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

特許第5704951号
(P5704951)

(45)発行日　平成27年4月22日(2015.4.22)　(24)登録日　平成27年3月6日(2015.3.6)

(51)Int.Cl.		F I		
G06F 21/12	(2013.01)	G06F	21/12	
G06F 21/62	(2013.01)	G06F	21/62	309
G06F 21/77	(2013.01)	G06F	21/62	390
		G06F	21/77	

請求項の数 13　(全 26 頁)

(21)出願番号　特願2011-27652(P2011-27652)
(22)出願日　平成23年2月10日(2011.2.10)
(65)公開番号　特開2012-168645(P2012-168645A)
(43)公開日　平成24年9月6日(2012.9.6)
審査請求日　平成26年1月22日(2014.1.22)

(73)特許権者　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100095957
弁理士　亀谷　美明
(74)代理人　100096389
弁理士　金本　哲男
(74)代理人　100101557
弁理士　萩原　康司
(74)代理人　100128587
弁理士　松本　一騎
(72)発明者　森田　直
東京都港区港南１丁目７番１号　ソニー株式会社内

最終頁に続く

(54)【発明の名称】情報処理装置、情報処理方法及びコンピュータプログラム

(57)【特許請求の範囲】
【請求項１】
耐タンパ性能を有する環境において、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行部を備え、
前記プログラム実行部で実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、
前記プログラム実行部は、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、情報処理装置。
【請求項２】
前記セキュリティ属性には、前記変数または関数が規定されるデータ構造へのアクセス属性、認証鍵バージョン及び認証鍵テーブルへのポインタが設定される、請求項１に記載の情報処理装置。
【請求項３】
前記プログラム実行部で実行される前記コンピュータプログラムで定義された関数及び変数はシステム鍵で認証することにより定義の変更が可能である、請求項１に記載の情報

処理装置。
【請求項４】
前記関数を定義する関数式をデータ構造に規定する場合は、該データ構造及び該関数内で利用する関数、変数の全ての認証鍵で認証されていることを要する、請求項１に記載の情報処理装置。
【請求項５】
外部装置との間で伝送されるバイナリデータを所定の形式にエンコードするバイナリデータ変換部を備える、請求項１に記載の情報処理装置。
【請求項６】
前記バイナリデータ変換部は、前記認証処理のために伝送されるバイナリデータを変換した後のデータの先頭に前記所定の形式で用いられない符号を付加する、請求項５に記載の情報処理装置。
【請求項７】
不揮発性を有し、前記プログラム実行部で実行されるプログラムで使用される変数定義及び関数定義を保存する記憶部を備える、請求項１に記載の情報処理装置。
【請求項８】
前記プログラム実行部が実行するコンピュータプログラムは任意のタイミングで変数及び関数単位でのセキュリティ属性及び認証鍵の設定が可能である、請求項１に記載の情報処理装置。
【請求項９】
前記セキュリティ属性の設定により、関数を定義する情報の読み出し、変更がさらに制限される、請求項１に記載の情報処理装置。
【請求項１０】
少なくとも、前記プログラム実行部が耐タンパ性能を有する、請求項１に記載の情報処理装置。
【請求項１１】
前記情報処理装置はＩＣチップを内蔵したＩＣカードである、請求項１に記載の情報処理装置。
【請求項１２】
耐タンパ性能を有する環境において、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行ステップを含み、
前記プログラム実行ステップで実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、
前記プログラム実行ステップは、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、情報処理方法。
【請求項１３】
耐タンパ性能を有する環境において、コンピュータに、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行ステップを実行させ、
前記プログラム実行ステップで実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、
前記プログラム実行ステップは、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属

性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、コンピュータプログラム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、情報処理装置、情報処理方法及びコンピュータプログラムに関する。

【背景技術】
【０００２】
近年、ＩＣ（ｉｎｔｅｇｒａｔｅｄ　ｃｉｒｃｕｉｔ）を備え、非接触で通信を行える非接触アンテナモジュールを備えるカードが普及している。そのようなカードは、例えば、非接触ＩＣカードなどと称され、非接触で他の装置と通信を行うことが可能とされている。非接触ＩＣカードを用いた非接触通信は、例えば、交通乗車券、電子マネー、ＩＤカード、入退室管理などに用いられ、その用途は広がりつつある。
【０００３】
ＩＣチップ上でプログラムの実行が可能な非接触ＩＣカードには、それぞれの用途に適したＯＳ（ｏｐｅｒａｔｉｎｇ　ｓｙｓｔｅｍ）を備える構成となっている。かかる非接触ＩＣカードに組み込まれているＯＳとして、例えばＭＵＬＴＯＳ（Ｍｅｌ言語）やＪａｖａ　ＯＳ（Ｊａｖａ（商標）言語）等がある。これらのＯＳは、仮想マシン（ＶＭ：Ｖｉｒｔｕａｌ　Ｍａｃｈｉｎｅ）をＲＯＭ上に格納し、実行コードを不揮発性メモリに書き込むことで実現される。
【先行技術文献】
【特許文献】
【０００４】
【特許文献１】特開２００１－１８４４７２号公報
【発明の概要】
【発明が解決しようとする課題】
【０００５】
しかし、これらのＩＣカードに組み込まれるＯＳは、ファイルシステムに操作可能なＡＰＩを呼ぶプログラムをダウンロードして定義しているものであり、自らはコンパイルやデバッグを行う機能を有しておらず、また、ＯＳの利用中に動的に機能の再定義は行えない。
【０００６】
ＭＵＬＴＯＳやＪａｖａ　ＯＳは、セキュリティファイアーウォールとしてコンテナを定義し、そのコンテナの中に各サービス用のプログラムをダウンロードでき、そのコンテナを認証することでプログラムの実行が可能になるが、そのプログラム中の個々の変数や関数の保護は、そのプログラムの書き方に依存してしまう。
【０００７】
そして、ＩＣカード用のアプリケーションのプログラムを開発する際には、複数の開発ツールが用いられるため、そのツールの理解が必要となり、そのセットアップなどに時間がかかっていた。また、ＩＣカード用のアプリケーションのプログラムの開発プロセスにおいては、上記開発ツールを用いてコードの作成及びデバッグを行い、最終的にＩＣカードにプログラムをダウンロードして、ＩＣカード上で最終デバッグを行うことが必須である。従って、開発プロセスが複雑となり、かつ、ＩＣカード上での最終デバッグで手戻りが発生すると、プログラムの修正に時間がかかるという問題が生じていた。
【０００８】
また、アプリケーションプログラムをＩＣカードにダウンロードした後は、セキュリティ機能を利用するため、暗号化したバイナリデータをやり取りする必要が生じ、専用の解

釈ツールを用いることが必須となっていた。

【０００９】

ＩＣカードへアプリケーションプログラムを供給する方法として、例えば特許文献１が開示されているが、特許文献１に開示されている技術は、汎用性を増してアプリケーションプログラム（ゲーム）を入れ替え可能にする点は記載されているが、セキュリティ機能の利用については言及が無い。

【００１０】

そこで、本発明は、上記問題に鑑みてなされたものであり、本発明の目的とするところは、緻密なセキュリティを要するアプリケーションプログラムの開発及び実装を容易に行えることが可能な、新規かつ改良された情報処理装置、情報処理方法及びコンピュータプログラムを提供することにある。

【課題を解決するための手段】

【００１１】

上記課題を解決するために、本発明のある観点によれば、耐タンパ性能を有する環境において、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行部を備え、前記プログラム実行部で実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、前記プログラム実行部は、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、情報処理装置が提供される。

【００１４】

前記セキュリティ属性には、前記変数または関数が規定されるデータ構造へのアクセス属性、認証鍵バージョン及び認証鍵テーブルへのポインタが設定されるようにしてもよい。

【００１５】

前記プログラム実行部で実行される前記コンピュータプログラムで定義された関数及び変数はシステム鍵で認証することにより定義の変更が可能であるようにしてもよい。

【００１６】

前記関数を定義する関数式をデータ構造に規定する場合は、該データ構造及び該関数内で利用する関数、変数の全ての認証鍵で認証されていることを要していてもよい。

【００１７】

外部装置との間で伝送されるバイナリデータを所定の形式にエンコードするバイナリデータ変換部を備えていてもよい。

【００１８】

前記バイナリデータ変換部は、前記認証処理のために伝送されるバイナリデータを変換した後のデータの先頭に前記所定の形式で用いられない符号を付加するようにしてもよい。

【００１９】

不揮発性を有し、前記プログラム実行部で実行されるプログラムで使用される変数定義及び関数定義を保存する記憶部を備えていてもよい。

【００２０】

前記プログラム実行部が実行するコンピュータプログラムは任意のタイミングで変数及

び関数単位でのセキュリティ属性及び認証鍵の設定が可能であってもよい。
【００２３】
前記セキュリティ属性の設定により、関数を定義する情報の読み出し、変更がさらに制限されるようにしてもよい。
【００２４】
少なくとも、前記プログラム実行部が耐タンパ性能を有していてもよい。
【００２５】
前記情報処理装置はＩＣチップを内蔵したＩＣカードであってもよい。
【００２６】
また、上記課題を解決するために、本発明の別の観点によれば、耐タンパ性能を有する環境において、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行ステップを含み、前記プログラム実行ステップで実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、前記プログラム実行ステップは、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、情報処理方法が提供される。

【００２８】
また、上記課題を解決するために、本発明の別の観点によれば、耐タンパ性能を有する環境において、コンピュータに、手続き型言語により作成されたコンピュータプログラムのコードを解釈して実行するプログラム実行ステップを実行させ、前記プログラム実行ステップで実行されるコンピュータプログラムには関数単位でセキュリティ属性及び認証鍵が設けられており、さらに変数単位でセキュリティ属性及び認証鍵が設けられており、前記プログラム実行ステップは、前記変数の参照及び前記関数の実行に際して、前記認証鍵による認証処理を実行し、前記セキュリティ属性に基づいて前記変数の参照及び前記関数の実行を可能とし、前記コンピュータプログラムに定義された関数にセキュリティ属性が付加されている場合は、前記関数に対して指定された認証鍵で認証することにより、該関数の内部で利用されている他の関数や変数のセキュリティ属性設定に基づく認証を経ずに該関数を実行できる、コンピュータプログラムが提供される。

