータ９０で実行される一つ以上の実行可能イメージ１００に関して真正性チェックを可能にする。
【００１３】
コンピュータ９０は、Ｐｅｎｔｉｕｍ（登録商標）プロセッサ、Ｐｅｎｔｉｕｍ （登録商標）Ｐｒｏプロセッサ、８０５１プロセッサ、ＭＰＳ（登録商標）プロセッサ、Ｐｏｗｅｒ　ＰＣ（登録商標）プロセッサ、或いは、ＡＬＰＨＡ（登録商標）プロセッサのよ

うな従来の汎用型の単一チップ若しくはマルチチップのマイクロプロセッサを利用する。
【００１４】
コンピュータ９０は、オペレーティングシステム９５とメモリ１０８を具備する。オペレーティングシステムは、どのオペレーティングシステムのベンダーから供給されるものでもよく、例えば、ＵＮＩＸ（登録商標）、ＬＩＮＵＸ、ディスク・オペレーティング・システム（ＤＯＳ）、ＯＳ／２、Ｗｉｎｄｏｗｓ（登録商標）　３．Ｘ、Ｗｉｎｄｏｗｓ（登録商標）　９５、Ｗｉｎｄｏｗｓ（登録商標）　９８、及び、Ｗｉｎｄｏｗｓ（登録商標）　ＮＴなどを含む。説明の便宜上、本発明の実施例は、Ｗｉｎｄｏｗｓ（登録商標）　９５に関して記述される。
【００１５】
コンピュータ９０は、実行可能イメージ１００をコンピュータ９０へ供給する一つ以上の実行可能イメージ供給元１０７と通信する。図１に示されるように、典型的な実行可能イメージ供給元には、サーバ１１０、インターネット１１４、データベース１１８、ネットワーク１２２、ハードウェア装置１２８、リムーバル記憶装置１３０などが含まれる。
【００１６】
実行可能イメージ１００は、それ自体で、若しくは、他の実行可能イメージと共に、一つ以上のソフトウェア・アプリケーションを定義することができるデータオブジェクトである。ソフトウェア・アプリケーションには、例えば、ワード・プロセッサ、データベース、デジタル権利管理システム、個人ファイナンス・ユーティリティ、グラフィック・ツール、インターネット・ブラウザ、コンピュータゲーム、通信プログラム、認証プログラム、電子ウォレット、マルチメディア・レンダラ、契約マネージャ等が含まれる。更に、実行可能イメージ１００は、他の実行可能イメージと動的に連結可能である。例えば、Ｗｉｎｄｏｗｓ（登録商標）９５と共に使用するため開発された本発明の一実施例において、実行可能イメージは、動的リンクテーブル（ＤＬＬ）である。
【００１７】
インターネット１１４は、公衆インターネット、私設インターネット、セキュアインターネット、私設網、公衆網、付加価値網、イントラネット等のようなバリエーションがある。
【００１８】
ネットワーク１２２は、任意のタイプの電子的に接続されたコンピュータのグループを含み、例えば、イントラネット、ローカル・エリア・ネットワーク（ＬＡＮ）、ワイド・エリア・ネットワーク（ＷＡＮ）のようなネットワークを含む。更に、ネットワークへの接続は、例えば、遠隔モデム、イーサネット（登録商標）（ＩＥＥＥ　８０２．３）、トークンリング（ＩＥＥＥ　８０２．５）、ファイバ・ディストリビューテッド・データリンク・インタフェース（ＦＤＤＩ）、又は、非同期転送モード（ＡＴＭ）等によって行われる。コンピューティング装置は、デスクトップ、サーバ、ポータブル、ハンドヘルド、セットトップ、若しくは、その他の望ましい構成の型である。ハードウェア装置１２６は、論理チップ、ＲＯＭ、ＲＡＭ、スマートカード、或いは、中央処理ユニットなどである。リムーバブル媒体記憶装置１３０は、フレキシブルディスク、コンパクトディスク、ハードディスク、テープドライブ、ＲＯＭ、若しくは、その他の持続性の記憶媒体などでもよい。
【００１９】
図２は、検証部２０４を示すブロック図である。本発明の一実施例において、検証部２０４は、実行可能イメージ１００にフォーマットが類似している実行可能イメージである。本発明の更に別の実施例において、検証部２０４は、プログラムローダー２０８と一体化される。プログラムローダー２０８の一つの機能は、記憶装置１０５（図１）からメモリ１０８へ実行可能イメージ１００をコピーし、実行可能イメージ１００の実行前にコード及びデータポインタを適切なアドレスにバインドする。説明の便宜上、以下では、検証部２０４は、実行可能イメージ１００及びプログラムローダー２０８とは異なる別個のアプリケーションである場合を考える。

