(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2007-511810
(P2007-511810A)
(43)公表日　平成19年5月10日(2007.5.10)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 21/22	(2006.01)	G06F	9/06	660F	5B276

審査請求 未請求　予備審査請求 未請求　（全 23 頁）

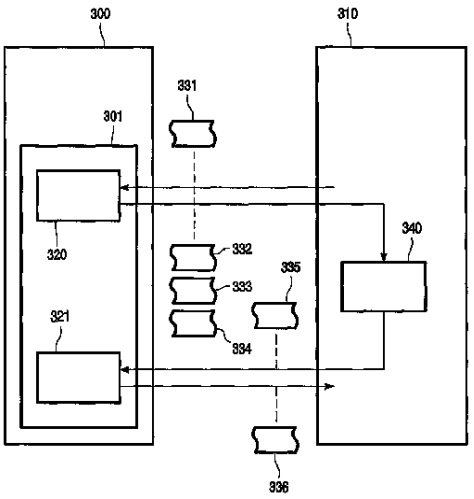
(21)出願番号	特願2006-530718 (P2006-530718)
(86)(22)出願日	平成16年5月6日 (2004.5.6)
(85)翻訳文提出日	平成17年11月14日 (2005.11.14)
(86)国際出願番号	PCT/IB2004/002342
(87)国際公開番号	WO2004/102302
(87)国際公開日	平成16年11月25日 (2004.11.25)
(31)優先権主張番号	60/471,452
(32)優先日	平成15年5月16日 (2003.5.16)
(33)優先権主張国	米国 (US)

(71)出願人　590000248
コーニンクレッカ　フィリップス　エレクトロニクス　エヌ　ヴィ
オランダ国　５６２１　ベーアー　アインドーフェン　フルーネヴァウツウェッハ　１
(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】乱数関数を利用した実行証明

(57)【要約】
物理的乱数関数は、評価が容易であるが特徴付けするのが困難な関数である。制御された物理的乱数関数は、分離不可な手法によりＰＵＦに物理的に結合されたセキュリティアルゴリズムにより制御されるセキュリティプログラムを介する場合のみアクセス可能なＰＵＦである。ＣＰＵＦは、特定の計算が特定のプロセッサ上で実行されたことを証明する証明書が生成される証明された実行を可能にする。本発明は、任意の第三者が検証可能な実行証明を生成する付加的レイヤを提供する。この実行証明はまた、セキュアなメモリ及び中断可能なプログラム実行を提供するのに有用である。



【特許請求の範囲】
【請求項１】
プログラム命令の実行の真正性を証明する方法であって、
乱数関数を有するセキュリティ装置上のセキュリティプログラムの制御の下、プログラム命令を実行するステップと、
前記乱数関数を利用して、制御されたインタフェースを介し前記乱数関数にアクセスすることにより、第１モードで動作する前記セキュリティプログラムの実行中に証明結果を計算するステップと、
前記乱数関数を利用して、前記制御されたインタフェースを介し前記乱数関数にアクセスすることにより、第２モードで動作する前記同一のセキュリティプログラムの実行中に前記証明結果を検証するステップと、
を有し、
前記乱数関数は、前記制御されたインタフェースを介し前記セキュリティプログラムからのみアクセス可能であり、前記制御されたインタフェースは少なくとも１つのプリミティブ関数を有し、前記プリミティブ関数は、該プリミティブ関数を呼び出す前記セキュリティプログラムの少なくとも一部の表現の少なくとも一部に依存する出力を返す前記乱数関数にアクセスすることを特徴とする方法。
【請求項２】
請求項１記載の方法であって、
第三者が前記証明結果を検証可能な実行証明として受け取ることを特徴とする方法。
【請求項３】
請求項１記載の方法であって、
前記セキュリティプログラムは、前記第１モードで動作する場合、
アプリケーションプログラム出力を生成するアプリケーションプログラム入力によりアプリケーションプログラムを実行するステップと、
前記制御されたインタフェースを介し前記乱数関数を用いて暗号化により結果を取得し、前記アプリケーションプログラム入力の少なくとも一部、前記アプリケーションプログラム出力の少なくとも一部、及び前記アプリケーションプログラムの少なくとも一部の少なくとも１つに対しメッセージ認証コードを生成するステップと、
前記暗号化及びメッセージ認証された結果を有する証明結果を生成するステップと、
を実行し、
前記セキュリティプログラムは、前記第２モードで動作する場合、
検証対象となる証明結果を受け取るステップと、
前記証明結果の暗号化及びメッセージ認証された結果のメッセージ真正性を少なくとも部分的に検証するステップと、
を実行することを特徴とする方法。
【請求項４】
請求項３記載の方法であって、
前記セキュリティプログラムはさらに、前記第２モードで動作する場合、前記証明結果の文字列による暗号化及びメッセージ認証された結果の少なくとも部分的に検証されたメッセージ真正性の少なくとも一部を前記セキュリティ装置のユーザに送信するステップを実行することを特徴とする方法。
【請求項５】
請求項１記載の方法であって、
前記証明結果は、以降の利用を想定した格納データを有し、該格納データが前記セキュリティ装置上に実行されているセキュリティプログラムからのものであることを検証することを可能にすることを特徴とする方法。
【請求項６】
請求項５記載の方法であって、
前記証明結果に含まれる格納データは、前記セキュリティプログラムの表現に依存する

出力を有するプリミティブ関数を利用して計算される鍵により暗号化されることを特徴とする方法。
【請求項７】
請求項１記載の方法であって、
前記証明結果は、前記セキュリティプログラムの以降の継続を想定する状態情報を有することを特徴とする方法。
【請求項８】
請求項７記載の方法であって、
前記証明結果に含まれる状態情報は、前記セキュリティプログラムの表現に依存する出力を有するプリミティブ関数を利用して計算される鍵により暗号化されることを特徴とする方法。
【請求項９】
請求項１記載の方法であって、
前記動作モードは、前記セキュリティプログラムに入力を供給することにより、前記セキュリティ装置のユーザにより選択されることを特徴とする方法。
【請求項１０】
請求項１記載の方法であって、
前記セキュリティプログラムは、該セキュリティプログラムが前記セキュリティ装置により実行されることを前記セキュリティ装置のユーザに証明する認証された実行を提供する第２セキュリティプログラムの一部として実行されることを特徴とする方法。
【請求項１１】
請求項１記載の方法であって、
前記乱数関数は、複雑な物理的システムを有することを特徴とする方法。
【請求項１２】
請求項１記載の方法であって、
前記乱数関数は、プリミティブ関数
ＧｅｔＳｅｃｒｅｔ（Ｃｈａｌｌｅｎｇｅ）＝ｈ（ｈ（Ｐｒｏｇｒａｍ），ｆ（Ｃｈａｌｌｅｎｇｅ））
を介し（ただし、ｆ（．）は前記乱数関数であり、ｈ（．）は実質的に公衆に利用可能な乱数ハッシュ関数である）アクセス可能であることを特徴とする方法。
【請求項１３】
請求項１記載の方法であって、
前記鍵の計算は、前記セキュリティプログラムの入力の一部を前記乱数関数への入力として利用することを特徴とする方法。
【請求項１４】
乱数関数と、
プロセッサ及びメモリを有する処理装置と、
を有するシステムであって、
当該システムに請求項１記載の方法を実現させるよう構成されるコンピュータ可読命令を実行することを特徴とするシステム。
【請求項１５】
請求項１記載の方法をコンピュータに実現させるコンピュータ可読命令を有することを特徴とするコンピュータプログラム。
【請求項１６】
請求項１記載の方法をコンピュータに実現させることを特徴とするコンピュータ実行可能命令。
【請求項１７】
請求項１記載の方法により生成される証明結果を伝搬することを特徴とする信号。
【発明の詳細な説明】
【技術分野】