【発明の効果】
【００３０】
以上説明したように本発明によれば、緻密なセキュリティを要するアプリケーションプログラムの開発及び実装を容易に行え、また、ネットワークを介したクライアントＰＣより新たなアプリケーションを安全に組み込むことが可能な、新規かつ改良された情報処理装置、情報処理方法及びコンピュータプログラムを提供することにある。
【図面の簡単な説明】
【００３１】
【図１】本発明の一実施形態にかかる情報処理システム１の構成例を示す説明図である。
【図２】本発明の一実施形態にかかる情報処理装置１００のハードウェア構成を示す説明図である。
【図３】リスト処理モジュールが定義できる、シンボルと呼ぶデータ構造を示す説明図である。
【図４】リスト構造を構成するためのコンスセル４１０の構成例を示す説明図である。
【図５】シンボル４００の名前領域４０１に格納される名前を格納するための名前格納テーブル４２０の構造例を示す説明図である。

【図６】認証鍵を格納する認証鍵テーブル４３０の構造例を示す説明図である。
【図７】図３に示したシンボルと、図４に示したコンスセルと、図５に示した名前格納テーブルと、図６に示した認証鍵テーブルとの対応関係を示す説明図である。
【図８】ＣＰＵ１２０が実行するリスト処理モジュールの起動時の処理について示す流れ図である。
【図９】ＣＰＵ１２０が実行するリスト処理モジュールのモード遷移について示す説明図である。
【図１０】シンボルの登録シーケンスとセキュリティ機能の活性化シーケンスについて示す流れ図である。
【図１１】シンボルの登録シーケンスとセキュリティ機能の活性化シーケンスについて示す流れ図である。
【図１２】セキュリティ設定された変数のリスト構造例を示す説明図である。
【図１３】セキュリティ設定された関数のリスト構造例を示す説明図である。
【図１４】情報処理装置の変形例である情報処理装置１１００のハードウェア構成を示す説明図である。
【図１５】従来のアプリケーション開発モデルについて示す説明図である。
【図１６】本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルについて示す説明図である。
【図１７】本発明の一実施形態に係る開発装置３００のハードウェア構成を説明するためのブロック図である。
【発明を実施するための形態】
【００３２】
以下に添付図面を参照しながら、本発明の好適な実施の形態について詳細に説明する。なお、本明細書及び図面において、実質的に同一の機能構成を有する構成要素については、同一の符号を付することにより重複説明を省略する。
【００３３】
なお、説明は以下の順序で行うものとする。
＜１．本発明の一実施形態＞
［１－１．システム構成例］
［１－２．ハードウェア構成例］
［１－３．プログラム構造例］
［１－４．情報処理装置の変形例］
［１－５．アプリケーション開発モデルの比較］
［１－６．開発装置のハードウェア構成］
＜２．まとめ＞
【００３４】
＜１．本発明の一実施形態＞
［１－１．システム構成例］
まず、本発明の一実施形態にかかるシステム構成例について説明する。図１は、本発明の一実施形態にかかる情報処理システム１の構成例を示す説明図である。以下、図１を用いて本発明の一実施形態にかかる情報処理システム１の構成例について説明する。
【００３５】
図１に示したように、本発明の一実施形態にかかる情報処理システム１は、情報処理装置１００と、リーダライタ（Ｒ／Ｗ）２００と、開発装置３００と、を含んで構成される。
【００３６】
情報処理装置１００は、内部に耐タンパ性を有し、近接非接触通信機能を備えるＩＣチップが内蔵されている。情報処理装置１００は、例えば、かかるＩＣチップが内蔵されたＩＣカードや、携帯電話、携帯端末等の形態をとることができる。そして、かかるＩＣチップは、リーダライタ２００との間で近接非接触通信を実行することで、リーダライタ２

００との間で情報の授受を実行することができる。
【００３７】
リーダライタ２００は、主に駅の自動改札機やキャッシュレジスター等に接続して設けられるものであり、ＩＣチップが内蔵された情報処理装置１００との間で近接非接触通信を実行するものである。リーダライタ２００が、情報処理装置１００との間で近接非接触通信を実行することで、例えばリーダライタ２００と接続されている自動改札機がゲートを開いたり、リーダライタ２００と接続されているキャッシュレジスターが商品代金の支払い処理を行ったりすることができる。
【００３８】
開発装置３００は、情報処理装置１００に内蔵されるＩＣチップに組み込まれるコンピュータプログラムの開発に用いられる装置である。ＩＣチップに組み込むためのコンピュータプログラムを作成するユーザは、開発装置３００を用いて、ＩＣチップに組み込んで実行させるためのコンピュータプログラムのソースコードの作成を行うことができる。開発装置３００で作成されたコンピュータプログラムのソースコードは、開発装置３００に接続したリーダライタ２００を介して情報処理装置１００に転送し、情報処理装置１００に内蔵されたＩＣチップに組み込まれる。情報処理装置１００は、ＩＣチップに組み込まれたコンピュータプログラムのソースコードを解釈して、当該コンピュータプログラムを実行することができる。
【００３９】
本実施形態にかかる情報処理装置１００に内蔵されるＩＣチップは、ソースコードで定義されている変数及び関数にそれぞれ独立してセキュリティが設定されるプログラムを実行することが出来るように構成される。これにより、本実施形態にかかる情報処理装置１００に内蔵されるＩＣチップは、組み込まれているプログラムの実行権限のないアプリケーション等による当該プログラムの実行を防ぐことができるので、プログラムのセキュアな実行が可能になる。
【００４０】
なお、図１では、開発装置３００で作成されたコンピュータプログラムのソースコードは、開発装置３００に接続したリーダライタ２００を介して情報処理装置１００に転送する構成が図示されているが、本発明においては、開発装置から情報処理装置にコンピュータプログラムを組み込むには必ずしもリーダライタを介する必要は無く、開発装置から直接情報処理装置にコンピュータプログラムを転送してもよい。
【００４１】
以上、図１を用いて本発明の一実施形態にかかる情報処理システム１の構成例について説明した。次に、本発明の一実施形態にかかる情報処理装置１００のハードウェア構成例について説明する。
【００４２】
［１－２．ハードウェア構成例］
図２は、本発明の一実施形態にかかる情報処理装置１００のハードウェア構成を示す説明図である。以下、図２を用いて、本発明の一実施形態にかかる情報処理装置１００のハードウェア構成について説明する。
【００４３】
図２に示した情報処理装置１００は、例えばＩＣカードに内蔵されるチップのように全体が耐タンパ性を有するハードウェア構造となっているものである。図１に示したように、本発明の一実施形態にかかる情報処理装置１００は、ＮＶＭ（不揮発性メモリ、Ｎｏｎ Ｖｏｌａｔｉｌｅ Ｍｅｍｏｒｙ）１１０と、ＣＰＵ（Ｃｅｎｔｒａｌ Ｐｒｏｃｅｓｓｉｎｇ Ｕｎｉｔ）１２０と、ＲＯＭ（Ｒｅａｄ Ｏｎｌｙ Ｍｅｍｏｒｙ）１３０と、ＲＡＭ（Ｒａｎｄｏｍ Ａｃｃｅｓｓ Ｍｅｍｏｒｙ）１４０と、ＢＡＳＥ６４モジュール１５０と、暗号処理モジュール１６０と、乱数発生モジュール１７０と、シリアルＩ／Ｏインタフェース１８０と、を含んで構成される。
【００４４】

ＮＶＭ１１０は、情報処理装置１００の初期化時に、ＲＯＭ１３０に予め書きこまれている組み込み関数が、シンボルとして記録されるものである。また、ＮＶＭ１１０は、ユーザが定義した変数（ユーザ定義変数）や関数（ユーザ定義関数）も記憶されるものである。ＮＶＭ１１０は、電源がオフになっても記憶された情報を保持しておくことができるので、情報処理装置１００の電源が再投入されても再初期化は行われず、登録されているシンボルがそのまま保持される。

【００４５】

ＣＰＵ１２０は、情報処理装置１００の動作を制御するものであり、ＲＯＭ１３０に予め記録されているオペレーティングシステムソフトの読み出し命令を実行することで、当該オペレーティングシステムを実行することができる。ＣＰＵ１２０は、オペレーティングシステムの実行に際しては、ＲＡＭ１４０をワークエリアとして用いることができる。ここで、ＲＯＭ１３０に記録されているオペレーティングシステムソフトとしては、例えば、手続き型プログラミング言語を解釈して実行できるものであり、そのようなプログラミング言語として、例えばＬＩＳＰやＲｕｂｙ、Ｐｙｔｈｏｎ等がある。

【００４６】

本実施形態にかかる情報処理装置１００のＲＯＭ１３０は、インタプリタとして上記手続き型プログラミング言語の基本機能に加え、セキュリティ機能が付加されたものが格納される。これにより、情報処理装置１００にアプリケーションプログラムを組み込む際に、事前にコンパイルする必要がなく、またセキュリティ機能が付加されているので、アプリケーションが利用される情報処理装置１００そのものでデバッグが可能となる。従って、開発ステップの短縮に繋がって、短期間でのアプリケーションプログラムの開発が可能となる。

【００４７】

ＢＡＳＥ６４モジュール１５０は、情報処理装置１００の外部に設けられた装置（以下単に「外部装置」とも称する）との通信のために、バイナリデータに対してＢＡＳＥ６４変換を施したり、またその逆の変換を施したりするモジュールである。本実施形態にかかる情報処理装置１００は、外部装置との通信は、シリアルＩ／Ｏインタフェース１８０を介したシリアルデータ通信を実行する。本実施形態では、情報処理装置１００の外部からリスト処理モジュールと通信する場合、入力されるデータは全てＡＳＣＩＩ文字列で行えるよう、バイナリデータはＢＡＳＥ６４モジュール１５０でＢＡＳＥ６４コードに変換される。

【００４８】

情報処理装置１００と外部装置（例えばリーダライタ２００）とで相互認証及びその後の通信を行う場合、やり取りされるバイナリ形式の暗号化データはＢＡＳＥ６４モジュール１５０でＢＡＳＥ６４変換されるが、その先頭にはＢＡＳＥ６４では用いない符号（例えば‘：’や’～’等）を付けることで、一般のリスト処理式とは識別してやり取りするようにしてもよい。

【００４９】

シリアルＩ／Ｏインタフェース１８０を介したシリアルデータ通信は、キャリッジリターン符号が入力の区切りとなる。情報処理装置１００は、外部装置との通信に用いるコードとしてＡＳＣＩＩコードを用いて外部装置との間で情報の授受を実行する。しかし、情報処理装置１００は、暗号化データなどのバイナリデータのやり取りに際しては、バイナリデータをＢＡＳＥ６４モジュール１５０でＢＡＳＥ６４変換する。そして、ＢＡＳＥ６４モジュール１５０は、バイナリデータをＢＡＳＥ６４変換する際には、上述したように、先頭にＢＡＳＥ６４では使用しない文字（符号）を識別文字として付加することができる。ＢＡＳＥ６４では使用しない文字（符号）を識別文字として先頭に付加することにより、ＣＰＵ１２０でプログラムを実行する際に、ＢＡＳＥ６４モジュール１５０で変換されたデータがバイナリデータであることをＣＰＵ１２０が認識することができる。