【００２０】
検証部２０４は、以下で詳述する選択された条件下で実行可能イメージ１００の真正性を検証する。熟練した技術者によって認められるように、検証部２０４及びプログラムローダー２０８は、典型的に、別々にコンパイルされ、単一の実行可能プログラムにリンクされる様々なサブルーチン、プロシージャ、定義文、及び、マクロにより構成される。したがって、以下の記述は、これらの事項の機能性を説明する便宜のため使用される。
【００２１】
検証部２０４及びプログラムローダー２０８（図１）は、Ｃ、Ｃ＋＋、ＢＡＳＩＣ、Ｐａｓｃａｌ、Ｊａｖａ（登録商標）、及び、ＦＯＲＴＲＡＮのような任意のプログラミング言語で記述することができる。Ｃ、Ｃ＋＋、ＢＡＳＩＣ、Ｐａｓｃａｌ、Ｊａｖａ（登録商標）、及び、ＦＯＲＴＲＡＮは、業界標準プログラミング言語であり、実行可能コードを作成するため、多数の市販コンパイラ及びインタープリターを使用することが可能である。
【００２２】
図２は、実行可能イメージ２００に接続された後の実行可能イメージ１００の説明図である。尚、実行可能イメージ２００は、実行可能イメージ１００に関して既に説明したような同じデータオブジェクトのタイプにより構成され得る。検証部２０４の一つの機能は、実行可能イメージがメモリ１０８にロードされた後に、実行可能イメージ１００及び実行可能イメージ２００のような実行可能イメージの真正性を検証することである。
【００２３】
実行可能イメージの真正性を検証するプロセス（処理）は、図５及び６を参照して詳細に説明される。簡単に説明すると、検証部２０４は、実行可能イメージ１００がメモリ１０８にロードされる前に、実行可能イメージ１００を解析し、実行可能イメージ１００に関する対照デジタル署名を生成する。実行可能イメージ１００がロードされた後、検証部２０４は、実行可能イメージ１００が改ざんされていないことを保証するため、真正デジタル署名を生成する。更に、検証部２０４は、実行可能イメージ１００とのバインディングが他の実行可能イメージへ不正に書き換えられていないことを保証するため、実行可能イメージ１００と実行可能イメージ２００の間のバインディングを調べる。
【００２４】
図３は、動的リンクライブラリとして構築され、記憶装置１０５（図１）に格納された実行可能イメージ１００の一実施例の内部構造を詳細に説明するブロック図である。
【００２５】
実行可能イメージ１００は、ヘッダセクション３００、コードセクション３０４、データセクション３０８、インポートテーブル３１２、再配置（リロケーション）テーブル３１６、及び、エクスポートテーブル３２０を含む多数のセクションを収容する。尚、実行可能イメージ１００は、後で詳述される多数のポインタを収容することに注意する必要がある。一般的に、ポインタは、メモリ１０８内のロケーションを、メモリ１０８（図１）に関して絶対的に、或いは、その他のロケーションに対して相対的に特定する参照情報（リファレンス）である。
【００２６】
ヘッダセクション３００は、実行可能イメージ１００内の他のセクション及び／又はテーブルの相対ロケーションを特定する。コードセクション３０４は、実行可能イメージ３００に対するコンパイルされた機械コードを含む。例えば、コードセクション３０４は、コンピュータ９０（図１）のための機械命令を含む。図３に示されるように、コードセクション３０４は、実行可能イメージ１００内、並びに、実行可能イメージ１００以外の他のセクションを参照する命令を含む。ブロック３２０に示されるように、コードセクション３０４は、数値”２”をグローバル変数”ｉ”に割当てるための命令を含む。しかし、記憶装置１０５（図１）に格納されている変数”ｉ”の実際のアドレスは、メモリ１０８に規定されていない。なぜならば、実行可能イメージ１００は、未だメモリ１０８（図１）にロードされていないからである。更に、コードセクション３０４は、巻数ｆｏｏ（）