【０００１】
本発明は、プログラム実行の真正性を証明するための方法、当該方法を実現するよう構成されたシステム、当該方法を実現するコンピュータプログラムプロダクツ、当該方法を実現するコンピュータ実行可能な命令、及び当該方法により生成される証明結果を伝搬する信号に関する。
【背景技術】
【０００２】
電子取引などの用途では、ユーザまたは第三者により特定のプロセッサ上で実際に計算（またはプログラム）が実行されたことを検証することが必要とされるかもしれない。Ｂｌａｉｓｅ Ｇａｓｓｅｎｄ、Ｄｗａｉｎｅ Ｃｌａｒｋｅ、Ｍａｒｔｅｎ ｖａｎ Ｄｉｊｋ及びＳｒｉｎｉｖａｓ Ｄｅｖａｄａｓによる「Ｃｏｎｔｒｏｌｌｅｄ Ｐｈｙｓｉｃａｌ Ｒａｎｄｏｍ Ｆｕｎｃｔｉｏｎｓ」（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ １８ｔｈ Ａｎｎｕａｌ Ｃｏｍｐｕｔｅｒ Ｓｅｃｕｒｉｔｙ Ａｐｐｌｉｃａｔｉｏｎｓ Ｃｏｎｆｅｒｅｎｃｅ，Ｄｅｃｅｍｂｅｒ，２００２）（以下では「先行技術文献」と呼ぶ）では、認証された実行とは、計算出力と共に、特定プロセッサチップのユーザに特定の計算が当該特定プロセッサチップ上で実行され、当該計算が実行され、所与の計算結果を生成したことを証明する証明書（電子証明書と呼ばれる）を生成するプロセスとして定義される。
【０００３】
電子証明書の生成及び検証のため先行技術文献で用いられるフレームワークは、物理的乱数関数（Ｐｈｙｓｉｃａｌ Ｒａｎｄｏｍ Ｆｕｎｃｔｉｏｎ）の概念に基づき構成されている。物理的乱数関数（ＰＵＦ）は、複雑な物理的システムを用いて評価される乱数関数である。略語ＰＵＦ（ＰＲＦの代わりに）を用いることは、発音が容易であるという利点と共に、擬似乱数関数（Ｐｓｅｕｄｏ－Ｒａｎｄｏｍ Ｆｕｎｃｔｉｏｎ）との混同を避けるためである。ＰＵＦは様々な手法により実現可能である。ＰＵＦの実現形態の一部は、各生成サンプル（例えば、各半導体チップ）が異なる機能を実現するような方法により生成することが容易である。これは、ＰＵＦが認証された識別アプリケーションにより利用されることを可能にする。
【０００４】
ＰＵＦは、チャレンジとレスポンスをマップし、物理的装置により実現され、以下の２つの性質を有する関数である。すなわち、（１）ＰＵＦは評価が容易であり、物理的装置は短時間に当該関数を容易に評価することができる。（２）ＰＵＦは、多項数のもっともらしい物理的測定値（特に、選ばれたチャレンジ－レスポンスペアの決定）から、もはやセキュリティ装置（へのアクセス）を有しない、多項量のリソースのみが利用可能な攻撃者は、ランダムに選ばれたチャレンジに対するレスポンスに関する無視できる量の情報しか抽出することができないということを特徴付けることは困難である。上記定義では、「短い」及び「多項（ｐｏｌｙｎｏｍｉａｌ）」という表現は、装置のサイズに対するものであり、セキュリティパラメータである。特に「短い」とは、リニアまたは低次元の多項を意味する。「もっともらしい」という表現は、測定技術における現在の技術状態に対するものであり、方法が改良されるに従い変わる可能性がある。
【０００５】
ＰＵＦの例として、シリコンＰＵＦ（Ｂｌａｉｓｅ Ｇａｓｓｅｎｄ、Ｄｗａｉｎｅ Ｃｌａｒｋｅ、Ｍａｒｔｅｎ ｖａｎ Ｄｉｊｋ及びＳｒｉｎｉｖａｓ Ｄｅｖａｄａｓによる「Ｓｉｌｉｃｏｎ Ｐｈｙｓｉｃａｌ Ｒａｎｄｏｍ Ｆｕｎｃｔｉｏｎｓ」（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ９ｔｈ ＡＣＭ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｃｏｍｐｕｔｅｒ ａｎｄ Ｃｏｍｍｕｎｉｃａｔｉｏｎｓ Ｓｅｃｕｒｉｔｙ，Ｎｏｖｅｍｂｅｒ，２００２））、オプティカルＰＵＦ（Ｐ．Ｓ．Ｒａｖｉｋａｎｔｈ，Ｍａｓｓａｃｈｕｓｅｔｔｓ Ｉｎｓｔｉｔｕｔｅ ｏｆ Ｔｅｃｈｎｏｌｏｇｙ，Ｐｈｙｓｉｃａｌ Ｏｎｅ－Ｗａｙ Ｆｕｎｃｔｉｏｎｓ，２００１）、及びデジタルＰＵＦがあげられる。シリコンＰＵＦは、製造過程に起因するチップ間変動を利用する。オプティカル

ＰＵＦは、コヒーレント光（レーザ）ビームにより照射された光構造により生成されるスペックルパターンの予測不可能性を利用する。デジタルＰＵＦは、改ざん耐性環境が暗号化及び認証のために利用する秘密鍵を保護する従来シナリオを表す。

【０００６】

ＰＵＦは、セキュリティ装置内で分離不可な方法によりＰＵＦに物理的リンクするセキュリティアルゴリズムを介してのみアクセス可能な場合に、「制御された」ものであると定義される（制御ＰＵＦまたはＣＰＵＦ）（すなわち、当該アルゴリズムを回避しようとする試みは、ＰＵＦの破壊を招くであろう）。特に、当該セキュリティアルゴリズムは、ＰＵＦに与えられたチャレンジを制限することが可能であり、外部に与えられるレスポンスに関する情報を制限することができる。制御は、ＰＵＦが単なる認証された識別用途以上のものとなるのを可能にする基本的アイデアである。

【０００７】

先行技術文献には、ＣＰＵＦの例が説明されている。セキュリティプログラムは、ＰＵＦがセキュリティプログラムからの２つのプリミティブな関数ＧｅｔＳｅｃｒｅｔ（．）とＧｅｔＲｅｓｐｏｎｓｅ（．）を介する場合に限りアクセス可能となるように、ＰＵＦにリンクされたセキュリティアルゴリズムの制御下で使用される。ＧｅｔＳｅｃｒｅｔ（．）は、ＰＵＦへの入力がプリミティブ関数が実行されるセキュリティプログラムの表現に依存することを保証する。ＧｅｔＲｅｓｐｏｎｓｅ（．）は、ＰＵＦの出力がプリミティブ関数が実行されるセキュリティプログラムの表現に依存することを保証する。この依存性のため、ＰＵＦの入出力は、上記プリミティブ関数が異なるセキュリティプログラム内で実行される場合、異なるものとなるであろう。さらに、上記プリミティブ関数は、先行技術文献に記載されるように、新たなチャレンジ－レスポンスペアの生成が規制可能であると共に、セキュアなものとすることが可能であることを保証する。

【０００８】

認証された実行は（先行技術文献で記載されるような）、ユーザにのみ知られている秘密のＰＵＦチャレンジ－レスポンスペアに基づきユーザが出力を計算可能なチャレンジに対しＧｅｔＳｅｃｒｅｔ（．）プリミティブを利用する。このようにして、ユーザがＰＵＦアルゴリズムと共に特定のプロセッサチップ上のアルゴリズムを実行したことを証明するため、当該出力はユーザに対し利用可能である。

【０００９】

しかしながら、ユーザは当該出力を用いて、プログラムが特定のプロセッサ上でアクティブに実行されたことを第三者に証明することはできない。なぜならば、ユーザはチャレンジ－レスポンスペアを用いて自ら当該結果を生成することができるためである。しかしながら、例えば電子取引システムでは、プログラム（番組を視聴するため料金を支払うプログラムなど）が特定のプロセッサ上で実行されたことを第三者に実際に証明することができることがしばしば望ましい。

【発明の開示】

【発明が解決しようとする課題】

【００１０】

従って、本発明の課題は、任意の第三者により検証可能な証明書としての電子証明と呼ばれる特定プロセッサ上の特定の計算に対する実行証明として利用可能な証明結果の生成を可能にする方法を提供することである。

【課題を解決するための手段】

【００１１】

上記課題は、プログラム命令の実行の真正性を証明する方法であって、乱数関数を有するセキュリティ装置上のセキュリティプログラムの制御の下、プログラム命令を実行するステップと、前記乱数関数を利用して、制御されたインタフェースを介し前記乱数関数にアクセスすることにより、第１モードで動作する前記セキュリティプログラムの実行中に証明結果を計算するステップと、前記乱数関数を利用して、前記制御されたインタフェースを介し前記乱数関数にアクセスすることにより、第２モードで動作する前記同一のセキ

ュリティプログラムの実行中に前記証明結果を検証するステップとを有し、前記乱数関数は、前記制御されたインタフェースを介し前記セキュリティプログラムからのみアクセス可能であり、前記制御されたインタフェースは少なくとも１つのプリミティブ関数を有し、前記プリミティブ関数は、該プリミティブ関数を呼び出す前記セキュリティプログラムの少なくとも一部の表現の少なくとも一部に依存する出力を返す前記乱数関数にアクセスすることを特徴とする方法により実現される。

【００１２】

セキュリティプログラムは、同一または異なる実行による異なる動作モードにより実行可能である。同一のプログラムに少なくとも２つの動作モードを有することにより、セキュリティプログラムは、異なるプログラム実行において乱数関数を効果的に利用することが可能である。乱数関数にアクセスするプリミティブ関数は異なるモードで動作する同一のセキュリティプログラムである少なくともセキュリティプログラムの一部の表現に依存するため、乱数関数へのアクセスは、当該異なるモードによるセキュリティプログラムに対し保証され、他の何れのセキュリティプログラムも乱数関数により提供されるセキュリティに妥協して乱数関数にアクセスすることはできない。従って、「マルチモード」プログラムは、その他のモードの機能がセキュリティプログラムの最初の実行中にすでに明らかに規定及び制限されるとき、効果的なコンセプトである。