【００５０】

また、ＣＰＵ１２０で実行されるプログラムにロックを掛ける際に、そのロックに用い

る鍵の設定等のシンボル式とバイナリデータとが混在するような場合には、シンボル式とバイナリデータとで異なる文字（符号）を割り当てることで、ＣＰＵ１２０でプログラムを実行する際にＣＰＵ１２０がそのシンボル式とバイナリデータとの違いを認識することができる。

【００５１】

暗号処理モジュール１６０は、入力されてくるデータに対して、指定された鍵を用いて暗号化処理を施して出力したり、入力されてくる暗号化データに対して、指定された鍵を用いて復号処理を施して出力したりするものである。

【００５２】

乱数発生モジュール１７０は、外部からの（例えばＣＰＵ１２０からの）乱数発生指示に基づいて、適当な乱数を発生させてその乱数を出力するものである。乱数発生モジュール１７０が発生する乱数は、例えば、情報処理装置１００から認証したい相手（例えばリーダライタ２００）に送信して暗号文を生成してもらい、相手（例えばリーダライタ２００）から返信された暗号文を復号した結果が情報処理装置１００から送った乱数と一致するか否かを判定することで相手の鍵が正しいか否かを判断するために用いられる。

【００５３】

シリアルＩ／Ｏインタフェース１８０は、外部装置からのシリアルデータをパケットとして認識して、適切なデータを抽出したり、また、情報処理装置１００の内部から外部装置へ出力されるデータをパケットデータとして構成し、シリアルデータとして出力したりする機能を有するものである。

【００５４】

図２に示すような構成を有する情報処理装置１００が、全体として耐タンパ性を有することにより、機器や回路の中身が外部からは分析しにくくなる。そして、ＣＰＵ１２０が実行するコンピュータプログラムには、変数・関数それぞれに対して独立してセキュリティが設定される。

【００５５】

なお、図２には図示しないが、情報処理装置１００は、リーダライタ２００との間で近接非接触通信を実行するためのアンテナや変調・復調回路等が備えられる。

【００５６】

以上、図２を用いて、本発明の一実施形態にかかる情報処理装置１００のハードウェア構成について説明した。次に、図２に示した情報処理装置１００で実行される、コンピュータプログラムの構造について説明する。

【００５７】

［１－３．プログラム構造例］

図３～図６は、本発明の一実施形態にかかる情報処理装置１００で実行される、コンピュータプログラムの構造例を示す説明図である。以下、図３～図６を用いて、本発明の一実施形態にかかる情報処理装置１００で実行される、コンピュータプログラムの構造例について説明する。

【００５８】

なお、以下においては、情報処理装置１００で実行されるコンピュータプログラムの言語としてＬＩＳＰを前提として説明するが、本発明においては、プログラミング言語として使用可能である言語はかかる例に限定されず、拡張機能または標準機能において、変数毎及び関数毎にそれぞれ独立してセキュリティが設定できるように構成することが可能な、手続き型のプログラミング言語であればいかなるものであっても良い。

【００５９】

ＣＰＵ１２０は、コンピュータプログラムの実行に際しては、開発装置３００で開発され、情報処理装置１００に組み込まれたプログラムのソースコードを解釈して実行するためのリスト処理モジュールをロードする。図３は、リスト処理モジュールが定義できる、シンボルと呼ぶデータ構造を示す説明図である。

【００６０】

図３に示したように、リスト処理モジュールが定義できるシンボル４００は、名前領域４０１と、変数定義領域４０２と、関数定義領域４０３と、セキュリティ属性領域４０４と、で構成される。

【００６１】

名前領域４０１は、印刷可能な文字デーブルを指し示すものである。名前領域４０１には、そのシンボルが変数を規定するものであればその変数名が格納され、関数を規定するものであればその関数名が格納されるものである。図３では、名前領域４０１を「ｐｎａｍｅ」で指し示している。

【００６２】

変数定義領域４０２は、そのシンボルが単純変数を規定するものであればその値を格納し、リスト変数を規定するものであればそのリストを指し示す値が格納されるものである。図３では、変数定義領域４０２を「ｖａｌｕｅ」で指し示している。

【００６３】

関数定義領域４０３は、そのシンボルが関数を規定するものであれば、その関数実態が格納されるものである。図３では、関数定義領域４０３を「ｆｕｎｃｔｉｏｎ」で指し示している。

【００６４】

セキュリティ属性領域４０４は、そのシンボルについてのセキュリティ属性に関する情報が格納されるものである。セキュリティ属性としては、例えば変数の読み出し属性、変数の変更属性、関数の実行属性がある。セキュリティ属性領域４０４には、そのシンボルへのアクセス権限を示すアクセスフラグと、そのシンボルへアクセスするための認証鍵を格納するテーブルを指し示す値とが格納される。

【００６５】

図３に示したシンボル４００に加えて、リスト構造を構成するためのコンスセルと呼ぶセルが連続的に定義されている。図４は、リスト構造を構成するためのコンスセル４１０の構成例を示す説明図である。コンスセル４１０は、図４に示したように、ＣＡＲスロット４１１およびＣＤＲスロット４１２と呼ばれる２つのポインタからなるオブジェクトである。図４では、ＣＡＲスロット４１１としてｃａｒ０～ｃａｒ９、ＣＤＲスロット４１２としてｃｄｒ０～ｃｄｒ９を示している。もちろん、それぞれのスロットの数はかかる例に限定されないことは言うまでもない。

【００６６】

シンボル４００の名前領域４０１に格納される名前を格納するためのテーブルも設けられる。図５はシンボル４００の名前領域４０１に格納される名前を格納するための名前格納テーブル４２０の構造例を示す説明図である。図５に示した名前格納テーブル４２０には、「ｅｖａｌ」、「ｓｅｔｑ」、「ｃｏｎｓ」、「ｄｅｆｕｎ」、「ｏｓａｉｆｕ」といった名称が格納されており、その実態であるシンボルと１対１に対応している。符号４２１は名称「ｅｖａｌ」が格納される領域であり、符号４２２は名称「ｓｅｔｑ」が格納される領域であり、符号４２３は名称「ｃｏｎｓ」が格納される領域であり、符号４２４は名称「ｄｅｆｕｎ」が格納される領域であり、符号４２５は名称「ｏｓａｉｆｕ」が格納される領域である。名前格納テーブル４２０の外部からシンボル名が名前格納テーブル４２０に対して入力されると、名前格納テーブル４２０に格納されているその入力されたシンボル名に対するシンボルが指し示されて評価される。なお、「ｏｓａｉｆｕ」は、情報処理装置１００に電子マネー機能を組み込む際にその電子マネーの残高を表す変数であるとする。

【００６７】

そして、シンボル４００のセキュリティ属性領域４０４に格納される、認証鍵を格納するテーブルを指し示す値に対応するテーブルも設けられる。図６は、認証鍵を格納する認証鍵テーブル４３０の構造例を示す説明図である。図６は、認証鍵テーブル４３０において、認証鍵がバージョン番号（ｋｖ１～ｋｖ５）で管理されている状態を示すものである。符号４３１は鍵「ｋｅｙ１」が格納される領域であり、符号４３２は鍵「ｋｅｙ２」が

格納される領域であり、符号４３３は鍵「ｋｅｙ３」が格納される領域であり、符号４３４は鍵「ｋｅｙ４」が格納される領域であり、符号４３５は鍵「ｋｅｙ５」が格納される領域である。

【００６８】

図３から図６に示したこれらのテーブルは、開発装置３００で開発されたコンピュータプログラムの、情報処理装置１００での実行に先立って、ＮＶＭ１１０に生成されて記憶される。これにより、情報処理装置１００の電源がオフになっても、これらのテーブルの内容は保持されたままの状態を維持することができる。

【００６９】

図７は、図３に示したシンボルと、図４に示したコンスセルと、図５に示した名前格納テーブルと、図６に示した認証鍵テーブルとの対応関係を示す説明図である。上述したように、シンボルは、印刷可能な名前のテーブルを指し示す領域と、値若しくは値のリストを指し示す領域と、関数属性と、セキュリティ属性とを持つ。関数属性は、関数のタイプ及びその関数の実態を指し示すポインタを持ち、セキュリティ属性は、セキュリティフラグと、鍵バージョンと、鍵を指し示すポインタとを持つ。なお、図７では、図６に示した認証鍵テーブル４３０において、符号４３１で示した鍵「ｋｅｙ１」及び符号４３２で示した鍵「ｋｅｙ２」が使用される様子が図示されている。

【００７０】

図３に示したシンボルと、図４に示したコンスセルと、図５に示した名前格納テーブルと、図６に示した認証鍵テーブルとは、いずれも図２に示した情報処理装置１００のような、耐タンパ機能を持ったハードウェアで守られることで、値の不正な取得や改ざんから守られることになる。

【００７１】

このように、リスト処理モジュールの一般的構造はシンボルと呼ばれ、数値若しくは数値を保持したリストへのポインタと、関数定義であれば関数へのポインタと、印刷可能な文字列を格納したテーブルを指し示すポインタと、で構成されている

【００７２】

そして本実施形態では、これに加えて、セキュリティ属性と２種類の暗号化鍵情報を保持するテーブルへのポインタをシンボルに付加している。一方の鍵へのポインタはマスター鍵を指し、もう一方の鍵へのポインタは、当該シンボルのアクセス鍵（認証鍵）を指している。マスター鍵は当該シンボルのセキュリティ属性やアクセス鍵を変更する場合に、前もって相互認証機能により認証されているべき鍵を指している。シンボルに保持した情報の内容評価、内容変更、関数実行においてそのシンボルに設定されたセキュリティフラグが立っている場合は、シンボルに付加された一方の鍵で認証してあることがそのシンボル利用の条件となる。もう一方の鍵は、当該シンボルの鍵を変更する場合に、その権限を確認する権限認証鍵を示している。アクセス情報の変更には、その権限認証鍵で認証された状態でなければならない。

【００７３】

また、図４に示したような、コンスセルと呼ばれる、シンボルとシンボルの関係を表す２組のポインタがあり。それぞれのポインタは、シンボル又は他のシンボルを指すコンスセルを指し示す構造となっている。