を呼び出すための命令を含む。プロシージャｆｏｏ（）への呼び出しは、インポートテーブル３１２内のロケーションを参照するインポートポインタを含む。

【００２７】

データセクション３０８は、コードセクション３０４で特定されたグローバル変数を記憶するため使用される。インポートテーブル３１２は、実行可能テーブル１００を別の実行可能テーブルへ接続する際にプログラムローダー２０８（図２）を補助するため、様々な情報の項目を含む。インポートテーブル３１２は、他の実行可能イメージによって保持されるプロシージャ毎にその実行可能イメージの名前のような識別子と、外部に保持されたプロシージャのアドレスを参照する一つ以上の外部ポインタと、を含む。インポートテーブル３１２は、記憶装置１０５（図１）に記憶されているので、未だプロシージャｆｏｏ（）のメモリアドレスを参照しない。

【００２８】

再配置テーブル３１６は、実行可能イメージ１００のローディング後の「フィクシングアップ（用意）」を必要とするコードセクション３０４の各部分の位置を特定する。用語「フィクシングアップ」は、未解決のポインタが適当なデータロケーション及び／又はコードロケーションを参照するように、メモリの実行可能イメージ１００を変更するプロセスを表すため使用されている。ポインタがプログラムローダー２０８によってフィクシングアップされると、ポインタは選択されたアドレスに「バインド（結合）」される。

【００２９】

エクスポートテーブル３２０は、実行可能イメージ１００によって公然と利用可能にされた各プロシージャを特定する。尚、実行可能イメージ１００は、デバッグ情報、又は、ローディングプロセス及び／又はリンクプロセスを補助するため用いるその他のテーブル他の情報を含み得ることに注意する必要がある。

【００３０】

検証部２０４（図２）は、記憶装置１０５に保存されている実行可能イメージ１００に関して対照デジタル署名を決定する。この対照署名を生成する処理は、図６に関して詳述する。本発明の一実施例によれば、検証部２０４は、プログラムローダー２０８（図２）によってフィクシングアップを必要とするポインタを除く実行可能イメージ１００の全体に関して対照デジタル署名を決定する。本発明の他の一実施例では、検証部２０４は、プログラムローダー２０８によってフィクシングアップを必要とするアドレスを除く実行可能イメージ１００のコードセクション３０４及び／又はインポートテーブル３１２のような選択されたセクションに関して紹介デジタル署名を決定する。

【００３１】

図４は、実行可能イメージ２００に関して用意された（フィクシングアップされた）後の実行可能イメージ１００を示すブロック図である。図４を参照することによってわかるように、変数”ｉ”に対するデータポインタは、再配置テーブル３１６に収容されたフィクスアップ情報に従って、データセクション３０８のアドレスに結び付けられる。更に、インポートテーブル３１２のデータセクション３０８は、実行可能テーブル２００のエクスポートテーブルに結び付けられる。第２の実行可能イメージ２００のエクスポートテーブルの参照アドレスへのエクスポートポインタは、実行可能イメージ２００内にあるプロシージャｆｏｏ（）の実際のロケーションに結び付けられる。

【００３２】

フィクシングアップされた後、検証部２０４（図２）は、実行可能イメージ１００に関して様々な真正性検査を行う。これらの各真正性検査は、図５及び６を参照して以下で詳述される。簡単に説明すると、検証部２０４は以下の機能を実行する。

【００３３】

第一に、検証部２０４は、対照デジタル署名を生成するため使用された同じアドレスに関して真正デジタル署名を決定する。真正デジタル署名が対照デジタル署名と異なる場合、検証部２０４（図２）は、実行可能イメージ１００が不正に改ざんされたことを想定する。

【００３４】
第二に、検証部２０４は、インポートテーブルによって参照されたアドレスが改ざんされていないことを保証するため、インポートテーブル３１２の各バインディングを調べる。インポートテーブル３１２が不正に変更された場合、選択された巻数へのプロシージャ呼び出しは、実行可能イメージ１００の供給元によって予定されたルーチンとは別の信用できないルーチンへルート変更される場合がある。このような信用できないルーチンは、偶然に、若しくは、意図的に、間違ったデータ、又は、有害なデータを実行可能イメージ１００へ戻す可能性がある。

【００３５】
図５は、実行可能イメージ１００の真正性を検証する処理を示すフローチャートである。ステップ６００から始まり、検証部２０４（図２）は、実行可能モジュール１００（図１）の真正を判定する要求を受信する。本発明の一実施例において、この要求はプログラムローダー２０８によって生成される。本発明の他の実施例では、この要求は、オペレーティングシステム９５（図１）によって生成される。本発明の更に別の一実施例では、この要求は、コンピュータ９０（図１）上で動く実行可能イメージ（図示せず）によって生成される。本発明の更に別の一実施例では、この要求は、検証部２０４内のルーチンによって生成される。