【００１３】

当該出力をセキュリティプログラムの表現に依存させることにより、セキュリティ装置上で実行される他の任意のセキュリティプログラムが制御されたインタフェースを介し同一の入力に対し異なる結果を取得することが（ほとんど）保証される。例えば、不正な証明結果を生成するため情報を取得するようハッカーにより設計された他の任意のセキュリティプログラムは、当該制御されたインタフェースを介し役に立たない結果のみを取得することになる（表現方法に応じて高い確率により）。これは、当該結果がハッカーにより利用されたセキュリティプログラムともとのセキュリティプログラムに対し異なるセキュリティプログラム表現に依存しているためである。

【００１４】

セキュリティプログラムの表現は、ハッシュまたは他の署名、若しくはその一部とすることができる。通常、セキュリティプログラムの表現は、セキュリティプログラム全体をカバーするが、特殊ケースでは（例えば、セキュリティプログラムが乱数関数に関係しない大きな部分を含む場合）、プリミティブ関数の入出力の呼び出し及び処理を取り扱うセキュリティプログラムの部分に表現を制限することが効果的であるかもしれない。

【００１５】

セキュリティプログラムの実行中、当該セキュリティプログラムの表現に依存する出力を有するプリミティブ関数を利用して、鍵が導出可能である。この鍵は、証明結果（の一部）を暗号化するのに利用可能である。この鍵により暗号化された結果は何れも、同一のセキュリティプログラムの以降の実行を除き、同一または異なるモードでは役に立たない。

【００１６】

セキュリティプログラムは、典型的には、セキュリティ装置のユーザにより提供される。これはまた、異なるサブシステムまたは他のシステムとすることが可能である。

【００１７】

以降の利用のため特定のセキュリティプログラムの迅速な抽出を可能にするため、任意的には以降の実行が可能なユーザのパーミッションに関する情報と共に、同一または異なるモードによるセキュリティプログラムの以降の実行のため、プログラムコードまたはそのハッシュコードが格納可能である。

【００１８】

当該方法を利用して、ＣＰＵＦは任意の第三者により検証可能な証明書である電子証明書（ｅ－ｐｒｏｏｆ）と呼ばれる実行証明を証明結果として生成するのに利用可能である。例えば、ＳＴＢアプリケーションでは、放送局はＳＴＢと通信し、コンテンツを当該所

有者（またはそれをレンタルした者）に販売する。放送局は、所有者のＳＴＢを利用し、所有者と放送局との間の取引を含むプログラムの認証された実行を所望する。放送局は、所有者がコンテンツを購入したことを任意の仲介者が確認できることを所望する。これには、実行の証明が必要とされる。他のアプリケーションとしては、電子商取引、電子バンキング及び電子ビジネスにおける電子取引がある。

【００１９】

本発明の第１実施例が請求項２に記載される。この証明結果は、セキュリティ装置のもとのユーザからの介入を必要とすることなく、任意の第三者に対しセキュリティ装置による（おそらく以降の）検証用の証明書として電子証明書と呼ばれる実行証明として利用可能である。電子証明書へのアクセスを有する任意の第三者は、証明結果及びセキュリティ装置を利用して、セキュリティ装置が実際に電子証明書を生成したか検証することが可能である。

【００２０】

本発明の第１実施例の効果的な実現形態が、請求項３に記載される。この変形では、第１モードにより証明結果を生成するセキュリティプログラムはまた（本実施例では、実行モードと呼ぶ）、実行モードにより実際のアプリケーションプログラムを実行する。第２モードでは（検証モードと呼ぶ）、セキュリティプログラムは、証明結果を検証し、任意的には解読する。セキュリティプログラムが実行モードの一部としてアプリケーションプログラムを実行すると、証明結果はアプリケーションプログラムの真正な実行に拡張する。この証明結果は、どのアプリケーションプログラムが実行されたか、及び／またはどの入出力がそれぞれ利用されたかの検証が生成されることをカバーすることを保証するため、アプリケーションプログラム（の一部）、アプリケーションプログラム入力（の一部）及びアプリケーションプログラム出力（の一部）を含むようにしてもよい。さらに、これは仲介モードによる当該情報の（部分的）復元を可能にするものであってもよい。アプリケーション、その目的及びセキュリティポリシーに応じて、これらの解読された部分を出力するか否かはセキュリティプログラム次第である。

【００２１】

本発明の第１実施例のさらなる実現形態が、請求項４に記載されている。本変形では、第１モードにより動作した特定のセキュリティプログラムにより生成された証明結果を所持する第三者は、証明結果を検証し（任意的には解読することにより抽出し）、その後第三者に確認の証拠（任意的には解読された結果）を送信するため第２モードにより以降において実行される同一のセキュリティプログラムを実行するための入力として、セキュリティプログラムに証明結果を送信する。当該情報は、証明結果が第１モードにより同一のセキュリティ装置および同一のセキュリティプログラムにより計算されたことを第三者に確信させるために想定される。

【００２２】

本発明の第２実施例が、請求項５に記載される。この第２実施例では、証明結果はセキュアなデータ格納に利用される格納データ（おそらく暗号化された）を有する。第１モードとメモリストアと、及び第１モードとメモリロード動作とを関連付けることにより、（暗号化）データを有する証明結果をメモリ位置に格納することが可能である。このようにして、セキュアでない物理的メモリを、メモリアクセスに応答してメモリコンテンツを検証／認証することによりセキュアなものとすることができる。第１モードと第２モードの両方が、セキュリティプログラムの同一または異なる実行において複数回利用することが可能となる。

【００２３】

本発明の第２実施例の可能な実現形態が、請求項６に記載される。証明結果に格納されているデータを暗号化することにより、当該データはセキュリティプログラム自体による以外は、証明結果から抽出することは不可能となる。当該データ（の一部）が外部に出力されるか否かは、セキュリティプログラム次第である。

【００２４】

本発明の第３実施例が、請求項７に記載される。本実施例は、それがプログラム実行に関する状態情報を格納することを可能にするとき効果的である。これは、中断、スタンバイまたは電源オフなど（意図的か否かによらず）の後に、プログラムが同一状態により確実に実質的に継続されることを可能にする。継続により、状態情報は、第１モードで動作する同一のプログラムから生成されたものであることが信頼される前に、第２モードで検証される。

【００２５】

本発明の第３実施例のさらなる実現形態が、請求項８に記載される。証明結果の状態情報を暗号化することにより、状態情報はセキュリティプログラム自体による場合を除き、証明結果から抽出することはできない。

【００２６】

本発明の一実現形態が、請求項９に記載される。選択される動作モードがセキュリティプログラム自体でハードコード処理ができないため、異なる動作モード選択方法が求められる。動作モードを選択する大変簡潔かつ有用な方法は、セキュリティ装置のユーザにセキュリティプログラムへの入力として当該モードを供給させることである。

【００２７】

本発明の効果的な実現形態が、請求項１０に記載される。誰がセキュリティ装置に証明結果の生成または検証を求めているか、当該生成または検証が実際に同一のセキュリティ装置により実行されているかについてセキュリティ装置のユーザに証明するため、セキュリティプログラムは好ましくは、先行技術文献に記載されるような認証された実行を実現する第２のセキュリティプログラムの一部として実行される。

【００２８】

本発明のより詳細な実現形態が、請求項１１に記載される。本実現形態では、プリミティブ関数に用いられる乱数関数を実現するためＰＵＦが利用される。

【００２９】

本発明のより詳細な実現形態が、請求項１２に記載される。好ましくは（ほとんど）コリジョン・フリー（ｃｏｌｌｉｓｉｏｎ－ｆｒｅｅ）な乱数ハッシュ関数ｈ（．）が利用されるとき、当該プリミティブ関数が、証明結果を生成する第１モード、及び証明結果の検証または証明結果からの情報の取得を行う第２モードの両方で利用される鍵を確実に生成するのに効果的に利用可能である。請求項１に記載されるように、プログラムはセキュリティプログラムの関連部分のみを（セキュリティの観点から）表しているということが理解されるべきである。

【００３０】

本発明のより詳細な実現形態は、請求項１３に記載される。本実現形態で証明結果を計算するため生成された鍵はまた、入力変数の少なくとも一部に依存する。これは、（アプリケーション）プログラムの入力が、証明結果によりプロテクトされるように、セキュリティプログラムにおいてハードコード処理されることが不要となるという効果を有する。一部の入力が対象外となっているとき、すべての入力が考慮される必要はなく、セキュリティ装置とユーザとの間で秘密にされるべきであり（従って、第三者に通信せず）、あるいは異なるプログラム実行間で異なるものとされることが可能にされるべきである（動作を決定する入力は、もちろん利用されるべきでない）。