【００７４】

組み込み関数はＲＯＭ１３０に書き込まれており、情報処理装置１００の最初の電源投入で、ＲＯＭ１３０に書き込まれている組み込み関数を、ＮＶＭ１１０に作成したシンボルに定義する。その後の電源投入では、既に登録済みのシンボルは初期化されない。

【００７５】

上記の構成は、ユーザにより新たな関数が登録される場合においても、同様に機能する構造となっている。

【００７６】

ＣＰＵ１２０が実行するリスト処理モジュールは、シンボルを自由に登録し、そのシン

ボルに数値やリスト、関数を自由に登録できる構成となっている。そして、登録されるシンボルにセキュリティ機能を生かすために、そのシンボルに暗号鍵とアクセスフラグを登録しておく。ＣＰＵ１２０が実行するリスト処理モジュールには、システム鍵と呼ぶ暗号化鍵が最初に設定されている。そのシステム鍵で相互認証されたモードになっている状態（以下の説明のモード２に該当する状態）でのみ、新たに登録されたシンボルは、そのシンボル独自の鍵とアクセスフラグが設定されることができる。また、ＣＰＵ１２０が実行するリスト処理モジュールで実行されるコンピュータプログラムは、システム鍵で相互認証されたモードになっている状態に限り、使用される変数や関数の定義を変更可能な構成となっている。

【００７７】

ＣＰＵ１２０が実行するリスト処理モジュールは、関数シンボルの登録に際しては、その関数で利用するシンボルの全ての鍵で認証済みであることが登録条件となっている。そしてその後、リスト処理モジュールは、その登録された関数を利用する場合は、単にその関数実行鍵で認証されていれば良い構造になっている。

【００７８】

次に、ＣＰＵ１２０が実行するリスト処理モジュールの起動時の処理、及びリスト処理モジュールのモード遷移について説明する。図８は、ＣＰＵ１２０が実行するリスト処理モジュールの起動時の処理について示す流れ図である。また図９は、ＣＰＵ１２０が実行するリスト処理モジュールのモード遷移について示す説明図である。

【００７９】

まず、ＣＰＵ１２０がリスト処理モジュールを起動する時の処理について図８を参照しながら説明する。まず情報処理装置１００の電源がオンされると、リスト処理モジュールがＣＰＵ１２０にロードされる。そしてＣＰＵ１２０で実行されるリスト処理モジュールは、まず初期完了フラグがオンになっているかどうかを判断する（ステップＳ１０１）。

【００８０】

上記ステップＳ１０１の判断の結果、初期完了フラグがオンになっていなかった場合には、リスト処理モジュールは、ＲＯＭ１３０からＮＶＭ１１０へ関数及び変数を組み込み、シンボルを定義する。シンボルの定義が完了すると、リスト処理モジュールは初期化フラグをオンにする（ステップＳ１０２）。

【００８１】

上記ステップＳ１０２の処理が完了すると、または、上記ステップＳ１０１の判断の結果、初期完了フラグがオンになっていた場合には、続いてリスト処理モジュールは、情報処理装置１００の外部から入力される文字の読み込みと、シンボル生成と、リスト構造への翻訳とを実行する（ステップＳ１０３）。

【００８２】

上記ステップＳ１０３の処理が完了すると、続いてリスト処理モジュールは、リスト式を評価し、その結果をリスト構造とする（ステップＳ１０４）。そしてリスト処理モジュールは、リスト式の評価結果がエラーでなく、シンボル定義ならば、そのシンボル定義をＮＶＭ１１０に書き込む（ステップＳ１０５）。そしてリスト処理モジュールは、リスト構造結果のリスト式化と出力を行う（ステップＳ１０６）。

【００８３】

上記ステップＳ１０６で、リスト構造結果のリスト式化と出力を行うと、リスト処理モジュールは、上記ステップＳ１０３に戻って、情報処理装置１００の外部から入力される文字の読み込みと、シンボル生成と、リスト構造への翻訳とを実行する。この一連の流れによって、ＣＰＵ１２０が実行するリスト処理モジュールは、ＮＶＭ１１０に組み込まれた関数や変数を用いて、開発装置３００で作成されたコンピュータプログラムを実行することができる。

【００８４】

続いて、ＣＰＵ１２０が実行するリスト処理モジュールのモード遷移について、図９を参照しながら説明する。図９に示すように、ＣＰＵ１２０が実行するリスト処理モジュー

ルは、モード０、モード１、モード２を遷移しながら動作する。
【００８５】
モード０は、ロックされていないシンボルを利用した変数の参照及び変更、並びに関数の実行が可能なモードであり、通信は平文で行われる。
【００８６】
モード１は、モード０から下記モード２へ移行する途中の段階、すなわち認証途中の段階であり、リスト処理モジュールが通信相手に認証された状態である。このモードでは、リスト処理モジュールと通信相手とが未だ相互認証状態までは至っておらず、相互認証状態となるには、リスト処理モジュールが通信相手を認証する必要がある。なお、リスト処理モジュールを通信相手に認証してもらうための式として図９では「ａｕｔｈ１」式を用いている。
【００８７】
そしてモード２は、システム鍵と呼ぶ予め設定された暗号化鍵でリスト処理モジュールと通信相手とが相互認証された状態を示すモードであり、認証され、ロックが解除されたシンボルの認証フラグに応じて、変数の参照と関数の実行が可能であり、かつ、モード０の機能も動作可能である。また、ＣＰＵ１２０が実行するリスト処理モジュールで実行されるコンピュータプログラムは、システム鍵で相互認証されたモードになっている状態に限り、使用される変数や関数の定義を変更可能な構成となっている。通信はセッション鍵を用いた暗号文となるが、モード０の平文も受け付けることができる。なお、リスト処理モジュールが通信相手を認証するための式として図９では「ａｕｔｈ２」式を用いている。
【００８８】
モード２の状態で、リスト処理モジュールはリセット動作を実行すると、モード０に戻り、ロックされていないシンボルを利用した変数の参照及び変更、並びに関数の実行が可能な状態に戻る。なお、モード２からモード０へ遷移するための式として図９では「ｒｅｓｅｔ」式を用いている。
【００８９】
このように、モード０、モード１、モード２の３つのモードを遷移しながら動作することで、ＣＰＵ１２０がリスト処理モジュールは、ロックされていないシンボルの実行及びロックされているシンボルを解除した実行ができる。
【００９０】
以上、ＣＰＵ１２０が実行するリスト処理モジュールのモード遷移について、図９を参照しながら説明した。次に、本発明の一実施形態にかかる情報処理装置１００における、シンボルの登録シーケンスとセキュリティ機能の活性化シーケンスについて説明する。図１０及び図１１は、本発明の一実施形態にかかる情報処理装置１００における、シンボルの登録シーケンスとセキュリティ機能の活性化シーケンスについて示す流れ図である。
【００９１】
まず、図１０を用いて、アプリケーションの開発時におけるシンボルの登録シーケンスについて説明する。最初に、開発装置３００を用いたアプリケーションの開発が行われ、その開発の際に、当該アプリケーションで用いられる変数及び関数の登録が行われる（ステップＳ１１１）。アプリケーションで用いられる変数及び関数の登録が行われると、開発装置３００は、登録された変数及び関数が正常に機能するかどうかデバッグ処理を実行する（ステップＳ１１２）。
【００９２】
上記ステップＳ１１２におけるデバッグ処理によって、アプリケーションに問題が無くなれば、そのアプリケーションは開発装置３００から情報処理装置１００に組み込まれる。アプリケーションが情報処理装置１００に組み込まれる際には、まず、リスト処理モジュールにシステム鍵で相互認証を実行させる（ステップＳ１１３）。上述したように、システム鍵で相互認証され、モード２の状態になっている場合のみ、シンボルに対してシンボル独自の鍵及びアクセスフラグを設定することができる。

【００９３】
システム鍵による相互認証が完了すると、開発装置３００から情報処理装置１００へアプリケーションを組み込み、関数及び変数のシンボルに対して、必要なセキュリティ属性を設定する（ステップＳ１１４）。これにより、アプリケーションで用いられる関数や変数のシンボルに対して鍵をかけることができ、関数の実行権限や変数の参照権限の有る者に対してのみ、その関数の実行や変数の参照、値の変更を行わせることが可能になる。
【００９４】
次に、開発装置３００で開発したアプリケーションを情報処理装置１００に組み込んで、情報処理装置１００を発行する場合の流れについて、図１１を用いて説明する。
【００９５】
開発装置３００で開発したアプリケーションを情報処理装置１００に組み込んで、情報処理装置１００を発行する場合は、まずリスト処理モジュールに、システム鍵による相互認証を実行させる（ステップＳ１２１）。システム鍵による相互認証が完了すると、次に、リスト処理モジュールに、開発装置３００で開発されたプログラムを読み込ませて、シンボルの定義、変数の登録、関数の登録を実行させる（ステップＳ１２２）。リスト処理モジュールは、上記ステップＳ１２１による相互認証を行わずにプログラムを読み込むことはできない。上記ステップＳ１２１による相互認証を行うことで、リスト処理モジュールはセキュリティが設定されたシンボルを定義し、変数の登録、関数の登録を実行することができ、その関数の実行や変数の参照、値の変更を実行することが出来る。
【００９６】
次に、本発明の一実施形態にかかる情報処理装置１００で実行されるリスト処理モジュールが参照できる、セキュリティ設定された変数のリスト構造例について説明する。図１２は、セキュリティ設定された変数のリスト構造例を示す説明図である。
【００９７】
システム鍵で認証した後にシンボルを生成し、セキュリティフラグ、鍵バージョン及び鍵を設定する。セキュリティ属性（Ｓ－ｆｌａｇ）は３種類指定することができる。図１２では、Ｘが関数実行ロックを、Ｍが内容更新ロックを、Ｅが内容読み出しロックを、それぞれ示している。関数実行ロックは、認証鍵で認証が取れていなければ関数が実行できないことを現し、内容更新ロックは、認証鍵で認証が取れていなければ変数の値を更新できないことを現し、内容読み出しロックは、認証鍵で認証が取れていなければ変数の値を参照できないことを現す。図１２は、２つの変数「ｏｓａｉｆｕ」「ｌｏｇ」のリスト構造例を示したものである。以下では、この２つの変数「ｏｓａｉｆｕ」「ｌｏｇ」のリスト構造について説明する。
【００９８】
図１２に示した変数「ｏｓａｉｆｕ」は、電子マネー残高を格納するための変数である。変数「ｏｓａｉｆｕ」は、変数名（ｐｎａｍｅ）として「ｏｓａｉｆｕ」が指定され、電子マネー残高が格納される値（ｖａｌｕｅ）には初期値として０が指定されている。そして変数「ｏｓａｉｆｕ」は、セキュリティ属性として内容更新ロック（Ｍ）及び内容読み出しロック（Ｅ）が指定されている。これにより、変数「ｏｓａｉｆｕ」の値は、認証鍵で認証が取れていなければ内容の参照及び変更が出来なくなる。
【００９９】
そして、変数「ｏｓａｉｆｕ」には、マスター鍵へのポインタ及び変数「ｏｓａｉｆｕ」にアクセスするためのアクセス鍵のバージョン並びにアクセス鍵へのポインタの情報が格納される。上述したように、マスター鍵は、当該シンボル（ここでは変数「ｏｓａｉｆｕ」）のセキュリティ属性やアクセス鍵を変更する場合に、前もって相互認証機能により認証されているべき鍵のことである。図１２では、変数「ｏｓａｉｆｕ」にアクセスするためのアクセス鍵のバージョンが１であり、そのアクセス鍵へのポインタの情報が格納されている状態が示されている。
【０１００】
図１２に示した変数「ｌｏｇ」は、関数の実行結果を格納するための変数である。変数

