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

特許第5734685号
(P5734685)

(45)発行日　平成27年6月17日(2015.6.17)　(24)登録日　平成27年4月24日(2015.4.24)

(51)Int.Cl.　F I
G06F 21/12 (2013.01)　G06F 21/12
G06F 21/64 (2013.01)　G06F 21/64

請求項の数 6　外国語出願　(全 12 頁)

(21)出願番号　特願2011-20280(P2011-20280)
(22)出願日　平成23年2月2日(2011.2.2)
(65)公開番号　特開2011-170847(P2011-170847A)
(43)公開日　平成23年9月1日(2011.9.1)
審査請求日　平成26年1月16日(2014.1.16)
(31)優先権主張番号　10305164.5
(32)優先日　平成22年2月18日(2010.2.18)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　501263810
トムソン　ライセンシング
Ｔｈｏｍｓｏｎ　Ｌｉｃｅｎｓｉｎｇ
フランス国，　９２１３０　イツシー　レ　ムーリノー，　ル　ジヤンヌ　ダルク，　１－５
１－５，　ｒｕｅ　Ｊｅａｎｎｅ　ｄ’Ａｒｃ，　９２１３０　ＩＳＳＹ　ＬＥＳ　ＭＯＵＬＩＮＥＡＵＸ，　Ｆｒａｎｃｅ
(74)代理人　100070150
弁理士　伊東　忠彦
(74)代理人　100091214
弁理士　大貫　進介
(74)代理人　100107766
弁理士　伊東　忠重

最終頁に続く

(54)【発明の名称】インテグリティを実行中に確かめるソフトウェアを生成するプログラム、方法及び記憶媒体

(57)【特許請求の範囲】
【請求項１】
プロセッサによって実行される場合に、該プロセッサに、
複数のモジュールを有し、該複数のモジュールの夫々が、暗号化された状態及び暗号化されていない状態を含む少なくとも２つの状態のうちのいずれかの状態をとることができる自己書換型のバイナリを受け取るステップと、
前記複数のモジュールがとり得る状態の組み合わせごとに状態変数を生成するステップと、
コールグラフ解析を用いて、前記状態変数から、とり得る値を有する状態変数のサブセットを生成するステップと、
前記サブセットに含まれる状態変数ごとに、対応するチェックサムを生成するステップと、
前記サブセットに含まれる状態変数の数に基づき、前記チェックサムを前記バイナリに埋め込む方法を決定するステップと、
前記チェックサムを照合するための少なくとも１つの呼び出し関数を前記バイナリに挿入することによって、且つ、前記決定された方法に従って前記チェックサムを前記バイナリに埋め込むことによって、インテグリティが保護された自己書換型のバイナリを生成するステップと
を実行させる保護エンジンプログラム。
【請求項２】

前記チェックサムを埋め込む方法を決定するステップは、
前記プロセッサに、前記サブセットに含まれる状態変数の数に基づき、前記チェックサムの夫々を、そのチェックサムを照合するための呼び出し関数に埋め込む第１の方法、又は前記チェックサムをルックアップテーブルに含める第２の方法のいずれかを選択させることを含む、
請求項１に記載の保護エンジンプログラム。
【請求項３】
前記状態変数の夫々は、状態ビットのストリームを有し、該状態ビットの夫々は、前記複数のモジュールの夫々１つに対応し、該対応するモジュールの状態を示す、
請求項１に記載の保護エンジンプログラム。
【請求項４】
前記チェックサムは、ハッシュ値である、
請求項１乃至３のうちいずれか一項に記載の保護エンジンプログラム。
【請求項５】
請求項１乃至４のうちいずれか一項に記載の保護エンジンプログラムを記憶するコンピュータ可読記憶媒体。
【請求項６】
インテグリティが保護された自己書換型のバイナリを生成するための装置の作動方法であって、
複数のモジュールを有し、該複数のモジュールの夫々が、暗号化された状態及び暗号化されていない状態を含む少なくとも２つの状態のうちのいずれかの状態をとることができる自己書換型のバイナリを受け取るステップと、
前記複数のモジュールがとり得る状態の組み合わせごとに状態変数を生成するステップと、
コールグラフ解析を用いて、前記状態変数から、とり得る値を有する状態変数のサブセットを生成するステップと、
前記サブセットに含まれる状態変数ごとに、対応するチェックサムを生成するステップと、
前記サブセットに含まれる状態変数の数に基づき、前記チェックサムを前記バイナリに埋め込む方法を決定するステップと、
前記チェックサムを照合するための少なくとも１つの呼び出し関数を前記バイナリに挿入することによって、且つ、前記決定された方法に従って前記チェックサムを前記バイナリに埋め込むことによって、前記インテグリティが保護された自己書換型のバイナリを生成するステップと
を有する方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、概して、ソフトウェア、特に、ソフトウェアのインテグリティを確保すること、に関する。
【背景技術】
【０００２】
本項目は、以下で記載及び／又は請求される本発明の様々な態様に関連する技術の様々な側面を読む者に紹介することを目的とする。この議論は、本発明の様々な態様のより良い理解を助けるよう読む者に背景情報を提供するのに役立つと信じられている。従って、当然、これらの記述は、先行技術の容認としてではなく、この観点から読まれるべきである。
【０００３】
プログラムが意図されたように実行されることを確かにすることを目的にコンピュータプログラムのインテグリティを保護することは、ソフトウェアのプロバイダにとって、比

