(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4937921号
(P4937921)
(45)発行日　平成24年5月23日(2012.5.23)　(24)登録日　平成24年3月2日(2012.3.2)

(51)Int.Cl.　F I
H04L 9/08 (2006.01)　H04L 9/00 601C
H04L 9/00 601E

請求項の数 20　(全 17 頁)

(21)出願番号　特願2007-540747(P2007-540747)
(86)(22)出願日　平成17年11月11日(2005.11.11)
(65)公表番号　特表2008-520145(P2008-520145A)
(43)公表日　平成20年6月12日(2008.6.12)
(86)国際出願番号　PCT/IB2005/003385
(87)国際公開番号　WO2006/051404
(87)国際公開日　平成18年5月18日(2006.5.18)
審査請求日　平成20年10月27日(2008.10.27)
(31)優先権主張番号　PCT/IB2004/3705
(32)優先日　平成16年11月11日(2004.11.11)
(33)優先権主張国　国際事務局(IB)

(73)特許権者　397071791
サーティコム　コーポレーション
カナダ国　エル４ダブリュー　０ビー５　オンタリオ，ミシソーガ，タホー　ブールバード　４７０１，タホー　エー，　６ティーエイチ　フロア
(74)代理人　100107489
弁理士　大塩　竹志
(72)発明者　ブラウン，　ダニエル　アール．　エル．
カナダ国　オンタリオ州　エル５エヌ　１エックス８　ミシソーガ　パドルロード　６０３３

最終頁に続く

(54)【発明の名称】汎用鍵導出関数サポートのための安全インタフェース

(57)【特許請求の範囲】
【請求項１】
計算装置上で暗号化動作を計算する方法であって、前記方法は秘密値を含み、前記秘密値はデータリンクを介して前記計算装置と通信するモジュール上に格納されており、前記方法は、
ｉ）アプリケーションが前記計算装置上で実行され、前記モジュールによって提供された前記秘密値に対するポインタを用いて前記秘密値を参照することによって、前記モジュールに、前記秘密値を含む前記暗号化動作の少なくとも１つのサブルーチンを実行するように要求するステップと、
ｉｉ）前記モジュールが、前記ポインタによって参照された前記秘密値を利用して前記少なくとも１つのサブルーチンを実行して、１つ以上の暗号構成要素を得るステップと、
ｉｉｉ）前記モジュールが、前記１つ以上の暗号構成要素を前記アプリケーションに提供するステップと、
ｉｖ）前記アプリケーションが、前記１つ以上の暗号構成要素を利用して前記暗号化動作を完結するステップと
を含む、方法。
【請求項２】
前記暗号化動作は、鍵導出関数である、請求項１に記載の方法。
【請求項３】
前記モジュールによって実行される前記少なくとも１つのサブルーチンは、ハッシュ関

数を含む、請求項２に記載の方法。
【請求項４】
前記鍵導出関数は、ＡＮＳＩ　Ｘ９．６３に適合し、前記ハッシュ関数は、ＳＨＡ－１ハッシュ関数を含み、前記方法は、
前記アプリケーションが、前記ポインタを用いて前記秘密値に対して前記ＳＨＡ－１ハッシュ関数を計算するように前記モジュールに指示することによって、前記モジュールに、前記少なくとも１つのサブルーチンを実行するように要求することと、
前記モジュールが、前記秘密値に対して前記ＳＨＡ－１ハッシュ関数を実行してハッシュ出力を生成することと、
前記モジュールが、前記１つ以上の暗号構成要素として前記ハッシュ出力を提供することと、
前記アプリケーションが、さらなる入力として前記ハッシュ出力を得て、前記暗号化動作を完結して、前記暗号化動作から鍵を導出することと
をさらに含む、請求項３に記載の方法。
【請求項５】
前記方法は、
前記アプリケーションが、共有秘密を第１の半分と第２の半分とに分割するように前記モジュールに指示することによって、前記モジュールに、前記少なくとも１つのサブルーチンを実行するように要求することと、
前記モジュールが、前記第１の半分に対応する第１の半分ポインタと前記第２の半分に対応する第２の半分ポインタとを、第１の出力として提供することと、
前記アプリケーションが、前記第１の半分ポインタと前記第２の半分ポインタとのうちの１つのポインタを参照することによって、前記第１の半分と前記第２の半分とのうちの１つの半分に対してＳＨＡ－１ハッシュ関数を実行するように前記モジュールに指示することと、
前記モジュールが、前記アプリケーションによって参照された前記第１の半分と前記第２の半分とのうちの前記１つの半分に対して前記ＳＨＡ－１ハッシュ関数を実行して前記ＳＨＡ－１ハッシュ関数の結果を生成して、前記ＳＨＡ－１ハッシュ関数の結果に対応するＳＨＡ－１ハッシュ関数の結果ポインタを前記アプリケーションに戻すことと
をさらに含む、請求項２に記載の方法。
【請求項６】
前記アプリケーションが、第２のセットの入力と前記ＳＨＡ－１ハッシュ関数の結果ポインタとを前記モジュールに提供して、前記第２のセットの入力と、前記ＳＨＡ－１ハッシュ関数の結果ポインタによって参照された前記ＳＨＡ－１ハッシュ関数の結果とに基づいて、前記ＳＨＡ－１ハッシュ関数の１回以上の繰り返しを実行するように前記モジュールに指示することと、
前記モジュールが、前記指示と、前記第２のセットの入力と、前記ＳＨＡ－１ハッシュ関数の結果ポインタとを受けて、前記ＳＨＡ－１ハッシュ関数の結果ポインタによって参照された前記ＳＨＡ－１ハッシュ関数の前記結果を利用して、前記ＳＨＡ－１ハッシュ関数の前記１回以上の繰り返しを実行して、前記ＳＨＡ－１ハッシュ関数の前記１回以上の繰り返しの結果を第２の出力として戻すことと
をさらに含む、請求項５に記載の方法。
【請求項７】
前記アプリケーションが、前記第１の出力と前記第２の出力とを用いてＰ＿ＳＨＡ－１を計算して、前記鍵導出関数を完結して、前記鍵導出関数から鍵を導出することをさらに含む、請求項５に記載の方法。
【請求項８】
前記ＳＨＡ－１ハッシュ関数の前記１回以上の繰り返しの結果は鍵を含み、前記第２の出力は前記鍵に対するポインタを含み、前記方法は、前記アプリケーションが、続いて、前記鍵に対する前記ポインタを参照することによって前記鍵を用いて動作を実行するよう