【００３１】

本発明によるシステムが、請求項１４に記載されるように特徴付けされる。

【００３２】

本発明によるコンピュータ可読媒体などのコンピュータプログラムが、請求項１５に記載されるように特徴付けされる。

【００３３】

本発明によるコンピュータ実行可能命令が、請求項１６に記載されるように特徴付けされる。

【００３４】

本発明による信号が、請求項１７に記載されるように特徴付けされる。

【００３５】

本発明の上記及び他の特徴がさらに、実施例と概略図を参照して説明される。

【発明を実施するための最良の形態】

【００３６】

図面を通じて、同一の参照番号は同様または対応する特徴を示す。図面に示されている特徴の一部は、典型的にはソフトウェアにより実現され、またソフトウェアモジュールまたはオブジェクトなどのソフトウェアエンティティを表す。

【００３７】

図１は、ＰＵＦ１０４を有するセキュリティ装置１０３を用いたアプリケーションのための基本モデルを示す。当該モデルは、システム１００により実現され、セキュリティ装置１０３の内部またはその制御下のチップ１０５の計算機能を利用することを所望するユーザ１０１を有する。

【００３８】

ユーザとチップは、おそらく信頼性の低い公衆通信チャネルにより互いに接続されている。ユーザは人間だけでなく、異なるソフトウェア、ハードウェアまたは他の装置とすることも可能である。

【００３９】

セキュリティ装置１０３は、プロセッサ１１１とメモリ１１２を有し、コンピュータプログラムプロダクツ１１３からのコンピュータにより実行可能な命令を実行するよう構成された処理装置１１０により実現することが可能である。

【００４０】

先行技術文献には、各ＰＵＦに一意的なチャレンジ及びレスポンスの取扱いが記載されている。チャレンジが与えられると、ＰＵＦは対応するレスポンスを計算することができる。ユーザは、ＰＵＦにより当初生成されたＣＲＰ（チャレンジ－レスポンスペア）のユーザ自身のプライベート（認証された）リストを所持している。当該リストは、（おそらくＰＵＦを除き）ユーザのみがリストの各チャレンジへのレスポンスを知っているという理由からプライベートである。ユーザのチャレンジはパブリックとすることができる。ユーザはセキュリティ装置により複数のＣＲＰを確立したと仮定される。

【００４１】

（限定数の）チャレンジに対するレスポンスのみが、ユーザに知られている。さらに、セキュリティ装置は、特定のチャレンジに対するレスポンスを計算するようにしてもよい。他の者が特定のチャレンジに対するレスポンスを復元するのを防ぐため、ＣＲＰの安全な管理方法が必要とされる。先行技術文献は、これを実現するのに制御されたＰＵＦのコンセプトを提案している。ＰＵＦは、それが分離不可な方法によりＰＵＦに物理的にリンクされているセキュリティアルゴリズムを介する場合に限りアクセス可能である場合に制御されたもの（制御されたＰＵＦまたはＣＰＵＦ）であると定義される（すなわち、当該アルゴリズムを回避しようとする任意の試みが、ＰＵＦの破壊を招くであろう）。特に、このセキュリティアルゴリズムは、ＰＵＦに与えられたチャレンジを制限し、外部に与えられるレスポンスに関する情報を制限することが可能である。制御は、ＰＵＦが単なる認証された識別プリケーション以上のものとなることを可能にする基本的アイデアである。ＰＵＦと制御されたＰＵＦは、スマートカード識別、認証された実行及びソフトウェアライセンス処理を実現するのに説明及び知られている。

【００４２】

中間者攻撃（ｍａｎ－ｉｎ－ｔｈｅ－ｍｉｄｄｌｅ　ａｔｔａｃｋ）を防ぐため、ＣＲＰ管理プロトコル中、ユーザは特定のチャレンジに対するレスポンスを求めることが禁止される。これは、ＣＲＰ管理プロトコルにおける懸念であり、当該プロトコルにおいて、セキュリティ装置はユーザにレスポンスを送信するためである。これは、セキュリティ装置がレスポンスをチャレンジ直接的には与えないように、ＰＵＦへのアクセスを制限することにより保証される。ＣＲＰ管理は、先行技術文献に記載されるように行われる。当該

アプリケーションプロトコルでは、レスポンスはメッセージ認証コード（ＭＡＣ）を生成するためなどさらなる処理のため内部的に利用されるだけであり、ユーザには送信されない。ＣＰＵＦは、プライベートな方法により（誰もプログラムが実行しているものを閲覧することができないか、あるいは少なくとも操作されているキーマテリアルは隠された状態にされる）、かつ真正な方法により（誰も検出されることなくプログラムが実行しているものを変更することができない）、ある形態のプログラム（さらに、セキュリティプログラム）を実行することが可能である。

【００４３】

ＣＰＵＦの制御は、ＰＵＦがセキュリティプログラムを開始する場合のみ、より詳細には、２つのプリミティブ関数ＧｅｔＲｅｓｐｏｎｓｅ（．）とＧｅｔＳｅｃｒｅｔ（．）を利用することによりアクセス可能となるよう設計される。本発明を実現するのに利用可能なプリミティブ関数は、

ＧｅｔＲｅｓｐｏｎｓｅ（ＰＣ）＝ｆ（ｈ（ｈ（ＳＰｒｏｇｒａｍ），ＰＣ））

ＧｅｔＳｅｃｒｅｔ（Ｃｈａｌｌｅｎｇｅ）＝ｈ（ｈ（ＳＰｒｏｇｒａｍ），ｆ（Ｃｈａｌｌｅｎｇｅ））

として定義される。ただし、ｆはＰＵＦ、ｈはパブリックに利用可能なランダムハッシュ関数（または実際には、擬似乱数関数）である。これらのプリミティブ関数では、ＳＰｒｏｇｒａｍは、真正な方法により実行されているプログラムのコードである。装置ユーザは、このようなセキュリティプログラムを実行するようにしてもよい。ここで、ｈ（ＳＰｒｏｇｒａｍ）は、ハードコード処理された値（一部のケースでは、Ｃｈａｌｌｅｎｇｅ）などを含むプログラムに含まれるあらゆるものを含むことに留意されたい。セキュリティ装置は、ｈ（ＳＰｒｏｇｒａｍ）を計算し、以降においてＧｅｔＲｅｓｐｏｎｓｅ（．）とＧｅｔＳｅｃｒｅｔ（．）が呼び出されると当該値を利用する。ｈ（ＳＰｒｏｇｒａｍ）の計算は、セキュリティプログラムの開始前（開始直前）、あるいはプリミティブ関数の最初のインスタンス化前に実行可能である。先行技術文献に示されるように、上記２つのプリミティブ関数は、ＧｅｔＲｅｓｐｏｎｓｅ（．）は実質的にＣＲＰの生成のため利用され、ＧｅｔＳｅｃｒｅｔ（．）はＣＲＰから共有された秘密を生成することを所望するアプリケーションにより利用されるセキュアなＣＲＰ管理を実現するのに十分である。

【００４４】

以降では、以下の記号が用いられる。

・Ｅ（ｍ，ｋ）は、鍵ｋによるメッセージｍの暗号化である。
・Ｄ（ｍ，ｋ）は、鍵ｋによるメッセージｍの解読である。
・Ｍ（ｍ，ｋ）は、鍵ｋによるメッセージｍのＭＡＣ処理である。
・Ｅ＆Ｍ（ｍ，ｋ）は、鍵ｋによるメッセージｍの暗号化及びＭＡＣ処理である。
・Ｄ＆Ｍ（ｍ，ｋ）は、ＭＡＣが一致する場合、鍵ｋによるメッセージｍの解読である。ＭＡＣが一致しない場合、それはＭＡＣが一致していないメッセージを出力し、解読は行わない。

【００４５】

認証された実行の概念は、先行技術文献に記載されている。当該技術は、いくつかの特定の実現形態により図示されるであろう。検証された実行は、いわゆる電子証明書を用いて提供される。セキュリティ装置上の入力Ｉｎｐｕｔによるプログラムの電子証明書は、セキュリティ装置のユーザが、ＸＰｒｏｇｒａｍの出力結果がセキュリティ装置上のＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）により生成されたものであるかかなりの確率で効率的にチェックすることができるように、セキュリティ装置上のＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）により効率的に生成される文字列として定義される。セキュリティ装置上でのＸＰｒｏｇｒａｍの実行を要求するユーザは、セキュリティ装置の所有者によることなく、自らがセキュリティ装置を製造したことを保証することができるセキュリティ装置のメーカーの信用性に頼ることが可能である。

