(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2004-530346
(P2004-530346A)
(43)公表日　平成16年9月30日(2004.9.30)

(51)Int.Cl.[7]	F I		テーマコード（参考）
H04L 9/10	H04L 9/00	621A	5B017
G06F 12/14	G06F 12/14	320B	5J104
	G06F 12/14	540B	
	G06F 12/14	540P	

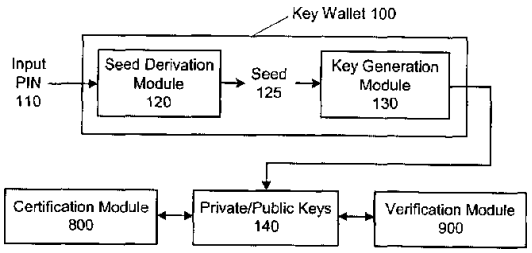
審査請求 未請求　予備審査請求 有　（全 58 頁）

(21)出願番号	特願2002-577330(P2002-577330)
(86)(22)出願日	平成14年3月29日(2002.3.29)
(85)翻訳文提出日	平成15年9月29日(2003.9.29)
(86)国際出願番号	PCT/US2002/009812
(87)国際公開番号	WO2002/080445
(87)国際公開日	平成14年10月10日(2002.10.10)
(31)優先権主張番号	60/280,629
(32)優先日	平成13年3月29日(2001.3.29)
(33)優先権主張国	米国(US)
(31)優先権主張番号	09/874,795
(32)優先日	平成13年6月5日(2001.6.5)
(33)優先権主張国	米国(US)

(71)出願人　500297557
アルコット　システムズ　インコーポレイテッド
アメリカ合衆国　９５０５４　カリフォルニア州　サンタ　クララ　パトリック　ヘンリー　ドライブ　３２００　スイート　２００
(74)代理人　100077481
弁理士　谷　義一
(74)代理人　100088915
弁理士　阿部　和夫
(72)発明者　ジェフリー　アール．ハード
アメリカ合衆国　９５０１４　カリフォルニア州　クパチーノ　パークウッド　ドライブ　１０２７４　アパートメント　４
最終頁に続く

(54)【発明の名称】セキュアな暗号鍵の生成、証明、および使用のための方法および装置

(57)【要約】
公開鍵署名システムにおいて使用される秘密鍵（１４０）などの秘密データが、「生成カモフラージュ」技術を使用してデジタルウォレット（１００）の中でセキュアにされる。この技術では、秘密鍵（１４０）は、暗号化された形態でさえ、デジタルウォレット（１００）の中に必ずしも記憶される必要がない。代わりに、ウォレットは、ユーザが自身のあらかじめ選択されたＰＩＮを入力した場合に正しい秘密鍵を再生成する秘密鍵生成ファンクション（１３０）を含む。ユーザが誤ったＰＩＮを入力した場合、誤った秘密鍵（１４０）が出力される。そのような秘密鍵（１４０）は、秘密鍵フォーマットの使用、および／または秘密鍵（１４０）に対応する擬似公開鍵の使用を介して正しい秘密鍵（１４０）と容易に区別することができないように構成することができる。本明細書で説明する技術は、秘密鍵以外の他の形態の再生成可能な秘密データにも適用可能である。



【特許請求の範囲】
【請求項１】
ユーザのアクセスコードを使用してセキュアにされた、前記ユーザのために秘密データを再生成するためのデジタルウォレットであって、
（ａ）入力アクセスコードを受け取るためのコンピュータによって実装される入力部と、
（ｂ）前記秘密データの少なくとも一部分を生成するのに使用可能なシードを導出するための前記入力部に動作上、接続されたシード導出モジュールと、
（ｃ）（ｉ）前記ユーザの前記秘密データのシードベースの初期設定によって以前に使用された所定のデータ生成プロトコルを実施し、
（ｉｉ）シード－アクセスコード関係の表現を含み、
（ｉｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードをデジタル処理することによって出力データを生成するように構成され、
（ｉｖ）前記出力データは、前記入力アクセスコードが前記ユーザのアクセスコードに等しい場合、前記ユーザの秘密データの前記少なくとも一部分を再生成するシードベースのデータ生成モジュールと、
（ｄ）前記出力データの前記生成が、前記秘密データの前記少なくとも一部分のいずれの形態のいずれの記憶にも依存せずに行われるステップと
を含むことを特徴とするウォレット。
【請求項２】
前記出力データは、前記入力アクセスコードが前記ユーザのアクセスコードに等しくない場合、前記ユーザの秘密データの前記少なくとも一部分を再生成しないことを特徴とする請求項１に記載のウォレット。
【請求項３】
前記出力データは、前記秘密データの前記少なくとも一部分の特徴的外観を有することを特徴とする請求項２に記載のウォレット。
【請求項４】
前記アクセスコードは、ＰＩＮであり、前記秘密データは、非対称暗号鍵を含むことを特徴とする請求項１に記載のウォレット。
【請求項５】
前記出力データは、非対称暗号鍵の特徴的外観を有することを特徴とする請求項４に記載のウォレット。
【請求項６】
前記アクセスコードは、ＰＩＮであり、前記秘密データは、対称暗号鍵を含むことを特徴とする請求項１に記載のウォレット。
【請求項７】
前記シード－アクセスコード関係は、前記導出されたシードが前記入力アクセスコードに等しいように恒等関係（ｉｄｅｎｔｉｔｙ　ｒｅｌａｔｉｏｎｓｈｉｐ）であることを特徴とする請求項１に記載のウォレット。
【請求項８】
前記シード－アクセスコード関係は、前記入力アクセスコードの埋込みが行われたバージョンとして前記導出されたシードを表現することを特徴とする請求項１に記載のウォレット。
【請求項９】
前記シード－アクセスコード関係は、ユーザのアクセスコードによってマスクされた前記初期シードのバージョンを含むことを特徴とする請求項１に記載のウォレット。
【請求項１０】
（ｉ）前記初期シードの前記マスクされたバージョンは、前記初期シードと前記ユーザのアクセスコードのＸＯＲを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記マスクされたバージョンを前記導出されたシードとＸ

ＯＲ演算するステップを含むことを特徴とする請求項９に記載のウォレット。
【請求項１１】
前記初期シードの前記記憶されているマスクされたバージョンをユーザの新しいアクセスコードとＸＯＲ演算されたユーザの古いアクセスコードとＸＯＲ演算された前記初期シードの前記記憶されているマスクされたバージョンの値で置き換えることによって前記ユーザの新しいアクセスコードで前記ユーザの古いアクセスコードを更新するためのプログラムコードをさらに含むことを特徴とする請求項１０に記載のウォレット。
【請求項１２】
（ｉ）前記シード－アクセスコード関係は、前記導出されたシードを形成するように前記入力アクセスコードと連結することができる前記初期シードの切り捨てられた（ｔｒｕｎｃａｔｅｄ）バージョンを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記切り捨てられたバージョンを前記入力アクセスコードと連結するステップを含むことを特徴とする請求項１に記載のウォレット。
【請求項１３】
（ｉ）前記シード－アクセスコード関係は、前記入力アクセスコードの複数の可能な値と前記導出されたシードの対応する複数の可能な値の値、ならびに前記入力アクセスコードの前記複数の可能な値と前記導出されたシードの対応する前記複数の可能な値の間の関連を含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記入力アクセスコードに対応する前記導出されたシードの前記可能な値をルックアップし、出力するステップを含むことを特徴とする請求項１に記載のウォレット。
【請求項１４】
（１）前記シード導出モジュールが、前記データ生成モジュールと合併され、
（２）前記出力データは、前記導出されたシードを含むことを特徴とする請求項１３に記載のウォレット。
【請求項１５】
前記秘密データは、前記ユーザの秘密鍵を含み、前記出力データは、秘密鍵の特徴的外観を有することを特徴とする請求項５に記載のウォレット。
【請求項１６】
前記ユーザの秘密鍵に対応する前記ユーザの公開鍵は、擬似公開されていることを特徴とする請求項５に記載のウォレット。
【請求項１７】
前記擬似公開鍵を含むデジタル証明書をさらに含むことを特徴とする請求項１６に記載のウォレット。
【請求項１８】
前記デジタル証明書は、許可を受けた検証者による以外は検証可能でない証明者の鍵の下で暗号化された前記ユーザの擬似公開鍵の暗号化されたバージョンを含むことを特徴とする請求項１７に記載のウォレット。
【請求項１９】
ネットワークを介してローミングする（ｒｏａｍｉｎｇ）ユーザが遠隔でアクセスすることができるように構成されることを特徴とする請求項１に記載のウォレット。
【請求項２０】
前記ユーザのために秘密データをセキュアに記憶し、再生成するためのコンピュータによって実施される方法であって、
（ａ）入力アクセスコードを受け取るステップと、
（ｂ）前記受け取られた入力アクセスコードを使用することによって前記秘密データの少なくとも一部分を生成するように使用可能なシードを導出するステップと、
（ｃ）シード－アクセスコード関係の表現を獲得するステップと、

（ｄ）（ｉ）前記シード－アクセスコード関係に従い、
（ｉｉ）前記ユーザの前記秘密データのシードベースの初期設定によって以前に使用された所定のデータ生成プロトコルを実行することにより、
（ｉｉｉ）前記入力アクセスコードが前記ユーザのアクセスコードに等しい場合、前記ユーザの秘密データの前記少なくとも一部分を再生成して出力データを生成する前記導出されたシードをデジタル処理するステップと、
（ｅ）前記出力データの前記生成が、前記秘密データの前記少なくとも一部分のいずれの形態のいずれの記憶にも依存せずに行われるステップと
を含むことを特徴とする方法。
【請求項２１】
前記出力データは、前記入力アクセスコードが前記ユーザのアクセスコードに等しくない場合、前記ユーザの秘密データの前記少なくとも一部分を再生成しないことを特徴とする請求項２０に記載の方法。
【請求項２２】
前記出力データは、前記秘密データの前記少なくとも一部分の特徴的外観を有することを特徴とする請求項２１に記載の方法。
【請求項２３】
前記アクセスコードは、ＰＩＮであり、前記秘密データは、非対称暗号鍵を含むことを特徴とする請求項２０に記載の方法。
【請求項２４】
前記シード－アクセスコード関係は、前記導出されたシードが前記入力アクセスコードに等しいように恒等関係であることを特徴とする請求項２０に記載の方法。
【請求項２５】
前記シード－アクセスコード関係は、前記入力アクセスコードの埋込みが行われたバージョンとして前記導出されたシードを表現することを特徴とする請求項２０に記載の方法。
【請求項２６】
前記シード－アクセスコード関係は、ユーザのアクセスコードによってマスクされた前記初期シードのバージョンを含むことを特徴とする請求項２０に記載の方法。
【請求項２７】
（ｉ）前記初期シードの前記マスクされたバージョンは、前記初期シードと前記ユーザのアクセスコードのＸＯＲを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記マスクされたバージョンを前記導出されたシードとＸＯＲ演算するステップを含むことを特徴とする請求項２６に記載の方法。
【請求項２８】
（ｉ）前記シード－アクセスコード関係は、前記導出されたシードを形成するように前記入力アクセスコードと連結することができる前記初期シードの切り捨てられた（ｔｒｕｎｃａｔｅｄ）バージョンを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記切り捨てられたバージョンを前記入力アクセスコードと連結するステップを含むことを特徴とする請求項２０に記載の方法。
【請求項２９】
（ｉ）前記シード－アクセスコード関係は、前記入力アクセスコードの複数の可能な値と前記導出されたシードの対応する複数の可能な値の値、ならびに前記入力アクセスコードの前記複数の可能な値と前記導出されたシードの対応する前記複数の可能な値の間の関連を含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記入力アクセスコードに対応する前記導出されたシードの前記可能な値をルックアップし、出力するステップを含むことを特徴とする請求項２０に記載の方法。
【請求項３０】

（１）前記シードを前記導出するステップと前記所定のデータ生成プロトコルを前記実施するステップが、共通の動作に合併され、
（２）前記出力データは、前記導出されたシードを含むことを特徴とする請求項２９に記載の方法。
【請求項３１】
（ａ）入力アクセスコードを受け取るためのコンピュータ論理命令と、
（ｂ）前記受け取られた入力アクセスコードを使用することによって前記秘密データの少なくとも一部分を生成するように使用可能なシードを導出するためのコンピュータ論理命令と、
（ｃ）シード－アクセスコード関係の表現を獲得するためのコンピュータ論理命令と、
（ｄ）（ｉ）前記シード－アクセスコード関係に従い、
（ｉｉ）前記ユーザの前記秘密データの少なくとも一部分のシードベースの初期設定によって以前に使用された所定のデータ生成プロトコルを実行することにより、
（ｉｉｉ）前記入力アクセスコードが前記ユーザのアクセスコードに等しい場合、前記ユーザの秘密データの前記少なくとも一部分を再生成して出力データを生成する前記導出されたシードをデジタル処理するためのコンピュータ論理命令と、
（ｅ）前記出力データの前記生成が、前記秘密データの前記少なくとも一部分のいずれの形態のいずれの記憶にも依存せずに行われるためのコンピュータ論理命令とを含む前記ユーザのために秘密データをセキュアに記憶し、再生成するようにコンピュータ上で実行可能なプログラムを記憶していることを特徴とするコンピュータ可読媒体。
【請求項３２】
前記出力データは、前記入力アクセスコードが前記ユーザのアクセスコードに等しくない場合、前記ユーザの秘密データの前記少なくとも一部分を再生成しないことを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項３３】
前記出力データは、前記秘密データの前記少なくとも一部分の特徴的外観を有することを特徴とする請求項３２に記載のコンピュータ可読媒体。
【請求項３４】
前記アクセスコードは、ＰＩＮであり、前記秘密データは、非対称暗号鍵を含むことを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項３５】
前記シード－アクセスコード関係は、前記導出されたシードが前記入力アクセスコードに等しいように恒等関係であることを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項３６】
前記シード－アクセスコード関係は、前記入力アクセスコードの埋込みが行われたバージョンとして前記導出されたシードを表現することを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項３７】
前記シード－アクセスコード関係は、ユーザのアクセスコードによってマスクされた前記初期シードのバージョンを含むことを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項３８】
（ｉ）前記初期シードの前記マスクされたバージョンは、前記初期シードと前記ユーザのアクセスコードのＸＯＲを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記マスクされたバージョンを前記導出されたシードとＸＯＲ演算するステップを含むことを特徴とする請求項３７に記載のコンピュータ可読媒体。
【請求項３９】