に前記モジュールに指示することをさらに含む、請求項５に記載の方法。
【請求項９】
暗号化装置であって、
前記暗号化装置は、
暗号化動作を計算するためのアプリケーションを実行するデバイスであって、前記暗号化動作の少なくとも１つのサブルーチンは秘密値を利用する、デバイスと、
前記秘密値を格納し、プロセッサを含むモジュールであって、前記プロセッサは、前記少なくとも１つのサブルーチンを実行して、前記秘密値を用いて暗号構成要素を生成して、前記暗号構成要素から結果を生成するように動作し、前記モジュールは、第１のインターフェース機能と第２のインターフェース機能とを提供するインターフェースを含み、前記第１のインターフェース機能は、前記秘密値を参照されるようにするが明らかにはされないようにすることができ、前記第２のインターフェース機能は、前記プロセッサに前記第１のインターフェース機能によって参照された秘密を利用して前記少なくとも１つのサブルーチンを実行するように指示する、モジュールと、
前記アプリケーションから前記モジュールへの指示を可能にして、前記モジュールから前記アプリケーションへの前記結果の転送を可能にする、第１のモジュールと前記アプリケーションとの間のデータ接続と、
を含み、前記アプリケーションは前記結果を用いて前記暗号化動作の計算を完結するようにさらに動作可能である、装置。
【請求項１０】
前記モジュールと前記アプリケーションとは、単一のデバイスによって備えられている、請求項９に記載の装置。
【請求項１１】
前記モジュールは、改竄防止保護によって保護されている、請求項９に記載の装置。
【請求項１２】
前記モジュールは、前記アプリケーションによって提供されるジャバ実行言語を有するプログラムを含み、前記プログラムは、複数の暗号化動作からサブルーチンを実行するためのものであり、前記モジュールは、前記サブルーチンの結果が前記アプリケーションに出力される前に安全なアルゴリズムを用いて処理されることを確実にする、請求項９に記載の装置。
【請求項１３】
前記アプリケーションは、前記プログラムをデジタル署名するためのデジタル署名モジュールをさらに含み、前記モジュールは、前記デジタル署名モジュールによって生成された署名を検証するための公開検証鍵をさらに含む、請求項１２に記載の装置。
【請求項１４】
前記暗号化動作は鍵導出関数であり、前記少なくとも１つのサブルーチンはハッシュ関数である、請求項９に記載の装置。
【請求項１５】
前記鍵導出関数はＡＮＳＩ　Ｘ９．６３に適合し、前記装置は、
前記アプリケーションであって、前記第１のインターフェース機能を用いて前記秘密値に対して前記ハッシュ関数を計算するように前記モジュールに指示することによって、前記モジュールに前記少なくとも１つのサブルーチンを実行するように要求するように動作する、アプリケーションと、
前記モジュールであって、前記秘密値に対して前記ハッシュ関数を実行してハッシュ出力を生成する、モジュールと、
前記モジュールであって、前記ハッシュ出力を前記１つ以上の暗号構成要素として提供する、モジュールと、
前記アプリケーションであって、前記ハッシュ出力をさらなる入力として得て前記暗号化動作を完結させて前記暗号化動作から鍵を導出する、アプリケーションと
をさらに含む、請求項１４に記載の装置。

【請求項１６】
前記装置は、
前記アプリケーションであって、共有秘密を第１の半分と第２の半分とに分割するように前記モジュールに指示することによって、前記モジュールに前記少なくとも１つのサブルーチンを実行するように要求するように動作する、アプリケーションと、
前記モジュールであって、前記第１のインターフェース機能として、前記第１の半分に対応する第１の半分ポインタと前記第２の半分に対応する第２の半分ポインタとを、第１の出力として提供するように動作する、モジュールと、
前記アプリケーションであって、前記第１の半分ポインタと前記第２の半分ポインタとのうちの１つのポインタを参照することによって、前記第１の半分と前記第２の半分とのうちの１つの半分に対してＳＨＡ－１ハッシュ関数を実行するように前記モジュールに指示するように動作する、アプリケーションと、
前記モジュールであって、前記アプリケーションによって参照された前記第１の半分と前記第２の半分とのうちの前記１つの半分に対して前記ＳＨＡ－１ハッシュ関数を実行して前記ＳＨＡ－１ハッシュ関数の結果を生成して、前記ＳＨＡ－１ハッシュ関数の結果に対応するＳＨＡ－１ハッシュ関数の結果ポインタを、前記アプリケーションに戻すように動作する、モジュールと
をさらに含む、請求項９に記載の装置。
【請求項１７】
前記アプリケーションであって、第２のセットの入力と前記ＳＨＡ－１ハッシュ関数の結果ポインタとを前記モジュールに提供して、前記第２のセットの入力と、前記ＳＨＡ－１ハッシュ関数の結果ポインタによって参照された前記ＳＨＡ－１ハッシュ関数の結果とに基づいて、前記ＳＨＡ－１ハッシュ関数の１回以上の繰り返しを実行するように前記モジュールに指示するように動作する、アプリケーションと、
前記モジュールであって、前記指示と、前記第２のセットの入力と、前記ＳＨＡ－１ハッシュ関数の結果ポインタとを受けて、前記ＳＨＡ－１ハッシュ関数の結果ポインタによって参照された前記ＳＨＡ－１ハッシュ関数の前記結果を利用して、前記ＳＨＡ－１ハッシュ関数の前記１回以上の繰り返しを実行して、前記ＳＨＡ－１ハッシュ関数の前記１回以上の繰り返しの結果を第２の出力として戻すように動作する、モジュールと
をさらに含む、請求項１６に記載の装置。
【請求項１８】
暗号化動作を計算する方法であって、前記方法は秘密値を含み、前記秘密値は暗号化装置のモジュールにアクセス可能であり、前記モジュールはデータ接続を介して、前記暗号化装置上で実行するアプリケーションと通信し、前記モジュールは、第１のインターフェース機能と第２のインターフェース機能とを提供するインターフェースを含み、前記第１のインターフェース機能は、前記秘密値を前記アプリケーションによって参照されるようにするが明らかにはされないようにすることができるポインタを提供し、前記第２のインターフェース機能は、入力値として受け取り、前記アプリケーションからのポインタは、秘密を利用して前記暗号化動作の少なくとも１つのサブルーチンを、前記モジュール上で実行し、前記方法は、
前記アプリケーションが、前記秘密値を利用して前記少なくとも１つのサブルーチンから結果を得るように前記モジュールに要求するステップであって、前記秘密値は、前記ポインタを用いて前記第１のインターフェース機能を通して前記アプリケーションによって参照され、前記少なくとも１つのサブルーチンは、前記第２のインターフェース機能を通して参照される、ステップと、
前記アプリケーションが、前記インターフェースを通して前記モジュールから前記結果を受け取るステップであって、前記結果は、前記ポインタによって参照された前記秘密を利用して前記暗号化動作の前記少なくとも１つのサブルーチンを実行することによって前記モジュールによって生成される、ステップと、
前記アプリケーションが、前記モジュールによって生成された前記結果を利用して前記