【００３６】
ステップ６０４へ進み、検証部２０４（図２）は、実行可能イメージ１００内の各ポインタを特定する。本発明の一実施例において、実行可能イメージ１００内の選択されたセクションだけの真正性が決定されるべき場合、検証部２０４は、実行可能イメージ１００の選択されたセクションに収まるポインタだけを特定する。例えば、検証部２０４は、コードセクション（図３）又はインポートテーブル（図３）に収まるポインタだけを特定するように構成することが可能である。

【００３７】
本発明の一実施例において、検証部２０４は、これらの各ポインタ（図３）を特定するため、再配置テーブル３１６を準備する。図３を参照して説明したように、再配置テーブル３１６は、実行可能イメージ１００のベースに関して相対的にアドレスのロケーションを特定する。再配置テーブル３１６を調べることによって、検証部２０４は、ローディング中にプログラムローダー２０８によって変更されるコードセクション３０４の部分を特定することが可能である。

【００３８】
ステップ６０８へ進み、検証部２０４（図２）は、実行可能イメージ１００（図１）に対する対照デジタル署名を生成する。ここで使用されるデジタル署名は、選択されたデータ集合の内容を特定する任意の方法論を包含するように規定される。最も簡単な形式では、デジタル署名は、署名されるべきデータから選択されたデータ集合の完全なコピーを含み得る。しかし、デジタル署名は、選択されたデータ集合に適用されたハッシング関数の結果でも構わない。更に、デジタル署名はデジタル証明書でもよい。熟練した技術者であれば、対照デジタル署名を生成するために多数の標準的なハッシュ関数の中の任意のハッシュ関数を使用可能であることが分かるであろう。

【００３９】
更に、ステップ６０８に関して説明すると、本発明の一実施例によれば、検証部２０４は、プログラムローダー２０８（図２）によるフィクシングアップを必要とするアドレスを除く実行可能イメージ１００の全体の内容に基づいて対照デジタル署名を決定する。本発明の他の実施例によれば、検証部２０４は、プログラムローダー２０８によるフィクシングアップを必要とするアドレスを除くコードセクション３０４及び／又はインポートテーブル３１２のような選択されたセクションの内容に基づいて対照デジタル署名を決定する。

【００４０】
ステップ６１２へ進むと、検証部２０４（図２）は、後で取り出すため対照デジタル署

名を記憶する。本発明の一実施例において、検証部２０４は、記憶装置１０５（図１）に対照デジタル署名を格納する。本発明の他の実施例において、検証部２０４は、リソーステーブル（図示せず）のような実行可能イメージ１００の選択されたセクションに対照デジタル署名を格納する。本発明の更に別の実施例において、対照デジタル署名は実行可能イメージ１００に付加される。本発明の更に別の実施例において、対照デジタル署名は、データベース、ウェブ・サーバ、若しくは、ネットワーク１２２（図１）に記憶される。本発明の更に別の実施例において、対照デジタル署名は、実行可能イメージ１００がコンピュータ９０へ供給される前に作成される。この実施例の場合、対照デジタル署名は、上述の何れの方法で生成してもよい。

【００４１】

次に、ステップ６１４において、検証部２０４（図２）は、実行可能イメージ１００がプログラムローダー２０８によってメモリ１０８（図２）にロードされる後若しくは前に、実行可能イメージ１００の真正署名を決定する。このステップにおいて、検証部２０４は、ステップ６０８の間に検証部２０４によって適用されたハッシュ関数を再適用する。検証部２０４は、タイマーの満了、実行可能イメージ１００の真正性セルフテストの終了、コンピュータ９０（図１）のアイドル時間の終了、又は、実行可能イメージ１００の要求時のような一つ以上の選択された条件の発生時に真正デジタル署名を決定するように構成することができる。