較的一般的である。しかし、通常、ハッカーは、様々な方法で、実行すべきプログラムに不正侵入しようとする。一例として、ハッカーは、ときどき、必要なアクセス権原を有さずにプログラムを使用することができるように、プログラムのアクセス制御機能を回避すべくコードを書き換えることを望む。

【０００４】

プログラムのインテグリティを保証する先行技術に係る方法は、コードの少なくとも幾つかの部分に対してシグニチャ（「チェックサム」ともいう。）を計算することである。例えば、シグニチャは、コードの部分に対して計算されて、秘密キーにより署名されたハッシュ値であってよい。当業者には当然のことながら、多くの他の可能性が存在する。プログラム実行中に、コードのシグニチャは少なくとも一度計算される。セキュリティ・レベルを高めるために、シグニチャを計算する関数が入れ子にされ、それにより、各関数のインテグリティは少なくとも１つの他の関数によって確かめられる。このようにして、ただ１つの関数が損なわれないままである場合、それは、少なくとも１つの他の関数を改ざんすることを検出する。更なる詳細及び説明は、例えば、米国特許出願公報第２００３／１８８２３１号明細書（特許文献１）及び欧州特許第１９４２４３１号明細書（特許文献２）で見つけられ、また、米国特許出願公開第２００２／１３８７４８号明細書（特許文献３）は、どのように再配置可能コードについてインテグリティを決定すべきかを教示している。同様の解決法は、Ｄ．Ａｕｃｓｍｉｔｈ、Ｔａｍｐｅｒ Ｒｅｓｉｓｔａｎｔ Ｓｏｆｔｗａｒｅ：Ａｎ Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｗｏｒｋｓｈｏｐ ｏｎ Ｉｎｆｏｒｍａｔｉｏｎ Ｈｉｄｉｎｇ（非特許文献１）で見つけられる。この文献において、暗号化されたインテグリティ関数は復号化されて、特別の暗号化されていないモジュールのインテグリティを確かめるために使用される。

【０００５】

予想通りに、このような保護に対処する方法は、ハッカーではなく学究的な興味よって見出されてきた（例えば、Ｇｌｅｎ Ｗｕｒｓｔｅｒ等、“Ａ Ｇｅｎｅｒｉｃ Ａｔｔａｃｋ ｏｎ Ｃｈｅｃｋｓｕｍｍｉｎｇ Ｂａｓｅｄ ｏｎ Ｓｏｆｔｗａｒｅ Ｔａｍｐｅｒ Ｒｅｓｉｓｔａｎｃｅ”、ＳＰ’０５：Ｐｒｏｃｅｅｄｉｎｇ ｏｆ ｔｈｅ ２００５ ＩＥＥＥ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｓｅｃｕｒｉｔｙ ａｎｄ Ｐｒｉｖａｃｅ、１２７～１３８頁、米国ワシントンＤＣ、２００５年、ＩＥＥＥ Ｃｏｍｐｕｔｅｒ Ｓｏｃｉｅｔｙ（非特許文献２））。かかる方法はコードの２つのコピーを用いる。１つのコピーは変更されておらず、もうひとつのコピーは変更されている。変更されているコードのコピーは実行され、チェックサムが必要とされるときには、これは変更されていないコードのコピーを用いて計算される。このようにして、当該方法は、コードの変更を可能にし、同時に、正しいチェックサム、すなわち、変更されていないコードに対応するチェックサム、を提供することが可能である。なお、攻撃はあらゆる種類のプロセッサにおいて行われ得ないことが認知されるべきである。