（ｉ）前記シード－アクセスコード関係は、前記導出されたシードを形成するように前記入力アクセスコードと連結することができる前記初期シードの切り捨てられた（ｔｒｕｎｃａｔｅｄ）バージョンを含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記初期シードの前記切り捨てられたバージョンを前記入力アクセスコードと連結するステップを含むことを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項４０】
（ｉ）前記シード－アクセスコード関係は、前記入力アクセスコードの複数の可能な値と前記導出されたシードの対応する複数の可能な値の値、ならびに前記入力アクセスコードの前記複数の可能な値と前記導出されたシードの対応する前記複数の可能な値の間の関連を含み、
（ｉｉ）前記シード－アクセスコード関係に従って前記導出されたシードを前記処理するステップは、前記入力アクセスコードに対応する前記導出されたシードの前記可能な値をルックアップし、出力するステップを含むことを特徴とする請求項３１に記載のコンピュータ可読媒体。
【請求項４１】
（１）前記シードを前記導出するステップと前記所定のデータ生成プロトコルを前記実施するステップが、共通の動作に合併され、
（２）前記出力データは、前記導出されたシードを含むことを特徴とする請求項４０に記載のコンピュータ可読媒体。
【請求項４２】
ユーザのアクセスコードの下で前記ユーザの生成カモフラージュされたアクセス規制されたデータをカモフラージュするための方法であって、
（ａ）生成証印（ｉｎｄｉｃｉａ）に従って生成プロトコルを使用することによってユーザのアクセス規制されたデータを初期設定するステップと、
（ｂ）前記生成証印と前記ユーザのアクセスコードの間の所定の関係をメモリの中に記憶するステップと、
（ｃ）（ｉ）前記アクセス規制されたデータの許可を受けたユーザによって再生成可能であるが、前記アクセス規制されたデータの許可を受けていないユーザによっては再生成不可能であるように、
（ｉｉ）前記カモフラージュするステップは、前記生成証印と前記ユーザのアクセスコードの間の前記所定の関係を記憶するステップを含み、
（ｉｉｉ）前記記憶されている生成証印－アクセスコード関係に従って入力されたアクセスコードのコンピュータベースの処理を介して前記アクセス規制されたデータの前記少なくとも一部分の後のアクセスを可能にし、
（ｉｖ）前記アクセス規制されたデータの前記少なくとも一部分のいずれの形態のいずれの記憶にも依存せず前記アクセス規制されたデータの前記少なくとも一部分をカモフラージュするステップと、
（ｄ）前記アクセス規制されたデータの前記カモフラージュされた少なくとも一部分をデジタルウォレットの中に記憶するステップと、
（ｅ）前記デジタルウォレットを前記ユーザに提供するステップと
を含むことを特徴とする方法。
【請求項４３】
ユーザのアクセスコードの下で前記ユーザの生成カモフラージュされたアクセス規制されたデータをカモフラージュするための方法であって、
（ａ）生成認証に従って生成プロトコルを使用することによってユーザのアクセス規制されたデータを初期設定するステップと、
（ｂ）前記アクセス規制されたデータの許可を受けたユーザによって再生成可能であるが、前記アクセス規制されたデータの許可を受けていないユーザによっては再生成不可能であるように前記アクセス規制されたデータの少なくとも一部分を生成カモフラージュする

ステップと、
（ｃ）前記アクセス規制されたデータの前記生成カモフラージュされた少なくとも一部分をデジタルウォレットの中に記憶するステップと、
（ｄ）前記デジタルウォレットを前記ユーザに提供するステップと
を含むことを特徴とする方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、アクセス規制されたデータをセキュアにすることに関し、より具体的には、暗号鍵の生成、証明、および使用をセキュアにすることに関する。
【背景技術】
【０００２】
暗号データセキュリティ技術は、鍵を使用してデータを暗号化することによってデータをセキュアにする。暗号化解除されたデータは、その鍵を使用することでしか回復することができない。鍵は、悪意のある侵入者が、かなり大量のコンピューティングリソースを使用してさえ、網羅的な試行錯誤によってその鍵を推測できないだけ十分に長いように選択される。したがって、データのセキュリティは、鍵のセキュリティに移されている。
【０００３】
ＲＳＡなどの非対称暗号法では、各ユーザが、マッチする１対の鍵である秘密鍵と公開鍵を保持する。秘密鍵と公開鍵は、秘密鍵を使用して暗号化されたメッセージ（例えば、メッセージ、データ、符号、および情報の他の暗号鍵または暗号表現を含む任意の他のデジタル式に表現可能な情報）が、公開鍵を使用してしか暗号化解除することができず、またその逆の場合も同様であることで、固有のマッチするペアを形成する。秘密鍵と公開鍵の間におけるこの１対１対応を使用して、電子メッセージおよび電子取引のためのデジタル署名を生成することができる。電子メッセージに署名するのに、ユーザは、単にメッセージを自身の秘密鍵を使用して暗号化するだけであることが可能である。次に、ユーザは、自身の公開鍵を暗号化されたメッセージに付加し、メッセージを受け手に送る。代替として、ユーザが自身の公開鍵をメッセージに付加するのではなく、受け手が、公開鍵のディレクトリの中でそのユーザの公開鍵をルックアップすることも可能である。いずれの場合も、署名を検証するのに、受け手は、付加された公開鍵を使用してメッセージを暗号化解除し、暗号化解除に成功した場合、受け手は、メッセージの出所について確信する。
【０００４】
前述したとおり、送り手は、メッセージに署名するのにメッセージ全体を自身の秘密鍵で暗号化しなければならず、これは、計算上の費用が高い。これに対処するのに、メッセージの固定された長さ、例えば、１２８ビット長の短いハッシュを計算した後、そのハッシュ値を暗号化するだけで十分である。ハッシュ関数が、ＭＤ５などの良好な関数である場合、２つの別個のメッセージが同一のハッシュ値を有する確率は、極めて小さい。したがって、デジタル署名法は、通常、メッセージのハッシュを計算し、ハッシュ値だけを暗号化する。暗号化されたハッシュ値、および送り手の公開鍵が、受け手への伝送に先立って元のメッセージに付加される。署名を検証するのに、受け手は、まず、受け取られたメッセージのハッシュを計算する。計算されたハッシュ値が、暗号化されたハッシュの暗号化解除された形態と同一である場合、受け手は、メッセージの出所について確信する。
【０００５】
以上において、署名検証プロセスの強度は、メッセージに付加された公開鍵が、実際に所有者とされる人の公開鍵であるという受け手の確信に依存する。マッチする１対の鍵を生成できる人は誰でも、ユーザになりすますことが、そのようななりすましを防止する手段が存在しない限り、可能である。このため、公開鍵は、しばしば、証明機関、または略してＣＡと呼ばれる第三者の公証人によって証明される。証明機関の例は、ＶｅｒｉｓｉｇｎやＥｎｔｒｕｓｔなどの商業エンティティである。ＣＡは、被証明者（ｃｅｒｔｉｆｅｅ）の公開鍵を被証明者の身元に結合し、次に、結合されたメッセージにＣＡの秘密鍵を

使用して署名して、被証明者の公開鍵の証明書を形成する。したがって、証明書保持者は、メッセージを受け手に送信するのに先立って、自身の公開鍵証明書を暗号化されたメッセージに付加する。送り手の身元、および送り手の公開鍵の真正性を調べるため、受け手は、ＣＡの公開鍵を使用して送り手の公開鍵証明書上のＣＡの署名を検証する。少数の広く信頼されているＣＡしか存在しないので、ＣＡの公開鍵は、信頼できる仕方で、容易に受け手に供与される。したがって、受け手と送り手が以前の関係を全く有していない場合でも、受け手と送り手が共通のＣＡをともに信頼する限り、受け手が送り手の署名を検証できることで、公開鍵署名は、他人対他人の証明に使用することができる。

【０００６】

ユーザの署名が固有であり、偽造が不可能であることは、ユーザが自身の秘密鍵を秘密に保つ能力に非常に強く依存する。ユーザの秘密鍵にアクセスを有する人は誰でも、完全に匿名でそのユーザになりすますことができる。したがって、電子商取引およびその他のアプリケーションに関するデジタル署名の広汎な使用は、秘密鍵のセキュアな記憶のための技術を要する。現在、秘密鍵は、スマートカード、Ｆｏｒｔｅｚｚａカード、ＰＣＭＣＩＡカード、およびその他の小型ハードウェアデバイスなどのハードウェアデバイス上に物理的に分離することによって記憶すべきであると広く信じられている。スマートカードは、マイクロプロセッサおよびいくらかのメモリを含むクレジットカードサイズのカードである。ユーザの秘密鍵および公開鍵証明書は、そのメモリ上に書き込まれる。カードを使用するのに、ユーザは、単に、ホストコンピュータに接続された適切なカード読取り装置にカードを挿入した後、自身のＰＩＮを入力してカードを起動する。正しいＰＩＮが入力された場合、カード上のプロセッサが、ホストコンピュータ上で使用するために秘密鍵を開放する。誤ったＰＩＮが入力された場合、プロセッサは、ユーザの秘密鍵を開放しない。一部の不正操作に耐性のあるスマートカードは、誤ったＰＩＮが数回の連続した起動の試行で入力された場合、永久にロックアップ（ｌｏｃｋ　ｕｐ）するように構成される。一部の高度なスマートカード（しばしば、クリプトカード（ｃｒｙｐｔｏｃａｒｄ）と呼ばれる）は、暗号処理を行うことができ、したがって、秘密鍵は、決してスマートカードを離れない。処理されるべきバイトがホストコンピュータからスマートカードに入り、処理され、ホストコンピュータに戻るように伝送される。残念ながら、クリプトカードでさえ、カード読取り装置との間でバイトを伝送してやり取りするためにホストコンピュータに頼らなければならず、したがって完全にセキュアではない。悪意のあるホストコンピュータが、伝送に先立って単にあるメッセージを別のメッセージに置き換え、ユーザが、実際には別のメッセージに署名しているのに、あるメッセージに署名していると思うようにすることが可能である。したがって、既存のクリプトカードでさえ、カード保持者を（例えば、悪意のあるホストコンピュータから）完全に保護することができない。

【０００７】

【特許文献１】

米国特許第６，１７０，０５８号

【特許文献２】

米国特許出願０９／１９６，４３０

【非特許文献１】

Schneier, Applied Cryptography, Wiley, 1996

【非特許文献２】

Menezes, Handbook of Applied Cryptography, CRC Press, 1997

【発明の開示】

【発明が解決しようとする課題】

【０００８】

スマートカードは、秘密鍵をセキュアに記憶する問題を解決するが、以下のいくつかの重要な欠点を抱えている。

【０００９】

１）高い初期費用：スマートカードは、スマートカード読取り装置の形態で高価な追加の

ハードウェアインフラストラクチャを必要とする。
２）管理上のオーバーヘッド：スマートカードは、配布および保守のために管理上のオーバーヘッドを要する。
３）低いユーザ利便性：ユーザは、スマートカードの不正操作に耐える特徴のため、スマートカードを複製すること、バックアップすること、または照合することができない。
【００１０】
追加のハードウェアを必要としないセキュアなソフトウェアベースの鍵ウォレット（ｗａｌｌｅｔ）が、以上にリストしたスマートカードの欠点のいくつかを軽減する。マイクロソフトの製品やネットスケープの製品などの製品において鍵の記憶のために今日、使用されている標準の技術は、しばしば、不正操作からの保護をほとんど提供せず、かなり簡単に侵入される可能性がある。具体的には、鍵ウォレットは、ユーザのＰＩＮを暗号鍵として使用して秘密鍵を暗号化された形態で記憶する。ＰＩＮは、ユーザが憶えているのに充分なだけ短くなければならず、例えば、６桁のコードでなければならない。そのようなソフトウェア鍵ウォレットをハッカが手に入れた場合、ハッカは、鍵ウォレットを開くコードを見つけ出すまで、数分間以内にパーソナルコンピュータ上で自動化された仕方で百万の可能な６桁のコードすべてを網羅的に試すことができる。見つけ出した時点で、ハッカは、完全に正しいＰＩＮを知り、ユーザの秘密鍵にアクセスを有する。したがって、ソフトウェアだけの鍵ウォレットを提供することの主な問題は、ＰＩＮは、ユーザが憶えているだけ十分に短くなければならないが、鍵ウォレットを不正操作に耐えるようにするだけ十分に長くしなければならないという競合する要件である。
【課題を解決するための手段】
【００１１】
本発明の様々な実施形態は、「生成カモフラージュ（ｇｅｎｅｒａｔｉｏｎ　ｃａｍｏｕｆｌａｇｉｎｇ）」と呼ぶ技術を使用して前述した問題に対処する。生成カモフラージュでは、鍵は、暗号化された形態でさえ、必ずしも記憶される必要がない。＊ただし、本技術の一部の実施形態は、便宜のため、および／または当面の特定の状況に応じて秘密鍵の表現を記憶することも可能である。しかし、秘密鍵の再生成は、秘密鍵の記憶された形態に依存せずに行われることが可能である。一部の実施形態では、生成カモフラージュシステムは、ユーザの鍵（またはその他のデータ）およびカモフラージュをユーザの鍵（またはその他のデータ）の一部分として記憶するように構成されることが可能である。再構成中、第１の部分が再生成され、第２の部分が、メモリから呼び出され、この２つの部分が結合されて、ユーザの鍵（またはその他のデータ）を共同で再構成する合成データが形成される。したがって、本明細書で使用する「生成」、「再生成」、または「生成カモフラージュ」への言及は、ユーザのデータのすべて、または一部に対する演算を含むと理解されたい。＊　代わりに、この技術は、ユーザが自身のＰＩＮを入力した場合に正しい秘密鍵を再生成する秘密鍵生成ファンクションを使用する。ユーザが誤ったＰＩＮを入力した場合、誤った秘密鍵が生成される。このようにして保護された秘密鍵について「生成カモフラージュされている」という言い方をする。この結果、生成カモフラージュ技術を使用するソフトウェアを獲得した（例えば、鍵保持者の鍵ウォレットを盗んだ）悪意のある侵入者は、正しいＰＩＮを推測しようと試みる際、暗号化された形態であれ、暗号化されていない形態であれ、いずれの形態の秘密鍵にもアクセスを有さない。生成カモフラージュを使用する鍵ウォレットの１つの例示的な実施形態では、悪意のある侵入者は、すべての（または実質的にすべての）可能なＰＩＮに対して有効に見える秘密鍵が生成されるため、もたらされる秘密鍵を見ることによって正しいＰＩＮを推測することが実質的に不可能であることを知る。
【００１２】
生成カモフラージュ技術は、入力シード値（ｓｅｅｄ　ｖａｌｕｅ）に基づいて秘密鍵出力を生成する秘密鍵生成ファンクションを使用する。入力シード値が、ユーザの秘密鍵を生成するのに最初に使用されたシード値と一致する場合、ユーザの秘密鍵はもちろん、再生成される。しかし、異なるシード値は、異なる秘密鍵を生成する。

【００１３】
一実施形態では、入力シード値は、排他ＯＲ演算に基づいて入力ＰＩＮ値から導出される。入力ＰＩＮおよび記憶されている値（ユーザの秘密鍵を生成するのに最初に使用されたシード値から導出された）は、正しいＰＩＮが入力された場合、入力シード値が元のシード値に一致するように排他ＯＲ演算を受ける。したがって、秘密鍵生成ファンクションは、入力として元のシード値を提供されると、ユーザの秘密鍵を生成する。
【００１４】
他の実施形態では、入力シード値は、入力ＰＩＮとの他のタイプの関係から導出される。
【００１５】
本明細書では、ＲＳＡ、ＤＳＳ、Ｅｌ－Ｇａｍａｌ、楕円曲線暗号システムなどの既存の公開鍵署名法、および関連する鍵の生成技術、検証技術、および証明技術と適合する「鍵ウォレット」の中で秘密鍵をセキュアにする例示的な適用例について述べる。
【００１６】
鍵ウォレットは、ユーザが、スマートカードを使用するのとほぼ同じやり方でＰＩＮを使用してロック解除するソフトウェアウォレットとして実装されることが可能である。配備される特定の構成に応じて、ソフトウェアベースのウォレットスキームの利点は、以下の一部またはすべてを含むことが可能である。
【００１７】
１）低い費用：システムは、必ずしも追加のハードウェアインフラストラクチャを必要としない。ウォレットは、フロッピー（登録商標）ディスク、ハードディスク、磁気ストライプカードを含み、スマートカード自体さえ含む任意のデジタル記憶媒体で実現することができる。
２）低い管理上のオーバーヘッド：ウォレットは、電子式に配布し、必要に応じて電子式に更新することができる。
３）高いユーザ利便性：ユーザは、ウォレットを複製すること、バックアップすること、および照合することができる。また、ウォレットは、電子式に伝送することが可能である。
４）不正操作に対する耐性：ウォレットの機能は、スマートカードの場合と同じ意味で不正操作に対する耐性を有することが可能である。
５）ユーザに不当な負担がかからない：ウォレットに対するユーザの体験は、スマートカードに対する体験と同じであり、ユーザは、異常に長いＰＩＮを必要とすること、またはＰＩＮを入力するのに極度の注意を必要とすること等がない。
【００１８】
もちろん、ソフトウェアだけの実施形態に多数の利点が存在するが、システムの部分またはすべてをハードウェアとソフトウェアの組合せで実施することさえ、または純粋にハードウェアで実施することさえ可能であり、配備および使用の大幅な柔軟性が提供される。
【００１９】
以上および以下の詳細な説明は、ＰＩＮを使用する秘密鍵のセキュアな生成を対象とする生成カモフラージュの例示的な実施形態について述べている。ただし、本明細書で開示する技術は、任意のデジタル式に表現可能なアクセスコードを使用する任意のアクセス規制されるデータ（ＡＣＤ）のセキュアな記憶のために一般的に使用可能であることが、当分野の技術者には認められよう。したがって、鍵ウォレット（または、代替として、デジタルウォレット）という用語は、本明細書で説明する例示的な実施形態だけでなく、生成カモフラージュされたアクセス規制されるデータをセキュアにするための任意のデバイスを指すものと一般的に理解されたい。
【００２０】
最後に、生成カモフラージュに関連するオプションの特徴は、前述した鍵ウォレットとともに使用可能な公開鍵証明書の生成および管理に関する。そのような特徴には、以下が含まれることが可能である。
【００２１】