【００４２】

判定ステップ６１８へ進み、検証部２０４（図２）は、（ステップ６０８で生成された）対照デジタル署名が（ステップ６１４で生成された）真正デジタル署名と一致するかどうかを判定する。対照デジタル署名が真正デジタル署名と一致しない場合、検証部２０４はステップ６２２へ進み、ステップ６２２において、検証部２０４はセキュリティ警報を開始する。ステップ６２２において、検証部２０４は、実行可能イメージ１００のアンロード、記憶装置１０５（図１）からメモリ１０８（図１）への実行可能イメージ１００の新しいコピーのロード、ネットワーク１２２からの実行可能イメージ１００の新バージョンのロード、記憶装置１０５からの実行可能イメージ１００の削除、コンピュータ９０（図１）の場所にいるユーザに対する警告表示、ネットワーク１２２を介した遠隔コンピュータ（図示せず）へのエラーメッセージ送信、或いは、実行可能イメージ１００によって実行された一つ以上の動作の取消などのような複数の機能を実行してもよい。

【００４３】

再度、判定ステップ６１８について説明すると、検証部２０４（図２）が、対照デジタル署名は真正デジタル署名と一致していることを判定した場合、検証部２０４は終了ステップ６２４へ進む。実施例に応じて、検証部２０４（図２）は、選択された条件の発生時に真正デジタル署名を再決定するため、ステップ６１４へ戻る。

【００４４】

図６は、図２に示された一方の実行可能イメージの真正性検査中に図２の検証部によって実行される別の処理を説明するフローチャートである。図６のフローチャートによって実行される処理は、図５のフローチャートに従って実行される処理とは異なり、単独で実行することも、図５で実行される処理と共に実行することも可能である。特に、図６には、実行可能イメージの各ポインタが適切なロケーションにバインドされていることを検証する処理が示されている。以下、実行可能イメージ１００のコードセクション３０４のインポートポインタ、及び、インポートテーブル３１２のエクスポートポインタに関する真正性検査処理について説明する。

【００４５】

ステップ７００から始まり、検証部２０４（図２）は、要求元から、実行可能モジュール１００（図１）の真正性を決定する要求を受信する。本発明の一実施例において、要求元はプログラムローダー２０８（図２）である。本発明の別の一実施例において、要求元はオペレーティングシステム（図１）である。本発明の更に別の実施例において、要求元はコンピュータ９０（図１）で動く実行可能イメージ（図示せず）である。本発明の更に

別の一実施例において、要求元は検証部２０４内のルーチンである。更に、要求は、一つ以上の選択された条件の出現時に、一つの要求元によって開始され得る。選択された条件には、タイマーの終了、コンピュータ９０（図１）に関するアイドル時間の検出、及び／又は、銀行取引のような重要な動作の実行前などが含まれる。

【００４６】

ステップ７０４へ進み、検証部２０４（図２）は、コードセクション３０４内の各インポートポインタを特定する。本発明の一実施例において、検証部２０４は、これらの各インポートテーブル（図３）を特定するため、再配置テーブル３１６を解析する。

【００４７】

次に、ステップ７０８において、検証部２０４（図２）は、各インポートポインタがインポートテーブル３１２内のロケーションに結び付けられているかどうかを判定するため、コードセクション３２０の各インポートポインタを調べる。このステップで、本発明の一実施例において、検証部２０４は、インポートテーブル３１２の開始アドレス及び終了アドレスを決定するためヘッダ３００を読み出す。

【００４８】

各インポートテーブルがインポートテーブル３１２（図３）内のロケーションに結び付けられていない場合、検証部２０４（図２）は、ステップ７１４へ進み、検証部２０４はセキュリティ警報を開始する。ステップ７１４において、検証部２０４は、実行可能イメージ１００のアンロード、記憶装置１０５（図１）からメモリ１０８（図１）への実行可能イメージ１００の新しいコピーのロード、実行可能イメージ１００の新しいコピーのロード、記憶装置１０５からの実行可能イメージ１００の削除、コンピュータ９０（図１）の場所にいるユーザに対する警告表示、ネットワーク１２２を介した遠隔コンピュータ（図示せず）へのエラーメッセージ送信、或いは、実行可能イメージ１００によって実行された一つ以上の動作の取消などのような複数の機能を実行してもよい。処理フローは、終了ステップ７１５へ進み、終了する。

【００４９】

再度、判定ステップ７１２について説明すると、検証部２０４（図２）が、コードセクション３０４内の各インポートポインタがインポートテーブル（図３）に結び付けられていると判定した場合、検証部２０４はステップ７１６へ進む。このステップ７１６において、検証部２０４は、インポートテーブル３１２内の各外部ポインタを特定する。

【００５０】

ステップ７２０へ進み、検証部２０４（図２）は、インポートテーブル３１２（図２）内の外部ポインタの結合ロケーションを決定する。本発明の一実施例において、検証部２０４は、検証部２０４によって保持される外部ポインタテーブル（図示せず）に結合ロケーションを格納する。