【０００６】

非特許文献２で提案されている対策は、コードを含むページについてアクセス権原を変更することである。コードを読み出す権利がそのコード自体についても解除される場合、これは、変更されていないコードを提供するよう接続される割り込みを（コードが自身を読み出そうとする場合に）引き起こす。

【０００７】

他の問題は、先行技術に係るチェックサムは自己書換コード（self-modifying code）に適合しないことである。これは、特定のチェックサムが、コードの１バージョンについてのみ、すなわち、変更の前又は後にのみ、有効であるためである。

【先行技術文献】

【特許文献】

【０００８】

【特許文献１】米国特許出願公報第２００３／１８８２３１号明細書

【特許文献２】欧州特許第１９４２４３１号明細書
【特許文献３】米国特許出願公開第２００２／１３８７４８号明細書
【非特許文献】
【０００９】
【非特許文献１】Ｄ．Ａｕｃｓｍｉｔｈ、Ｔａｍｐｅｒ Ｒｅｓｉｓｔａｎｔ Ｓｏｆｔｗａｒｅ：Ａｎ Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｗｏｒｋｓｈｏｐ ｏｎ Ｉｎｆｏｒｍａｔｉｏｎ Ｈｉｄｉｎｇ
【非特許文献２】Ｇｌｅｎ Ｗｕｒｓｔｅｒ等、“Ａ Ｇｅｎｅｒｉｃ Ａｔｔａｃｋ ｏｎ Ｃｈｅｃｋｓｕｍｍｉｎｇ Ｂａｓｅｄ ｏｎ Ｓｏｆｔｗａｒｅ Ｔａｍｐｅｒ Ｒｅｓｉｓｔａｎｃｅ”、ＳＰ’０５：Ｐｒｏｃｅｅｄｉｎｇ ｏｆ ｔｈｅ ２００５ ＩＥＥＥ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｓｅｃｕｒｉｔｙ ａｎｄ Ｐｒｉｖａｃｅ、１２７～１３８頁、米国ワシントンＤＣ、２００５年、ＩＥＥＥ Ｃｏｍｐｕｔｅｒ Ｓｏｃｉｅｔｙ
【発明の概要】
【発明が解決しようとする課題】
【００１０】
従って、ソフトウェア、特に、自己書換型であるコード、のインテグリティを保証するための改善された解決法が必要である。本発明は、かかる解決法を提供する。
【課題を解決するための手段】
【００１１】
第１の態様で、本発明は、自己書換ソフトウェアコードのインテグリティをその実行中に確かめる方法を対象とする。ソフトウェアコードは複数のモジュールを有し、各モジュールは、ソフトウェアコードの実行中に、少なくとも２つのとり得る状態、すなわち、暗号化された状態及び暗号化されていない状態、にあることができる。ソフトウェアコードを実行するプロセッサは、複数のモジュールのうちの１つを第１の状態から第２の状態に変えることによってソフトウェアコードを変更し、その変更されたソフトウェアコードについてチェックサムと変更されたソフトウェアコードを比較することによってソフトウェアコードのインテグリティ確かめる。
【００１２】
第１の望ましい実施形態で、チェックサムはハッシュ値である。
【００１３】
第２の望ましい実施形態で、チェックサムは、モジュールのインテグリティを確かめる関数に埋め込まれている。
【００１４】
第３の望ましい実施形態で、チェックサムはルックアップテーブルに含まれる。有利に、変更されたソフトウェアコードのインテグリティを確かめる関数は、ルックアップテーブルに含まれるチェックサムにアクセスするために、複数のモジュールの夫々の状態を示す状態変数を使用する。
【００１５】
第２の態様で、本発明は、自己書換ソフトウェアコードのインテグリティをその実行中に確かめる装置を対象とする。ソフトウェアコードは複数のモジュールを有し、各モジュールは、ソフトウェアコードの実行中に、少なくとも２つのとり得る状態、すなわち、暗号化された状態及び暗号化されていない状態、にあることができる。当該装置は、ソフトウェアコードを実行し、それによって、複数のモジュールのうちの１つを第１の状態から第２の状態に変えることによってソフトウェアコードを変更し、その変更されたソフトウェアコードについてチェックサムと変更されたソフトウェアコードを比較することによってソフトウェアコードのインテグリティ確かめるよう構成されたプロセッサを有する。
【００１６】
第１の望ましい実施形態で、チェックサムはハッシュ値である。