１）有限責任：公開鍵は、擬似公開であり、使用が、証明を行う機関によって許可された許可を受けた検証者に明示的に限定されることが可能である。また、これにより、実際問題として、証明を行う機関の法的責任が限定されることが可能である。
２）証明書取消し：許可を受けた検証者だけに有用な公開鍵証明書が生成された場合、以下に詳細に説明するとおり、許可を受けた検証者だけが、証明書のキャンセルを通知されればよいという点で、証明書の取消しが容易になる。

【００２２】
生成カモフラージュの適用例には、（ａ）コンピューティングリソースおよび記憶リソースに対するセキュアな遠隔／ローカルアクセスのための強い証明、（ｂ）ネットワーク上の多数のコンピュータを伴う環境における軽減されたユーザサインオン、（ｃ）ＩＰＳＥＣネットワークプロトコルを伴う、または伴わないファイアウォールを介するセキュアなアクセスの強い証明、（ｄ）セキュアな電子商取引のためのデジタル署名、（ｅ）電子支払い機構のためのデジタル署名、（ｆ）データベーストランザクションのためのデータベースに対するセキュアなアクセスおよび／またはデジタル署名、（ｇ）ルータおよびネットワークスイッチに対するセキュアなアクセス、（ｈ）分散サービスに対するセキュアなアクセス、および／または（ｉ）前述の適用例のいずれかに適用された、ただし制限されない、ユーザ（例えば、デジタルウォレットの）がソフトウェアまたはハードウェアで実行されるプログラムなどの計算エージェントによって代表される組み込まれたアプリケーションが含まれるが、以上には制限されないことが当分野の技術者には認められよう。

【発明を実施するための最良の形態】

【００２３】
提示する説明は、デジタル署名に関する秘密鍵をセキュアにする例示的な文脈において行うが、生成カモフラージュの技術は、他の形態のデータをセキュアにするのにも使用できることが、当分野の技術者には容易に認められよう。

【００２４】
図１は、鍵秘密鍵初期設定（図１に示さず）の態様、秘密鍵生成（または再生成）の態様、公開鍵証明の態様、および検証の態様を含む例示的な「生成カモフラージュ」システムの機能要素の概要を示している。もちろん、実際の実施形態は、システムがどのように配備されるかの状況に応じて、以上の態様の一部、またはすべてを含むことが可能である。

【００２５】
従来の暗号システムの場合と同様に、秘密鍵の初期設定は、シードベースのプロセスを介して行われる。詳細には、秘密鍵（および対応する公開鍵が、使用される特定の暗号プロトコルに適切な秘密鍵と公開鍵の間の指定された関係に従って）が、（非特許文献１）および（非特許文献２）で説明されるような周知の鍵生成プロトコルを使用してシードに相関するものとして生成される。

【００２６】
より高いセキュリティのため、シードは、好ましくは、ランダムまたは擬似ランダムである。シードは、システムのセキュリティ要件に応じて任意の長さであることが可能である。通常のシードは、現在、８バイトないし２０バイトまたはそれより多くのバイト数の範囲にある。より短いシードを埋め込むことによって、より長いシードを予期する鍵生成ファンクションを変更して、より短いシードを受け入れるようにすることは容易である。埋込みが固定である（所与のウォレットに関して）場合、生成は、再現可能である。すなわち、同一の入力シードは、同一の出力鍵をもたらす。

【００２７】
初期設定された秘密鍵は、デジタル署名を行う際に使用するために鍵保持者に提供される。従来の鍵暗号システムでは、秘密鍵（または秘密鍵の何らかの形態）が、ＰＩＮの保護の下で保持者の鍵ウォレットの中に記憶される。鍵保持者がＰＩＮを入力した場合、秘密鍵が使用のために開放され、誤ったＰＩＮが入力された場合、何も開放されない。鍵ウォレットの中に秘密鍵が記憶されること、および／または正しいＰＩＮが入力されたときだけに秘密鍵が開放されることにより、従来の鍵管理システムに関連するあるセキュリティ

リスクがもたらされる。
【００２８】
本明細書で開示する本発明の様々な実施形態では、秘密鍵は、暗号化された形態でさえ、必ずしも鍵ウォレットの中に記憶されない。代わりに、ユーザの鍵ウォレットは、ユーザが必要とするのに応じて秘密鍵を再生成することも可能にするモジュールを含む。より詳細には、鍵ウォレットは、まず、ユーザの秘密鍵を生成するのに使用された元のシードを再導出し、次に、導出されたシードを使用して秘密鍵を再生成する。
【００２９】
図１は、入力されたＰＩＮ１１０を受け取る鍵ウォレット１００を開示している。ＰＩＮ１１０は、入力されたＰＩＮからシードを導出するのに使用することができる元のシードの記憶されている表現、または元のシードと正しいＰＩＮの間の既知の関係を使用するシード導出モジュール１２０に送られる。入力されたＰＩＮが元のＰＩＮであった場合、導出されるシードは、元のシードである。入力されたＰＩＮが元のＰＩＮではなかった場合、導出されるシードは、元のシードではない。いずれの場合も、導出されるシードは、元のシードからユーザの秘密鍵を初期設定するのに使用されたのと同様の鍵生成ファンクションを含む鍵生成モジュール１３０に送られる。したがって、正しいＰＩＮが入力された場合、元のシードが再生成され、ユーザの秘密鍵が再生成される。しかし、誤ったＰＩＮが入力された場合、誤ったシードが生成され、誤った秘密鍵が生成される。このようにして、前述したシード導出機構および鍵生成機構が、鍵保持者によるアクセスおよび使用を許しながら、許可を受けていない人員（ハッカなどの）に対して秘密鍵を効果的にカモフラージュする。そのような仕方で保護された秘密鍵について「生成カモフラージュされている」という呼び方をして、そのための技術を「生成カモフラージュ」と呼ぶ。
【００３０】
前述したとおり、生成カモフラージュは、元のシードの記憶されている表現に、あるいは元のシードと正しいＰＩＮの間の既知の関係にアクセスを有するシード導出モジュールを使用する。シード導出モジュールの様々な実施形態を以下の「鍵ウォレット」セクションでより詳細に説明する。
【００３１】
鍵ウォレットは、以下の「鍵ウォレットに対する攻撃および攻撃に対する応答」セクションで説明するとおり、ある種の攻撃に耐える形態および／または形式で鍵を出力するように構成することが可能である。
【００３２】
最後に、鍵ウォレット１００は、さらに別のタイプの攻撃に対するさらなるセキュリティのための証明モジュール８００および／または検証モジュール９００と組合せで使用することも可能である。証明モジュール８００は、秘密鍵に対応する公開鍵を証明するのに使用され、検証モジュール９００は、秘密鍵によって生成された署名を検証するのに使用される。証明モジュールおよび検証モジュールは、それぞれ、以下の「証明モジュール」サブセクションおよび「検証モジュール」サブセクションでより詳細に説明する。
【００３３】
１．鍵ウォレット
生成カモフラージュ技術の第１の実施形態が、図２に概略で描かれている。図２は、入力ＰＩＮ１１０が、ユーザによって入力された後に一時的に記憶されるメモリ２１０（ハードウェア・メモリ、例えばＲＡＭ、またはバッファや他のデータ記憶フィールドなどのソフトウェア・メモリ）を含むシード導出モジュール２０１を含む鍵ウォレットを示している。この実施形態では、シードとＰＩＮの間の記憶されている関係は、シードをメモリ２１０の中に通すことによって実質的に実行される単位（つまり１を掛ける）ファンクションと見なすことができる。したがって、鍵生成モジュール１３０は、ＰＩＮ自体をシードとして使用して出力鍵１４０を生成する。したがって、鍵生成ファンクションは、通常、ＰＩＮと同じ長さのシードを受け入れる。
【００３４】

鍵生成ファンクション１３０は、例えば、ＤＥＳシステム、または任意の他の対称暗号システムのための鍵生成ファンクションであること、あるいは、ＲＳＡシステム、または任意の他の非対称鍵署名システムのための秘密鍵生成ファンクションであることが可能である。もちろん、鍵生成ファンクションは、生成される量が暗号鍵でない任意の他のデータ生成ファンクションであることも可能である。ファンクション１３０は、暗号の分野における機密データまたは機密情報を生成するためのファンクションに制限される必要はなく、任意の他の技術分野における秘密データまたは秘密情報を生成するファンクションを含むことも可能であることが、当分野の技術者には認められよう。

【００３５】

図３は、本発明の第２の実施形態による生成カモフラージュ技術を実施する鍵ウォレットを示している。この実施形態では、シード導出モジュール１２０は、メモリ２１０と埋込みモジュール３１０とを含む。この組合せは、シードがＰＩＮより長いために埋込みが必要とされるが、シードが基本的にＰＩＮである（第１の実施形態と同様に）場合に使用される。例えば、通常のＰＩＮは、ユーザが容易に憶えておくことができる適当な長さである。多くの適用例では、ＰＩＮは、４～６バイトの長さである。ただし、通常の秘密データ生成ファンクションに対する入力値は、しばしば、有効な秘密データを多くの無効なデータのなかに分散させる目的ではるかに長い。第２の実施形態では、埋込みファンクションは、単にゼロをＰＩＮに追加して（または別の仕方でＰＩＮに埋込みを行って）、もたらされる数が、鍵生成ファンクション１３３０に対する入力シード値の役割をするのに必要とされるバイト数を有するようにする。より一般的には、シード導出モジュール１２０内部のメモリ２１０および埋込みモジュール３１０は、単一のファンクションまたは複数のファンクションを表す他のモジュールで置き換えることも可能であることが、当分野の技術者には認められよう。

【００３６】

図４は、生成カモフラージュを実施する鍵ウォレットの第３の実施形態を示している。第３の実施形態は、ＲＳＡシステム、または任意の他の非対称署名システムに特に適用される。この第３の実施形態では、シード導出モジュールは、排他ＯＲ（「ＸＯＲ」）ファンクション４２０を含む。ＸＯＲファンクションは、２つの入力、第１のメモリ２１０の中に記憶された入力ＰＩＮ値（ｐｉｎ）、および第２のメモリ４１０の中に記憶されたマスクされた値（$seed_{masked}$）（すなわち、元のシードの記憶されている表現）に対して演算を行って鍵生成モジュール１３０に対する入力シード値（ｓｅｅｄ）を生成する。シードを導出するためのブール演算式は、

【００３７】

$seed = pin \ XOR \ seed_{masked}$

【００３８】

であり、マスクされたシード値（$seed_{masked}$）は、

【００３９】

$seed_{masked} = pin_{valid} \ XOR \ seed_{orig}$

【００４０】

によって与えられる有効なＰＩＮと元のシードの間の関係を表し、ただし、ｖａｌｉｄ＿ｐｉｎは、許可された有効なＰＩＮであり、ｓｅｅｄ＿ｏｒｉｇは、正しい秘密鍵を生成した生成ファンクションに対する元のシードである。したがって、有効なＰＩＮが入力ＰＩＮとして供給された場合、マスクされたシード値（$seed_{masked}$）に対する排他ＯＲ演算は、元のシード値（$seed_{orig}$）を明らかにし、鍵生成モジュール１３０は、正しい秘密鍵を再生成する。他方、誤ったＰＩＮが入力ＰＩＮとして供給された場合、排他ＯＲ演算は、元のシード値を明らかにすることができず、秘密鍵生成ファンクション１３０は、誤った秘密鍵を生成する。

【００４１】

以上の例のいくつかの実施形態では、ＰＩＮ値は、４～６バイトの長さしかないことが可能であり、他方、シード値は、２０バイトの長さにもなることが可能である。したがって

、ＸＯＲ演算が行われた場合、バイトサイズのミスマッチが存在する可能性がある。そのような場合、ＰＩＮ値は、シードの残りの部分をそのままにして、シードの最初のセグメントまたは最後のセグメント、あるいはその他のセグメントとＸＯＲ演算を行うことができる。

【００４２】

さらに、秘密鍵生成ファンクションは、独立の秘密鍵生成ファンクションであること、またはマッチする１対の公開鍵と秘密鍵を生成するファンクションであることが可能である。マッチする１対の公開鍵と秘密鍵を生成するファンクションである場合、公開鍵出力は、使用する必要がない。

【００４３】

前述した第３の実施形態により、ユーザによって選択されるＰＩＮを変更することが可能になる。ユーザが自身のＰＩＮを変更した際、新しいマスクされたシード値が、第２のメモリ４１０の中に記憶される。新しいマスクされたシード値は、例えば、以下のブール式に基づくことが可能である。

【００４４】

$seed_{masked\_new} = pin_{valid\_new}$ ＸＯＲ pin_{valid_old} ＸＯＲ $seed_{masked\_old}$

【００４５】

以上の例では、ＸＯＲファンクションは、シードのマスキングを行うのに使用される。ただし、他のファンクションを使用することも可能であり、したがって、マスキングは、ＰＩＮが関与する演算を使用してそれを元にシードを再生成することができるシードの記憶を含むものと当分野の技術者には容易に一般的に理解されよう。

【００４６】

図５は、本発明の第４の実施形態を示している。この実施形態では、バイト長ｍのユーザによって入力されたＰＩＮ（ｖａｌｉｄ＿ｐｉｎ）は、シードの最初のセグメントであり、シードの残りの部分は、メモリ２１０において、連結器５２０および鍵生成モジュール１３０とともに鍵ウォレットの中に記憶される。ユーザが入力ＰＩＮ（ｐｉｎ）を入力した際、連結器（ｃｏｎｃａｔｅｎａｔｏｒ）５２０が、入力ＰＩＮを記憶されている残りの部分に連結して、鍵生成ファンクション１３０に対する入力シード値を生成する。ユーザの正しいＰＩＮが入力された場合、元のシード値が再構成され、したがって、正しい秘密鍵値が再構成される。他方、誤ったＰＩＮが入力された場合、異なるシード値が生成され、したがって、誤った秘密鍵が生成される。