「ｌｏｇ」は、変数名（ｐｎａｍｅ）として「ｌｏｇ」が指定されている。そして、関数の実行結果を格納するための値（ｖａｌｕｅ）はサイクリックな構成となっている。図１２に示した構成では、変数「ｌｏｇ」は、関数の実行結果が５回分格納される。なお、サイクリックファイルの生成はリスト処理機能に組み込みこまれた関数を利用して設定する。

【０１０１】

そして変数「ｌｏｇ」は、セキュリティ属性として内容更新ロック（Ｍ）及び内容読み出しロック（Ｅ）が指定されている。これにより、変数「ｌｏｇ」の値は、認証鍵で認証が取れていなければ内容の参照及び変更が出来なくなる。図１２では、変数「ｌｏｇ」にアクセスするためのアクセス鍵のバージョンが２であり、そのアクセス鍵へのポインタの情報が格納されている状態が示されている。

【０１０２】

また、図１２に示した「Ｓｙｓｔｅｍ」は、モジュールの元権限を持つものである。っこの「Ｓｙｓｔｅｍ」に鍵が設定されていれば、各関数や変数オブジェクトを生成するに当たって、前もって認証されていることが条件となる。一方、「Ｓｙｓｔｅｍ」に鍵が設定されていなければ、前もって認証されている必要はなく、例えば非接触型ＩＣカードの技術方式の一つであるＦｅｌｉｃａ（登録商標）における、システムコードおよびモジュール情報を保持する変数となる。

【０１０３】

次に、本発明の一実施形態にかかる情報処理装置１００で実行されるリスト処理モジュールが参照できる、セキュリティ設定された関数のリスト構造例について説明する。図１３は、セキュリティ設定された関数のリスト構造例を示す説明図である。

【０１０４】

図１３は、関数「ｃｈａｒｇｅ」のリスト構造例を示したものである。以下では、この関数「ｃｈａｒｇｅ」のリスト構造について説明する。

【０１０５】

関数「ｃｈａｒｇｅ」を定義するためのＳ式は図１３に示した通りである。ここで、Ｓ式はＬＩＳＰで用いられる論理記述方式であり、シンボルを定義するために用いられる。このＳ式により、関数「ｃｈａｒｇｅ」は、図１２に示した変数「ｏｓａｉｆｕ」の値に、引数ｘで指定された値を加算する処理を実行するものである。また、このＳ式は、図１２に示した、セキュリティロックがかかっている２つの変数「ｏｓａｉｆｕ」「ｌｏｇ」を使用している。

【０１０６】

図１３に示した「ｃｈａｒｇｅ＊」「ｏｓａｉｆｕ＊」「ｌｏｇ＊」は、セキュリティロックがかかっている関数及び変数であることを示している。従って、関数「ｃｈａｒｇｅ」を実行するには、セキュリティロックがかかっている関数及び変数の全てにおいて、認証済みであることが要求される。

【０１０７】

このように、関数「ｃｈａｒｇｅ」の実行にはセキュリティロックがかかっている２つの変数「ｏｓａｉｆｕ」「ｌｏｇ」についても認証されている必要がある。しかし、関数「ｃｈａｒｇｅ」を実行する権限を有するものは、権限の移譲により、変数「ｏｓａｉｆｕ」「ｌｏｇ」についても値の参照や変更を実行する権限を有しているものと考え、関数「ｃｈａｒｇｅ」に設定された鍵で認証することで、関数「ｃｈａｒｇｅ」の実行が可能であり、改めて変数「ｏｓａｉｆｕ」「ｌｏｇ」について認証することを要しないものとする。

【０１０８】

以上、セキュリティ設定された関数のリスト構造例について説明した。このように、本実施形態にかかる情報処理装置１００のＣＰＵ１２０が実行するリスト処理モジュールは、セキュリティロックがかけられている変数の参照及び関数の実行を可能とする。また、情報処理装置１００で実行されるリスト処理モジュールに参照される変数や実行される関

数は、それぞれ独自にセキュリティ属性を設定して、セキュリティロックをかけることができる。

【０１０９】

これにより、情報処理装置１００で実行されるリスト処理モジュールが実行するプログラムで使用される変数や関数は、それぞれ独自にセキュリティ設定が可能となり、これにより緻密なセキュリティ実装が可能となる。また、情報処理装置１００で実行されるリスト処理モジュールは、インタプリタの基本機能に加えてセキュリティ機能を付加していることで、事前にコンパイルする必要がなく、また、実際に利用される情報処理装置１００でデバッグを行うことが可能になる。また、ＮＶＭ１１０に動的に変数や関数を直接定義することが可能になり、ＮＶＭ１１０に動的に変数や関数を直接定義することで柔軟なプログラム開発が可能になる。また、セキュリティ属性のセットにより、関数実行制限のみならず、関数定義情報の読み出し、変更も制限され、外部からの攻撃による関数定義情報の漏洩や改ざんからプログラムを保護することができる。

【０１１０】

セキュリティロックがかかった変数や関数は、それぞれ、対応する認証鍵で認証することで初めて利用することが出来る。そして、関数においては、その関数が内部で利用している引数として使用する、セキュリティロックがかかった変数や関数は、その関数を定義する際に認証されている必要がある。しかし、関数定義後の当該関数の利用においては、内部で利用している引数として用いる変数や関数は無条件で利用可能としているので、関数実行の際に毎回全ての認証鍵を用意して認証する必要がなくなり、運用が容易になる。

【０１１１】

具体例を挙げて説明すれば、上記の変数「ｏｓａｉｆｕ」にセキュリティを設定すると、当該変数「ｏｓａｉｆｕ」については、変数「ｏｓａｉｆｕ」に指定された鍵を用いて相互認証された状態でなければ、当該変数「ｏｓａｉｆｕ」に格納された値の読み出し及び変更が行えない。

【０１１２】

一方、例えば、変数「ｏｓａｉｆｕ」に格納された値の変更を実行する関数「ｃｈａｒｇｅ」にセキュリティを設定すると、関数「ｃｈａｒｇｅ」に指定された鍵を用いて相互認証された状態でなければ、当該関数「ｃｈａｒｇｅ」を実行することは出来ない。この場合において、関数「ｃｈａｒｇｅ」の実行について認証されているが、変数「ｏｓａｉｆｕ」に格納された値の読み出し及び変更について認証されていなければ、変数「ｏｓａｉｆｕ」に格納された値の読み出し及び変更を直接実行することは出来ない。このように、変数や関数にそれぞれ独立してセキュリティを設定することで柔軟な運用を可能にするとともに、関数実行の際に、内部で使用されている全ての変数等についての認証鍵を用意して毎回認証する必要が無くなるので、運用が容易になる。

【０１１３】

また、情報処理装置１００で実行されるリスト処理モジュールが実行するプログラムで使用される変数や関数のセキュリティ機能は、任意の時点で行われるようにしてもよい。従って、開発装置３００を用いて開発されたプログラムに対する十分なデバッグが行われた後で、開発装置３００において変数や関数に対してセキュリティ機能を実装し、セキュリティ機能が実装された状態でさらなるデバッグが可能になる。

【０１１４】

暗号処理モジュール１６０で暗号化されたバイナリデータは、送信前に印刷可能な文字に変換してやり取りされる。これにより、暗号処理モジュール１６０で暗号化されたバイナリデータは、他のアプリケーション、例えばウェブアプリケーションなどから操作することが可能になる。

【０１１５】

［１－４．情報処理装置の変形例］

次に、本発明の一実施形態にかかる情報処理装置の変形例について説明する。図１４は、本発明の一実施形態にかかる情報処理装置の変形例である、情報処理装置１１００のハ

ードウェア構成を示す説明図である。以下、図１４を用いて、情報処理装置１１００のハードウェア構成について説明する。

【０１１６】

図２に示した情報処理装置１００は、装置全体が耐タンパ性を有していた。図１４に示したように、情報処理装置１１００は、ＮＶＭ１１１０と、セキュアＣＰＵ１１２０と、ＲＯＭ１１３０と、ＲＡＭ１１４０と、シリアルＩ／Ｏ１１８０と、を含んで構成される。情報処理装置１１００は、装置全体が耐タンパ性を有するのではなく、セキュアＣＰＵ１１２０のみが耐タンパ性を有している点で図２に示した情報処理装置１００と異なっている。このように、ＣＰＵを含む一部を耐タンパ機能で保護し、その他の機能との通信及び送受信データを暗号化技術で保護する構成を採ることもできる。また例えば、図２に示した情報処理装置１００を、耐タンパ環境を有する場所に設けて、変数の参照や関数の属性を実行させるような構成を採ることもできる。

【０１１７】

ＮＶＭ１１１０は、情報処理装置４００の初期化時に、ＲＯＭ１１３０に予め書きこまれている組み込み関数が、シンボルとして記録されるものである。また、ＮＶＭ１１１０は、ユーザが定義した変数（ユーザ定義変数）や関数（ユーザ定義関数）も記憶されるものである。ＮＶＭ１１１０は、電源がオフになっても記憶された情報を保持しておくことができるので、情報処理装置１１００の電源が再投入されても再初期化は行われず、登録されているシンボルがそのまま保持される。

【０１１８】

セキュアＣＰＵ１１２０は、情報処理装置１１００の動作を制御するものであり、ＲＯＭ１１３０に予め記録されているオペレーティングシステムソフトの読み出し命令を実行することで、当該オペレーティングシステムを実行することができる。ＣＰＵ１１２０は、オペレーティングシステムの実行に際しては、ＲＡＭ１１４０をワークエリアとして用いることができる。ここで、ＲＯＭ１１３０に記録されているオペレーティングシステムソフトとしては、例えば、手続き型プログラミング言語を解釈して実行できるものであり、そのようなプログラミング言語として、上述したように例えばＬＩＳＰやＲｕｂｙ、Ｐｙｔｈｏｎ等がある。