【００１７】
第２の望ましい実施形態で、チェックサムは、モジュールのインテグリティを確かめる関数に埋め込まれている。
【００１８】
第３の望ましい実施形態で、チェックサムはルックアップテーブルに含まれる。有利に、変更されたソフトウェアコードのインテグリティを確かめる関数は、ルックアップテーブルに含まれるチェックサムにアクセスするために、複数のモジュールの夫々の状態を示す状態変数を使用する。
【００１９】
第３の態様で、本発明は、インテグリティが保護された自己書換バイナリを生成する装置を対象とする。バイナリは複数のモジュールを有し、各モジュールは、ソフトウェアコードの実行中に、少なくとも２つのとり得る状態、すなわち、暗号化された状態及び暗号化されていない状態、にあることができる。当該装置はプロセッサを有し、プロセッサは、バイナリを受け取り、バイナリのチェックサムを、そのとり得る状態の夫々において生成し、少なくとも１つのチェックサム照合関数及び前記生成されたチェックサムをバイナリに挿入することによって、インテグリティが保護されたバイナリを生成するよう構成される。各チェックサム関数は、バイナリの状態を該バイナリの状態についてのチェックサムと比較することによって、インテグリティが保護されたバイナリの各状態のインテグリティを確かめるよう構成される。
【００２０】
第１の望ましい実施形態で、プロセッサは、更に、入れ子式に複数のチェックサム照合関数を挿入し、それにより、実行中に、各チェックサム照合関数のインテグリティが少なくとも１つの他のチェックサム照合関数によって確かめられるようにする、よう構成される。
【００２１】
第４の態様で、本発明は、自己書換型のインテグリティが保護されたバイナリを記憶しており、プロセッサによって実行される場合に、本発明の第１の態様に係る方法のステップを実行するコンピュータプログラムを対象とする。
【００２２】
第５の態様で、本発明は、プロセッサによって実行される場合に、複数のモジュールを有するバイナリの各状態についてチェックサムを生成し、各モジュールは前記バイナリの実行中に少なくとも２つのとり得る状態にあることができ、該とり得る状態は暗号化された状態及び暗号化されていない状態であり、少なくとも１つのチェックサム照合関数及び前記生成されたチェックサムを前記バイナリに挿入することによって、インテグリティが保護されたバイナリを生成し、各チェックサム関数は、前記バイナリの状態を該バイナリの状態についてのチェックサムと比較することによって、前記インテグリティが保護されたバイナリの各状態のインテグリティを確かめるよう構成される、命令を記憶しているコンピュータプログラムを対象とする。
【発明の効果】
【００２３】
本発明の実施形態によれば、ソフトウェア、特に、自己書換型であるコード、のインテグリティを保証するための改善された解決法が提供される。
【図面の簡単な説明】
【００２４】
【図１】本発明が実施されるコンピュータデバイスの例を表す。
【図２】本発明の望ましい実施形態を表す状態図である。
【図３】本発明の望ましい実施形態に従うインテグリティ照合のための方法を表す。
【図４】バイナリの保護を表す。
【図５】例となる状態遷移図を表す。
【発明を実施するための形態】