暗号化動作の計算を完結するステップと
を含む、方法。
【請求項１９】
前記暗号化動作は、鍵導出関数である、請求項１８に記載の方法。
【請求項２０】
前記鍵導出関数は、共有秘密値から公開鍵を生成するための公開鍵導出関数を含み、前記少なくとも１つのサブルーチンは、前記共有秘密値に適用される一方向性関数を含む、請求項１９に記載の方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は一般的には、暗号の分野に関する。特に、本発明は、多用途鍵導出関数サポートの提供に関する。
【背景技術】
【０００２】
ディフィーヘルマン（Diffie-Hellman：ＤＨ）鍵共有は、暗号における基本的な開発である。それは、公開鍵暗号の最初の機能する方法であり、それにより、前もって同意された秘密を設定せずに鍵配布を実現可能にする。
【０００３】
ＤＨ鍵共有の最も簡単な形態では、当事者はそれぞれ私有鍵ｘ、ｙを有し、そこから公開鍵α^{x}、α^{y}がそれぞれ導出できる。公開鍵を交換することにより、当事者それぞれは、私有鍵と公開鍵を組み合わせることにより共有秘密鍵α^{xy}を算出できる。私有鍵から公開鍵を導出するために使用される関数は、一方向性関数であり、一方向性関数では、公開鍵の計算は比較的簡単であるが、公開鍵から私有鍵を抽出するのは実行不可能である。そのような関数は、２つの大きな素数の積である大きな数の素因数分解の困難性、または有限体上における離散対数問題に基づいている。
【０００４】
ディフィーヘルマン（Diffie-Hellman：ＤＨ）鍵共有は、今日では広く使用されている。ＩＰＳｅｃプロトコルはＤＨ鍵共有を使用し、ＩＰＳｅｃは、ほとんどの企業において、従業員が公開インターネット上で離れたオフィスへ接続するのと共に、遠隔地から会社のネットワークに接続するのを可能にするために使用されている仮想プライベートネットワーク（ＶＰＮ）のほとんどで使用されている。
【０００５】
ディフィーヘルマン（Diffie-Hellman）鍵共有はまた、トランスポート層セキュリティ（ＴＬＳ）プロトコルにおけるＮＩＳＴ推奨オプションでもある。ＴＬＳプロトコルは、ＳＳＬプロトコルの後継プロトコルである。これらのプロトコルは今日では、オンラインバンキングのような、取扱いに慎重さを要するウェブ上の通信データの流れを安全にするために広く使用されている。
【０００６】
静的ＤＨ鍵共有は、私有鍵の１つが静的であるＤＨ鍵共有の変形であり、それは、複数回使用される長期間鍵であることを意味している。
【０００７】
私有鍵の鋭敏性のため、特に複数回使用される場合は、それは私有鍵操作を含む実践である私有鍵モジュールに通常位置している。一般的には、そのようなモジュールは私有鍵の抽出を防止する手段を含んでおり、もっと制限された範囲で、私有鍵操作の悪用を防止している。例えば、これらのモジュールは、ウイルスや、ワーム、およびトロイの木馬のような悪意のあるソフトウェアのロードを認めない特殊化されたハードウェアにおいて実践できる。一般的には、そのような改竄防止手段の実践は高価となる。従って、コストを下げるために、モジュールは一般的には、最低限の機能を有するように設計される。このように、最低限の機能には、改竄防止による保護が必要である。

【０００８】
簡単な例では、モジュールはスマートカードであってよい。スマートカードはユーザーにより所持される。ユーザーがある遠隔コンピュータから、ホームコンピュータのような接続先へ安全に接続したいと所望していると仮定する。ユーザーはスマートカードを、リモートコンピュータに取り付けられたスマートカードリーダーに入れる。その後、ホームコンピュータへの接続が行われる。ホームコンピュータは、問いかけを送ることでユーザーを認証する。リモートコンピュータはその問いかけをスマートカードに転送する。スマートカードは問いかけに署名し、それはホームコンピュータに転送されて戻される。ホームコンピュータは、その問いかけを検証して、リモートコンピュータを介しての、必要なアクセスをユーザーに提供する。これにより、ユーザーは異なるリモートコンピュータ間を動き回ることができる。しかし、リモートコンピュータはスマートカードからユーザーの私有鍵を抽出できてはならない。つまり、リモートコンピュータはユーザーがスマートカードをリーダー内に入れている間だけホームコンピュータに接続できるだけである。（これを達成するために、簡単な問いかけと応答より、より高度な方法が必要となる。そして、スマートカードはトラフィックの正規の認証、またはすべてのトラフィックの暗号化と暗号解読さえも行うことが必要となることもある。）

【０００９】
安全性を更に高めるために、未処理のＤＨ共有秘密に適用された一方向性関数である鍵導出関数（ＫＤＦ）がしばしば指定される。ある規格では、ＫＤＦをＤＨ鍵共有と共に使用するよう指定している。しかし、規格が異なれば、推奨されるＫＤＦも異なる。例えば、ＡＮＳＩは、ＩＥＥＥや、ＳＳＬおよびＴＬＳのように、いくつかの異なるＫＤＦを指定しており、ＩＰＳｅｃも更にまた異なっている。

【００１０】
下記に、２つの規格化鍵導出関数の詳細を簡単に記述する。これらはＡＮＳＩ　Ｘ９．６３鍵導出およびＴＬＳ鍵導出関数である。

【００１１】
ＡＮＳＩ　Ｘ９．６３鍵導出は下記のように計算される。入力は３つの成分を有している。第１入力成分はＺであり、これは私有鍵モジュールと接続先、例えば、上記の簡単な例におけるホームコンピュータとの間で共有される秘密値である。この共有秘密値Ｚは、上記の例におけるリモートコンピュータのような、いかなるゲートウェイにも明かされてはならない。第２の入力成分は整数鍵datalenであり、これは生成されるべきキーイングデータのオクテットでの長さである。オプションである第３の入力成分は、オクテットストリングSharedInfoであり、共有秘密値Ｚを共有するエンティティにより共有されるいくつかのデータから構成されている。更に、SharedInfoにはまた、随意に抽象構文記法１（ＡＳＮ．１）の符号化が与えられ、これは５つのフィールド、つまり、アルゴリズム識別子、２つのエンティティそれぞれに対する随意の識別子、随意の公開共有情報、および随意の私有共有情報を含む。この入力についてのＫＤＦの評価はその後、下記のように進行する。

【００１２】
ＡＮＳＩ　Ｘ９．６３鍵導出関数の第１ステップは、ある一貫性チェックであり、入力の長さと、所望の出力長keydatalenに対して行われる。その後、４オクテット整数カウンタｊが値１で初期化される。一連のハッシュ値Ｋｊは、次のように計算される。Ｋｊ＝ＳＨＡ－１（Ｚ‖ｊ‖[SharedInfo]）、ここで‖は、連接を示し、[ ]はカッコで囲まれた入力はオプションであることを示している。これらの出力の数ｔは、keydatalenに依存する。ハッシュ値は連接されてオクテットストリングＫ’＝Ｋ１‖Ｋ２‖．．．‖Ｋｔを形成する。オクテットストリングは、最左端のkeydatalenオクテットを取ることにより短オクテットストリングKに切り詰められる。ＡＮＳＩ　Ｘ９．６３ＫＤＦの出力はＫである。

【００１３】
ＴＬＳ規格では、鍵導出関数は擬似乱数関数（ＰＲＦ）と称せられる。ＴＬＳ　ＰＲＦの構成はＡＮＳＩ　Ｘ９．６３ＫＤＦとはまったく異なり、次のように与えられる。この

構成は、最初に記述される補助構成ＨＭＡＣを利用する。

【００１４】

ＨＭＡＣ構成は、任意のハッシュ関数上に構築できる。ＨＭＡＣ構成が、ＭＤ５およびＳＨＡ－１のようなハッシュ関数と共に使用されると、その結果としての関数は、ＨＭＡＣ－Ｈａｓｈと命名され、ここでＨａｓｈはハッシュ関数の名前である。ＴＬＳ　ＰＲＦはＨＭＡＣ－ＳＨＡ－１とＨＭＡＣ－ＭＤ５を使用する。ＨＭＡＣの一般的な形式、つまりＨＭＡＣ－Ｈａｓｈは下記のように作動する。