【００４７】

前述した（また多数の他の）秘密鍵生成ファンクションおよびシード導出技術は、生成カモフラージュ技術と併せて使用できることが、当分野の技術者には容易に認められよう。さらに、例示的な実施形態は、ＰＩＮを使用する秘密鍵生成の文脈で説明してきたが、本明細書で開示する技術は、パスワード、またはユーザによって保持される任意のアクセスコードを使用して、実質的にあらゆる秘密データをカモフラージュすることに適用可能であることが、当分野の技術者には認められよう。したがって、本発明の範囲は、本明細書に開示する特定の実施形態に限定されず、頭記の特許請求の範囲の全幅（ｆｕｌｌ ｂｒｅａｄｔｈ）に限定されるものとする。

【００４８】

２．鍵ウォレットに対する攻撃および攻撃に対する応答

次に、ハッカがデジタル鍵ウォレットに対してしかける可能性がある種類の攻撃、およびそのような攻撃に抗する手段を列挙する。簡明にするため、説明は、ＲＳＡ公開鍵署名システムに関して提示する。ただし、説明の基本的な要素は、秘密データが特殊ファンクションによって生成されるその他の暗号システムおよび非暗号システムにも適用可能であることが、当分野の技術者には理解されよう。

【００４９】

ａ．力任せの攻撃（Ｂｒｕｔｅ Ｆｏｒｃｅ Ａｔｔａｃｋ）

悪意のあるハッカは、従来の鍵ウォレットのコピーを入手した場合、秘密鍵が開放されるまで、単に可能なＰＩＮのすべての組合せを試みる。従来の鍵ウォレットでは、秘密鍵は、正しいＰＩＮが入力された場合にだけ開放されるので、ハッカは、秘密鍵を開放したＰＩＮが正しいＰＩＮであることを知る。このタイプの攻撃に対抗するため、鍵ウォレットは、あらゆる入力ＰＩＮに応答していくらかの量を常に開放するように（または、いずれにしても、正しいＰＩＮが入力されたときにだけ開放を行う代わりに、相当な数の入力ＰＩＮに応答してある量を開放して）構成されることが可能である。さらに（形の整っていない鍵の攻撃（Ｉｌｌ－Ｆｏｒｍｅｄ　Ｋｅｙ　Ａｔｔａｃｋ）に関連して以下に説明するとおり）、鍵生成モジュールは、出力が秘密鍵の適切な形態で現れるように構成することができる。

【００５０】

ｂ．形の整っていない鍵の攻撃

別の攻撃は、悪意のあるハッカがすべての可能なＰＩＮを試みて、もたらされる秘密鍵の形態を検査する攻撃である。もたらされる秘密鍵の形が整っていない場合、ハッカは、そのような秘密鍵をもたらすＰＩＮが正しいＰＩＮであり得ないことを知る。したがって、鍵生成モジュールの前述した実施形態は、形の整った鍵を常に生成するように構成されることが可能であり、結果として、ハッカは、鍵の形が整っているかどうかを検査して正しいＰＩＮに辿り着くことができない。ＲＳＡシステム、およびその他の公開鍵暗号化システムにおいて形の整った鍵を生成する秘密鍵生成ファンクションは、周知であり、さらに詳述することはしない。有効に見える秘密鍵の特性（また、したがって、生成するための技術）は、当技術分野で周知であり（例えば、（非特許文献１）、（非特許文献２）、および参照により本明細書に組み込まれている（特許文献１）における秘密鍵フォーマットの説明を参照されたい）、詳述する必要はない。

【００５１】

ｃ．公知の公開鍵の攻撃

この攻撃では、悪意のあるハッカは、公開鍵証明書ディレクトリの中で容易に入手可能である可能性がある２つの情報、（ａ）ユーザの鍵ウォレット、および（ｂ）ユーザの公開鍵にアクセスを有する。この攻撃を図６に図示している。ハッカは、鍵ウォレット１００に対してすべての可能なＰＩＮ１１０を試みる。各ＰＩＮに関して、ハッカは、出力された秘密鍵６１０を使用して、恣意的に選択されたサンプルメッセージ６２０を暗号化し、次に、暗号化されたメッセージをユーザの公開鍵を使用して暗号化解除する。暗号化解除されたメッセージ６３０が元のサンプルメッセージ６２０に一致した場合、ハッカは、正しいＰＩＮを発見し、ユーザの正しい秘密鍵を再生成したことを知る。

【００５２】

この攻撃に対抗するため、生成カモフラージュの一実施形態は、公開鍵が真に公開されていることを許さない。便宜上、そのような限定された配布の公開鍵を「擬似公開鍵」と呼び、そのような擬似公開鍵を含む証明書を「擬似公開証明書」と呼ぶことにする。具体的には、擬似公開証明書は、ユーザの擬似公開鍵を暗号化された形態で含む。許可を受けた関係者だけが、擬似公開鍵にアクセスしてユーザの署名を検証することができる。これは、誰でも公開鍵署名を検証することができる公開鍵証明書の従来の使用法とは対照的である。もちろん、鍵ウォレット、および本発明のその他の態様または実施形態を単独で従来の証明書とともに使用することも可能であるが、本明細書で説明するとおり、擬似公開鍵および擬似公開証明書も使用される場合、さらに高いセキュリティが提供される。既存の証明書発行のデバイスおよび手続きを本発明の以上の実施形態に対応するように容易に適応させることが可能なことが、当分野の技術者には容易に理解されよう。したがって、証明モジュールのこの実施形態の特定のハードウェアおよび／またはソフトウェア実装形態を詳述する必要はない。むしろ、従来の証明書との違いだけを以下に説明する。従来の証明書には、いくつかのフォーマットがあり、そのなかで最も注目に値するのが、Ｘ．５０９フォーマットおよびその改訂版であるが、すべての従来のフォーマットの基本的な要素は、本発明との関係で見た場合、同様であることが、当分野の技術者である読者には認め

られよう。

【００５３】

擬似公開鍵には、擬似公開鍵証明書および擬似公開鍵検証技術が関わる。これらに関する例示的な実施形態を以下に説明する。

【００５４】

ｉ．証明書モジュール

図１を再び参照すると、生成カモフラージュ鍵ウォレットと組合せで使用可能な証明書モジュール１３０の一実施形態が、従来の公開鍵証明書とはいくぶん異なる公開鍵証明書を生成する。基本的に、本明細書で使用される公開鍵は、従来の方法のように真に公開ではなく、限定された配布（例えば、イントラネットを介する組織内部、あるいはそれ以外で、閉じた企業または擬似公開の企業内部の）に向けられている。

【００５５】

従来の公開鍵証明書と擬似公開証明書の１つの可能な実施形態を図７に並べて示している。従来の証明書は、ユーザの身元証明情報、ユーザの公開鍵、およびユーザの公開鍵の証明機関（ＣＡ）による署名を含み、署名は、ユーザの公開鍵のハッシュのＣＡの秘密鍵の下での暗号化を含む。

【００５６】

例示的な擬似公開証明書は、従来の証明書と同じフォーマットを有する可能性がある。ただし、擬似公開鍵を含む証明書７００の本文は、許可を受けた検証者だけが読み取ることができるように暗号化される。例えば、一実施形態では、暗号化は、許可を受けた検証者の公開鍵の下で行うことが可能である。対応する秘密鍵にアクセスを有する認証サーバだけが、ユーザの証明書のラップを解除して（ｕｎｗｒａｐ）ユーザの公開鍵にアクセスすることができる。数名の許可を受けた検証者が存在する場合、証明書の本文は、その検証者の各人の公開鍵によってそれぞれが暗号化された擬似公開鍵のいくつかの暗号化されたコピーを担持することが可能である。この手法を使用する各企業または各エンティティは、自らの擬似公開証明書をサポートするように前述した証明モジュールを有する証明書サーバを有する。公開鍵が暗号化され、許可を受けた検証者だけによって暗号化解除が行われることが可能であるようにする擬似公開証明書の構成は、様々な暗号アルゴリズムを使用して多くの異なる仕方で実現できることが、当分野の技術者には理解されよう。例えば、擬似公開鍵証明書の代替の実施形態では、公開鍵は、ＤＥＳ鍵によって暗号化され、ＤＥＳ鍵は、許可を受けた検証者の公開鍵によって暗号化される。

【００５７】

次に、もたらされた証明書に従来の証明書と同様に証明機関によって署名が行われる。鍵管理における２つの利点の可能性を提供するのが、公開鍵の擬似公開の性質である。第１に、証明機関は、公開鍵証明書を使用することが誰に許されているかを明確に認知しているので、ＣＡの法的責任は、実際問題として、限定される可能性がある。これは、ＣＡが、証明書を誰が使用するかについて事前の知識を全く有さない従来の証明書とは対照的である。第２に、公開鍵証明書を取り消すことが容易になる。というのは、ＣＡは、公開鍵証明書を使用することが許可された検証者に通知するだけでよいからである。

【００５８】

提案する形態の証明書は、図８に示すとおり証明書サーバとして動作する証明モジュールによって発行される。当分野の技術者には理解されるとおり、証明書サーバは、ソフトウェアで、ハードウェアで、またはソフトウェアとハードウェアの組合せで実装することができる一連のモジュールを含む。証明されることを所望するユーザは、入力８１０などのデジタル署名された要求を証明書サーバ８００にサブミットする。そのような要求は、通常、証明されるべきユーザの公開鍵、ならびにユーザの名前、または他の身元証明属性を含む。証明書サーバは、サブミットされた公開鍵を使用してユーザのデジタル署名を検証する。署名が正しく検証された場合、サーバは、データベースの中でユーザの身元情報を調べ、次に、提案する形態の公開鍵証明書８２０を出力として発行する。ユーザ身元データベースは、証明書を要求するユーザの身元を検証する情報の他のソースによって置き換

えることもできることが当分野の技術者には認められよう。
【００５９】
擬似公開証明書サーバの代替の実施形態には、変更ユニットが従来の証明書サーバに付加されることが関わることが可能である。そのようなアドオン（ａｄｄ－ｏｎ）ユニットが、従来の証明書サーバの入力または出力に対して処理を行うことが可能である。変更ユニットは、入力に対して処理を行う場合、ユーザの公開鍵を暗号化することによって証明書の要求を再パッケージ化し、暗号化された公開鍵を身元証明属性のなかに組み込む。次に、変更ユニットは、ダミー（ｄｕｍｍｙ）公開鍵を要求に付加し、関連する秘密鍵を使用してその要求に署名し、要求を従来の証明書サーバに転送する。従来の証明書サーバの出力は、身元証明属性の１つとしてユーザの暗号化された公開鍵を含む証明書である。変更ユニットは、従来の証明書サーバの出力に対して処理を行う場合、元の場所で証明書の中の公開鍵指数を暗号化し、次に、証明書サーバの署名に変更された証明書の新規の署名で上書きを行うことによって従来の証明書サーバによって生成された従来の証明書を再パッケージ化する。さらに別の代替の実施形態も可能であることが、当分野の技術者には認められよう。
【００６０】
ｉｉ．検証モジュール
ＣＡによる擬似公開鍵の証明は、その擬似公開鍵に対応する秘密鍵の保持者と通信する、または取引を行う関係者に配布するための擬似公開証明書の生成をもたらす。そのような関係者は、対応する秘密鍵を使用して暗号化されたメッセージを暗号化解除するのに擬似公開鍵に依拠するのに先立って、未知の秘密鍵保持者から受け取られた擬似公開鍵を検証することを望む。
【００６１】
図１を再び参照すると、擬似公開鍵および擬似証明書と組合せで使用可能な検証モジュール１２０の一実施形態が開示されている。検証モジュールは、従来のシステムの検証モジュールとは２つの点で異なる。検証モジュールは、公開鍵証明書の擬似公開の性質を遵守し、ユーザの署名を検証する前に証明書からユーザの公開鍵を抽出するのに適切なステップをとる。例示的な実施形態では、このステップには、証明書保持者の暗号化された擬似公開鍵を含む証明書を受け取るステップ、および許可を受けた検証者の秘密鍵を使用して擬似公開鍵を暗号化解除するステップが含まれる。次に、検証モジュールは、擬似公開鍵を使用して証明書保持者によって送られたメッセージの中のデジタル署名を検証する。代替の実施形態では、ＤＥＳ鍵が、擬似公開鍵を暗号化するのに使用されている場合、ＤＥＳ鍵が、まず、検証者の秘密鍵を使用して暗号化解除され、次に、ＤＥＳ鍵を使用して擬似公開鍵が暗号化解除される。暗号化解除機構がどのようなものであれ、検証モジュールは、不正なハッカによる、例えば、鍵ウォレットの誤ったアクセスコードに対応する誤った候補秘密鍵でメッセージに署名するハッカによる侵入の試みを検出する論理も含むことが可能である。そのような場合において、不正なハッカが、正当なユーザの擬似公開証明書を盗むか、または別の仕方で入手して、その証明書を誤った候補秘密鍵で署名された虚偽のメッセージとともに送る可能性がある。証明書の中の正当なユーザの正しい擬似公開鍵と誤った候補秘密鍵の間の不整合により、不正なユーザの検出が可能になる。詳細には、一実施形態では、特定のユーザの署名が３回の連続する試行で検証されなかった場合、検証モジュールは、侵入が進行中であると結論し、さらなる調査を待ってそのユーザのアクセス特権を凍結する。アクセスを凍結することに加えて（代わりに）、検証モジュールは、試みられた侵入の操作者に警告する警報を鳴らすことが可能である。検証モジュールにおいて侵入の試みを検出する他の方法、および侵入が検出された際の他の可能な対処の仕方が存在する。当分野の技術者には理解されるとおり、検証モジュールは、ソフトウェアで、ハードウェアで、またはソフトウェアとハードウェアの組合せで実装することができる一連の論理モジュールの折り合いをつける。
【００６２】
３．変更形態、拡張形態、および代替の実施形態

以上、生成カモフラージュの様々な態様を説明してきた。１つの例示的な実施形態では、鍵ウォレット、シード生成－鍵生成モジュール、鍵検証モジュール、および鍵証明モジュールをすべて一緒に使用して暗号鍵の記憶と使用のためのセキュアな技術が提供されたが、代替の実施形態では、モジュールのすべては必要としない特定のアプリケーションのためにシステム全体の様々なサブセットを組み合わせることもできることが、当分野の技術者には認められよう。

【００６３】

さらに、以上は、例示的なソフトウェアベースのシステムに関して説明してきたが、これは、絶対に必要とされるわけではない。例えば、モジュールの一部またはすべては、マイクロコードおよびＰＬＡまたはＲＯＭ、汎用プログラミング言語および汎用マイクロプロセッサ、あるいはＡＳＩＣを使用して配備することもできる。つまり、本明細書で説明した技術は、ソフトウェアそのものに制限されず、純粋なソフトウェアで、ソフトウェアとハードウェアの組合せで、あるいはハードウェアだけであることも含め、実質的にあらゆる形態の論理で配備することが可能である。

【００６４】

鍵ウォレットは、ユーザが担持すること、または代替として、遠隔に記憶され、参照により本明細書に組み込まれている１９９８年１１月１９日に出願した（特許文献２）で開示しているような「ローミング（ｒｏａｍｉｎｇ）」技術を使用して、ネットワークを介してユーザに「オンザフライ（ｏｎ　ｔｈｅ　ｆｌｙ）」でダウンロードされることが可能である。