【００２５】
本発明の望ましい特徴について、添付の図面を参照して、限定されない例を用いて記載する。
【００２６】
図１は、本発明が実施されるコンピュータデバイス（コンピュータ）１００の例を表す。コンピュータ１００は、例えば、標準のパーソナルコンピュータ（ＰＣ）といった、計算を実行することができる如何なる種類のコンピュータ又はデバイスであってもよい。コンピュータ１００は、少なくとも１つのプロセッサ１１０と、ＲＡＭメモリ１２０と、ユーザと対話するためのユーザインターフェース（ＵＩ）１３０と、デジタルデータ担体１５０からソフトウェアプログラムを読み出すための第２のインターフェース（Ｉ／Ｏ）１４０とを有する。当業者には明らかなように、図示されているコンピュータは、明りょうさのために簡単化されており、実際のコンピュータは、更に、ネットワーク接続や持続記憶デバイス等の機能を有する。
【００２７】
本発明の主たる発明概念は、各状態が保護されているプログラムコードの状態に対応するところの、プログラムの実行中の有限状態機械（finite state machine）の使用である。１つの状態から他の状態に移るとき、コードの少なくとも１つのモジュールは動的に変更される。有限状態機械はそれ自体当該技術でよく知られているので、それについて詳細には記載しない。モジュールの限定されない例は、コードのサブセクション、コードのチャンク（chunks）、及び関数である。改ざんから保護するために、各状態についてチェックサムを計算することが可能であり、これにより、コードを改ざんすることを防ぐことができる。
【００２８】
図２は、本発明の望ましい実施形態を表す状態図である。説明の明りょうさのために、例となる状態図は３つの状態（Ｓ１、Ｓ２及びＳ３）しか含まず、各状態はチェックサムに関連付けられており、コードは３つのモジュール（Ｍ１、Ｍ２及びＭ３）しか有さない。当業者には明らかなように、本発明は、より多くの（又はより少ない）状態及びモジュールにも等しく適用され、それに対応する異なった状態遷移が可能である。
【００２９】
例において、初期状態はＳ１である。この状態において、コードの全てのモジュールは、それらの初期状態にあり、状態は第１チェックサム、すなわち、チェックサムＶ１、に関連付けられている。この第１チェックサムにより、少なくとも１つのモジュールのインテグリティ照合が可能となる。
【００３０】
実行により状態Ｓ１から状態Ｓ２へのシフトが起こる場合、第１のモジュールＭ１は変更され、変更されたモジュールＭ’１が生成される。望ましくは、この時点で（チェックサムＶ２を用いて）チェックサムが確認されるが、その確認は、夫々の状態遷移に適用することができる、後の段階で行われてもよい。
【００３１】
同様に、実行により状態Ｓ２から状態Ｓ３へのシフトが起こる場合、第２のモジュールＭ２が変更され、変更されたモジュールＭ’２が生成される。変更されたモジュールＭ’１は、その変更されていない形態Ｍ１に戻り、それにより、コードは３つのモジュールＭ１、Ｍ’２及びＭ３を有する。前に説明されたように、望ましくは、この時点で（チェックサムＶ３を用いて）チェックサムが確認される。
【００３２】
状態Ｓ３から状態Ｓ２へ、更に状態Ｓ２から状態Ｓ１へ移ると、変更は逆方向に行われる。例えば、状態Ｓ２から状態Ｓ１に進む場合、変更されたモジュールＭ’１は、その変更されていない形態Ｍ１に戻る。また、かかる場合に、望ましくは、遷移後に（関連するチェックサムを用いて）チェックサムが確認される。
【００３３】

図３は、本発明の望ましい実施形態に従うインテグリティ照合のための方法を表す。方法は、通常のプログラム実行から始まることができる（ステップ３１０）。或る時点で、状態を変化させることが決定される（ステップ３２０）。少なくとも１つのモジュールが変更され（ステップ３３０）、インテグリティがチェックされる（ステップ３４０）。この後、通常のプログラム実行がステップ３１０で戻る。

【００３４】

モジュールは、例えば、モジュールの解読によって（その場合には、復号化キーが必要である。）、又はモジュールの一部のバイトを変更することによって（例えば、置換によって）、変更されてよい。前者の場合、コードは、その最初の状態に戻るよう暗号化される（対称的な暗号アルゴリズムでは、復号化キーと同じ暗号キーを必要とする。）。後者の場合、バイトは、例えば、バックワード置換（backwards permutation）によって、最初の構成に戻される。

【００３５】

図４は、バイナリ４１０の保護を表す。保護エンジン４２０は、とり得る状態及びそれらの夫々のチェックサム値を列挙するよう機能する。これらのチェックサム値は、識別を可能にする何らかのリファレンスとともに、保護されたバイナリ４３０に挿入される。保護エンジン４２０は、例えばパーソナルＰＣ等の如何なる種類の適切なコンピュータデバイスであってもよい。それは、望ましくは、プロセッサ及びメモリ等（図示せず。）を有する。コンピュータプログラムプロダクト４１５は、例えばＣＤ－ＲＯＭ又はＵＳＢメモリ等であって、プロセッサによって実行される場合に本願で記載されるようにバイナリを保護する命令を記憶する。

【００３６】

バイナリ４１０を保護するために、保護エンジン４２０は、バイナリ４１０を解析して、とり得る状態を計算する。この目的のために、有利に、保護エンジン４２０は、コールグラフ解析及び状態解析を使用する。状態が生成されると、保護エンジン４２０は、その状態に係るチェックサムを計算する。

【００３７】

次いで、保護エンジン４２０は、少なくとも１つのチェックサム呼び出し（invocation）ポイント４３２と、チェックサムが確認される各状態ごとに１つである、チェックサムの（単一値を有することができる）テーブル４３４とを有する保護されたバイナリ４３０を生成する。望ましくは、チェックサムは、容易なアクセスを可能にするある種の識別子（例えば、状態１、状態２・・・といった表示、又は“暗号化されたモジュール１、復号化されたモジュール２、暗号化されたモジュール３”のラインに沿った表示）と関連付けられる。望ましくは、チェックサムテーブル４３４は、保護されたバイナリの少なくとも一部についてハッシュ値の形をとるが、他のチェックサムも想定されてよい（例えば、バイナリの一定範囲における特定の文字の数）。チェックサムは、例えば暗号化によって、保護されるが、先に記載されたように、チェックサム照合関数を入れ子にすることが好ましく、それにより、各関数のインテグリティは少なくとも１つの他の関数によって確認される。