【００１５】

ＨＭＡＣへの入力は、秘密鍵Ｋとメッセージｍである。出力はタグＴである。ＨＭＡＣタグは、Ｔ＝Ｈａｓｈ（Ｃ＋Ｋ‖Ｈａｓｈ（（Ｄ＋Ｋ）‖Ｍ））として計算され、ここで‖は、連接を表わし、＋は、よく知られたビット単位の排他的論理和（ＸＯＲ）演算を表わし、ＣとＤは、ＨＭＡＣアルゴリズムで決定された一定のビットストリングである。より正確には、鍵Ｋは、その長さがＣとＤの長さになるまでゼロビットを埋め込まれるが、ＫがＣとＤより長いときは除外され、その場合は、Ｋは鍵のハッシュと置換される。これは次のように記述される。

Ｔ＝ＨＡＭＣ－Ｈａｓｈ（Ｋ，Ｍ）

【００１６】

関数ＨＭＡＣ－Ｈａｓｈは、Ｐ＿Ｈａｓｈと呼ばれる、ＴＬＳ　ＰＲＦにおける別の補助ハッシュ一般構成に使用される。Ｐ＿Ｈａｓｈの構成は下記の通りである。

Ｐ＿Ｈａｓｈ（Ｚ，シード）＝ＨＡＭＣ－Ｈａｓｈ（Ｚ，Ａ（１）‖シード）‖ＨＭＡＣ－Ｈａｓｈ（Ｚ，Ａ（２）‖シード）‖ＨＭＡＣ－Ｈａｓｈ（Ｚ，Ａ（３）‖シード）．．．

ここで‖は、連接を表わし、Ａ（）は下記のように定義される。

Ａ（０）＝シード；Ａ（ｊ）＝ＨＭＡＣ＿Ｈａｓｈ（Ｚ，Ａ（ｊ－１））

【００１７】

Ｐ＿Ｈａｓｈは、必要な回数繰り返して、必要な量のデータを生成できる。ＡＮＳＩＸ９．６３と同様に、ＨＭＡＣタグの結果としての連接が、必要なデータ量よりも長い場合は、最終（最右端）バイトが切断される。

【００１８】

ＴＬＳ　ＰＲＦは下記のように定義される。

ＰＲＦ（Ｚ，ラベル，シード）＝Ｐ＿ＭＤ５（Ｓ１，ラベル‖シード）＋Ｐ＿ＳＨＡ－１（Ｓ２，ラベル‖シード）

ここで、通常のように、＋は、排他的論理和を表わし、‖は、連接を表わす。値Ｓ１とＳ２はオクテットストリング秘密Ｚを半々に分割することで得られ、左半分は、Ｓ１であり右半分は、Ｓ２であり、左半分が大きく、秘密は奇数個のオクテットを有する。

【００１９】

アルゴリズムで指定されるように、ＭＤ５の出力は、１６オクテットであり、一方、ＳＨＡ－１の出力は２０オクテットであり、関数Ｐ＿ＭＤ５は、一般的にはＰ＿ＳＨＡ－１よりも多くの繰り返しを使用する。

【００２０】

ＴＬＳ　ＰＲＦはＴＬＳプロトコルにおいては、幅広く使用される。例えば、プレマスター秘密からマスター秘密を導出するのに使用され、またマスター鍵から暗号鍵を導出などに使用される。

【００２１】

ＫＤＦに関する規格の不一致が、ＫＤＦなしのＤＨ鍵共有をサポートするか、単に限定された数のＫＤＦをサポートするかのどちらかに対する、モジュールの実践者への大きなインセンティブを作り出している。

【００２２】

標準公開鍵暗号規格（ＰＫＣＳ）＃１１：暗号トークンインタフェース（ｃｒｙｐｔｏｋｉ）は、私有鍵モジュールのクラスである、スマートカードのようなトークンのインタ

フェースをアドレスする。この規格においては、２、３のＫＤＦがサポートされるが、提供されるインタフェースは一般的には、ＫＤＦとの順応性がない。規格ＦＩＰＳ　１４０－２もまた私有鍵モジュールに対する条件を指定する。それは明示的に、未処理ＤＨ共有秘密値のような暗号値は、私有鍵モジュールの安全境界を離れてはならないことを要求するが、鍵導出に対する正確な機構は提供しない。

【００２３】

本発明の発明者は、静的ＤＨ私有鍵の不適切な再使用は、最終的には敵対者による私有鍵の再生という結果になり得ることを発見した。より正確にいえば、静的ＤＨ鍵共有を介して確立された共有秘密が、鍵導出関数（ＫＤＦ）の適用なしに使用された場合、敵対者は、複数の異なる共有鍵が確立されて使用される攻撃を仕掛けることができ、それにより静的ＤＨ私有鍵を再生できる。

【００２４】

本発明の発明者の最近の発見は、ＫＤＦなしにＤＨを実践できるという選択の余地があることは、安全性のリスクであり得ることを意味している。減少された数のＫＤＦをサポートすることはあまりにも限定的すぎる。例えば、新しいアプリケーション規格を使用するためだけにハードウェアのアップグレードが必要になることもある。

【００２５】

規格が鍵導出関数について一致しないため、ＤＨ私有鍵操作を実行するモジュールは、ある程度は、複数の異なるＫＤＦ規格をサポートしなければならない。１つのアプローチは、モジュールがすべてのＫＤＦアルゴリズムを実践することであるが、それは、モジュールが複数の異なるＫＤＦをサポートしなければならないので高価であり、モジュールは、新しいＫＤＦが登場したときにそれらをサポートできないので限定的である。反対のアプローチは、モジュールが保護されていないアクセスを未処理ＤＨ私有鍵操作に提供し、そのモジュールを使用するアプリケーションにＫＤＦを適用させることである。しかし、これは私有鍵を、最近発見された攻撃に対して脆弱にさせる。

【００２６】

本発明の目的は、上記の不都合な点を削除し、または緩和することである。

【発明の開示】

【００２７】

一般的に言えば、モジュールを使用するアプリケーションにより示されるように、本発明によりモジュールがＫＤＦアルゴリズムの一部を実行することを可能にする。これにより、モジュールが各必要なＫＤＦに対してＫＤＦ全体を実践する必要はなくなる。その代わり、ほとんどのＫＤＦに共通の再使用可能部分のみが実践される。更に、新しいＫＤＦが必要となったときに、それらがモジュールが実践したＫＤＦの部分上に構築されれば、それらをサポートできる。

【００２８】

このように、静的ＤＨ私有鍵操作への未処理アクセスは、一般的に安全性のリスクがあまりにも高すぎる傾向があるので、モジュール上では許可されない。その代わり、モジュールは、すべての予見可能なＫＤＦと同様に、すべての既存の関心対象であるＫＤＦをサポートするのに十分なほど順応性のあるインタフェースを提供する。これは、安全私有鍵モジュール上の既存および予見可能なＫＤＦの共通部分を実践することによりなされる。今日のほとんどのＫＤＦは、ハッシュ関数上に構築されている。従来は、ほとんどの私有鍵モジュールは、少なくともハッシュ関数を実践する必要があった。これは、ハッシュ関数が、デジタル署名のような、多くのアルゴリズムの安全性に対して非常に重要であるので、改竄防止を考慮するためにも重要である。

【００２９】

これに対する代替として、モジュールはまた、ＳＨＡ－１の圧縮関数へのアクセスを簡単に提供することもできる。アプリケーションは、この圧縮関数を使用して、いくつかの必要な埋め込み（Padding）をし、ある適切なチェインニング（Chaining）を行うだけで、ＳＨＡ－１を計算できる。これは更に、実践モジュールを単純化し、その順応性を高め