【００６５】

さらに、様々な実施形態または態様をＲＳＡ暗号法（公開鍵および／または擬似公開鍵、および公開証明書および／または擬似公開証明書に関して）、またはＤＥＳ暗号法（擬似公開証明書上の擬似公開鍵のＰＩＮ暗号化およびＰＩＮ記憶に関して）に関して説明してきたが、そのような例示的な暗号技術に対する多くの変更形態および拡張形態が可能であることが、当分野の技術者には認められよう。より一般的には、前述した処理のそれぞれは、多くの種類の非対称暗号化または対称暗号化、ならびにＣＲＣ、ハッシュ、メッセージダイジェスト、または他の一方向関数を含む多種類の暗号技術から実施することが可能である。例えば、非対称暗号化処理は、完全性が主要な関心事である（オプションとして鍵付きの）１方向関数で、または対称セッション鍵の暗号化の後に続く、平文暗号化のためのセッション鍵の使用で、また当分野の技術者には周知の他の代替策で置き換えることができる。

【００６６】

さらに、多くの適用例では、入力されるＰＩＮは、およそ４～６バイトであり、通常、シード値よりも小さい。ただし、シード値の方がＰＩＮ値より小さいことも可能である。これには、例えば、ＰＩＮの多対１ハッシュを行い、このより小さいバイトシーケンスをシードとして使用することによって対処することができる。したがって、シードの長さとＰＩＮの長さの間の関係について絶対的な要件は存在しない。むしろ、この関係は、配備されるシステムの選択および要件によって決まる。

【００６７】

また、例示的な実施形態は、秘密鍵を保護するＰＩＮに関して説明してきたが、生成カモフラージュの同じ技術を他のタイプのアクセスコードおよび暗号表現とともに使用して、あらゆる再生成可能なアクセス規制されたデータを保護することができる。例えば、本明細書で開示した例示的な実施形態は、秘密鍵を生成するのに使用された元のシードの記憶されている表現、または元のシードとユーザのＰＩＮの間の既知の関係にアクセスを有するシード導出モジュールに依拠する。しかし、システムは、ユーザの秘密鍵（または、より一般的には、アクセス規制されたデータ）が、シード値に基づいていないか、またはシード値が、生成カモフラージュデジタルウォレットが実装される時点で知られていない場合でも、機能する。より詳細には、アクセス規制されたデータが、オペランド（ｏｐｅｒａｎｄ）が受け取られた時点で鍵生成（または、より一般的には、データ生成）モジュー

ル内部で導出可能であるものと想定されたい。これは、シード導出モジュールを使用して、オペランドが、ユーザのＰＩＮの相関として表現可能であり、したがって、ユーザのＰＩＮから再生成することができるアクセス規制されたデータとオペランドの間の任意の１対１マッピングで実現可能である。

【００６８】

以上に開示するＸＯＲシード導出モジュール機能を使用する第１の例として、シード導出モジュールは、$operand_{masked}$＝ｐｉｎ ＸＯＲ ｏｐｅｒａｎｄを記憶し、したがって、オペランドは、ｏｐｅｒａｎｄ＝$operand_{masked}$ ＸＯＲ ｐｉｎとして導出可能である。シード導出モジュール機能の他の例示的な実施形態（例えば、ｏｐｅｒａｎｄ＝埋込みを有する、または有さないｐｉｎ）も、この目的で使用可能である。以上は、オペランドが「シード」に機能的に等価であり、またオペランドとアクセス規制されたデータの間の１対１マッピングは、鍵生成モジュールにおけるシードベースの鍵生成ルーチンに機能的に等価であることを示している。したがって、「シード」という用語は、ユーザの秘密鍵を初期設定するのに使用された量に排他的に限定されるべきでなく、この段落で説明したタイプの他の量に限定されるべきである。同様に、「鍵」という用語は、暗号化処理に使用可能な量に排他的に限定されるべきでなく、他のアクセス規制されたデータに一般的に限定されるべきである。

【００６９】

第２の例として、ＰＩＮ（または、より一般的に、アクセスコード）空間は、ｐｉｎｓｐａｃｅ＿ｓｉｚｅ（例えば、６桁のＰＩＮの場合、ｐｉｎｓｐａｃｅ＿ｓｉｚｅ＝１，０００，０００）のサイズである。シード導出モジュールは、最大で、何らかの順序でリストされたｐｉｎｓｐａｃｅ＿ｓｉｚｅのＰＩＮを記憶することができ、また各ＰＩＮに関して、対応するシードを記憶することができる。これらのＰＩＮ－シードのペアは、正しいＰＩＮに対応する正しい（つまり初期設定）シードだけでなく、誤ったＰＩＮに対応する誤ったシードの（いくつかの、またはすべての）可能性の高い値も含む。鍵を開放するようにＰＩＮを入力した際、そのＰＩＮは、リストへの索引として解釈され、指し示されたシードが選択される。正しいＰＩＮを入力した場合、正しいシードが導出される。誤ったＰＩＮを入力した場合、誤ったシードが導出される。次に、シードは、鍵（または、より一般的に、データ）生成モジュールに対する入力として使用される。

【００７０】

第３の例として、入力ＰＩＮに対応するシードを記憶する代わりに、生成カモフラージュウォレットが、２ステップ（シード導出に加えて鍵生成）のプロセスの必要性なしに、入力ＰＩＮに対応する鍵を直接に記憶する第２の例を考慮されたい。これは、単純化された形態の生成カモフラージュと見なすことができる。前の説明では、シードにＰＩＮを足し、生成機能を足したものがともに、鍵を生成する。本例では、リスト、ＰＩＮ、および選択機能が、鍵を生成する。生成カモフラージュ機構は、リストを明示的に記憶し、生成ルーチンが、リストの中の項目を選択する手段に縮減される、または単純化されるようにすることにより、明示的なシードを撤廃した。このようにして、シード導出機能は、鍵生成機能に合併されており、したがって、導出されるシードは、出力される鍵と同じである。この例は、データ項目のリストが明確に記憶され、ＰＩＮが、その項目の１つを選択するのに索引として使用される生成カモフラージュの変種と見なすことができる。

【００７１】

したがって、生成カモフラージュの一般的な技術の以上に開示した（また、さらに別の可能な）すべての実施形態および実装形態に鑑みて、本発明の範囲は、頭記の特許請求の範囲によってのみ限定されるものとする。

【図面の簡単な説明】

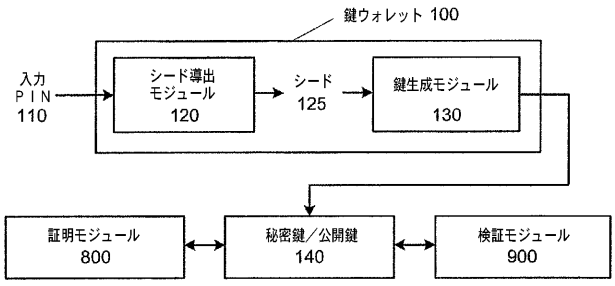
【００７２】

【図１】暗号鍵ウォレットおよびシード生成と鍵生成のサブシステム、鍵証明サブシステム、および検証サブシステムを示す概略図である。

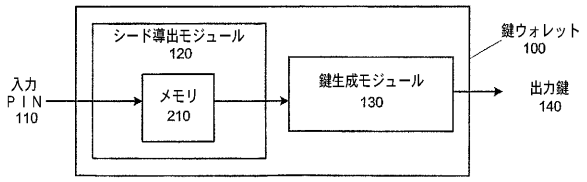
【図２】生成カモフラージュ技術を実施する鍵ウォレットの第１の実施形態を示す図であ

る。
【図３】生成カモフラージュ技術を実施する鍵ウォレットの第２の実施形態を示す図である。
【図４】生成カモフラージュ技術を実施する鍵ウォレットの第３の実施形態を示す図である。
【図５】生成カモフラージュ技術を実施する鍵ウォレットの第４の実施形態を示す図である。
【図６】擬似公開証明書によって対処が行われる周知の公開鍵攻撃を示す図である。
【図７】従来の例示的な擬似公開証明書を示す図である。
【図８】例示的な証明書サーバの実施形態を示す図である。

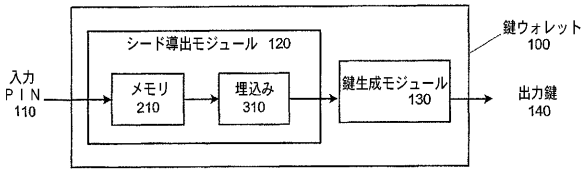
【図１】



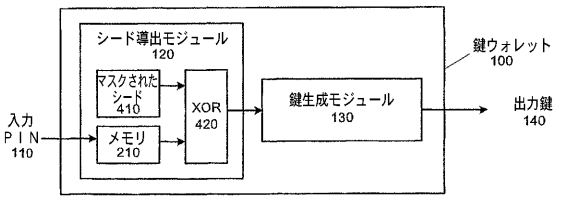
【図２】



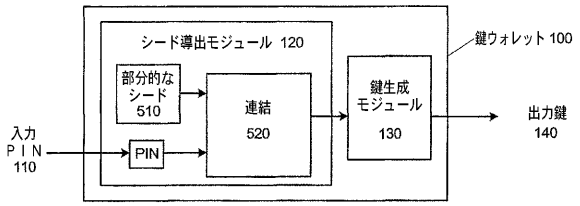
【図３】


シード導出モジュール 120
メモリ 210
埋込み 310
鍵生成モジュール 130
入力 PIN 110
鍵ウォレット 100
出力鍵 140

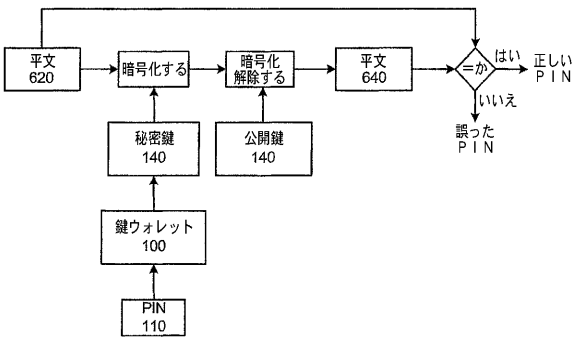
【図４】


シード導出モジュール 120
マスクされたシード 410
メモリ 210
XOR 420
鍵生成モジュール 130
入力 PIN 110
鍵ウォレット 100
出力鍵 140

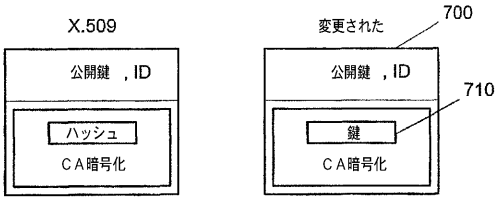
【図５】


シード導出モジュール 120
部分的なシード 510
連結 520
PIN
鍵生成モジュール 130
入力 PIN 110
鍵ウォレット 100
出力鍵 140

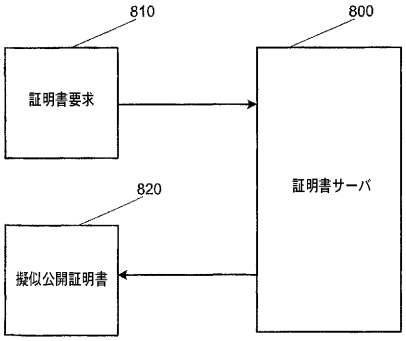
【図６】


平文 620
暗号化する
暗号化解除する
平文 640
=か
はい
正しい PIN
いいえ
誤った PIN
秘密鍵 140
公開鍵 140
鍵ウォレット 100
PIN 110

【図７】


X.509
公開鍵 ，ID
ハッシュ
ＣＡ暗号化
変更された
700
公開鍵 ，ID
710
鍵
ＣＡ暗号化

【図８】


810
800
証明書要求
証明書サーバ
820
擬似公開証明書

【国際公開パンフレット】

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 October 2002 (10.10.2002)

PCT

(10) International Publication Number
WO 02/080445 A1

(51) International Patent Classification⁷: H04L 9/00

(21) International Application Number: PCT/US02/09812

(22) International Filing Date: 29 March 2002 (29.03.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/280,629 29 March 2001 (29.03.2001) US
09/874,795 5 June 2001 (05.06.2001) US

(71) Applicant: ARCOT SYSTEMS, INC. [US/US]; Suite 200, 3200 Patrick Henry Drive, Santa Clara, CA 95054 (US).

(72) Inventor: HIRD, Geoffrey, R.; 10274 Parkwood Drive, Apt.4, Cupertino, CA 95014 (US).

(74) Agents: YANG, Joseph et al.; Skadden, Arps, Slate, Meagher & Flom LLP, 525 University Avenue, Palo Alto, CA 94301 (US).

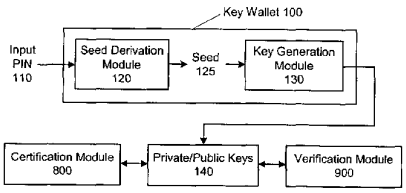
(81) Designated States (national): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— with international search report
— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD AND APPARATUS FOR SECURE CRYPTOGRAPHIC KEY GENERATION, CERTIFICATION AND USE



(57) Abstract: A confidential datum, such as a private key (140) used in public key signature systems, is secured in a digital wallet (100) using a "generation camouflaging" technique. With this technique, the private key (140) is not necessarily stored in the digital wallet (100), not even in an encrypted form. Instead, the wallet contains a private key generation function (130) that reproduces the correct private key when the user inputs his or her pre-selected PIN. If the user inputs an incorrect PIN, an incorrect private key (140) is outputted. Such private key (140) can be configured so that it cannot be readily distinguished from the correct private key (140) through the use of private key formatting, and/or the use of pseudo-public keys corresponding to the private key (140). The techniques described herein are also applicable to other forms of regeneratable confidential data besides private keys.

WO 02/080445 A1

METHOD AND APPARATUS FOR SECURE CRYPTOGRAPHIC KEY GENERATION, CERTIFICATION AND USE

FIELD OF THE INVENTION

The invention relates generally to securing an access-controlled datum and, more specifically, to secure cryptographic key generation, certification and use.

BACKGROUND

Cryptographic data security techniques secure data by encrypting the data with a key. The decrypted data can only be recovered using the key. The key is selected to be sufficiently long that a malicious intruder cannot guess the key by exhaustive trial and error, even with the use of substantially large amounts of computing resources. Therefore, the security of the data has been transferred to the security of the key.

In asymmetric cryptographic methods such as RSA, each user holds a matched pair of keys, a private key and a public key. The private key and the public key form a unique and matched pair in that messages (e.g., messages, data, code, and any other digitally representable information including other cryptographic keys or cryptographic representations of information) that are encrypted with the private key can only be decrypted with the public key and vice versa. This one-to-one correspondence between the private key and the public key can be used to create digital signatures for electronic messages and transactions. In order to sign an electronic message, a user could simply encrypt the message with his private key. He would then attach his public key to the encrypted message and send it to the recipient. Alternatively, the user would not attach his public key to the message, but the recipient could look up the user's public key in a directory of public keys. In either case, to verify the signature, the recipient would decrypt the message using the attached public key, and if the decryption is successful, the recipient is confident of the origin of the message.

As described above, the sender would have to encrypt the entire message with his private key to sign it, which is computationally expensive. To address this, it suffices to compute a short hash of fixed length, say 128 bits long, of the message and then encrypt

the hash value. If the hash function is a good one, such as MD5, the chances of two distinct messages having the same hash value are extremely small. Therefore, digital signature methods typically compute hashes of messages, and encrypt only the hash value. The encrypted hash value and the public key of the sender are attached to the original message prior to transmission to the recipient. To verify the signature, the recipient would first compute the hash of the received message. If the computed hash value is the same as the decrypted form of the encrypted hash, the recipient is confident of the origin of the message.