【００３８】

状態のチェックサムが決定論的に前の状態のチェックサムから決定される場合に、チェックサムテーブル４３４を用いずに行うことも可能である点に留意すべきである。チェックサムは、また、保護されたバイナリにおいて難読化され、又は別なファイルに記憶されてもよい。

【００３９】

次いで、以下の限定されない例を考える。記載及びその理解を容易にするよう、例は、単一コード領域（「チェックサム範囲」と呼ばれる。）が、望ましくはプログラム実行中に複数の照合ポイントで起こるチェックサム照合によって保護される場合に限定される。当業者には明らかなように、本発明は、複数のチェックサム範囲の場合に容易に拡張可能である。例は、更に、モジュールが、２つの異なった状態、すなわち、暗号化された状態

及び復号化された状態、にあることができる関数に対応する特定の場合に限定される。

【００４０】

このようにして、コードには特有の状態遷移が存在し、各状態は、望ましくは、保護されたバイナリにおける少なくとも１つの照合ポイントで確認される。

【００４１】

例において、各関数が２つのとり得る状態（暗号化された状態及び復号化された状態）のうちの１つにある場合、状態変数Ｓ（望ましくは、ビットストリーム）において状態ビットＳｉとして状態を表すことが可能である。状態変数Ｓは、アプリケーションの関数コールグラフに依存して、とり得ない値を表すことができる。そのようなものとして、状態変数Ｓのとり得る値のサブセットのみがアプリケーションについて有効であってよい。状態変数Ｓの各値は、保護エンジンによって計算されるチェックサム値に対応し、これらのチェックサム値は、保護されたバイナリに（有利にハッシュ値として）埋め込まれる。次いで、ハッシュ・チェックサムは、識別子として状態変数Ｓを用いて調べられてよい。

【００４２】

関数が暗号化又は復号化されるときはいつでも、すなわち、状態が変化する場合には、状態変数Ｓは、適切な状態ビットＳｉを反転（flip）させることによって、更新される。

【００４３】

図５は、状態ビットＳ１、Ｓ２に夫々関連付けられた２つの関数Ｆ１、Ｆ２を伴う、例となる状態遷移図を表す。各関数は、暗号化され（Ｓ１、Ｓ２によって表され）且つ復号化されてよく、４つの異なる状態を生ずる。

・Ｓ０：Ｆ１及びＦ２は暗号化－（Ｓ１Ｓ２）
・Ｓ１：Ｆ１は暗号化、Ｆ２は復号化－（Ｓ１Ｓ２）
・Ｓ２：Ｆ１は復号化、Ｆ２は暗号化－（Ｓ１Ｓ２）
・Ｓ３：Ｆ１及びＦ２は復号化－（Ｓ１Ｓ２）

最初に、プログラムは、メイン関数５１０を実行する。このとき、状態ＳはＳ０＝Ｓ１Ｓ２である。メイン関数は関数Ｆ１及びＦ２を呼び出すことができる。関数Ｆ１が呼び出される場合、暗号化されているＦ１は復号化される（５２０）。その後に、Ｆ１の実行５３０が始まる。このとき、状態ＳはＳ２＝Ｓ１Ｓ２である。同様に、関数Ｆ２が呼び出される場合、暗号化されているＦ２は復号化され（５５０）、Ｓ２は反転する。その後に、Ｆ２の実行（５６０）が始まる。このとき、状態ＳはＳ１＝Ｓ１Ｓ２である。

【００４４】

関数Ｆ１（５３０）には２つの可能性が存在する。Ｆ１の実行が終わり、Ｆ１はＦ２を呼び出す。最初の場合では、Ｆ１は暗号化され（５４０）、Ｓ１は反転され、実行はメイン関数５１０に戻る。そして、状態はＳ０＝Ｓ１Ｓ２に戻る。

【００４５】

第２の場合では、Ｆ２は復号化され（５５０）、状態ビットＳ２は反転され、Ｆ２の実行５６０が始まる。しかし、メイン関数５１０が直接Ｆ２を呼び出す場合と対照的に、Ｆ１も復号化されている。このことは、状態がＳ３＝Ｓ１Ｓ２であることを意味する。従って、Ｆ２の実行中、状態はＳ１又はＳ３のいずれかである。