る。例えば、ある追加的順応性は、あるＡＮＳＩ決定的乱数生成子は、関数ＳＨＡ－１全体の代わりに、ＳＨＡ－１圧縮関数を使用するということである。より一般的には、鍵導出のような乱数生成は、一般的に秘密および秘密でない入力の混合に対するハッシュ関数値計算法の組み合わせを含む。従って、本発明は、複数のＫＤＦをサポートすることに限定されないだけでなく、複数の決定的乱数生成子もサポートできる。

【００３０】

更に高い順応性のために、モジュールは、ＳＨＡ－１圧縮関数のサブ演算のいくつかのような、より微小な演算をサポートできる。しかし、これらのサブ演算がＳＨＡ－１圧縮関数以外のある目的で再使用されるとは思われない。また、これら個々のサブ演算は、ＳＨＡ－１の完全な安全性は提供せず、従って、モジュール上の秘密をアプリケーションに明らかにする可能性もあり、それは避けなければならない。しかし、この原理に対する例外は、新しいハッシュ関数の２つの対である。これはＳＨＡ－３８４とＳＨＡ－５１２の対と、ＳＨＡ－２２４とＳＨＡ－２５６の対である。これらの対のそれぞれは、多くの共通部分を有しており、本質的に単一共通関数で実践できる。アプリケーションは、入力と出力を共通関数に対してのみ処理して、所望のハッシュ関数を得る。

【００３１】

ＴＬＳの用語においては擬似乱数関数（ＰＲＦ）として知られているＴＬＳ鍵導出の場合、２つのハッシュ関数が使用される。その１つはＳＨＡ－１で他の１つはＭＤ５である。ＰＲＦ－ＴＬＳを秘密Ｚに適用するために、秘密は２つの部分Ｓ１とＳ２に半々にされる。そして、ＭＤ５に基づくＰＲＦがＳ１に適用されて、ＳＨＡ１に基づく関数がＳ２に適用される。モジュールが、高価となる可能性のあるＭＤ５とＳＨＡ１両者の実践をしなくていいように、モジュールはアプリケーションにＳ１を明らかにする機構を提供し、Ｓ２をモジュール内に保持する。モジュールはＳ２についてのＳＨＡ１の計算を行い、アプリケーションはＳ１に対してＭＤ５の計算を行うことができる。

【００３２】

ＴＬＳ内の１つ以外の他のＫＤＦはそのような方法で秘密を分割することは予想されないが、規格が今後どのような方向に進むかを予測することは困難になる傾向がある。従って、モジュールが秘密を分割する一般的方法をサポートすることは有効である。従って、モジュールに対するインタフェースは、アプリケーションが秘密のその部分が公開されるように要求できる機構を含む。十分な秘密が、秘密のまま残り、アプリケーションが秘密の異なる部分に対する複数の要求ができないような方法でモジュールは実践される。

【００３３】

新しい規格は出現し続けているので、また、規格はＫＤＦと乱数生成子を再設計し続けているので、ハードウェアモジュールへの順応性があり、安全なインタフェースは、モジュールの使用可能性の拡大に重要な価値を提供する。そうでなければ、モジュールはあまりにも早くに廃れてしまう危険性がある。

【発明を実施するための最良の形態】

【００３４】

下記の説明と、そこで説明される実施形態は、本発明の原理の特別な実施形態の例を示すことにより提供される。これらの例は、本発明のそれらの原理を説明する目的のため提供されるのであって、限定的ではない。下記の説明では、類似の部分には、本明細書と図を通してそれぞれ、同一の参照番号が記される。

【００３５】

図１を参照すると、私有鍵モジュール装置５０で保護された、ユーザー装置４０と接続先１００の間の接続が示されている。ユーザー装置４０と接続先１００の間の接続は、一般的には安全ではなく、オープンである。例えば、接続は、インターネットなどの公衆ネットワーク８０へのリンク７０と、公衆ネットワークから接続先１００へのリンク９０から構成されている。それぞれのリンクは、有線リンクであっても、ワイヤレスリンクであっても、その両者の組み合わせであってもよい。一般的には、私有鍵モジュール装置５０は、スマートカードやトークンのような、現地の装置に挿入できる自己完結型装置である

か、またはアプリケーションが起動されるユーザー装置４０である。モジュール装置５０は、アプリケーションにより起動されると、ユーザー装置４０と協働して、リンク７０上の通信を保護する。

【００３６】

この作動モードにおいて、私有鍵モジュール装置５０は、ユーザー装置４０と接続先装置１００の間の接続を保護するための私有鍵機能を提供する。しかし、私有鍵モジュール装置５０は、カスタム私有鍵モジュールなので、ユーザー装置４０のような典型的なユーザーコンピュータの保護とは別のある追加的な保護を必要とする。鍵導出関数（ＫＤＦ）を、ユーザー装置４０で起動しているアプリケーションで部分的に、私有鍵モジュール装置５０上で実行されているモジュール内で部分的に実践することにより、安全性が強化される。ユーザー装置４０と私有鍵モジュール装置５０がここでは別個の装置として記述されているが、それらは単一の物理的装置に統合できるということは理解されよう。例えば、私有鍵モジュール装置５０は、特別埋め込みチップセットとしてユーザー装置４０上に常駐できる。

【００３７】

ユーザー装置４０は典型的に複数のアプリケーションを起動し、ＣＰＵ４２とメモリ装置４４を利用して異なる機能を実行する。ユーザー装置４０は、ＣＰＵ４２上で起動している通信アプリケーションの指令のもとで、リンク７０を管理するための通信モジュール４５を含む。安全な通信を確立するために、通信アプリケーションは、ＫＤＦのような私有鍵機能を要求する、上記に検討したプロトコルの１つのような、確立された保護プロトコルを実践する。順応性を維持しながら、選択されたＫＤＦの計算を促進するために、ＫＤＦの導出は、離散したサブルーチンに分離され、私有鍵上での作動を要求するサブルーチンが私有鍵モジュール５０により実行される。ユーザー装置４０によりバランスが取られ、それにより未処理私有鍵データは、ユーザー装置４０を通してアクセスされない。

【００３８】

図２を参照すると、ユーザー装置４０上で起動しているアプリケーション１０で部分的に実践され、私有鍵モジュール５０上で起動しているアプリケーション２０で部分的に実践される鍵導出関数（ＫＤＦ）を有する保護システムの実践例が示されている。ＫＤＦは２つの部分に分割される。私有鍵モジュール５０は、ＫＤＦの構成要素２４を生成し、アプリケーション１０はこれらの構成要素を使用してＫＤＦのバランス２２を計算する。私有鍵モジュール２０は、データを交換し、アプリケーション１０と通信するモジュールインタフェース２６を有している。モジュールインタフェース２６は更に２つのインタフェース関数、第１インタフェース関数２８と、第２インタフェース関数３０を有している。

【００３９】

有利なことであるが、ディフィーヘルマン（Diffie-Hellman）共有秘密値Ｚのような、ある秘密値は、私有鍵モジュール２０内で判定される。Ｚの長さはアプリケーション１０に知られているが、Ｚの値は知られていない。アプリケーション１０は、ハンドルを有しており、それにより、秘密Ｚを参照することができ、このように私有鍵モジュール２０にＺから値を導出することを依頼できる。