【０１１９】

そしてセキュアＣＰＵ１１２０は、図２に示した暗号処理モジュール１６０の機能も有している。セキュアＣＰＵ１１２０は、内部に暗号化鍵を保持しており、ＲＯＭ１１３０に書きこまれるプログラムのコードや、ＮＶＭ１１１０及びＲＡＭ１１４０に記録されるデータを暗号化したり、上記コードやデータを読み込む際に暗号化されているコードやデータを復号して処理したりする機能を有する。またセキュアＣＰＵ１１２０は、図２に示した乱数発生モジュール１７０の機能も有しており、乱数発生指示に基づいて、適当な乱数を発生させることができる。

【０１２０】

シリアルＩ／Ｏインタフェース１１８０は、外部装置からのシリアルデータをパケットとして認識して、適切なデータを抽出したり、また、情報処理装置１１００の内部から外部装置へ出力されるデータをパケットデータとして構成し、シリアルデータとして出力したりする機能を有するものである。

【０１２１】

以上、図１４を用いて、情報処理装置４００のハードウェア構成について説明した。この情報処理装置１１００のように、装置全体ではなく、装置の一部分のみが耐タンパ性を有していれば、上述のリスト処理モジュールを、当該耐タンパ性を有する部分に実行させることで、かかる装置は、変数・関数それぞれに対して独立してセキュリティが担保されたプログラムの実行が可能となる。

【０１２２】

以上、リスト処理モジュールにセキュリティ属性を付加し、さらに耐タンパ機能で保護することで、セキュアなアプリケーション開発及び実行が容易に実現できる情報処理装置

について説明した。ここで、従来のアプリケーション開発モデルと、本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルとの違いについて説明する。

【０１２３】

［１－５．アプリケーション開発モデルの比較］

図１５は、従来のアプリケーション開発モデルについて示す説明図である。図１５を用いて従来のアプリケーション開発モデルについて説明すれば、以下の通りである。まず、ＩＣカードに実行させるプログラムを作成すると、そのプログラムをコンパイラに通してクラスファイルを生成する必要がある。その後、シミュレータを用いて当該クラスファイルのデバッグを行い、バグが存在していれば、作成したプログラムの修正を行う必要がある。

【０１２４】

デバッグが完了したクラスファイルは、ライブラリファイルと共にコンバータに通され、ＩＣカード等のＩＣチップが内蔵されたデバイスに実行させるためのアプリケーションファイルが生成される。そして、ＩＣカードの動作環境をエミュレートするカードエミュレータに生成されたアプリケーションファイルを組み込み、カードエミュレータ上でデバッグを行う。この段階でバグが存在していればプログラムの修正、クラスファイルへのコンパイル及びアプリケーションファイルへのコンバートを行う必要がある。

【０１２５】

カードエミュレータでのデバッグが完了すると、ようやくＩＣカードにアプリケーションファイルを組み込むことになる。そして、ＩＣカードでのデバッグを行って、この段階でバグが存在していればプログラムの修正、クラスファイルへのコンパイル、アプリケーションファイルへのコンバート及びカードエミュレータでのデバッグを行う必要がある。

【０１２６】

このように、従来のアプリケーション開発モデルにおいては、プログラムの作成とデバッグを、複数の段階で繰り返す必要があり、完成したプログラムをＩＣチップ等の耐タンパ環境に実装するまでの工数が多くかかっていた。

【０１２７】

図１６は、本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルについて示す説明図である。図１６を用いて本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルについて説明すれば、以下の通りである。まず、情報処理装置１００に実行させるためのプログラムを開発装置３００で開発し、そのまま情報処理装置１００に組み込む。開発装置３００での開発の際には、プログラムで使用する変数や関数にセキュリティを予め設定することもでき、プログラムで使用する変数や関数に対するセキュリティ機能は後から設定することもできる。

【０１２８】

プログラムが組み込まれた情報処理装置１００は、開発装置３００で開発されたプログラムを、耐タンパ環境で実行されるリスト処理モジュールで実行する。その際に、従来のアプリケーション開発モデルで必要であったプログラムのコードの事前コンパイルは必要なく、プログラムを情報処理装置１００で直接実行することができる。これにより情報処理装置１００は、外部からプログラムのコードやプログラムで使用されるデータを覗き見られることなくセキュアに実行することができる。

【０１２９】

そして、情報処理装置１００でプログラムを実行させてバグが存在することが明らかになった場合には、開発装置３００でプログラムのデバッグを行うが、デバッグに伴う工数の増加は、図１５に示した従来のアプリケーション開発モデルに比べると格段に少なく済むのは明らかである。従って、本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルは、従来のアプリケーション開発モデルに比べて、完成したプログラムをＩＣチップ等の耐タンパ環境に実装するまでの開発工数を大幅に削減でき、セキュアなアプリケーション開発を容易に行うことができる。

【０１３０】
そして本発明の一実施形態にかかる情報処理システムによるアプリケーション開発モデルには、プログラムを作成する時点であらかじめ変数や関数に対するセキュリティを設定して、情報処理装置１００にプログラムを実行させることもできるが、変数や関数に対するセキュリティを設定しない状態で情報処理装置１００にプログラムを実行させて、動作に問題が無ければ情報処理装置１００にプログラムを組み込み、組み込んだプログラムにセキュリティを設定するような運用にしてもよい。
【０１３１】
［１－６．開発装置のハードウェア構成］
次に、図１７を参照しながら、本発明の一実施形態に係る開発装置３００のハードウェア構成について、詳細に説明する。図１７は、本発明の一実施形態に係る開発装置３００のハードウェア構成を説明するためのブロック図である。
【０１３２】
開発装置３００は、主に、ＣＰＵ９０１と、ＲＯＭ９０３と、ＲＡＭ９０５と、ホストバス９０７と、ブリッジ９０９と、外部バス９１１と、インタフェース９１３と、入力装置９１５と、出力装置９１７と、撮像装置９１８と、ストレージ装置９１９と、ドライブ９２１と、接続ポート９２３と、通信装置９２５とを備える。
【０１３３】
ＣＰＵ９０１は、演算処理装置および制御装置として機能し、ＲＯＭ９０３、ＲＡＭ９０５、ストレージ装置９１９、またはリムーバブル記録媒体９２７に記録された各種プログラムに従って、開発装置３００内の動作全般またはその一部を制御する。ＲＯＭ９０３は、ＣＰＵ９０１が使用するプログラムや演算パラメータ等を記憶する。ＲＡＭ９０５は、ＣＰＵ９０１の実行において使用するプログラムや、その実行において適宜変化するパラメータ等を一次記憶する。これらはＣＰＵバス等の内部バスにより構成されるホストバス９０７により相互に接続されている。
【０１３４】
ホストバス９０７は、ブリッジ９０９を介して、ＰＣＩ（Ｐｅｒｉｐｈｅｒａｌ Ｃｏｍｐｏｎｅｎｔ Ｉｎｔｅｒｃｏｎｎｅｃｔ／Ｉｎｔｅｒｆａｃｅ）バスなどの外部バス９１１に接続されている。
【０１３５】
入力装置９１５は、例えば、マウス、キーボード、タッチパネル、ボタン、スイッチおよびレバーなどユーザが操作する操作手段である。また、入力装置９１５は、例えば、赤外線やその他の電波を利用したリモートコントロール手段（いわゆる、リモコン）であってもよいし、開発装置３００の操作に対応した携帯電話やＰＤＡ等の外部接続機器９２９であってもよい。さらに、入力装置９１５は、例えば、上記の操作手段を用いてユーザにより入力された情報に基づいて入力信号を生成し、ＣＰＵ９０１に出力する入力制御回路などから構成されている。開発装置３００のユーザは、この入力装置９１５を操作することにより、開発装置３００に対して各種のデータを入力したり処理動作を指示したりすることができる。
【０１３６】
出力装置９１７は、例えば、ＣＲＴディスプレイ装置、液晶ディスプレイ装置、プラズマディスプレイ装置、ＥＬディスプレイ装置およびランプなどの表示装置や、スピーカおよびヘッドホンなどの音声出力装置や、プリンタ装置、携帯電話、ファクシミリなど、取得した情報をユーザに対して視覚的または聴覚的に通知することが可能な装置で構成される。出力装置９１７は、例えば、開発装置３００が行った各種処理により得られた結果を出力する。具体的には、表示装置は、開発装置３００が行った各種処理により得られた結果を、テキストまたはイメージで表示する。他方、音声出力装置は、再生された音声データや音響データ等からなるオーディオ信号をアナログ信号に変換して出力する。
【０１３７】
撮像装置９１８は、例えばディスプレイ装置の上部に設けられており、開発装置３００

のユーザの静止画像または動画像を撮影することが出来る。撮像装置９１８は、例えばＣＣＤ（Ｃｈａｒｇｅ Ｃｏｕｐｌｅｄ Ｄｅｖｉｃｅ）イメージセンサまたはＣＭＯＳ（Ｃｏｍｐｌｅｍｅｎｔａｒｙ Ｍｅｔａｌ Ｏｘｉｄｅ Ｓｅｍｉｃｏｎｄｕｃｔｏｒ）イメージセンサを備えており、レンズで集光した光を電気信号に変換することで静止画像または動画像を撮影することができる。

【０１３８】

ストレージ装置９１９は、開発装置３００の記憶部の一例として構成されたデータ格納用の装置であり、例えば、ＨＤＤ（Ｈａｒｄ Ｄｉｓｋ Ｄｒｉｖｅ）等の磁気記憶部デバイス、半導体記憶デバイス、光記憶デバイス、または光磁気記憶デバイス等により構成される。このストレージ装置９１９は、ＣＰＵ９０１が実行するプログラムや各種データ、および外部から取得した音響信号データや画像信号データなどを格納する。

【０１３９】

ドライブ９２１は、記録媒体用リーダライタであり、開発装置３００に内蔵、あるいは外付けされる。ドライブ９２１は、装着されている磁気ディスク、光ディスク、光磁気ディスク、または半導体メモリ等のリムーバブル記録媒体９２７に記録されている情報を読み出して、ＲＡＭ９０５に出力する。また、ドライブ９２１は、装着されている磁気ディスク、光ディスク、光磁気ディスク、または半導体メモリ等のリムーバブル記録媒体９２７に記録を書き込むことも可能である。リムーバブル記録媒体９２７は、例えば、ＤＶＤメディア、Ｂｌｕ－ｒａｙメディア、コンパクトフラッシュ（登録商標）（ＣｏｍｐａｃｔＦｌａｓｈ：ＣＦ）、メモリースティック、または、ＳＤメモリカード（Ｓｅｃｕｒｅ Ｄｉｇｉｔａｌ ｍｅｍｏｒｙ ｃａｒｄ）等である。また、リムーバブル記録媒体９２７は、例えば、非接触型ＩＣチップを搭載したＩＣカード（Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ ｃａｒｄ）または電子機器等であってもよい。