【００４６】

このようにして、Ｆ２の実行が終わると、Ｆ２は暗号化され（５７０）、状態ビットＳ２は反転され、実行は関数の呼び出しに戻る。呼び出す関数がメイン関数である場合、メイン関数５１０は実行を引き継ぐ。このとき、状態はＳ０＝Ｓ１Ｓ２である。他方で、呼び出す関数がＦ１であった場合、Ｆ１の実行５３０が再開する。このとき、状態はＳ２＝Ｓ１Ｓ２である。

【００４７】

図５で、チェックサム呼び出しポイントは、理解を簡単にするために、省略されているが、当然、夫々の関数（メイン関数、Ｆ１、Ｆ２）は、望ましくは、状態に関連付けられ

たチェックサムを確認する（場合により、暗号化／復号化関数の部分としての）少なくとも１つのチェックサム呼び出しポイントを有する。

【００４８】

チェックサム呼び出しポイントの最も簡単な実施は、プログラムが現在の状態Ｓを読み出し、対応するチェックサム値をチェックサムテーブルにおいて見つけ出し、記憶されているチェックサム値に対して照合されるべきチェックサムを生成することである。照合が失敗である場合、望ましくは、実行は中止される。

【００４９】

当業者に明らかなように、攻撃者はチェックサム値を改ざんしようと試みる。このような理由で、他のチェックサムを含む呼び出し関数と同様にチェックサムテーブルを保護することが好ましく、これにより、これらは照合される。

【００５０】

可能であればいつでも、呼び出しポイントコードにおいてチェックサム値をインラインにすることによって、セキュリティのレベルを高めることが可能である。この場合、保護エンジンは、有利に、下記の３つの場合の間を区別するようコールグラフを解析する。

１．保護される関数が入力される場合、状態は一意である。すなわち、単一の実行経路が関数に通ずる。この場合、チェックサム値は一意であり、呼び出しポイントコードは、（望ましくは、難読な方法で）チェックサム値を埋め込むコードを有してよい。例えば、

```
if(checksum_value equals checksum_ref) then (do_something)
```

２．保護される関数が入力される場合、少数（５未満）のとり得る状態が存在する。この場合も、呼び出しポイントコードにチェックサム値を埋め込むことが可能である。例えば、

```
if(S equals S1 and checksum_value equals checksum_S1) or
if(S equals S2 and checksum_value equals checksum_S2)
then (do_something)
```

３．とり得る状態が多数存在する場合には、チェックサムテーブルが望ましい解決法である。例えば、

```
if(checksum_value equals LookupInChecksumTable(S)) then (do_something)
```

明細書において、モジュールは、２つのとり得る状態（例えば、暗号化及び復号化）を有するとして記載されてきた。当業者に明らかなように、モジュールは複数の（特に、２よりも多い）状態を有することが可能である。

【００５１】

本発明は、上記の攻撃に打ち勝つことができるマルチステート・ソフトウェアコードのインテグリティを確かめる方法を提供することができると認められる。

【００５２】

明細書並びに（必要に応じて）特許請求の範囲及び図面で開示されている夫々の特徴は、独立して、又は何らかの適切な組合せにおいて、提供されてよい。ハードウェアで実施されると記載される特徴は、ソフトウェアでも実施されてよく、また、その逆も同様である。特許請求の範囲に現れる参照符号は、単なる例示として用いられ、特許請求の範囲の適用範囲を限定するものではない。

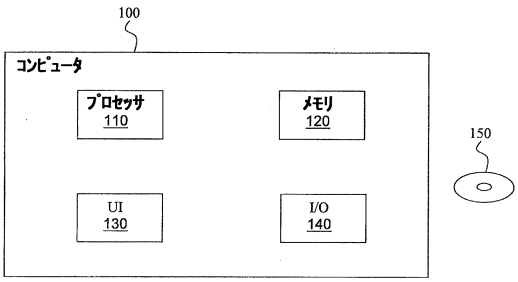
【符号の説明】

【００５３】

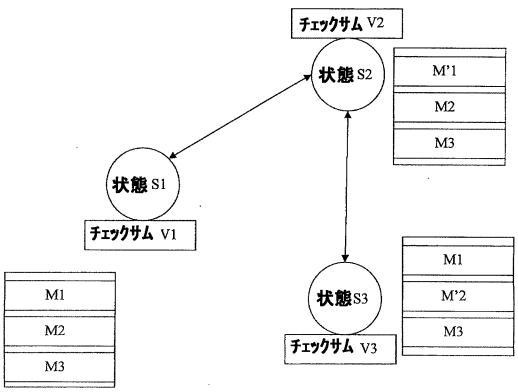
１００　コンピュータ
１１０　プロセッサ
１２０　ＲＡＭメモリ
１３０　ユーザインターフェース（ＵＩ）