【００４０】

第１インタフェース関数２８は、整数と、秘密Ｚのハンドルから構成される入力を有している。この整数は、アプリケーション１０に開示されるＺのオクテット数を定義する。これはＴＬＳ　ＰＲＦ内のＳ１値である。この機能を実行するときは、私有鍵モジュール２０は、秘密の最小数のオクテットをＳ２として保持させることができ、それにより、アプリケーション１０は、全体の秘密を知ることはない。この最小数は、アプリケーションの意図された保護レベルに対して適切に選択される。８０ビットの保護レベルに対しては１０オクテットであってもよい。第１インタフェース関数２８がいったん呼び出されると、秘密は永久的にＳ２に切断され、私有鍵モジュール２０は、Ｓ２の更なる切断を許可しない。Ｓ２を参照するためのハンドルまたはポインタがアプリケーション１０に設けられる。好ましくは、Ｚが更なる計算では使用されないので、Ｚを参照するハンドルまたはポ

インタは再使用される。この後、私有鍵モジュール２０は、第１インタフェース関数２８が呼び出された後に、秘密Ｚ＝Ｓ２を設定する。随意に、私有鍵モジュール２０は、単にＳ２を指し示している新しいハンドルを作成して、この新しいハンドルをアプリケーション１０に出力して、アプリケーション１０が後でＳ２を参照できるようにすることができる。値Ｓ１は常に第１インタフェース関数２８の出力の一部であり、それにより、アプリケーション１０、つまり、アプリケーション１０に含まれるＫＤＦの第１部分２２は、ＴＬＳ　ＰＲＦ内で使用されるＭＤ５計算のような、必要ないかなる計算もＳ１に対して実行できる。

【００４１】

第２インタフェース関数３０は、２つの値ＸとＹと秘密Ｚのハンドルから構成される入力を有する。第１値は、秘密Ｚと同一の長さのオクテットストリングである。第２インタフェース関数３０の出力は下記の通りである。

ＳＨＡ－１（Ｘ＋Ｚ‖Ｙ）

【００４２】

第２インタフェース関数３０は、ＡＮＳＩ　Ｘ９.６３　ＫＤＦとＴＬＳ　ＰＲＦの両者が構築できる基本的暗号動作である。第１インタフェース関数２８の出力Ｓ１と、第２インタフェース関数３０の出力、つまり、ＳＨＡ－１のハッシュ値から、アプリケーション１０は、ＫＤＦ計算を完結でき、鍵を導出できる。

【００４３】

ユーザー装置４０は一般的に、ＣＰＵ４２と、ＣＰＵ４２にアクセス可能なメモリ装置４４と、これもまたＣＰＵ２０にアクセス可能な格納媒体４６と、ある入力および出力装置（図示せず）を有する。理解されるであろうが、ユーザー装置４０は、他のプログラム可能な計算装置であってもよい。アプリケーション１０は、ＣＰＵ４２上で実行される。アプリケーション１０は、格納媒体４６上に格納でき、ユーザー装置４０に永久的に設置してもよく、あるいはユーザー装置４０から取外しできてもよく、あるいは、ユーザー装置４０にリモートアクセスできてもよい。アプリケーション１０はまた、ＣＰＵ４２に直接搭載することもできる。ＫＤＦの出力は、ユーザー装置４０から接続先１００への接続を保護するために必要である。

【００４４】

私有鍵モジュール５０は一般的に、ＣＰＵまたはマイクロプロセッサ５２と、ＣＰＵ５２にアクセス可能なメモリ装置５４と、これもまたＣＰＵ５２にアクセス可能な格納媒体５６を有する。私有鍵モジュール２０はＣＰＵ５２上で実行される。私有鍵モジュール５０は、格納媒体５６上に格納してもよく、またはメモリ装置５２に直接搭載してもよい。私有鍵モジュール５０は秘密私有鍵を、そのメモリ装置５４またはその格納媒体５６内に格納できる。理解されるであろうが、私有鍵モジュール５０は、私有鍵モジュール５０がキーボード付きのスマートカードである場合のキーボードのような、ユーザーが秘密私有鍵を入力する入力手段を有してもよい。

【００４５】

より揮発性の高いデータを格納するために使用される傾向にあるメモリ装置５４と、より永続的なデータを格納するために使用される傾向にある格納媒体５６を、ここでは区別してきたが、私有鍵モジュール５０は、揮発性データおよび永続性データの両者を格納するための単一のデータ格納装置のみを有してもよい。同様に、ユーザー装置４０は、揮発性データおよび永続性データの両者を格納するための単一のデータ格納装置のみを有してもよい。

【００４６】

データリンク６０は、必要なときに、アプリケーション１０と私有鍵モジュール５０の間の通信チャネルを提供する。データリンク６０は、有線でも、ワイヤレスでもよい。それはユーザー装置４０と私有鍵モジュール５０間の直接接続であってもよい。データリンク６０は永続的であってもよく、より好ましくは、要求により確立される接続である。一般的に、データリンク６０は、オープンリンクではなく、保護されているリンクである。

【００４７】
上記したように、私有鍵モジュール２０は全体のＫＤＦを実践しない。私有鍵モジュール５０内で生成されたＫＤＦの構成要素２４は、再使用可能な部分のみと、安全性に基本的な暗号操作を行う部分しか実践しない。これにより、安全性を損ねることなく順応性を促進できる。例えば、ＤＨプロトコルを実践するときは、静的ＤＨ私有鍵操作への未処理アクセスはモジュール上では許可されない。その代わりモジュールは、すべての予見可能なＫＤＦと同様に、すべての関心対象の既存ＫＤＦをサポートするのに十分順応性のあるインタフェースを提供する。これを最も効率的に行う１つの方法は、既存および予見可能なＫＤＦの共通部分を実践することである。今日のほとんどのＫＤＦは、将来はそのいくつかはブロック暗号から構築されるであろうことは予見されるが、ハッシュ関数上で構築されている。ほとんどの私有鍵モジュールは、少なくともハッシュ関数をサポートすべきであり、これはデジタル署名のような、多くのアルゴリズムの安全性に対してハッシュ関数が非常に重要であることによる。幸いなことに、規格化されているハッシュ関数の数は、ＫＤＦよりも少ない。例えば、ハッシュ関数ＳＨＡ－１は、別個のＡＮＳＩ、ＩＰＳｅｃ、およびＴＬＳ鍵導出関数のような、いくつかの異なるＫＤＦをサポートするために再使用できる。ＴＬＳ鍵導出はまた、別のハッシュ関数であるＭＤ５を使用するが、これは、下記に更に説明するように、モジュール５０の外部で扱うことができる。

【００４８】
図３を参照すると、ＳＨＡ－１操作を使用して生成されたＫＤＦに対して、アプリケーション１０は、私有鍵モジュール５０に、ハッシュ関数への入力としてどの入力を供給すべきかを指令する。入力のいくつかは、秘密であり、アプリケーションには知らされない。これを指定するために、アプリケーション１０は、ハンドルまたはポインタ５７を介してそのような秘密入力を参照する。公開入力がアプリケーション１０により直接提供されてもよい。各ＫＤＦ独自の入力のフォーマットは、モジュールにより提供される一般的フォーマットインタフェースにより指定される。私有鍵モジュール５０がアプリケーション１０に提供するハッシュ出力は、より多くのハッシュ関数の呼び出しへの更なる入力としてアプリケーション１０により再使用することもできる。これは、多くのＫＤＦが、１つのハッシュ呼び出しの出力が、別のハッシュ呼び出しの入力に供給されるチェーニング機構に基づいているからである。