【００４６】

図２は、計算がセキュリティ装置上で直接的に実行される認証された実行の簡単な例を示す。ユーザであるＡｌｉｃｅは、Ｂｏｂのコンピュータ２０１上で計算量の大きなプログラムＰｒｏｇｒａｍ（Ｉｎｐｕｔ）を実行することを所望している。Ｂｏｂのコンピュータは、ＰＵＦ２０３を有するセキュリティ装置２０２を有する。Ａｌｉｃｅはすでにセキュリティ装置によりＣＲＰ２１１のリストを確立したと仮定される。（Ｃｈａｌｌｅｎｇｅ，Ｒｅｓｐｏｎｓｅ）をＢｏｂのＰＵＦに対するＡｌｉｃｅのＣＲＰの１つであるとする。第１の実現形態の変形では、Ａｌｉｃｅはセキュリティ装置２０２に（Ｃｈａｌｌｅｎｇｅ，Ｅ＆Ｍ（（ＸＰｒｏｇｒａｍ，Ｉｎｐｕｔ），ｈ（ｈ（ＣＰｏｇｒａｍ），Ｒｅｓｐｏｎｓｅ））に等しい入力Ｉｎｐｕｔ２３２と共に以下のプログラムＣＰｒｏｇｒａｍ２３１を（通信２２１により）送信する。

【００４７】

【表１】

```
CProgram1(Inputs):
begin program
var Challenge,EM,XProgram,Input,Result,Certificate;
                (Challenge,EM)=Inputs;
                Secret=GetSecret(Challenge);
                (XProgram,Input)=D&M(EM,Secret);
                Abort if the MAC does not match;
                Result=XProgram(Input);
                Certificate=M(Result,Secret);
                Output(Result,Certificate);
end program
```

第２の実現形態の変形では、Ａｌｉｃｅはセキュリティ装置に（Ｅ＆Ｍ（（ＸＰｒｏｇｒａｍ，Ｉｎｐｕｔ），ｈ（ｈ（ＣＰｒｏｇｒａｍ），Ｒｅｓｐｏｎｓｅ））に等しい入力Ｉｎｐｕｔと共に以下のプログラムＣＰｒｏｇｒａｍ２を送信する。この変形は、ＣＰｒｏｇｒａｍ２のＣｈａｌｌｅｎｇｅの値をハードコード処理するとき、よりロウバストなものとなる。すなわち、Ｃｈａｌｌｅｎｇｅの値はプリミティブ関数において利用される。

【００４８】

【表２】

```
CProgram2(Inputs):
begin program
const Challenge=...;
var EM,XProgram,Input,Result,Certificate;
                (EM)=Inputs;
                Secret=GetSecret(Challenge);
                (XProgram,Input)=D&M(EM,Secret);
                Abort if the MAC does not match;
                Result=XProgram(Input);
                Certificate=M(Result,Secret);
                Output(Result,Certificate);
end program
```

Ｒｅｓｕｌｔ＝ＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）とすることにより、ＲｅｓｕｌｔはＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）の出力の一部となることは理解される。電子証明が必要とされないより多くの出力があるかもしれない。Ｏｕｔｐｕｔ（．．．）は、通信２２２に示されるようなＣＰＵＦからの結果２３３を送信するのに利用される。セキュリティ装置から送信されるものは何れも、全世界に潜在的には視聴されうる（メーカーがセキュリティ装置の物理的な所持をしているブーストラップ処理期間を除き）。プログラムのセキュアな設計は、Ｒｅｓｕｌｔに暗号化された形式により配置された結果を生成する。この暗号化は、従来の暗号化により、あるいはＳｅｃｒｅｔを用いることにより実行可能である。後者の場合、ＳｅｃｒｅｔはＩｎｐｕｔに含まれる。

【００４９】

ＡｌｉｃｅのＣＲＰはプライベートであるため、他の何れの者もＳｅｃｒｅｔ、すなわち、Ｓｅｃｒｅｔを有するＭＡＣを生成することができない。ＭＡＣはプログラムの２つの箇所で利用される。第１のＭＡＣは、プログラムによりチェックされ、Ｉｎｐｕｔの真正性を保証する。第２のＭＡＣは、Ａｌｉｃｅによりチェックされ、セキュリティ装置から受け取ったメッセージの真正性を保証する。Ａｌｉｃｅとは別に、セキュリティ装置はプログラムを実行することによりＣｈａｌｌｅｎｇｅが与えられると、Ｓｅｃｒｅｔを生成することができる。このことは、ＲｅｓｕｌｔとＣｅｒｔｉｆｉｃａｔｅがセキュリティ装置のＣＰｒｏｇｒａｍにより生成されたことを意味する。言い換えると、ＣＰｒｏｇｒａｍは、入力としてＩｎｐｕｔにより認証された実行を行う。これは、Ｃｅｒｔｉｆｉｃａｔｅが電子証明書であることを証明するものである。

【００５０】

従って、電子証明書はセキュアなリモート計算に利用可能である。Ｃｅｒｔｉｆｉｃａｔｅが一致する場合、これはＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）がセキュリティ装置上で（ＣＰｒｏｇｒａｍ（Ｉｎｐｕｔｓ）により）実行されたことをＡｌｉｃｅに証明する。

【００５１】

先行技術文献に記載されるような認証された実行は、ＸＰｒｏｇｒａｍの実行を実行のプルーフとして証明するため、第三者に対しＡｌｉｃｅにより利用可能ではない。ＡｌｉｃｅのＣＲＰを利用して、Ａｌｉｃｅは、任意の結果であるＲｅｓｕｌｔに対し電子証明書Ｃｅｒｔｉｆｉｃａｔｅを偽ることが可能である。この事実から、ＡｌｉｃｅはＣｈａｌｌｅｎｇｅに関するレスポンスを利用することによりＳｅｃｒｅｔを計算することができる。ＡｌｉｃｅがＣＲＰを計算できる（ＭＡＣをチェックするため）という事実により、Ａｌｉｃｅは当該電子証明書をＡｌｉｃｅがＢｏｂのセキュリティ装置上でＸＰｒｏｇ

ｒａｍ（Ｉｎｐｕｔ）を（ＣＰｒｏｇｒａｍ（Ｉｎｐｕｔｓ）において）実行したことを第三者に証明するための実行のプルーフとして利用することができなくなる。

【００５２】

本発明の第１実施例では、任意の第三者に対し実行のプルーフとして利用可能な証明結果が利用される。セキュリティ装置上で結果Ｒｅｓｕｌｔを生成する入力Ｉｎｐｕｔによるプログラム$XPrograma$の電子証明書ＥＰｒｏｏｆは、入力ＥＰｒｏｏｆ及び$XPrograma$によるセキュリティ装置と任意の仲介者との間にプロトコルＡ１が存在するように、セキュリティ装置上でＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）により生成される文字列として定義され、ＥＰｒｏｏｆがＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）によりセキュリティ装置上で生成されたか否か高い確率で正確かつ効率的に決定可能な補助情報であるかもしれず、正しく生成された場合には、セキュリティ装置上のＸＰｒｏｇｒａｍ（Ｉｎｐｕｔ）によりＥＰｒｏｏｆと共に以前に生成された結果Ｒｅｓｕｌｔを高い確率により復元する。プロトコルＡ１は、仲介者プロトコルと呼ばれる。以下の例は、電子証明書がセキュリティ装置のユーザと所有者の両方により利用可能であることを示している。

【００５３】

実行の証明をサポートするため、実行の証明を生成するための追加的なプログラムレイヤにより認証された実行の解を拡張することが必要とされる。ユーザであるＡｌｉｃｅは、ＰＵＦを有する１つのセキュリティ装置を有するＢｏｂのコンピュータ上でアプリケーションプログラムＡＰｒｏｇｒａｍを実行することを所望する。Ａｌｉｃｅは既に、Ｂｏｂのセキュリティ装置によりＣＲＰを確立している。

【００５４】

本実施例が利用可能な第１の例として、図３に参照されるように、Ａｌｉｃｅが配信者３１０であり、ＢｏｂがセキュリティＳＴＢ３００の所有者であるＳＴＢ（セットトップボックス）アプリケーションを考える。プログラムＡ３２０では、Ｂｏｂはあるサービスを購入する。Ａｌｉｃｅは、取引詳細３３２、電子証明書３３３（電子証明書は、取引詳細と電子証明書の両方の真正性を検証する）及び電子証明書３３４を受信する。Ａｌｉｃｅはステップ３４０において、電子証明書が一致するかチェックする。一致する場合、電子証明書はＢｏｂのＳＴＢにより生成されたものであることを知ることとなり、プログラムＢにおいて当該取引を続ける。電子証明書がＢｏｂがサービスを購入したことの確認として利用することができる。なぜならば、仲介者が取引詳細を復元することができるからである。プログラムＢ３２１では、Ｂｏｂは自らが要求したサービスに属するコンテンツ３３５を受信する。当該コンテンツは、ＣＲＰを用いることにより暗号化されてもよい。Ａｌｉｃｅは、プログラムＢにおいてＢｏｂのアクションの第２の電子証明書３３６を受信する。第１の例では、ＢｏｂはプログラムＢのコンテンツを送信するため、Ａｌｉｃｅの約束の証明書を受信しないかのように見える。しかしながら、Ａｌｉｃｅだけでなく Ｂｏｂもまた、第１の電子証明書を利用することが可能である。任意の第三者は、ＢｏｂのＳＴＢがプログラムＡにより符号化されたプロトコルの実行に成功したことをチェックすることが可能であり、それ自体プログラムＢによりコンテンツをＢｏｂに送信するＡｌｉｃｅの約束である。例えば、Ｂｏｂは電子証明書を利用して、割引やアップグレードの資格を与えるあるサービスを購入したことを第三者（特に、Ａｌｉｃｅ）に確信させることができる。