１４０　第２のインターフェース（Ｉ／Ｏ）
１５０　デジタルデータ担体１５０
４１０　バイナリ
４１５　コンピュータプログラムプロダクト
４２０　保護エンジン４２０
４３０　保護されたバイナリ４３０
４３２　チェックサム呼び出しポイント
４３４　チェックサムテーブル
Ｆ１、Ｆ２　関数
Ｍ１～Ｍ３，Ｍ’１～Ｍ’３　モジュール
Ｓ０～Ｓ３　状態
Ｖ１～Ｖ３　チェックサム

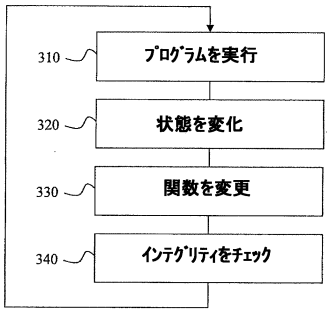
【図１】



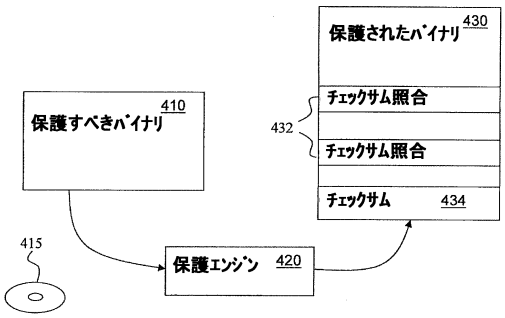
【図２】



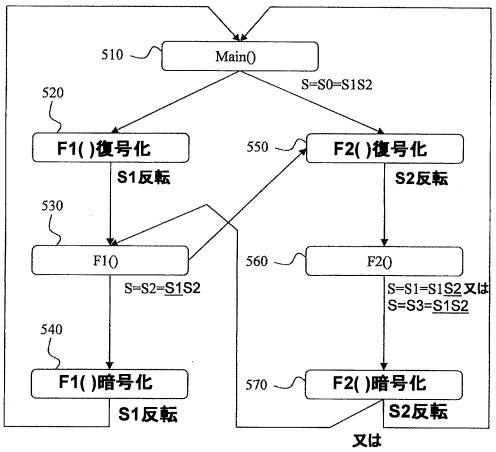
【図３】



【図４】



【図５】


510
Main()
S=S0=S1S2
520
F1( )復号化
S1反転
550
F2( )復号化
S2反転
530
F1()
S=S2=S1S2
560
F2()
S=S1=S1S2又は
S=S3=S1S2
540
F1( )暗号化
S1反転
570
F2( )暗号化
S2反転
又は

フロントページの続き

(72)発明者 シャルル サルモン－ルガニュール
フランス国 ３５５１０ セソン・セヴィニェ アヴェニュ・ド・ベル・フォンテーヌ １ テクニカラー・アールアンドディー・フランス テクニカラー・アールアンドディー・フランス（レンヌ）内
(72)発明者 アントワーヌ モンシフロ
フランス国 ３５５１０ セソン・セヴィニェ アヴェニュ・ド・ベル・フォンテーヌ １ テクニカラー・アールアンドディー・フランス テクニカラー・アールアンドディー・フランス（レンヌ）内

審査官 戸島 弘詩

(56)参考文献 韓国登録特許第１０－０７７２８８１（ＫＲ，Ｂ１）
国際公開第２００９／１５０４７５（WO，Ａ１）
国際公開第２００９／１２５２２０（WO，Ａ１）
特表２００８－５３２１１３（ＪＰ，Ａ）
特表２０１１－５２５６５０（ＪＰ，Ａ）
特表２０１１－５１６９５３（ＪＰ，Ａ）
特開２００１－１７５４６６（ＪＰ，Ａ）
特開２００２－２９７５４２（ＪＰ，Ａ）
米国特許出願公開第２００５／０２１０２８７（ＵＳ，Ａ１）
特表２００２－５０７３０７（ＪＰ，Ａ）
特表２０１２－５２６３１０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ２１／００－２１／８８
Ｇ０９Ｃ１／００－５／００
Ｈ０４Ｋ１／００－３／００
Ｈ０４Ｌ９／００－９／３８