【００４９】
１つの実施形態において、私有鍵モジュール５０は、ＳＨＡ－１の実践と単純インタフェースを含む。代替実施形態においては、私有鍵モジュール２０は、汎用目的の実行環境を含む。

【００５０】
または、インタフェースが実践されて、それにより、モジュールがＴＬＳ　ＰＲＦとＡＮＳＩ　Ｘ９.６３　ＫＤＦの両者を、過度に未処理私有鍵操作を開示することなく（それにより本発明の発明者により発見された攻撃を回避する）サポートできる。

【００５１】
ＡＮＳＩ　Ｘ９.６３　ＫＤＦとＴＬＳ　ＰＲＦのサポートにおける操作において、ＡＮＳＩ　Ｘ９.６３　ＫＤＦは、ハッシュ関数ＳＨＡ－１から、共有秘密値に基づいて計算された一連のハッシュ値を計算し、ハッシュ値の連接から形成されるオクテットストリングを切断することにより、共有秘密値から鍵を導出し、一方、ＴＬＳ　ＰＲＦは、ハッシュ関数ＭＤ５とハッシュ関数ＳＨＡ－１の両者の計算を含む、はるかに複雑な構造を有している。

【００５２】
モジュールインタフェース２６の目的は、ハッシュ関数ＭＤ５を実践しないことである。ハッシュ関数ＳＨＡ－１のみが私有鍵モジュール２０上で、つまり、ＫＤＦの第２部分２４上で実践される。従って、私有鍵モジュール２０を使用するアプリケーション１０は、ＫＤＦのその第１部分２２においてＭＤ５を実践することに責任がある。安全性の観点からは、これは重大な欠点とはならない。これは、ＭＤ５ハッシュ関数は、十分な安全性

を提供するとは普遍的には考えられていないからであり、一方では、ＳＨＡ－１ハッシュ関数は、最高の安全性レベル（これらの高いレベルはＳＨＡ－２５６またはＳＨＡ－１の別の後継プロトコルを必要とする）以外のすべてのレベルに対して、鍵導出の目的のために、十分な安全性を提供することが普遍的に受け入れられる傾向にあるからである。

【００５３】

ＡＮＳＩ　Ｘ９.６３　ＫＤＦのサポートにおける操作の概要は、図４に示されている。そのような操作においては、アプリケーション１０はＸ＝０とＹ＝ｊ‖[SharedInfo]を選択し、ここでｊは、アプリケーションが維持する４オクテットカウンタである。アプリケーション１０は、その後、Ｘ、Ｙと、Ｚに対するハンドルで機能３０を呼び出す。私有鍵モジュール５０のアプリケーション２０は、その後、アプリケーション１０により供給されたＸとＹに対する値と、Ｚに対するハンドルを使用して、上述され、図４に示された式に従ってＳＨＡ－１を計算する。アプリケーション１０は、これにより計算されたＳＨＡ－１値を得ることができ、これを使用して、ＡＮＳＩ　Ｘ９.６３　ＫＤＦを構築して鍵を導出する。

【００５４】

ＴＬＳ　ＰＲＦのサポートにおけるアプリケーション１０と２０の操作は、図５に示されている。アプリケーション１０は、関数２８に関して上述したように、共有秘密ＺをＳ１とＳ２に半々にする（図５のパート１）ために、第１インタフェース関数２８を呼び出す。アプリケーション１０はこの後、第２インタフェース関数３０を呼び出して、Ｓ２に基づいてハッシュ値を計算し（図５のパート２）、この上述の構成を使用して、第１および第２インタフェース関数２８、３０の出力からＰ＿ＳＨＡ－１を計算する（図５のパート３）。パート２と３は下記に説明する。

【００５５】

図５に示されたＴＬＳ－ＰＲＦの操作のパート２で使用された関数ＨＭＡＣ－ＳＨＡ－１を構築するために、アプリケーション１０はまず、Ｘ＝ＤとＹ＝Ｍと、鍵Ｋに対するハンドルで第２インタフェース関数３０を呼び出し、それによりＴ１＝ＳＨＡ－１（（Ｄ＋Ｋ）‖Ｍ）が与えられる（Ｄの値は、一定であると一般に知られており、従って、アプリケーション１０に利用できる）。そして、アプリケーション１０は、Ｋに対する同一ハンドルでＸ＝ＣとＹ＝Ｔを設定して、Ｔ＝ＳＨＡ－１（Ｃ＋Ｋ）‖Ｔ１）＝ＨＭＡＣ－ＳＨＡ（Ｋ，Ｍ）を得る（Ｃの値は、Ｄのように公開である）。

【００５６】

鍵Ｋをゼロビットで埋める必要がある場合は、アプリケーション１０が、第２入力Ｙに、定数ＣとＤの適切なオクテットとの排他的論理和が演算された必要なゼロビットを提供することでこれを行う。鍵Ｋが最初に圧縮が必要なほど長い場合は、アプリケーション１０は、これをＸ＝０とＹ＝０と設定してハッシュ鍵を得ることで行うことができる。この場合、アプリケーション１０は、必要な情報をすべて有しているので、随意的にそれ自身上で計算の残りを実行することができ、または更に第３インタフェース機能を使用して、上記のハッシュ出力を、新しいハンドルの別の秘密として指定できる。

【００５７】

図３に示されたＴＬＳ－ＰＲＦをサポートする操作のパート３において関数Ｐ＿ＳＨＡ－１を構築するために、アプリケーション１０は、今度はパート１での出力として提供されたＳ１と、上記の構成を使用して、秘密鍵が私有鍵モジュール２０に閉じ込められている、ＨＭＡＣ＿ＳＨＡ－１を計算する。これは、ＨＭＡＣ＿ＳＨＡ－１を繰り返し適用することでＡ（０）、Ａ（１）、Ａ（２）を計算することを含み、それらはその結果、ＨＭＡＣ＿ＳＨＡ－１を更に適用することでＰ＿ＳＨＡ－１の出力を形成する。

【００５８】

出力Ｐ＿ＳＨＡ－１は、ＫＤＦを構築して、鍵を導出するのに使用される。

【００５９】

上記の例は、私有鍵モジュール２０内で導出された鍵はアプリケーション１０への出力として導出されると仮定している。これに対する代替例では、導出された鍵は、私有鍵モ

ジュール２０内に留まり、出力はその鍵への単なるハンドルまたはポインタである。この優位点は、すべての鍵が私有鍵モジュール２０上に保持できることであり、それによりモジュールの所有者に、アプリケーションが、長期私有鍵は言うまでもなく、導出されたセッション鍵さえも乱用できないことを更に確信させる。

【００６０】

代替実施形態において、私有鍵モジュール２０は、更に高いレベルの順応性を有している。私有鍵モジュール２０は、ジャバスクリプト（javascript）またはジャバ（java）のような、ある単純な実行言語をサポートしてもよく、それにより、カード上で実行される操作に広大な普遍性を与える。つまり、アプリケーション１０はプログラムを私有鍵モジュール２０に供給して、それを私有鍵モジュール２０が実行する。プログラムは、モジュールにあるときは、秘密に自由にアクセスできる。安全のために、私有鍵モジュール２０は、モジュールからのすべての出力が、ＳＨＡ－１のようなハッシュアルゴリズムや、ＡＥＳのような対称暗号操作の一部のような、認可されている安全性アルゴリズムを経ることを確実にする。これにより、悪意あるプログラムが試みることができるほとんどの悪用を防止できる。