In the foregoing, the strength of the signature verification process depends on the recipient's confidence that the public key attached to the message is indeed the public key of the purported owner. Anybody who can generate a matched pair of keys can masquerade as the user, unless there exists a means to prevent such a masquerade. To this end, public keys are often certified by third-party notaries called *certifying authorities* or CAs for short. Examples of certifying authorities are commercial entities such as Verisign and Entrust. The CA binds a certifiee's public key with the certifiee's identity, and then signs the combined message with the CA's private key, to form the certifiee's public key certificate. Thus, a certificate holder would attach his public key certificate to the encrypted message prior to sending the message to the recipient. To check the sender's identity and the authenticity of his public key, the recipient verifies the CA's signature on the sender's public key certificate, using the CA's public key. Since there would only be a small number of widely trusted CAs, the CA's public key would be reliably and easily available to the recipient. Thus, public key signatures can be used for stranger-to-stranger authentication in that even if the recipient and the sender have no prior relationship, the recipient can verify the sender's signature as long as the recipient and the sender both trust a common CA.

The uniqueness and unforgeability of a user's signature depend very strongly on the ability of the user to keep his private key private. Anybody who has access to the private key of a user can masquerade as that user with complete anonymity. Hence, widespread use of digital signatures for electronic commerce and other applications will require technology for the secure storage of private keys. At present, it is widely believed that private keys are best stored by physically isolating them on hardware devices such as smart cards, Fortezza cards, PCMCIA cards and other compact hardware devices. Smart cards are credit-card sized cards that contain a microprocessor and some memory. The

user's private key and public key certificate are written onto the memory. To use the card, the user would simply insert the card into an appropriate card reader connected to a host computer, and then enter his PIN to activate the card. If the correct PIN were entered, the on-card processor would release the private key for use on the host computer. If an incorrect PIN were entered, the processor would not release the user's private key. Some tamper-resistant smart cards are configured so that if incorrect PINs are entered on several consecutive activation attempts, the card locks up permanently. Some sophisticated smart cards (often called cryptocards) can perform cryptographic operations, so that the private key need never leave the smart card. The bytes to be processed enter the smart card from the host computer, are processed, and are then transmitted back to the host computer. Unfortunately, even cryptocards must rely on the host computer for transmitting the bytes back and forth from the card reader and are therefore not perfectly secure. A malicious host computer could simply substitute one message for another prior to transmission, so that the user thinks he is signing one message, while in fact he is signing another. Therefore, even existing cryptocards cannot completely protect the cardholder (*e.g.*, against malicious host computers).

While the smart card solves the problem of securely storing private keys, it suffers from several significant drawbacks:

1) *High Initial Cost*: Smart cards require expensive additional hardware infrastructure in the form of smart card readers;
2) *Administrative Overhead:* Smart cards require administrative overhead for distribution and upkeep; and
3) *Low User Convenience:* The user cannot duplicate, backup or collate smart cards, owing to their tamper proof features.

A secure *software-based* key wallet that does not require additional hardware would mitigate some of the drawbacks of the smart card listed above. The standard technology that is used today for key storage, in products such as those of Microsoft and Netscape, offers very little protection against tampering, and can be broken into rather easily. Specifically, these key wallets store the private key in encrypted form, using the user's PIN as the encryption key. The PIN must be short enough for the user to remember, say a six-digit code. If such a software key wallet falls into the hands of a hacker, the hacker could exhaustively try all one million possible six-digit codes in an automated fashion on a personal computer within a few minutes, until he finds the code that opens the

key wallet. At this point, the hacker knows that he has exactly the correct PIN, and has access to the user's private keys. Thus, the primary problem with providing a software-only key wallet are the competing requirements that the PIN be short enough for the user to remember, but long enough to make the key wallet tamper resistant.

5

SUMMARY

Various embodiments of the invention address the above-described problems using a technique which we shall refer to as "generation camouflaging." In generation camouflaging, the key is not necessarily required to be stored, not even in its encrypted form.[1] Instead, this technique employs a private key generation function that reproduces the correct private key when the user inputs his PIN. If the user inputs an incorrect PIN, a wrong private key is produced. We refer to private keys protected in this manner as being "generation camouflaged." As a result, a malicious intruder who obtains the software (e.g., steals the key holder's key wallet) using the generation camouflaging technique will not have access to any form of the private key, encrypted or unencrypted, when trying to guess the correct PIN. In one exemplary embodiment of a key wallet using generation camouflaging, the malicious intruder will find it virtually impossible to guess the correct PIN by looking at the resulting private key, because for every (or virtually every) possible PIN, a valid-looking private key is produced.

10 15 20

---

[1] Although some embodiments of the technology could also store a representation of the private key, as a matter of convenience and/or depending on the specific circumstances at hand. However, the regeneration of the private key can occur without dependence on the stored form of the private key. In some embodiments, the generation camouflage system might be configured to store part of a user's key (or other datum) and camouflage a part of a user's key (or other datum). During reconstruction, the first part would be regenerated, the second part would be recalled from memory, and the two parts would be combined to form a composite datum that collectively reconstitutes the user's key (or other datum). Thus, as used herein, references to "generation," "regeneration," or "generation camouflage" should be understood to include operations on all or part of a user's datum.

The generation camouflaging technique employs a private key generation function that produces a private key output based on an input seed value. If the input seed value matches the seed value that was originally used to produce the user's private key, the user's private key is, of course, reproduced. However, a different seed value will produce a different private key.

In one embodiment, the input seed value is derived from an input PIN value based on an exclusive-OR operation. The input PIN and a stored value (which is derived from the seed value that was originally used to produce the user's private key) undergo the exclusive-OR operation such that, when the correct PIN is input, the input seed value matches the original seed value. Consequently, the private key generation function, having the original seed value provided thereto as an input, reproduces the user's private key.

In other embodiments, the input seed value is derived from other types of relationships with the input PIN.

In this specification, we discuss an exemplary application of securing private keys in a "key wallet," compatible with existing public key signature methods such as RSA, DSS, El-Gamal, elliptic curve cryptosystems, and their associated key generation, verification and certification technologies.

A key wallet may be implemented as a software wallet, which the user would unlock, using a PIN, in much the same way as he would use a smart card. Depending on the particular configuration deployed, the advantages of a software-based wallet scheme may include some or all of the following:

1) *Low Cost:* The system does not necessarily require additional hardware infrastructure. The wallet can be embodied in any digital storage medium including floppy disks, hard disks, magnetic stripe cards, and even smart cards themselves;
2) *Low Administrative Overhead:* The wallet can be distributed electronically and updated electronically as required;
3) *High User Convenience:* The user can duplicate, backup, and collate wallets. Wallets can also be transmitted electronically;
4) *Tamper Resistance:* The wallet can be tamper resistant in functionally the same sense as a smart card; and

5) *No Undue Burden on User:* The user's experience with the wallet would be the same as that with the smart card, and the user would not require unusually long PINs, or require extreme care in entering PINs, etc.

Of course, although there are many advantages to a software-only implementation, parts or all of the system could even be implemented in a combination of hardware and software, or even purely in hardware, providing great flexibility for deployment and usage.

The foregoing and the following detailed description describe an exemplary implementation of generation camouflaging, directed at secure generation of private keys using a PIN. Those skilled in the art will realize, however, that the techniques disclosed herein are usable generally for secure storage of any access-controlled datum (ACD) using any digitally representable access code. Therefore, the term *key wallet* (or, alternatively, *digital wallet*) should be understood generally to refer to any device for securing generation camouflaged access-controlled data, rather than only the exemplary embodiments described herein.

Finally, optional features related to generation camouflaging pertain to the creation and management of public-key certificates usable with the above key wallet. Such features may include:

1) *Limited liability:* The public key may be pseudo-public, with its use being limited explicitly to authorized verifiers who are authorized by the certifying authority. This could also, as a practical matter, limit the legal liability of the certifying authority.

2) *Certificate Revocation:* If public-key certificates are created which are only useful to authorized verifiers, then revocation of the certificates is facilitated to the extent that only the authorized verifiers need to be notified of the cancellation of a certificate., as will be explained in detail below.

Persons skilled in the art will recognize that applications of generation camouflaging include, but are not limited to: (a) strong authentication for secure remote/local access to computing and storage resources; (b) reduced user sign-on in environments with multiple computers on a network; (c) strong authentication for secure access through firewalls with or without the IPSEC network protocol; (d) digital signatures for secure electronic commerce transactions; (e) digital signatures for electronic payment mechanisms; (f) secure access to databases and/or digital signatures for database transactions; (g) secure access to routers and network switches; (h) secure access to

distributed services; and/or (i) embedded applications where the user (*e.g.*, of the digital wallet) is represented by a computational agent, such as a program running in software or hardware, as applied to, but not limited to, any of the aforementioned applications.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 is a schematic overview of a cryptographic key wallet and seed and key generation, key certification, and verification subsystems.

Figure 2 illustrates a first embodiment of a key wallet that implements the generation camouflaging technique.

Figure 3 illustrates a second embodiment of a key wallet that implements the generation camouflaging technique.

Figure 4 illustrates a third embodiment of a key wallet that implements the generation camouflaging technique.

Figure 5 illustrates a fourth embodiment of a key wallet that implements the generation camouflaging technique.

Figure 6 illustrates a known public key attack that is addressed by a pseudo-public certificate.

Figure 7 illustrates a conventional and an exemplary pseudo-public certificate.

Figure 8 illustrates an exemplary certificate server embodiment.

DETAILED DESCRIPTION

While the discussion that we will present is in the exemplary context of securing private keys for digital signatures, those skilled in the art will readily recognize that the technique of generation camouflaging can be used to secure other forms of data.

Figure 1 gives a schematic overview of functional elements of an exemplary "generation camouflage" system which includes key private key initialization (not shown in Figure 1), private key generation (or regeneration), public key certification, and verification aspects. Of course, actual embodiments could include some or all of these aspects, depending on the circumstances of how the system is deployed.

As in conventional cryptographic systems, the initialization of a private key occurs via a seed-based process. In particular, a private key (and its corresponding public key, in

accordance with a specified relationship between private and public keys appropriate to the particular cryptographic protocol in use) is generated as a function of a seed using known key generation protocols such as those described in Schneier, *Applied Cryptography*, Wiley, 1996 and Menezes, *Handbook of Applied Cryptography*, CRC Press, 1997.

For enhanced security, the seed is preferably random or pseudo-random. The seed can be of any length, depending on the security requirements of the system. Typical seeds currently range from 8 bytes to 20 or more bytes. It is easy to modify a key generation function that expects a longer seed so that it accepts a shorter seed, by padding the shorter seed. If the padding is fixed (for a given wallet), the generation is reproducible: the same input seed will produce the same output key(s).

The initialized private key is provided to the key holder for use in performing digital signatures. In conventional key cryptographic systems, the private key (or some form thereof) is stored in the key holder's key wallet under protection of a PIN. When the key holder inputs the PIN, the private key is released for use; if an incorrect PIN is entered, nothing is released. Storage of the private key in the key wallet, and/or its release only upon entry of the correct PIN leads to certain security risks associated with to conventional key management systems.

In the various embodiments of the invention disclosed herein, the private key is not necessarily stored in the key wallet, not even in an encrypted form.[1] Rather, the user's key wallet contains a module that also allows it to regenerate the private key as needed by the user. More particularly, the key wallet first re-derives the original seed used to create the user's private key, then uses the derived seed to regenerate the private key.

Figure 1 discloses a key wallet **100** receiving an inputted PIN **110**. The PIN **110** is sent to a seed derivation module **120** which uses a stored representation of the original seed, or a known relationship between the original seed and the correct PIN, that can be used to derive a seed from the inputted PIN. If the inputted PIN is the original PIN, the derived seed is the original seed. If the inputted PIN is not the original PIN, the derived seed is not the original seed. In either case, the derived seed is sent to a key generation module **130** containing similar key generation functionality as was used to initialize the user's private key from the original seed. Thus, if the correct PIN is entered, the original seed is reproduced and the user's private key is regenerated. However, if the incorrect PIN is inputted, an incorrect seed is produced, and an incorrect private key is generated. In this

fashion, the aforementioned seed derivation and key generation mechanisms effectively camouflage the private key against unauthorized personnel (such as hackers) while allowing access and use by the key holder. We refer to a private key protected in such a fashion as being "generation camouflaged" and to the techniques therefor as "generation camouflaging."

As stated above, generation camouflaging uses a seed derivation module that has access to either a stored representation of the original seed, or a known relationship between the original seed and the correct PIN. Various embodiments of seed derivation modules are described in greater detail in the "Key Wallet" section below.

The key wallet can also be configured to output keys in forms and/or formats in a manner resistant to certain kinds of attacks, as described in the "Attacks on Key Wallets and Responses Thereto" section below.

Finally, key wallet **100** can be used in connection with a certification module **800** and/or a verification module **900** for added security against still other types of attacks. Certification module **800** is used to certify the public key corresponding to the private key, and verification module **900** is used to verify signatures created by the private key. The certification and verification modules are described in greater detail, respectively, in the "Certification Module" and "Verification Module" subsections below.

1. Key Wallet

A first embodiment of the generation camouflaging technique is depicted schematically in **Figure 2**. Figure 2 illustrates a key wallet including a seed derivation module **120** including a memory **210** (which may include a hardware memory, for example RAM, or a software memory such as a buffer or other data storage field) in which an input PIN **110** is temporarily stored after being input by the user is stored. In this embodiment, the stored relationship between the seed and the PIN may be regarded as a unit (or multiply-by-one) function that is effectively implemented by passing the seed through memory **210**. Thus, the key generation module **130** uses the PIN itself as the seed to produce output key(s) **140**. The key generation function would, therefore, typically accept a seed of the same length as the PIN.

The key generation function **130** may be, for example, a key generation function for DES or any other symmetric cryptography system, or a private key generation function

for RSA or any other asymmetric key signature system. Of course, the key generation function could be any other data generation function where the quantity being generated is not a cryptographic key. Those skilled in the art will recognize that the function **130** need not be limited to a function for generating confidential data or information in the field of cryptography, but may also include a function that generates confidential data or information in any other technical field.

Figure 3 illustrates a key wallet that implements the generation camouflaging technique according to a second embodiment of the invention. In this embodiment, seed derivation module **120** includes a memory **210** and a padding module **310**. This combination is used where the seed is essentially the PIN (as in the first embodiment), although padding is required because the seed is longer than the PIN. For example, a typical PIN is of a reasonable length that can be easily remembered by the user. In many applications, the PIN is 4-6 bytes long. However, an input value to a typical confidential data generation function is often much longer for purposes of spreading out the valid confidential data among many invalid ones. In the second embodiment, the padding function may simply add zeroes to the PIN (or otherwise pad the PIN) so that the resulting number will have the requisite number of bytes to serve as the input seed value to the key generation function **1330**. More generally, those skilled in the art will appreciate that the memory **210** and padding module **310** in seed derivation module **120** could be replaced by other modules representing a single function or a plurality of functions.

Figure 4 illustrates a third embodiment of a key wallet that implements generation camouflaging. The third embodiment has specific application to the RSA or any other asymmetric signature system. In this third embodiment, the seed derivation module includes an exclusive-OR ("XOR") function **420**. The XOR function operates on two inputs, an input PIN value (*pin*) stored in a first memory **210** and a masked value ($seed_{masked}$) (i.e., a stored representation of the original seed) stored in a second memory **410**, to generate the input seed value (*seed*) to a key generation module **130**. The boolean expression for deriving the seed is:

$$seed = pin\ XOR\ seed_{masked},$$

with the masked seed value ($seed_{masked}$) representing a relationship between the valid PIN and the original seed given by:

$seed_{masked} = pin_{valid}$ XOR $seed_{orig}$,

where *valid_pin* is an authorized valid PIN, and *seed_orig* is the original seed to the generation function which produced the correct private key. Thus, when the valid PIN is supplied as the input PIN, the exclusive-OR operation on the masked seed value ($seed_{masked}$) unmasks the original seed value ($seed_{orig}$), and the key generation module **130** reproduces the correct private key. On the other hand, when an incorrect PIN is supplied as the input PIN, the exclusive-OR operation fails to unmask the original seed value, and the private key generation function **130** produces an incorrect private key.