【００５１】

判定ステップ７２２へ進み、検証部２０４は、外部ポインタの結合ロケーションがインポートテーブル３１２（図２）によって特定された一方の実行可能イメージ内に存在するかどうかを判定する。説明の便宜上、インポートテーブル３１２によって特定された実行可能イメージは、一まとめにしてエクスポート用実行可能イメージと呼ぶ。

【００５２】

本発明の一実施例において、検証部２０４は、メモリ１０８（図１）内のエクスポート用実行可能イメージのロケーションを決定するため、オペレーティングシステム（図１）を呼び出す。例えば、Ｗｉｎｄｏｗｓ（登録商標）　９５は、実行可能イメージの名前が与えられた場合に、実行可能イメージのベースアドレスを返すＧｅｔＭｏｄｕｌｅＨａｎｄｌｅ（）という名称のプロシージャ呼び出しを提供する。このベースアドレスを使用して、検証部２０４は、エクスポート用実行可能イメージのヘッダのロケーション、並びに、他のセクションのロケーションを特定することができる。

【００５３】

更に、ステップ７２２について説明すると、検証部２０４が、各外部ポインタはエクス

ポート用実行可能イメージに結合されていない、と判定した場合、検証部２０４は、既に詳述したステップ７１４へ進む。しかし、検証部２０４が、各外部ポインタはエクスポート用実行可能イメージに結合されている、と判定した場合、検証部２０４は、終了ステップ７１５へ進み、終了する。

【００５４】

有利的には、本発明は、実行可能イメージの静的検証、動的検証、及び、ランタイム検証が可能であり、実行可能イメージのソースコード又はオブジェクトコードに変更を加える必要が無い。本発明は、実行可能イメージが真正であり、ロード後に改ざんされていないことを保証する。対照デジタル署名が決定された後、検証部２０４は、次に、データオブジェクトが変更されていないことを保証するため、真正デジタル署名を生成可能である。更に、検証部２０４は、実行可能イメージ内の各ポインタが正確なロケーションに結合されているかどうかを判定可能である。

【００５５】

例えば、コードセクション３０４のデータポインタに関して、検証部２０４は、データポインタがデータセクション３０８内のロケーションを参照することを保証するため、再配置テーブル３１６を検査可能である。更に、例えば、インポートテーブル３１２の外部ポインタに関して、検証部２０４は、外部ポインタがインポートテーブル３１２に名前が含まれている信頼できる実行可能イメージを参照することを保証可能である。実行可能イメージの名前はプログラムローダー２０８（図２）によって変更されないので、検証部２０４は、信頼できる実行可能イメージの名前が対照デジタル署名及び真正デジタル署名の使用によって変更されないことを保証可能である。

【００５６】

上記の詳細な説明では、種々の実施例に適用されるような本発明の新規な特徴が図示され、記載され、指摘されているが、例示された装置又は方法の形式及び細部に関する様々な省略、置換、及び、変更は、本発明の精神から逸脱すること無く、当業者によってなされることが明らかである。本発明の範囲は、上記の説明ではなく、特許請求の範囲に記載された事項によって表される。特許請求の範囲に記載された事項の均等物の意味及び範囲に含まれる全ての変更は、本発明の範囲に包含される。