【０１４０】

接続ポート９２３は、例えば、ＵＳＢ（Ｕｎｉｖｅｒｓａｌ Ｓｅｒｉａｌ Ｂｕｓ）ポート、ｉ．Ｌｉｎｋ等のＩＥＥＥ１３９４ポート、ＳＣＳＩ（Ｓｍａｌｌ Ｃｏｍｐｕｔｅｒ Ｓｙｓｔｅｍ Ｉｎｔｅｒｆａｃｅ）ポート、ＲＳ－２３２Ｃポート、光オーディオ端子、ＨＤＭＩ（Ｈｉｇｈ－Ｄｅｆｉｎｉｔｉｏｎ Ｍｕｌｔｉｍｅｄｉａ Ｉｎｔｅｒｆａｃｅ）ポート等の、機器を開発装置３００に直接接続するためのポートである。この接続ポート９２３に外部接続機器９２９を接続することで、開発装置３００は、外部接続機器９２９から直接音響信号データや画像信号データを取得したり、外部接続機器９２９に音響信号データや画像信号データを提供したりする。

【０１４１】

通信装置９２５は、例えば、通信網９３１に接続するための通信デバイス等で構成された通信インタフェースである。通信装置９２５は、例えば、有線または無線ＬＡＮ（Ｌｏｃａｌ Ａｒｅａ Ｎｅｔｗｏｒｋ）、Ｂｌｕｅｔｏｏｔｈ、またはＷＵＳＢ（Ｗｉｒｅｌｅｓｓ ＵＳＢ）用の通信カード、光通信用のルータ、ＡＤＳＬ（Ａｓｙｍｍｅｔｒｉｃ Ｄｉｇｉｔａｌ Ｓｕｂｓｃｒｉｂｅｒ Ｌｉｎｅ）用のルータ、または、各種通信用のモデム等である。この通信装置９２５は、例えば、インターネットや他の通信機器との間で、例えばＴＣＰ／ＩＰ等の所定のプロトコルに則して信号等を送受信することができる。また、通信装置９２５に接続される通信網９３１は、有線または無線によって接続されたネットワーク等により構成され、例えば、インターネット、家庭内ＬＡＮ、赤外線通信、ラジオ波通信または衛星通信等であってもよい。

【０１４２】

ユーザは、かかる開発装置３００を用いて、情報処理装置１００で実行させるコンピュータプログラムを開発することができる。開発装置３００を用いて作成されたコンピュータプログラムは、例えば、開発装置３００に接続したリーダライタ２００を介して情報処理装置１００に組み込むことができる。

【０１４３】

＜２．まとめ＞

以上説明したように本発明の一実施形態によれば、情報処理装置１００で実行されるリスト処理モジュールを、変数・関数それぞれ独立してセキュリティを設定できるよう構成し、当該リスト処理モジュールを耐タンパ機能で保護することにより、セキュアなアプリケーション開発が容易となる、プログラミングオペレーティングシステムを実現することができる。また、プログラムで使用する変数や関数はそれぞれ独自にセキュリティ設定が可能であるので、緻密なセキュリティの実装を容易に実現することができる。

【０１４４】

上記リスト処理モジュールは、インタプリタの基本機能を備え、さらにセキュリティ機能を付加することで、情報処理装置１００にプログラムを実行させる際には、事前のコンパイルの必要がなく、プログラムの実行に利用するＩＣチップそのものデバッグを実行することが可能である。従って、従来のアプリケーション開発モデルに比べて、開発ステップが少なく、短期間でのアプリケーション開発が可能となる。また、情報処理装置１００の内部に設けられる不揮発性メモリに、動的に変数や関数を直接定義することが可能であり、柔軟なアプリケーション開発も可能となる。

【０１４５】

本発明の一実施形態にかかる情報処理装置１００で実行するアプリケーションで使用される、セキュリティロックのかかった変数や関数は、それぞれ設定された認証鍵で認証して初めて利用可能となる。そのため、関数定義においてその関数が引数として利用するセキュリティロックのかかった変数、関数は、関数定義時に認証されている必要がある。しかし、関数定義後のその関数の利用においては、定義した関数のセキュリティ認証のみを行うことで、内部で利用している引数として用いる変数、関数は無条件に利用可能とすることができる。これにより、毎回全ての認証鍵を用意して認証する必要が無くなり、アプリケーションの運用を容易にすることができる。

【０１４６】

スクリプト言語はアプリケーションをダウンロードした後で、容易にプログラムの書き換えが出来ることが特徴となっているが、これがオープンプラットフォームでは改ざんの脅威になっている。本発明の一実施形態にかかる情報処理装置１００の構成は、そのプログラムの改ざんも防ぐことが可能となる。

【０１４７】

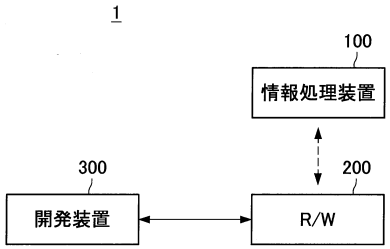
以上、添付図面を参照しながら本発明の好適な実施形態について詳細に説明したが、本発明はかかる例に限定されない。本発明の属する技術の分野における通常の知識を有する者であれば、特許請求の範囲に記載された技術的思想の範疇内において、各種の変更例または修正例に想到し得ることは明らかであり、これらについても、当然に本発明の技術的範囲に属するものと了解される。

【符号の説明】

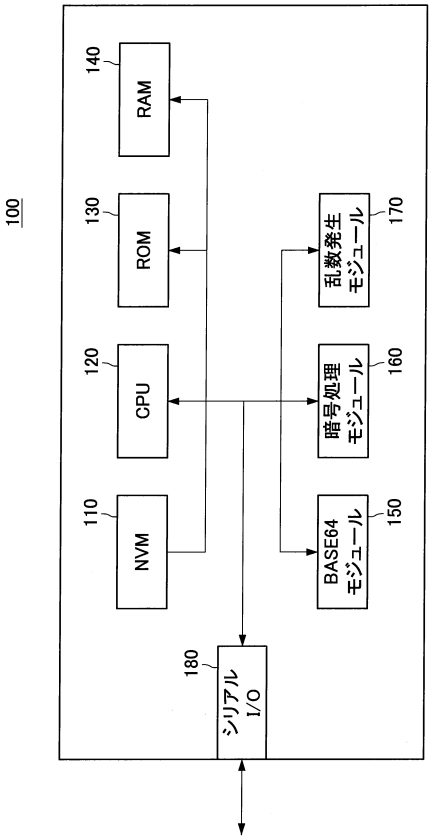
【０１４８】

１００　情報処理装置
１１０　ＮＶＭ
１２０　ＣＰＵ
１３０　ＲＯＭ
１４０　ＲＡＭ
１５０　ＢＡＳＥ６４モジュール
１６０　暗号処理モジュール
１７０　乱数発生モジュール
１８０　シリアルＩ／Ｏインタフェース
２００　リーダライタ
３００　開発装置