【００５５】

第２の例として、Ａｌｉｃｅが入力の一部としてタイムスタンプによりＢｏｂのセキュリティ装置上でプログラムを実行することを所望していると仮定する。当該実行の結果は、タイムスタンプが正確な実行時間を表しているというＢｏｂの同意によるタイムスタンプのコピーを含む。例えば、当該プログラムは、Ｂｏｂが同意しているか問い合わせ、Ｂｏｂが同意していない場合には中断されるよう設計される。正しい電子証明書が与えられると、仲介者はこの結果を抽出する。すなわち、仲介者はタイムスタンプをチェックし、Ｂｏｂ及び／またはＡｌｉｃｅの主張が依然として有効であるか検証することができる。

【００５６】

第３の例として、異なるモードを有するプログラムＰｒｏｇｒａｍ’を仮定する。そのモードに応じて、Ｐｒｏｇｒａｍ’はＥＰｒｏｏｆがＰ上の入力Ｉｎｐｕｔによるプログラムの電子証明書であるプロセッサＰ上において（Ｒｅｓｕｌｔ，ＥＰｒｏｏｆ）＝Ｐｒｏｇｒａｍ（Ｉｎｐｕｔ）を計算するか、あるいは、Ｐｒｏｇｒａｍ’はＥＰｒｏｏｆが有効な電子証明書であるかチェックする仲介者の役割を果たし、有効である場合には、Ｒｅｓｕｌｔを再構成する。仲介者の役割において、ＥＰｒｏｏｆはＰｒｏｇｒａｍ’の次のモードへのチケットとして利用されてもよい。当該技術は、限定受信を実現する。

【００５７】

図４は、異なるプログラムレイヤを示す。ＥＰｒｏｇｒａｍ４０３と呼ばれる実行の証明書を生成または検証する本発明によるプログラムは、ユーザと第三者の両方がセキュリティ装置において実行が行われたことを確信できるように、ＰＵＦ４０１を有するセキュリティ装置４００における認証された実行プログラムＣＰｒｏｇｒａｍ１（４０２）（またはＣＰｒｏｇｒａｍ２（４０２））のＸＰｒｏｇｒａｍ部分として実行される。ＥＰｒｏｇｒａｍは、実行モード４０４と仲介モード４０５の両方を有する。

【００５８】

実行モードでは、ＥＰｒｏｇｒａｍはＡｌｉｃｅが興味を有する結果だけでなく、電子証明書も計算する（ＡＰｒｏｇｒａｍ４０６において）。Ａｌｉｃｅは、プログラムがＢｏｂのセキュリティ装置上で正確に実行されたことを確信するため、認証された実行を利用する（ＣＰｒｏｇｒａｍのＸＰｒｏｇｒａｍ部分としてＥＰｒｏｇｒａｍを実行することにより）。仲介者は、認証された実行を用いて、仲介モードによりＥＰｒｏｇｒａｍを実行することにより電子証明書をチェックすることができる。主要なアイデアは、ＧｅｔＲｅｓｐｏｎｓｅ（．）プリミティブが、両方のモードを含む完全なプログラムＥＰｒｏｇｒａｍのハッシュに依存するということである。この結果、実行モードによりプログラムＥＰｒｏｇｒａｍにより生成された電子証明書は（ＧｅｔＲｅｓｐｏｎｓｅ（．）プリミティブを介し取得される鍵により）、仲介モードによりプログラムＥＰｒｏｇｒａｍにより解読可能である。

【００５９】

セキュリティは、第１にＧｅｔＲｅｓｐｏｎｓｅ（．）プリミティブの解読、ハッシュの解読及びＧｅｔＲｅｓｐｏｎｓｅ（．）が規定されるＰＵＦの解読を行う困難さにより、第２に暗号化及びＭＡＣ　Ｅ＆Ｍ（．）プリミティブを解読する困難さにより決定される。

【００６０】

以下において、ＥＰｒｏｇｒａｍのいくつかの変形が与えられる。一部のプログラムは、入力の一部をハードコード処理することから、柔軟性は低下するが、ロウバスト性は向上する。証明結果に与えられる出力量もまた異なる。これらのアルゴリズムの任意の変形が実現可能である。

【００６１】

第１の変形では、ＡｌｉｃｅはＡＰｒｏｇｒａｍ（Ｉｎｐｕｔ）を実行し、実行の証明書を受け取ることを所望し、Ｍｏｄｅ４３２が「実行モード」、ＰＣ４３３が乱数文字列、ＥＰｒｏｇｒａｍ１が以下に規定されるＩｎｐｕｔｓ＝（（ＡＰｒｏｇｒａｍ，Ｉｎｐｕｔ，ＰＣ），Ｍｏｄｅ）（ＡＰｒｏｇｒａｍ４３５、Ｉｎｐｕｔ４３４）であるＥＰｒｏｇｒａｍ１（Ｉｎｐｕｔｓ）（４３１）を実行する。ＰＣは、秘密鍵ＫＥ（実行モードにおける）またはＫＡ（仲介モードにおける）を生成するため、乱数関数のチャレンジを計算するため、「ｐｒｅ－ｃｈａｌｌｅｎｇｅ」としてＧｅｔＲｅｓｐｏｎｓｅ（．）により利用される。Ａｌｉｃｅは、認証された実行技術を利用して、上述のように、ＣＰｒｏｇｒａｍ４３０を利用してＢｏｂのセキュリティ装置上でＥＰｒｏｇｒａｍ１（Ｉｎｐｕｔｓ）を実行する。Ａｌｉｃｅは、セキュリティ装置から取得するすべての出力の真正性を検証するため、電子証明書をチェックする。生成された電子証明書は、Ｐｒｏｇｒａｍ（Ｉｎｐｕｔ）により生成された結果４３８だけでなく生成された電子証明書４３６の

証明書である。

【００６２】

【表３】

```
EProgram1(Inputs):
begin program
var X,Mode,AProgram,Input,PC,Result,KE,KA,Checkbit,EMResult,EProof,Results;
        (X,Mode)=Inputs;
        If Mode is execution mode:
        begin
                (AProgram,Input,PC)=X;
                Result=AProgram(Input);
                KE=GetResponse(PC);
                EMResult=E&M(Result,KE);
                EProof=(PC,EMResult);
                Results=(Result,EProof);
        end
        If Mode is arbitration mode:
        begin
                EProof=X;
                (PC,EMResult)=EProof;
                KA=GetResponse(PC);
                Result=D&M(EMResult,KA);
                CheckBit=(MAC of EMResult matches);
                Results=(Result,CheckBit);
        end
        Output(Results);
end program
```

第１実施例の第２の変形では、Ａｌｉｃｅは、乱数文字列ＰＣをロウバスト性を向上させるためＥＰｒｏｇｒａｍにハードコード処理し、当該入力を有するプログラムが性格に利用されたことを以降において証明できるように、アプリケーションプログラムＡＰｒｏｇｒａｍのハッシュ値（部分）と、アプリケーションプログラム入力Ｉｎｐｕｔのハッシュ値をＥＰｒｏｏｆに組み込む。仲介モードでは、ＥＰｒｏｏｆは検証されるだけであり、ＡＰｒｏｇｒａｍ、Ｉｎｐｕｔ及びＲｅｓｕｌｔをカバーして、これらの何れも第三者ユーザには出力されない。

【００６３】

【表４】

```
EProgram2(Inputs):
begin program
const PC=...;
var X,Mode,Program,Input,Result,KE,KA,Checkbit,EMResult,EProof,Results;
                (X,Mode)=Inputs;
                If Mode is execution mode:
                begin
                        (Program,Input)=X;
                        Result=Program(Input);
                        KE=GetResponse(PC);
                        EMResult=E&M((Program,Input,Result),KE);
                        EProof=(PC,EMResult);
                        Results=(Result,EProof);
                End
                If Mode is arbitration mode:
                begin
                        EProof=X;
                        (PC,EMResult)=EProof;
                        KA=GetResponse(PC);
                        Result=D&M(EMResult,KA);
                        CheckBit=(MAC of EMResult matches);
                        Results=(CheckBit);
                end
                Output(Results);
end program
```

第１実施例の第３の変形では、乱数文字列ＰＣは省略され、これにより、計算が簡単化される。鍵ＫＥが、ＫＥ＝ＧｅｔＲｅｓｐｏｎｓｅ（）またはよりシンプルな（新しい）プリミティブ関数ＫＥ＝ｆ（ｈ（ＥＰｒｏｇｒａｍ３））によりＥＰｒｏｇｒａｍ３において計算される。