【図面の簡単な説明】

【００５７】

【図１】一つ以上の供給元から実行可能イメージを受信するように適合した本発明のコンピュータを示す上位レベルのブロック図である。

【図２】図１のコンピュータに搭載され、一つ以上の実行可能イメージの真正性を判定する機能を備えた検証部を示すブロック図である。

【図３】図２の一方の実行可能イメージの内部構造を示すブロック図である。

【図４】ローディング処理後にリンクされた図２の二つの実行可能イメージを示すブロック図である。

【図５】図２に示された一方の実行可能イメージの真正性チェック中に図２の検証部によって実行される処理のフローチャートである。

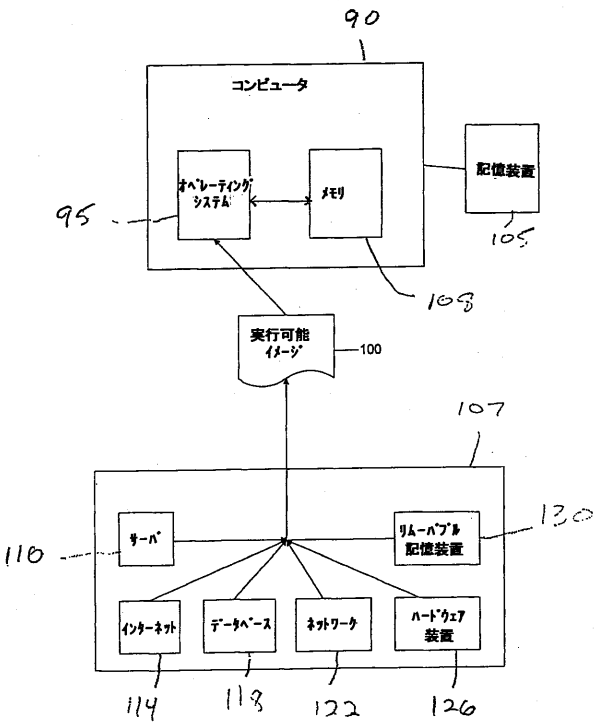
【図６】図２に示された一方の実行可能イメージの真正性チェック中に図２の検証部によって実行される別の処理のフローチャートである。

【符号の説明】

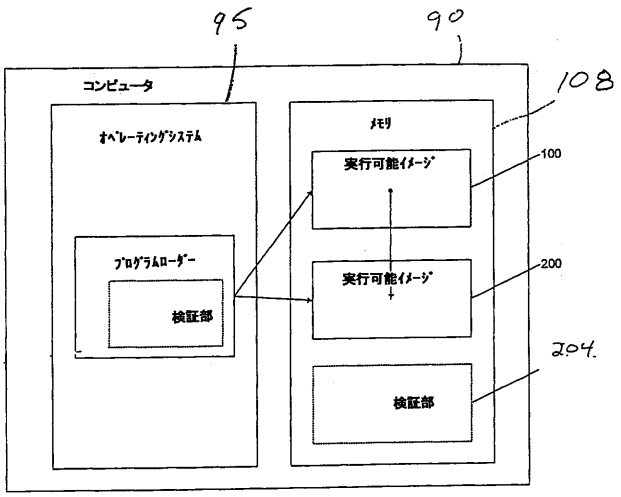
【００５８】

９０　コンピュータ
１００　実行可能イメージ
１０５　記憶装置
１０８　メモリ
２０４　検証部
２０８　プログラムローダー

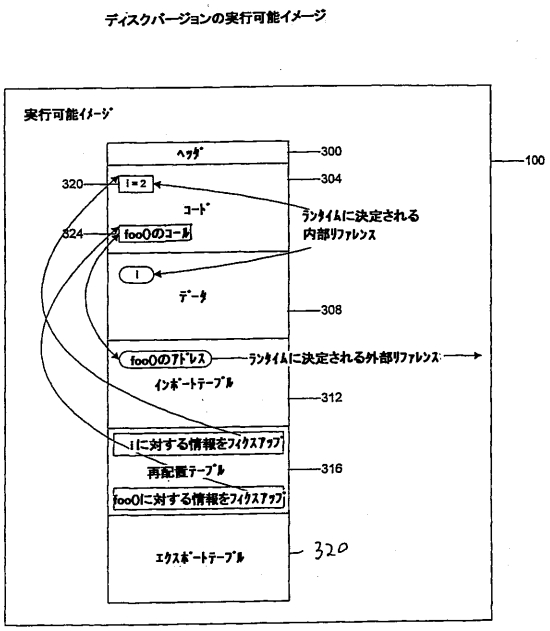
【図１】


90
コンピュータ
オペレーティングシステム
メモリ
記憶装置
95
105
108
実行可能イメージ
100
107
サーバ
リムーバブル記憶装置
130
110
インターネット
データベース
ネットワーク
ハードウェア装置
114
118
122
126