【００６１】

安全性を更に高めるために、私有鍵モジュール２０は、プログラムが、その公開検証鍵が既に安全に私有鍵モジュール２０にロードされている署名者によるデジタル署名を要求する。これは、私有鍵モジュール２０にロードされたプログラムを認証する１つの方法である。プログラムの認証により、プログラムが、モジュールの秘密を危険に晒すという目的の悪意ある実行プログラムでないことが保証される。プログラムの認証により、プログラムそれ自身が任意のアルゴリズムを実行するのに十分なほど信頼できるので、モジュール入力をあるハッシュ、または他のアルゴリズムに制限する必要がなくなる。

【００６２】

この代替実施形態の、第１実施形態に対する優位点は、それが、既存または新しい多様なハッシュをモジュール上で実行することを可能にするように、より高い順応性を提供することである。不都合な点は、モジュールが、一般実行言語と、場合によっては公開鍵インフラの一部、をサポートする必要があるということである。

【００６３】

本発明の種々の実施形態が詳細に記述された。当業者は、本発明の範囲から逸脱することなく、実施形態に対して、多くの修正、適合、および変形が可能であることを理解するであろう。上述した最も好適な実施形態における、またはそれに付け加える変更は、本発明の本質、精神、および範囲から逸脱することなく可能であるので、本発明はそれらの詳細には限定されず、付随する請求項にのみ限定される。

【図面の簡単な説明】

【００６４】

【図１】図１は、私有鍵モジュールにより保護されたユーザー装置と接続先との間の接続を示すブロック図である。

【図２】図２は、図１に示されたユーザー装置と、私有鍵モジュールにおける鍵導出関数の実践を示している模式図である。

【図３】図３は、私有鍵モジュール装置を示している模式図である。

【図４】図４は、鍵導出関数の１つの例を示しているフローチャートである。

【図５】図５は、鍵導出関数の別の例を示しているフローチャートである。

【符号の説明】

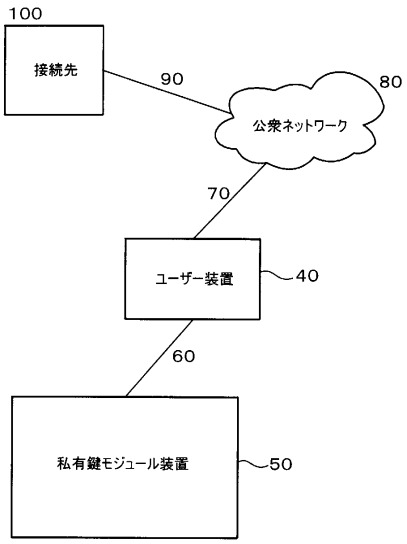
【００６５】

１０　アプリケーション
２０　アプリケーション
４０　ユーザー装置
５０　私有鍵モジュール装置
６０　データリンク

７０　リンク
８０　公衆ネットワーク
９０　リンク
１００　接続

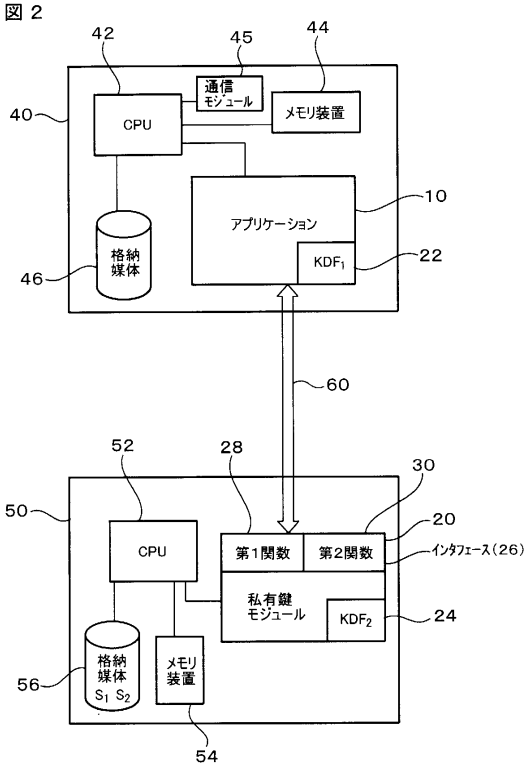
【図１】

図1



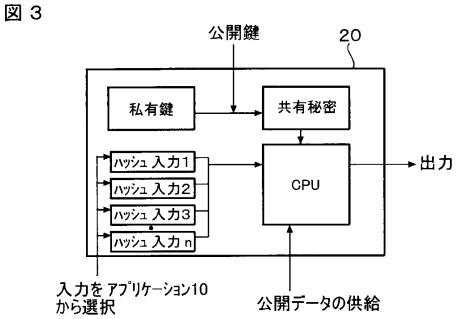
【図２】

図２



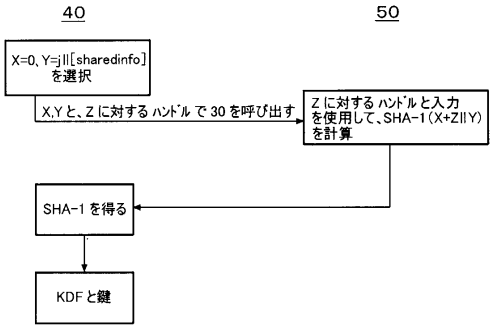
【図３】

図３



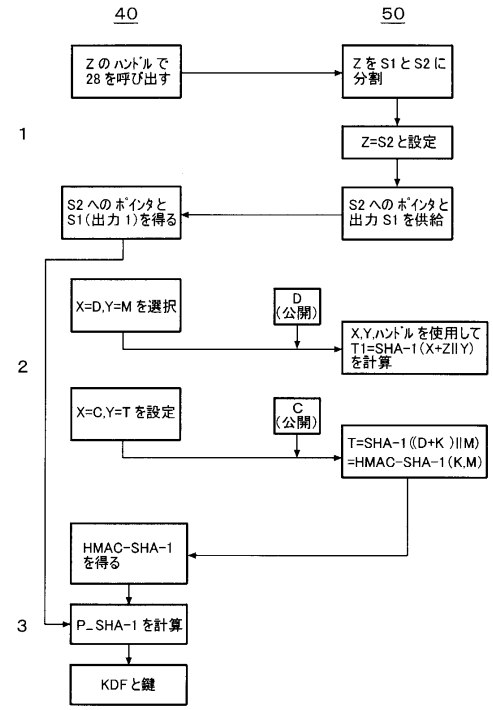
【図４】

図４



【図５】

図５



フロントページの続き

(72)発明者　ギャラント，　ロバート　ピー.
カナダ国　オンタリオ州　エル５エム　５エヌ１　ミシソーガ　ローズブッシュロード　４７８８
(72)発明者　ヴァンストーン，　スコット　エー.
カナダ国　オンタリオ州　エル０ピー　１ビー０　キャンプベルヴィル　ピー．オー．ボックス　４９０　パインビュートレイル　１０１４０

審査官　石田　信行

(56)参考文献　特開平０３－０７２７３７（ＪＰ，Ａ）
特開平０６－０１９３９３（ＪＰ，Ａ）
特開２００４－２９７５７８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
H04L　9/08
G09C　1/00