【００６４】

第１実施例の第４の変形では、ＰＣ及び任意的な他の入力パラメータが、ハードコード処理がなされないが（第２の変形のように）、プリミティブ関数における乱数関数への入力として依然として利用される。これは、例えば、ＰｒｏｇｒａｍとＩｎｐｕｔがＥＰｒｏｇｒａｍ３への入力として取得され、ＧｅｔＲｅｓｐｏｎｓｅ（．）への入力として利用されるＥＰｒｏｇｒａｍ３に示される。一部の入力が関心の対象外であるとき、必ずしもすべての入力が考慮される必要はなく、セキュリティ装置と当該セキュリティ装置のユーザとの間で秘密に保たれるべきであり（従って、第三者に通信されるべきでない）、あるいは異なるプログラム実行間で異なるものとされることが許容されるべきである（動作モードを決定する入力は、もちろん利用されるべきでない）。

【００６５】

【表５】

```
EProgram3(Inputs):
begin program
const PC=...;
var X,Mode,Program,Input,Result,KE,KA,Checkbit,EMResult,EProof,Results;
                (X,Mode)=Inputs;
                If Mode is execution mode:
                begin
                        (Program,Input)=X;
                        Result=Program(Input);
                        KE=GetResponse((PC,Program,Input));
                        EMResult=E&M((Program,Input,Result),KE);
                        EProof=(PC,EMResult);
                        Results=(Result,EProof);
                end
                If Mode is arbitration mode:
                begin
                        EProof=X;
                        (PC,EMResult)=EProof;
                        KA=GetResponse((PC,Program,Input));
                        Result=D&M(EMResult,KA);
                        CheckBit=(MAC of EMResult matches);
                        Results=(CheckBit);
                end
                Output(Results);
end program
```

仲介モードでは、仲介者は３つのステップから構成されるＢｏｂのセキュリティ装置によりプロトコルを実行する。ステップ１では、仲介者はステップ４５０における実行の証明書ＥＰｒｏｏｆをＡｌｉｃｅまたはＢｏｂから受信する。仲介者は、Ｍｏｄｅ４４２が仲介モードに等しいＩｎｐｕｔｓ＝（ＥＰｒｏｏｆ，Ｍｏｄｅ）（ＥＰｒｏｏｆ：４４４）を構成する。仲介者はまた、おそらくＡｌｉｃｅとＢｏｂから同じＥＰｒｏｇｒａｍとＣＰｒｏｇｒａｍを取得する（おそらく以前に実行されたように、本例では、ステップ４５１と４５２において仲介者に通信される）。ここで、仲介者はＰＣを必要としないということに留意されたい。

【００６６】

ステップ２において、仲介者はＣＰｒｏｇｒａｍ４４０による認証された実行の技術を利用して、Ｂｏｂのセキュリティ装置上でＥＰｒｏｇｒａｍ（Ｉｎｐｕｔｓ）（ＥＰｒｏｇｒａｍ：４４１）を実行する。仲介者は、電子証明書４４７をチェックし、セキュリティ装置から取得したＲｅｓｕｌｔｓの真正性を検証する。電子証明書がＲｅｓｕｌｔｓと一致する場合、仲介者はＢｏｂのセキュリティ装置が他者の介入なくＥＰｒｏｇｒａｍ（Ｉｎｐｕｔｓ）を実行し、それの入出力が改ざんされていないことを知る。特に、何れの

者も入力ＥＰｒｏｏｆ及びＭｏｄｅを変更していない。言い換えると、Ｂｏｂのセキュリティ装置は、ＥＰｒｏｏｆを用いて仲介モードによりＥＰｒｏｇｒａｍ（Ｉｎｐｕｔｓ）を実行した。仲介モードでは、Ｒｅｓｕｌｔ４４５はＯｕｔｐｕｔに完全または部分的に供給可能か、あるいは全く供給することができない。それはまた、Ｒｅｓｕｌｔから導出される情報により置換可能である。これは、アプリケーション及び仲介者に依存するものであってもよい。このとき、その決定はプログラムにより実現される。例えば、プライバシーの理由から、ＥＰｒｏｇｒａｍは結果の概要のみを仲介者に送信することも可能である。

【００６７】

ステップ３において、仲介者はＣｈｅｃｋＢｉｔ４４６が真であるか、すなわち、ＥＭＲｅｓｕｌｔのＭＡＣが一致するか検証する。一致する場合、仲介者はＢｏｂのセキュリティ装置が実行モードによりＥＰｒｏｏｆとＲｅｓｕｌｔを計算したと判断する。一致しない場合、仲介者はＢｏｂのセキュリティ装置は実行モードによりＥＰｒｏｏｆを計算しなかったと判断する。仲介モードでは、ＥＰｒｏｇｒａｍは、ＭＡＣが一致していないということを出力するか（Ｄ＆Ｍ（．）及びＣｈｅｃｋＢｉｔの定義を参照せよ）、あるいは、ＭＡＣが解読された結果と共に一致しているということを出力する。偽の電子証明書を生成するため、（偽の）結果ＦＲｅｓｕｌｔに対するＦＥＰｒｏｏｆ＝（ＦＰＣ，ＦＥＭＲｅｓｕｌｔ）は、いわゆる困難な問題である。

【００６８】

第２実施例では、電子証明書と類似した証明結果が、セキュアでない（おそらくオフチップ）物理的メモリを用いて、あるいは中断処理、ソフトウェアの著作権侵害による環境、暗号化されていないコンテンツが不正に配布される可能性がある環境などの困難な状況の下、ＰＵＦを有する特定のセキュリティ装置上の特定プログラムによるセキュアなメモリ制御を実現するのに利用可能である。

【００６９】

図５は、セキュアなメモリ実現形態を示す。本実施例では、第１モード５０１または実行モードが、メモリ５０３に結果を格納するのにセキュリティプログラム５００により使用され、第２モード５０２または仲介モードが、メモリをロードし、それの真正性をチェックするのにプログラムにより使用される。メモリはデータＤａｔａを位置Ａｄｄｒｅｓｓに格納すると仮定される。これは、単一のアドレスまたはアドレス範囲とすることが可能である。Ｄａｔａは（ＰＣ，Ｅ＆Ｍ（Ｄａｔａ，Ｋ））として暗号化形式により格納される。ここで、ＫはＧｅｔＲｅｓｐｏｎｓｅ（ＰＣ）に等しい。入力（Ａｄｄｒｅｓｓ，（ＰＣ，Ｄａｔａ））によるＳｔｏｒｅと呼ばれる手順は、データを格納し、Ｒｅｓｕｌｔ＝Ｄａｔａｍ、ＥＭＲｅｓｕｌｔ＝ＥＭＤａｔａ及びＥＰｒｏｏｆ＝（ＰＣ，ＥＭＤａｔａ）であるＥＰｒｏｇｒａｍにおける実行モードに対応する。入力ＡｄｄｒｅｓｓによるＬｏａｄと呼ばれる手順は、データをロードし、Ｒｅｓｕｌｔ＝Ｄａｔａ、ＥＭＲｅｓｕｌｔ＝ＥＭＤａｔａ及びＥＰｒｏｏｆ＝（ＰＣ，ＥＭＤａｔａ）であるＥＰｒｏｇｒａｍの仲介モードに対応する。

【００７０】

【表６】

```
MProgram
begin program
        Store(Address,Data):
        begin procedure
        var PC,KE,EMData;
                KE=GetResponse(PC);
                EMData=E&M(Data,KE);
                Store(PC,EMData)at Address;
        end procedure
        Load(Address):
        begin procedure
        var PC,EMData,KA;
                Load(PC,EMData)from Address;
                KA=GetResponse(PC);
                Data=D&M(EMData,KA);
                CheckBit=(MACof EMData matches);
                Output(Data,CheckBit);
        end procedure
end program
```

それのコードの一部として手順Ｓｔｏｒｅ（．）とＬｏａｄ（．）を有するプログラムＭＰｒｏｇｒａｍ（Ｉｎｐｕｔ）がメモリアクセスに対する手順を利用する場合、プログラム自体は両方のモードで実行される。Ｓｔｏｒｅ（．）とＬｏａｄ（．）のＧｅｔＲｅｓｐｏｎｓｅ（．）は何れも同一のＭＰｒｏｇｒａｍに依存する。ＭＰｒｏｇｒａｍがデータを格納する場合、それは第１モードで動作する。すなわち、データはメモリに暗号化された電子証明書形式により書き込まれる。それがデータをロードする場合、それは第２モードにより動作する。すなわち、出力されたＣｈｅｃｋＢｉｔが、Ｂｏｂのセキュリティ装置上で実行されるＭＰｒｏｇｒａｍから出力されるという意味で、データの真正性をチェックするのに利用される。この意味で、ＭＰｒｏｇｒａｍは完全にそれが処理するデータにより制御される。暗号化されていないデータを公衆に出力すべきか否かはＭＰｒｏｇｒａｍ次第である。このプログラムは効率的にデータの所有者になる。