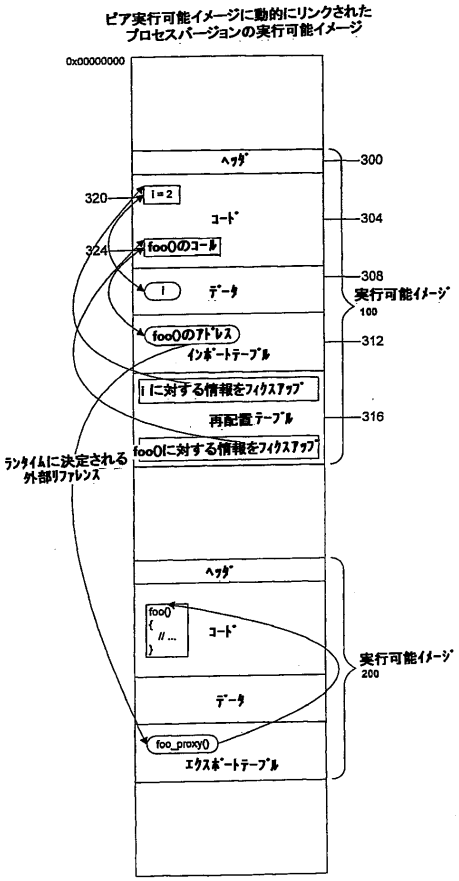
【図２】


95
90
コンピュータ
108
オペレーティングシステム
メモリ
実行可能イメージ
100
プログラムローダー
検証部
実行可能イメージ
200
204
検証部

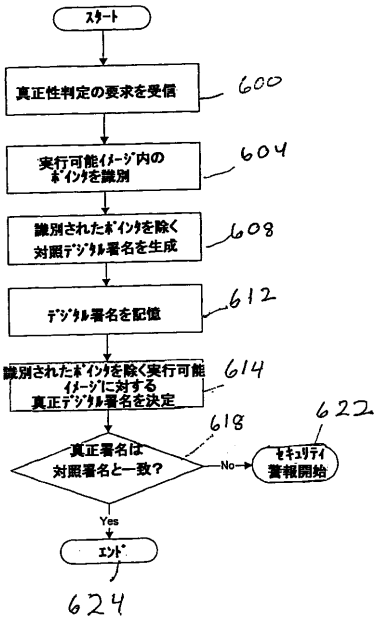
【図３】


ディスクバージョンの実行可能イメージ
実行可能イメージ
100
ヘッダ
300
320
i = 2
304
コード
ランタイムに決定される内部リファレンス
324
foo()のコール
i
データ
308
foo()のアドレス
ランタイムに決定される外部リファレンス
インポートテーブル
312
i に対する情報をフィクスアップ
再配置テーブル
316
foo()に対する情報をフィクスアップ
エクスポートテーブル
320

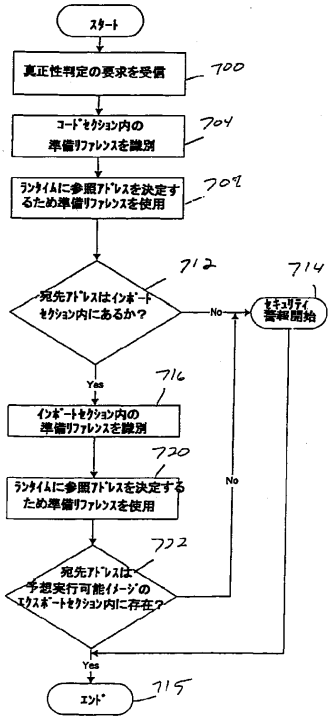
【図４】


ピア実行可能イメージに動的にリンクされたプロセスバージョンの実行可能イメージ
0x00000000
ヘッダ
300
320
i = 2
304
コード
324
foo()のコール
i
データ
308
実行可能イメージ
100
foo()のアドレス
インポートテーブル
312
i に対する情報をフィクスアップ
再配置テーブル
316
ランタイムに決定される外部リファレンス
foo()に対する情報をフィクスアップ
ヘッダ
foo()
{
//...
}
コード
実行可能イメージ
200
データ
foo_proxy()
エクスポートテーブル

【図５】


スタート
真正性判定の要求を受信
600
実行可能イメージ内の
ポインタを識別
604
識別されたポインタを除く
対照デジタル署名を生成
608
デジタル署名を記憶
612
識別されたポインタを除く実行可能
イメージに対する
真正デジタル署名を決定
614
真正署名は
対照署名と一致?
618
No
セキュリティ
警報開始
622
Yes
エンド
624

【図６】


スタート
真正性判定の要求を受信
700
コードセクション内の
準備リファレンスを識別
704
ランタイムに参照アドレスを決定する
ため準備リファレンスを使用
708
宛先アドレスはインポート
セクション内にあるか?
712
No
セキュリティ
警報開始
714
Yes
インポートセクション内の
準備リファレンスを識別
716
ランタイムに参照アドレスを決定する
ため準備リファレンスを使用
720
No
宛先アドレスは
予想実行可能イメージの
エクスポートセクション内に存在?
722
Yes
エンド
715