In certain implementations of the above example, the PIN value may be only 4-6 bytes long, whereas the seed values may be as long as 20 bytes. Therefore, there may be a mismatch of byte size when the XOR operations are performed. In such cases, it is understood that the PIN value can be XORed with an initial or final or other segment of the seed, leaving the rest of the seed untouched.

Further, the private key generation function may be a stand-alone private key generation function or it may be a function that generates a matching pair of public and private keys. In the latter case, the public key output need not be used.

The third embodiment described above permits changing of the user-selected PIN. When a user changes his PIN, a new masked seed value is stored in the second memory **410**. The new masked seed value may be based, for example, on the following boolean relationship:

$seed_{masked\_new} = pin_{valid\_new}$ XOR pin_{valid_old} XOR $seed_{masked\_old}$

In the foregoing examples, an XOR function is used to perform the masking of the seed. However, those skilled in the art will readily appreciate that other functions may also be used, so that masking should generally be understood to include storage of the seed in a form from which the seed can be reproduced using an operation involving a PIN.

Figure 5 illustrates a fourth embodiment of the invention. In this embodiment, the user-entered PIN (*valid_pin*) of byte length *m* is an initial segment of the seed, and the remainder of the seed is stored in the key wallet at memory **210** along with a concatentor **520** and a key generation module **130**. When the user enters an input PIN (*pin*), the

concatenator **520** concatenates the input PIN to the stored remainder to produce the input seed value to the key generation function **130**. When the user's correct PIN is entered, the original seed value and thus the correct private key value are reconstructed. On the other hand, when the wrong PIN is entered, a different seed value and thus an incorrect private key are produced.

Those skilled in the art will readily appreciate that all of the foregoing (and many other) private key generation functions and seed derivation techniques can be used in conjunction with generation camouflaging techniques. In addition, although the exemplary embodiments have been described in the context of private key generation using a PIN, those skilled in the art will realize that the techniques disclosed herein are applicable to camouflaging virtually any confidential datum, using passwords or any user-held access codes. It is therefore intended that the scope of the invention not be limited to the particular embodiments disclosed herein, but rather to the full breadth of the claims appended hereto.

2. Attacks on Key Wallets and Responses Thereto

We now enumerate the kinds of attacks which hacker may mount on digital key wallets, and means to resist such attacks. In the interest of clarity, the discussion will be set forth with respect to the RSA public key signature system. However, those skilled in the art will appreciate that the basic elements of the discussion are applicable to other systems as well, cryptographic and non-cryptographic, where a confidential data is generated by a special function.

a. Brute Force Attack

When a malicious hacker gets a copy of a conventional key wallet, the hacker simply tries every combination of possible PINs until a private key is released. Since in a conventional key wallet, a private key is released only when the correct PIN is input, the hacker will know that the PIN that released the private key is the correct PIN. To resist this type of attack, the key wallet may be configured to always release some quantity in response to any input PIN (or, in any event, releasing a quantity in response to a significant number of input PINs, instead of releasing only upon entry of the correct PIN).

In addition (as discussed below with respect to Ill-Formed Key Attacks), the key generation module may be configured so that its output appears in the proper form of a private key.

b. Ill-Formed Key Attack

Another attack is one in which the malicious hacker tries all possible PINs and examines the form of the resulting private key. If the resulting private key is not well formed, the hacker knows that the PIN producing such private key cannot be the correct PIN. Thus, the above-described embodiments of key generation modules may be configured to always produce well-formed keys, and as a result, the hacker cannot examine the well-formedness of the key to arrive at the correct PIN. Private key generation functions that generate well-formed keys in RSA and other public key encryption systems are well known and will not be discussed in further detail. Characteristics of (and, therefore, techniques for producing) valid-looking private keys are well known in the art (see, e.g., a description of private key formats in *Schneier*, *Menezes*, and U.S. Patent 6,170,058, which is hereby incorporated by reference) and need not be discussed in detail.

c. Known Public Key Attack

In this attack, the malicious hacker has access to two pieces of information: (a) the user's key wallet, and (b) the user's public key, as might be readily available in a public key certificate directory. The attack is shown pictorially in **Figure 6**. The hacker will try all possible PINs **110** on the key wallet **100**. For each PIN, he would use the outputted private key **610** to encrypt an arbitrarily chosen sample message **620**, and then decrypt the encrypted message with the user's public key. If the decrypted message **630** matches the original sample message **620**, the hacker knows that he has discovered the correct PIN and regenerated the user's correct private key.

To resist this attack, one embodiment of generation camouflaging does not permit public keys to be truly public. As a matter of convenience, we shall call such limited-distribution public keys "pseudo-public keys" and we shall call certificates containing such pseudo-public keys "pseudo-public certificates." Specifically, pseudo-public

certificates contain the user's pseudo-public key in encrypted form. Only authorized parties can access a pseudo-public key to verify the user's signature. This is in contrast with the conventional use of public key certificates, where anybody can verify a public key signature. Of course, the key wallet and other aspects or embodiments of the present invention could be used with conventional certificates alone, but even greater security is provided if pseudo-public keys and certification are also used, as described herein. Those skilled in the art will readily appreciate that existing certification issuance devices and procedures may readily be adapted to accommodate the foregoing embodiment of the present invention. Therefore, the specific hardware and/or software implementations of this embodiment of a certification module need not be described in detail. Rather, only the differences from the conventional certificates will be described below. Readers skilled in the art will recognize that conventional certificates come in several formats, most notable of which is the X.509 format and its revisions; however, the essential elements of all the conventional formats are similar, when viewed in relation to the present invention.

Pseudo-public keys, in turn, implicate pseudo-public key certification and pseudo-public key verification techniques. Exemplary embodiments therefor are described below.

i. Certification Module

Referring again to **Figure 1**, one embodiment of a certification module **130** usable in combination with a generation camouflaging key wallet creates public key certificates that are somewhat different from the conventional public key certificates. Essentially, public keys as used herein are not truly public as with conventional methods, but are meant for limited distribution (e.g., within organizations, across intranets or otherwise within closed or pseudo-public enterprises).

A conventional public key certificate and one possible embodiment of a pseudo-public certificate are shown side by side in **Figure 7**. The conventional certificate includes the user's identifying information, the user's public key, and a certificate authority's (CA's) signature of the user's public key, the signature comprising an encryption, under the CA's private key, of a hash of the user's public key.

The exemplary pseudo-public certificate may have the same format as the conventional certificate. However, the body of the certificate **700** containing the pseudo-public key is encrypted in a manner that is readable only by an authorized verifier. For

example, in one implementation, the encryption could be performed under the public key of the authorized verifier. Only authentication servers that have access to the corresponding private key can unwrap the user's certificate to access the user's public key. If there are several authorized verifiers, the body of the certificate could carry several encrypted copies of the pseudo-public key, each copy being encrypted by the public key of one of the verifiers. Each enterprise or entity employing this approach would have a certificate server having the above-described certification module to support its pseudo-public certificates. Those skilled in the art will appreciate that configuration of the pseudo-public certificate, so that the public key is encrypted and can be decrypted only by authorized verifiers, may be achieved in many different ways using a variety of cryptographic algorithms. For example, in an alternate embodiment of the pseudo-public key certificate, the public key would be encrypted by a DES key, and the DES key would be encrypted by the public key of the authorized verifier.

The resulting certificate would then be signed by the certifying authority similar to a conventional certificate. It is the pseudo-public nature of public keys that provides for two potential advantages in key management. Firstly, since the certifying authority is explicitly aware of who is authorized to use the public-key certificates, the legal liability of the CA could, as a practical matter, be limited. This is in contrast to the conventional certificate where the CA has no prior knowledge of who will use the certificate. Secondly, revoking a public-key certificate is facilitated, since the CA only has to notify those verifiers authorized to use the public-key certificates.

Certificates of the proposed form will be issued by the certification module, acting as a certificate server as shown in **Figure 8**. As those skilled in the art will appreciate, the certificate server comprises a series of modules that can be implemented in software, hardware, or a combination thereof. The user who wishes to be certified submits a digitally signed request for such as input **810** to the certificate server **800**. Such a request would typically contain the user's public key that is to be certified, along with his name or other identifying attributes. The certificate server would verify the user's digital signature using the submitted public key. If the signature verifies correctly, the server would check the user's identity information in a database, and then issue a public key certificate **820** of the proposed form as output. Those skilled in the art will recognize that the user identity database could be supplanted by other sources of information to verify the identity of the user requesting the certificate.

An alternate realization of the pseudo-public certificate server could involve a modification unit to be attached to a conventional certificate server. Such an add-on unit could operate on the input or the output of the conventional certificate server. In the event the modification unit operates on the input, it would repackage the request for the certificate by encrypting the users public key, and embed the encrypted public key among the identification attributes. The modification unit would then attach a dummy public key to the request, sign the request with the associated private key and pass on the request to the conventional certificate server. The output of the conventional certificate server would be a certificate containing the encrypted public key of the user as one of the identifying attributes. In the event the modification unit operates on the output of a conventional certificate server, the unit would repackage the conventional certificate produced by the conventional certificate server by encrypting the public-key exponent in the certificate in situ, and then overwriting the signature of the certificate server with a fresh signature of the modified certificate. Persons skilled in the art will appreciate that still other alternative embodiments are possible.

ii. Verification Module

Certification of a pseudo-public key by a CA results in the creation of pseudo-public certificates for distribution to parties communicating or conducting transactions with the holder of the private key corresponding to the pseudo-public key. Such parties will desire to verify pseudo-public keys received from unknown private key holders prior to relying on the pseudo-public keys for decrypting messages encrypted with the corresponding private keys.

Referring again to **Figure 1**, one embodiment of a verification module **120** usable in combination with pseudo-public keys and certificates is disclosed. The verification module differs in two ways from the verification module in conventional systems. The verification module respects the pseudo-public nature of the public key certificate, and takes appropriate steps to extract a user's public key from the certificate before verifying the user's signature. In an exemplary embodiment, these would include receiving a certificate containing an encrypted pseudo-public key of the certificate holder, and using the private key of an authorized verifier to decrypt the pseudo-public key. The verification module would then use the pseudo-public key to verify a digital signature in a message

sent by the certificate holder. In an alternative embodiment, if a DES key had been used to encrypt the pseudo-public key, the DES key would first be decrypted using the private key of the verifier, and in turn the DES key used to decrypt the pseudo-public key. No matter what the decryption mechanism, the verification module may also include logic to detect break-in attempts by fraudulent hackers, e.g., those signing messages with incorrect candidate private keys corresponding to the incorrect access codes of the key wallet. In such a case, a fraudulent hacker might steal or otherwise obtain the legitimate user's pseudo-public certificate and send the certificate along with a fraudulent message signed with the incorrect candidate private key. The inconsistency between the legitimate user's correct pseudo-public key in the certificate and the incorrect candidate private key allows detection of the fraudulent user. In particular, in one embodiment, if a particular user's signature is not verified in three successive attempts, the verification module concludes that a break-in might be in progress, and freezes the user's access privileges pending further investigation. In addition to (or instead of) freezing the access, the verification module might sound an alarm alerting an operator of the attempted break-in. There are other methods of detecting break-in attempts at the verification module, and other possible courses of action upon detecting a break-in. As those skilled in the art will appreciate, the verification module will compromise a series of logic modules that can be implemented in software, hardware, or a combination thereof.

3. Modifications, Enhancements and Alternate Embodiments

The foregoing has described various aspects of generation camouflaging. Although in one exemplary embodiment, the key wallet, the seed and key generation module, the key verification module and the key certification module are all used together to provide a secure technology for cryptographic key storage and use, those skilled in the art will appreciate that in alternative embodiments, various subsets of the whole system may also be combined for particular applications not requiring all of the modules.

In addition, although the foregoing has been described with respect to an exemplary software-based system, this is not strictly necessary. For example, some or all of the modules could be deployed using microcode and PLAs or ROMs, general purpose programming language and general purpose microprocessors, or ASICs. That is, the techniques described herein are not limited to software per se, but could be deployed in

virtually any form of logic, including pure software, a combination of software and hardware, or even hardware alone.

The key wallet may be carried by the user or, alternatively, stored remotely and downloaded over a network to a user "on the fly" using "roaming" techniques such as those disclosed in pending U.S. Patent Application 09/196,430, filed on Nov. 19, 1998, which is hereby incorporated by reference.

In addition, although various embodiments or aspects have been described with regard to RSA cryptography (for the public and/or pseudo-public keys and public and/or pseudo-public certificates) or DES cryptography (for the PIN encryption and storage of the pseudo-public key on the pseudo-public certificate), those skilled in the art will appreciate that many modifications and enhancements to such exemplary cryptographic technology are possible. More generally, each of the aforementioned operations can be implemented from a broad choice of cryptographic techniques, including many kinds of asymmetric or symmetric encryption as well as CRCs, hashes, message digests, or other one-way functions. For example, an asymmetric encryption operation could be replaced with a (optionally keyed) one-way function where integrity is the primary concern, or encryption of a symmetric session key followed by use of the session key for plaintext encryption, and various other alternatives that are well-known to those skilled in the art.

Furthermore, in many applications, the inputted PIN is about 4-6 bytes and is typically smaller than the seed value. However, it is also possible that the seed value may be smaller than the PIN value. This could be accommodated, for example, by doing a many-to-one hash of the PIN, and using this smaller byte sequence as the seed. Thus, there is no absolute requirement as to the relationship between the seed length and the PIN length. Rather, this relationship is determined by the choices and requirements of the system as deployed.

Also, although the exemplary embodiment has been described with respect to PINs protecting a private key, those skilled in the art will realize that the same technology of generation camouflaging can be used with other types of access codes and cryptographic representations to protect any regeneratable access-controlled datum. For example, the exemplary embodiments disclosed in this specification rely on the seed derivation module having access to a stored representation of the original seed used to create the private key, or a known relationship between the original seed and the user's PIN. However, the system is also operable where the user's private key (or, more generally, an access-

controlled datum) was either not based on a seed value or the seed value is not known at the time the generation camouflaging digital wallet is implemented. More particularly, suppose that the access-controlled datum is derivable within the key generation (or, more generally, data generation) module upon receipt of an operand. This is achievable with any one-to-one mapping between the access-controlled datum and the operand where the operand is representable as a function of, and is thus reproducible from, the user's PIN using the seed derivation module.

As a first example using the XOR seed derivation module functionality disclosed above, the seed derivation module would store a $operand_{masked} = pin$ XOR *operand*, so that the operand is derivable as $operand = operand_{masked}$ XOR *pin*. The other exemplary embodiments of seed derivation module functionality (e.g., *operand* = *pin* with or without padding) are also usable for this purpose. The foregoing shows that the operand is functionally equivalent to the "seed," and the one-to-one mapping between the operand and the access controlled datum is functionally equivalent to the seed-based key generation routine in the key generation module. Thus, the term "seed" should not be limited exclusively to a quantity that was used to initialize the user's private key, but rather to other quantities of the type described in this paragraph. Similarly, the term "key" should not be limited exclusively to a quantity usable for cryptographic operations, but rather to other access-controlled data generally.