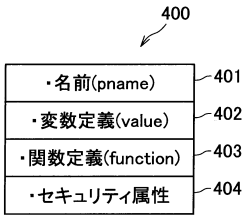
【図１】



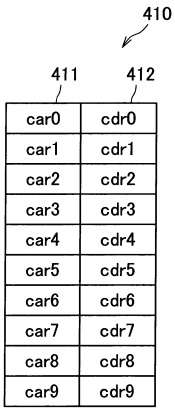
【図２】



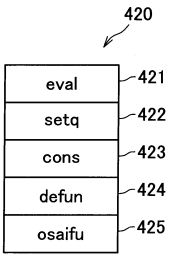
【図３】



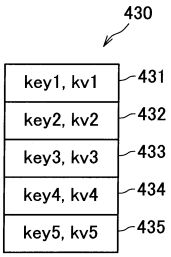
【図４】



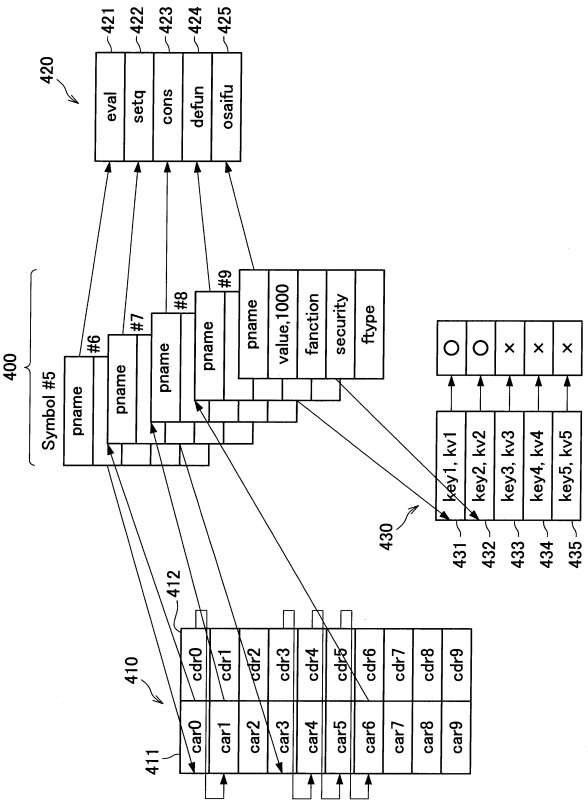
【図５】



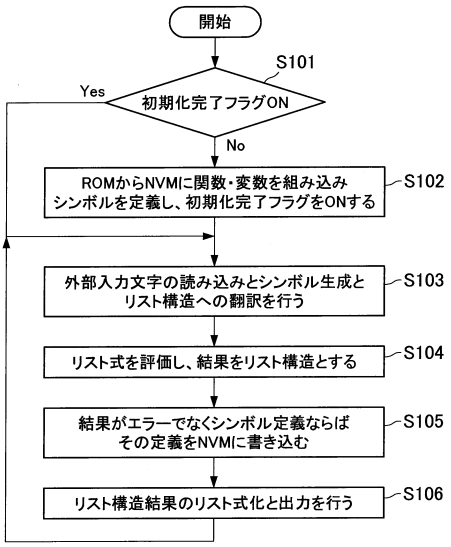
【図６】



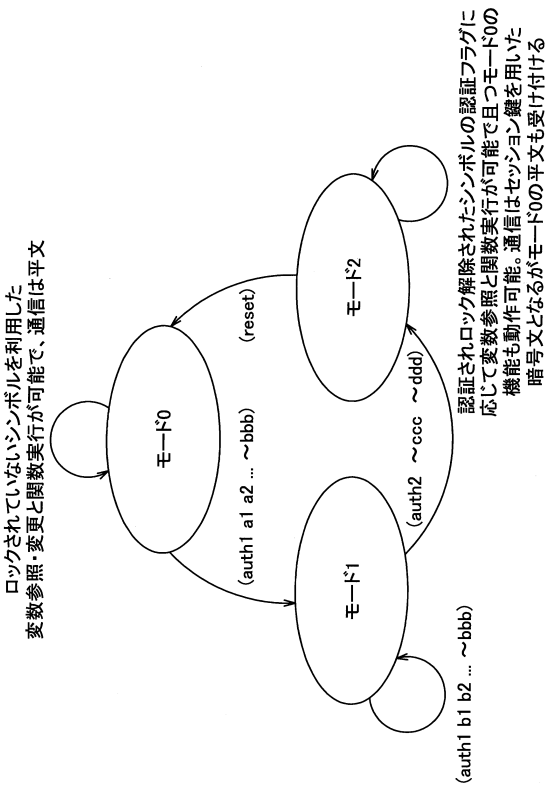
【図７】



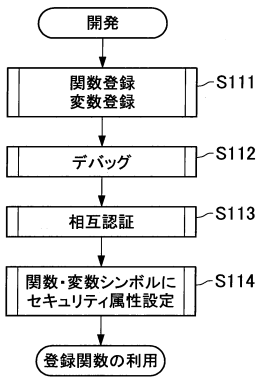
【図８】



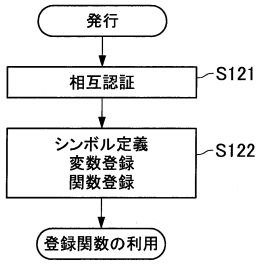
【図９】



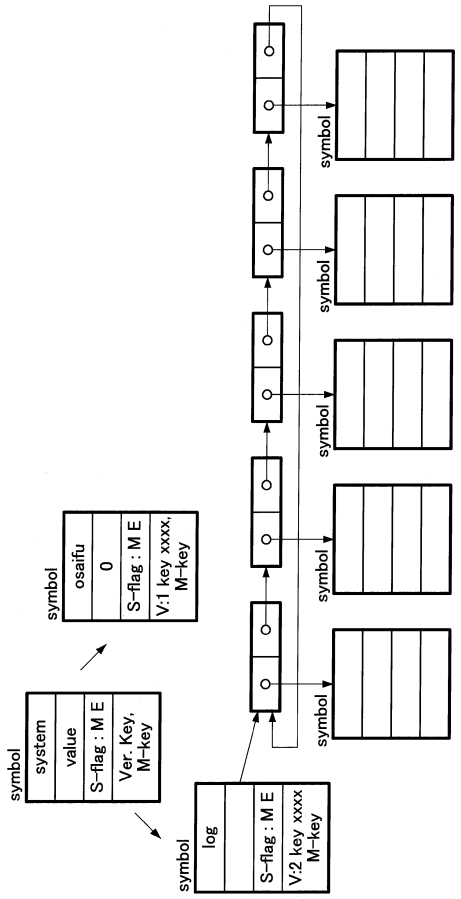
【図１０】



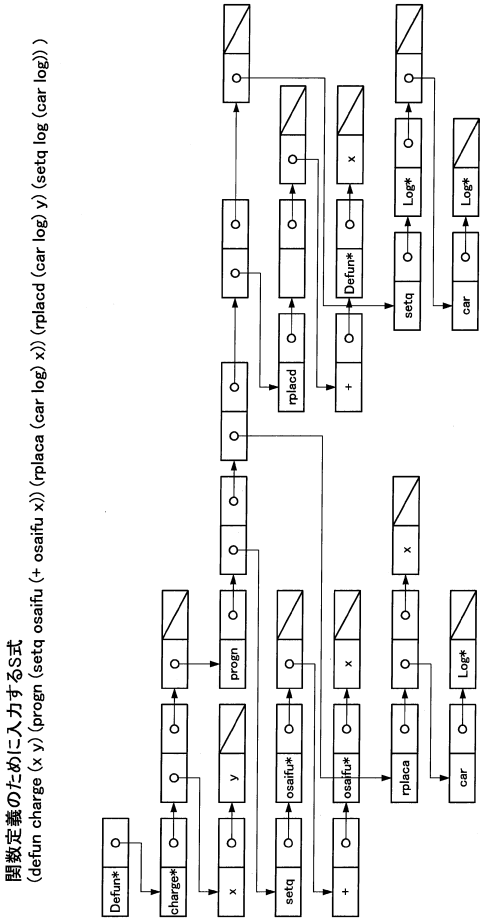
【図１１】



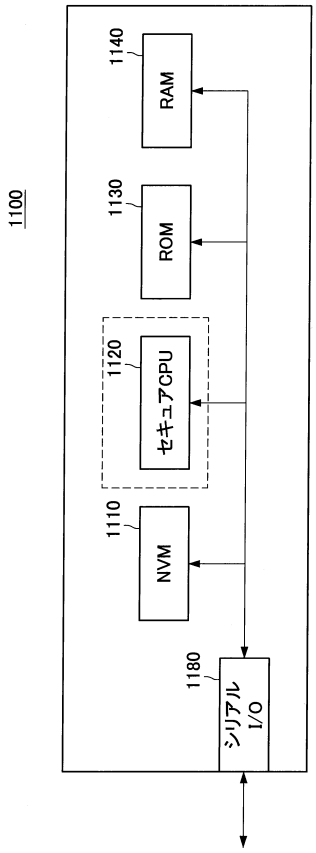
【図１２】



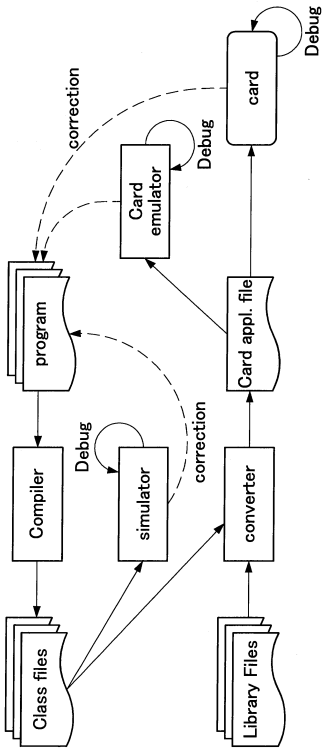
【図１３】



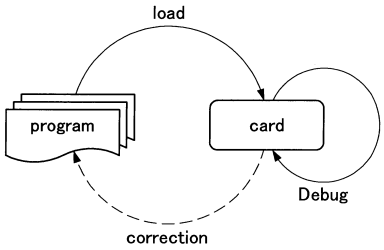
【図１４】



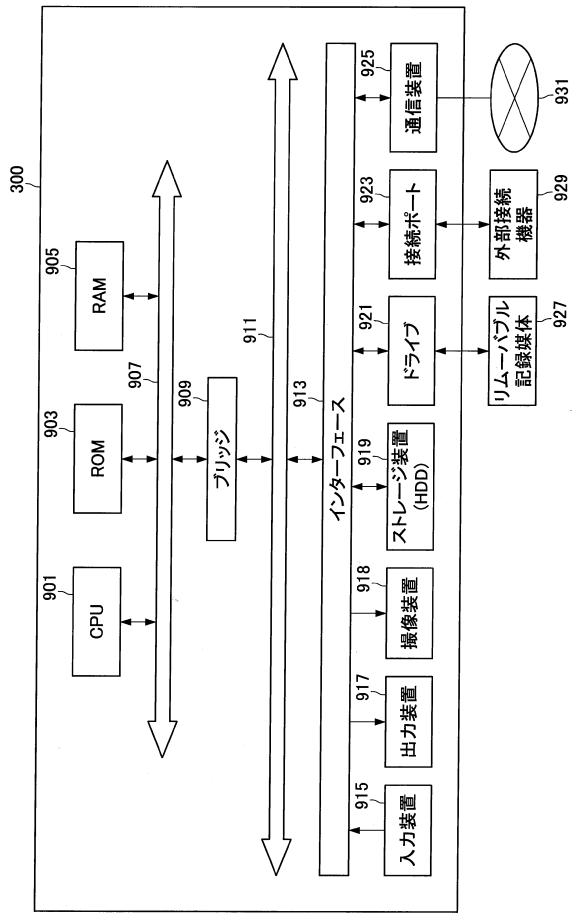
【図１５】



【図１６】


load
program
card
Debug
correction

【図１７】


300
901
CPU
903
ROM
905
RAM
907
909
ブリッジ
911
913
インターフェース
915
入力装置
917
出力装置
918
撮像装置
919
ストレージ装置
(HDD)
921
ドライブ
923
接続ポート
925
通信装置
927
リムーバブル
記録媒体
929
外部接続
機器
931

フロントページの続き

審査官　戸島　弘詩

(56)参考文献　特開２００２－１１７３８１（ＪＰ，Ａ）
特開２０１０－１０８１７０（ＪＰ，Ａ）
特開２００９－２９６４６３（ＪＰ，Ａ）
特開２００７－１５８３８３（ＪＰ，Ａ）
特表２００２－５０５４５９（ＪＰ，Ａ）
特開平０３－００６６４０（ＪＰ，Ａ）
特開２００８－０５９５９４（ＪＰ，Ａ）
特開平０５－２１０４９８（ＪＰ，Ａ）
特開２００６－２９５５１９（ＪＰ，Ａ）
金子　勇，プロトタイプベースオブジェクトファイルシステム，情報処理学会論文誌，日本，社団法人情報処理学会，１９９８年　9月１５日，第３９巻，第９号，第2671-2682頁
細見　格，デジタル情報流通アーキテクチャＭｅｄｉａＳｈｅｌｌとその利用・課金制御，情報処理学会研究報告，日本，社団法人情報処理学会，１９９８年　9月１９日，Ｖｏｌ．９８，Ｎｏ．８５，第49-56頁

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ２１／００－２１／８８
Ｇ０６Ｆ９／４４－９／４４５
Ｇ０６Ｋ１９／００－１９／１８