【００７１】

ここで、アドバーサリ（ａｄｖｅｒｓａｒｙ）が旧式のバージョンにより現在メモリを置き換え、未検出に進行するということに留意されたい。最新のメモリの真正性をチェックするため、プロセッサはタイマーカウンターを格納するためプライベートメモリを必要とする。このタイマーカウンターは、ＰＵＦから導出される鍵を有するＭＡＣと共に格納することが可能である。さらに、このアイデアは、Ｄａｖｉｄ Ｌｉｅ、Ｃｈａｎｄｒａｍｏｈａｎ Ｔｈｅｋｋａｔｈ、Ｍａｒｋ Ｍｉｔｃｈｅｌｌ、Ｐａｔｒｉｃｋ Ｌｉｎｃｏｌｎ、Ｄａｎ Ｂｏｎｅｈ、Ｊｏｈｎ Ｍｉｔｃｈｅｌｌ及びＭａｒｋ Ｈｏｒｏｗｉｔｚによる「Ａｒｃｈｉｔｅｃｔｕｒａｌ Ｓｕｐｐｏｒｔ ｆｏｒ Ｃｏｐｙ ａｎｄ Ｔａｍｐｅｒ Ｒｅｓｉｓｔａｎｔ Ｓｏｆｔｗａｒｅ」（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ９ｔｈ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ

Ａｒｃｈｉｔｅｃｔｕｒａｌ Ｓｕｐｐｏｒｔ ｆｏｒ Ｐｒｏｇｒａｍｍｉｎｇ Ｌａｎｇｕａｇｅｓ ａｎｄ Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍｓ（ＡＳＰＬＯＳ－ＩＸ），Ｎｏｖｅｍｂｅｒ，２００２，ｐ．１６９－１７７）、及びＢｌａｉｓｅ Ｇａｓｓｅｎｄ、Ｇ．Ｅｄｗａｒｄ Ｓｕｈ、Ｄｗａｉｎｅ Ｃｌａｒｋｅ、Ｍａｒｔｅｎ ｖａｎ Ｄｉｊｋ及びＳｒｉｎｉｖａｓ Ｄｅｖａｄａｓによる「Ｃａｃｈｅｓ ａｎｄ Ｍｅｒｋｌｅ Ｔｒｅｅｓ ｆｏｒ Ｅｆｆｉｃｉｅｎｔ Ｍｅｍｏｒｙ Ａｕｔｈｅｎｔｉｃａｔｉｏｎ」（Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ９ｔｈ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｓｙｍｐｏｓｉｕｍ ｏｎ Ｈｉｇｈ－Ｐｅｒｆｏｒｍａｎｃｅ Ｃｏｍｐｕｔｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅ，Ｆｅｂｒｕａｒｙ，２００３）に記載されるようなオフチップリソースメモリ認証スキームをセキュアに利用したより高度なアーキテクチャを利用して向上させることが可能である。

【００７２】

図６は、本発明の第３及び第４実施例のアーキテクチャを示す。

【００７３】

本発明の第３実施例では、中断の場合にセキュリティ装置６００上で実行されるプログラム６０１が、それのプログラム状態６０５をセキュアに格納することが可能となるように、証明結果がプログラム実行状態６０２を格納するのに利用される。中断が起こると、プログラム状態は暗号化される（第１モード６０３のように）。セキュリティ装置は、それの状態を外部に明らかにすることなく、以降においてそれの実行を続けるようにしてもよい。継続により、プログラム状態は検証、解読（第２モード６０４のように）及び復元される。このため、プログラムはフル制御状態となる。これは、中断に対しロウバストなセキュアな実行を可能にする。アプリケーションは、セキュア中断処理、ソフトウェア著作権侵害に対する耐性、及び暗号化されていないコンテンツの不正な配布に対する耐性である。

【００７４】

第４実施例では、以降の利用のため、プログラムは暗号化されたコンテンツ６０２または暗号化されたソフトウェア６０２を格納するようにしてもよい。当該コンテンツのみが再生し（または再生を継続し）、当該ソフトウェアのみが同一の特定のセキュリティ装置６００上で同一状態６０５により実行される（または実行を継続される）。これは、暗号化されていないコンテンツの不正な配布またはソフトウェア著作権侵害に対する耐性を可能にする。

【００７５】

ここで、セキュリティ装置の所有者（Ｂｏｂ）とセキュリティ装置のユーザ（Ａｌｉｃｅ）は同一の者であってもよいということに留意されたい。例えば、ＢｏｂはＰｒｏｇｒａｍ（Ｉｎｐｕｔ）によりＲｅｓｕｌｔを計算したことを電子証明書により他者に対し証明する。最終的には、本発明の効果は、Ａｌｉｃｅと仲介者の何れもがＰＵＦを搭載したセキュリティ装置を必要としないということである。

【００７６】

本発明は、デジタル、物理的またはオプティカルなすべてのＰＵＦに適用可能であるという意味において、一般に適用可能である。当該構成の詳細は、物理的ＰＵＦに対し与えられるが、デジタルまたはオプティカルＰＵＦに転送可能である。

【００７７】

他の実施例が可能である。上記説明では、「有する」という用語は他の要素またはステップを排除するものでなく、「ある」という用語は複数を排除するものでなく、単一のプロセッサまたは他のユニットがまた請求項に記載される複数の手段の機能を実現するようにしてもよい。

【図面の簡単な説明】

【００７８】

【図１】図１は、ＰＵＦを用いたアプリケーションのための基本モデルを示す。

【図２】図２は、認証された実行の使用例を示す。

【図３】図３は、実行証明書の使用例を示す。
【図４】図４は、認証された実行下で電子証明書を生成する異なるプログラムレイヤの概観を示す。
【図５】図５は、セキュアなメモリ実現形態を示す。
【図６】図６は、中断された処理を示す。

【図１】

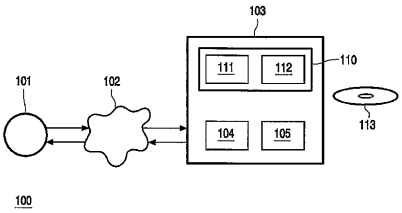


FIG. 1

【図２】

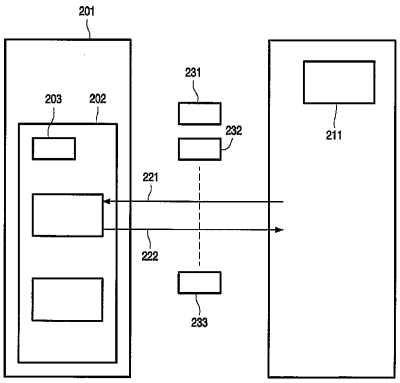


FIG. 2

【図３】

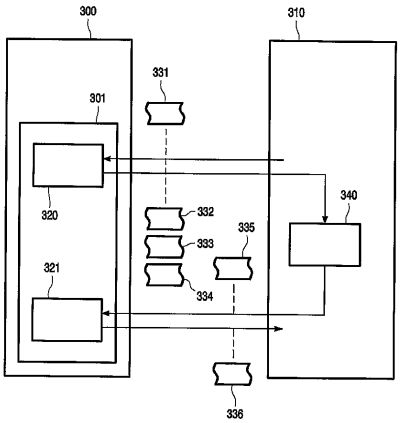


FIG. 3

【図４】

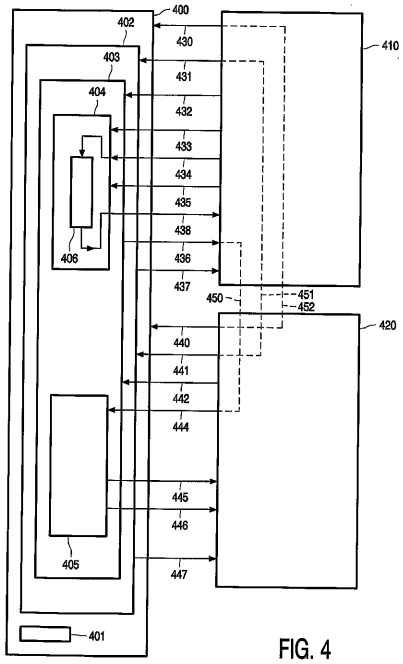


FIG. 4

【図５】

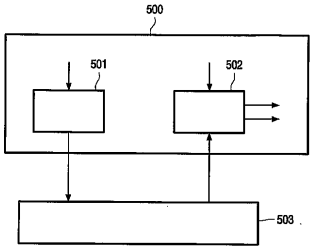


FIG. 5

【図６】

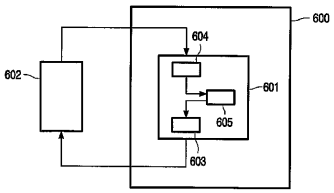


FIG. 6

フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

(72)発明者 ファン デイク, マールテン エリク
アメリカ合衆国 ニューヨーク州 １０５１０－８００１ ブライアクリフ・マナー ピー・オー・ボックス ３００１
(72)発明者 テュイルス, ピム テオ
アメリカ合衆国 ニューヨーク州 １０５１０－８００１ ブライアクリフ・マナー ピー・オー・ボックス ３００１

Ｆターム(参考) 5B276 FB06