As a second example, suppose the PIN (or, more generally, access code) space is of size *pinspace_size* (e.g., for 6 digit PINs, *pinspace_size* =1,000,000). The seed derivation module can store up to *pinspace_size* PINs listed in some order and, for each PIN, a corresponding seed. These PIN-seed pairs would include not only the correct (or initialization) seed corresponding to the correct PIN, but also (some or all) likely values of incorrect seeds corresponding to incorrect PINs. When one enters a PIN to release the key, the PIN is interpreted as an index into the list, and the indexed seed is selected. When one inputs the correct PIN, the correct seed is derived. When one inputs an incorrect PIN, an incorrect seed is derived. The seed is then used as input to the key (or, more generally, data) generation module.

As a third example, consider the second example where, instead of storing seeds corresponding to input PINs, the generation camouflaging wallet directly stores keys corresponding to input PINs without the need for a two step (seed derivation plus key generation) process. This can be regarded as a simplified form of generation camouflage.

In the previous discussions, the seed plus PIN plus generation functionality together produce the key. In this example, the list, PIN, and selection functionality produces the key. The generation camouflage mechanism has done away with the explicit seed, by storing the list explicitly, so that the generation routine has been reduced or simplified to a means of selecting an item in a list. In this way, the seed derivation functionality has been merged into the key generation functionality, so that the derived seed is the same as the outputted key. This example can be regarded as a variant of generation camouflage in which a list of data items is stored in the clear, and a PIN is used as an index to select one of the items.

In view of all the foregoing disclosed (and still other possible) embodiments and implementations of the general technique of generation camouflage, it is therefore intended that the scope of the invention be limited only by the claims appended below.

What is claimed is:

1. A digital wallet, secured with a user's access code, for reproducing a confidential datum for said user, said digital wallet comprising:
   (a) a computer-implemented input for receiving a input access code;
   (b) a seed derivation module operatively connected to said input, for deriving a seed usable to generate at least a portion of said confidential datum;
   (c) a seed-based data generation module
      (i) implementing a predetermined data generation protocol that was previously used by a seed-based initialization of said confidential datum of said user,
      (ii) containing a representation of a seed-access code relationship,
      (iii) configured to generate an output datum by digitally processing said derived seed in accordance with said seed-access code relationship, and
      (iv) said output datum reproducing said at least a portion of said user's confidential datum if said input access code equals said user's access code; and
   (d) said generation of said output datum occurring without dependence on any storage of any form of said at least a portion of said confidential datum.

2. The wallet of claim 1 where said output datum does not reproduce said at least a portion of said user's confidential datum if said input access code does not equal said user's access code.

3. The wallet of claim 2 where said output datum has the characteristic appearance of said at least a portion of said confidential datum.

4. The wallet of claim 1 where said access code is a PIN, and said confidential datum includes an asymmetric cryptographic key.

5. The wallet of claim 4 where said output datum has the characteristic appearance of an asymmetric cryptographic key.

6. The wallet of claim 1 where said access code is a PIN, and said confidential datum includes a symmetric cryptographic key.

7. The wallet of claim 1 where said seed-access code relationship is a identity relationship, so that said derived seed equals said input access code.

8. The wallet of claim 1 where said seed-access code relationship represents said derived seed as a padded version of said input access code.

9. The wallet of claim 1 where said seed-access code relationship includes a version of said initial seed masked by user's access code.

10. The wallet of claim 9 where:

(i) said masked version of said initial seed includes an XOR of said initial seed with said user's access code; and

(ii) said processing of said derived seed in accordance with said seed-access code relationship includes XORing said masked version of said initial seed with said derived seed.

11. The wallet of claim 10 further comprising program code for updating an user's old access code with a user's new access code by replacing said stored masked version of said initial seed with its value XORed with said user's old access code XORed with said user's new access code.

12. The wallet of claim 1 where:

(i) said seed-access code relationship includes a truncated version of said initial seed capable of being concatenated with said input access code to form said derived seed; and

(ii) said processing of said derived seed in accordance with said seed-access code relationship includes concatenating said truncated version of said initial seed with said input access code.

13. The wallet of claim 1 where:
    (i) said seed-access code relationship includes values of, and associations between, a plurality of possible values of said input access code and a corresponding plurality of possible values of said derived seed; and
    (ii) said processing of said derived seed in accordance with said seed-access code relationship includes looking up and outputting said possible value of said derived seed corresponding to said input access code.

14. The wallet of claim 13 where:
    (1) said seed derivation module is merged with said data generation module;
    (2) said output datum includes said derived seed.

15. The wallet of claim 5 where said confidential datum includes a private key of said user, and said output datum has the characteristic appearance of a private key.

16. The wallet of claim 5 where said user's public key corresponding to said user's private key is pseudo-public.

17. The wallet of claim 16 further comprising a digital certificate containing said pseudo-public key.

18. The wallet of claim 17 where said digital certificate includes an encrypted version of said user's pseudo-public key encrypted under a certifier's key which is not verifiable except by authorized verifiers.

19. The wallet of claim 1 configured to be remotely accessible to a roaming user across a network.

20. A computer-implemented method for securely storing and reproducing a confidential datum for said user, comprising:
    (a) receiving an input access code;
    (b) deriving a seed usable to generate at least a portion of said confidential datum by using said received input access code;

(c) obtaining a representation of a seed-access code relationship;

(d) digitally processing said derived seed

(i) in accordance with said seed-access code relationship,

(ii) by executing a predetermined data generation protocol that was previously used by a seed-based initialization of said confidential datum of said user,

(iii) thereby producing an output datum reproducing said at least a portion of said user's confidential datum if said input access code equals said user's access code;

(e) said generation of said output datum occurring without dependence on any storage of any form of said at least a portion of said confidential datum.

21. The method of claim 20 where said output datum does not reproduce said at least a portion of said user's confidential datum if said input access code does not equal said user's access code.

22. The method of claim 21 where said output datum has the characteristic appearance of said at least a portion of said confidential datum.

23. The method of claim 20 where said access code is a PIN, and said confidential datum includes an asymmetric cryptographic key.

24. The method of claim 20 where said seed-access code relationship is a identity relationship, so that said derived seed equals said input access code.

25. The method of claim 20 where said seed-access code relationship represents said derived seed as a padded version of said input access code.

26. The method of claim 20 where said seed-access code relationship includes a version of said initial seed masked by user's access code.

27. The method of claim 26 where:
(i) said masked version of said initial seed includes an XOR of said initial seed with said user's access code; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes XORing said masked version of said initial seed with said derived seed.

28. The method of claim 20 where:
(i) said seed-access code relationship includes a truncated version of said initial seed capable of being concatenated with said input access code to form said derived seed; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes concatenating said truncated version of said initial seed with said input access code.

29. The method of claim 20 where:
(i) said seed-access code relationship includes values of, and associations between, a plurality of possible values of said input access code and a corresponding plurality of possible values of said derived seed; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes looking up and outputting said possible value of said derived seed corresponding to said input access code.

30. The method of claim 29 where:
(1) said deriving said seed and said executing said predetermined data generation protocol are merged into a common operation; and
(2) said output datum includes said derived seed.

31. A computer-readable medium having stored thereon a program executable on a computer to securely store and reproduce a confidential datum for said user, the program comprising computer logic instructions for:
(a) receiving an input access code;

(b) deriving a seed usable to generate at least a portion of said confidential datum by using said received input access code;

(c) obtaining a representation of a seed-access code relationship;

(d) digitally processing said derived seed

(i) in accordance with said seed-access code relationship,

(ii) by executing a predetermined data generation protocol that was previously used by a seed-based initialization of said at least a portion of said confidential datum of said user,

(iii) thereby producing an output datum reproducing said at least a portion of said user's confidential datum if said input access code equals said user's access code;

(e) said generation of said output datum occurring without dependence on any storage of any form of said at least a portion of said confidential datum.

32. The computer-readable medium of claim 31 where said output datum does not reproduce said at least a portion of said user's confidential datum if said input access code does not equal said user's access code.

33. The computer-readable medium of claim 32 where said output datum has the characteristic appearance of said at least a portion of said confidential datum.

34. The computer-readable medium of claim 31 where said access code is a PIN, said confidential datum includes an asymmetric cryptographic key.

35. The computer-readable medium of claim 31 where said seed-access code relationship is a identity relationship, so that said derived seed equals said input access code.

36. The computer-readable medium of claim 31 where said seed-access code relationship represents said derived seed as a padded version of said input access code.

37. The computer-readable medium of claim 31 where said seed-access code relationship includes a version of said initial seed masked by user's access code.

38. The computer-readable medium of claim 37 where:
(i) said masked version of said initial seed includes an XOR of said initial seed with said user's access code; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes XORing said masked version of said initial seed with said derived seed.

39. The computer-readable medium of claim 31 where:
(i) said seed-access code relationship includes a truncated version of said initial seed capable of being concatenated with said input access code to form said derived seed; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes concatenating said truncated version of said initial seed with said input access code.

40. The computer-readable medium of claim 31 where:
(i) said seed-access code relationship includes values of, and associations between, a plurality of possible values of said input access code and a corresponding plurality of possible values of said derived seed; and
(ii) said processing of said derived seed in accordance with said seed-access code relationship includes looking up and outputting said possible value of said derived seed corresponding to said input access code.

41. The computer-readable medium of claim 40 where:
(1) said deriving said seed and said executing said predetermined data generation protocol are merged into a common operation; and
(2) said output datum includes said derived seed.

42. A method for camouflaging a user's generation-camouflaged access-controlled datum under said user's access code, comprising:

(a) initializing a user's access-controlled datum by using a generation protocol in accordance with a generation indicia;

(b) storing in a memory a predetermined relationship between said generation indicia and said user's access code;

(c) camouflaging at least a portion of said access-controlled datum

(i) such as to be reproducible by an authorized user thereof but non-reproducible by an unauthorized user thereof,

(ii) said camouflaging including storing said predetermined relationship between said generation indicia and said user's access code;

(iii) thereby allowing subsequent accessing of said at least a portion of said access controlled datum via computer-based processing of an inputted access code, in accordance with said stored generation indicia-access code relationship;

(iv) without dependence on any storage of any form of said at least a portion of said access-controlled datum;

(d) storing said camouflaged at least a portion of said access-controlled datum in a digital wallet; and

(e) providing said digital wallet to said user.

43. A method for camouflaging a user's generation-camouflaged access-controlled datum under said user's access code, comprising:

(a) initializing a user's access-controlled datum by using a generation protocol in accordance with a generation indicia;

(b) generation-camouflaging at least a portion of said access-controlled datum such as to be reproducible by an authorized user thereof but non-reproducible by an unauthorized user thereof;

(c) storing said generation-camouflaged at least a portion of said access-controlled datum in a digital wallet; and

(d) providing said digital wallet to said user.

Figure 1

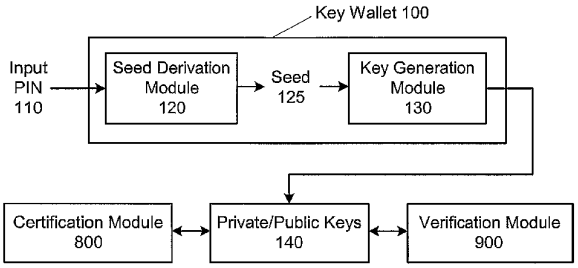


Figure 2

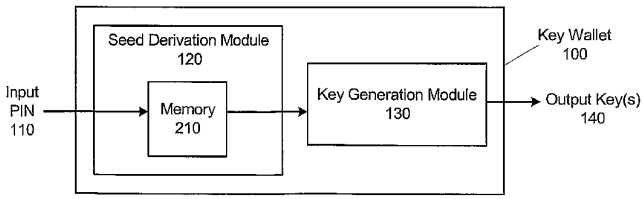


Figure 3

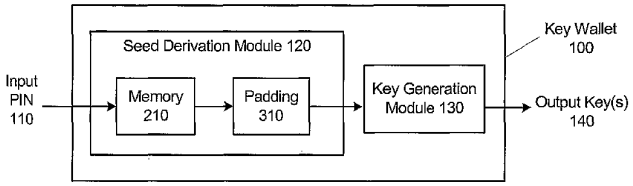


Figure 4

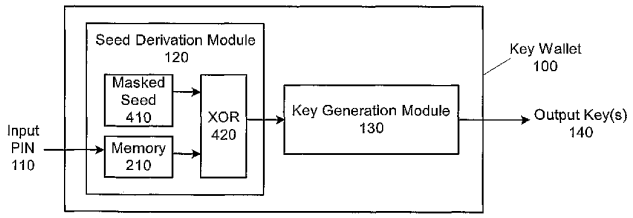


Figure 5

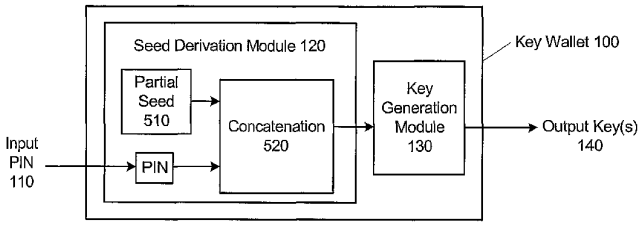


Figure 6

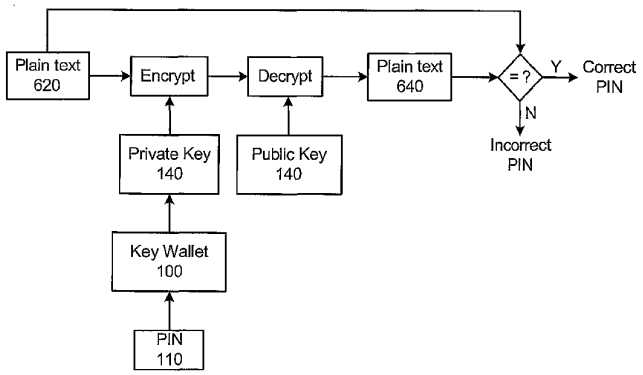


Figure 7

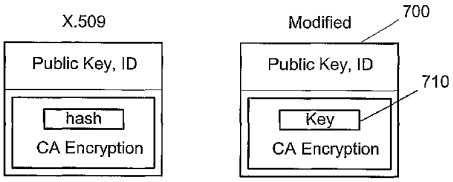
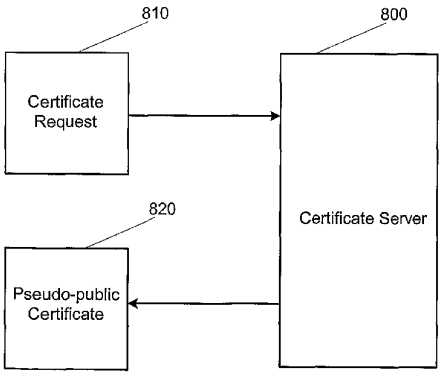


Figure 8



【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/09812

A. CLASSIFICATION OF SUBJECT MATTER
IPC(7) : H04L 9/00
US CL : 713/184
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
U.S. : Please See Continuation Sheet

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 6,170,058 B1 (KAUSIK) 02 JANUARY 2001 (2.1.2001), abstract, fig. 4-11, col. 1 lines 30-47, col. 4 lines 60-67, col. 5 lines 52-67, col. 6 lines 1-43,col. 7 lines 60-67, col. 8 lines 40-40-53, col. 9 lines 14-38, col. 10 lines 3-25, col. 11 lines 24-67, col. 12 lines 1-41, col. 13 lines 5-67.	1-43
Y	US 5,206,905 A (LEE et al.) 27 April 1993 (27.4.1993) abstract, fig. 5-8, col. 2 lines 48-67, col. 3 lines 25-50, claims 1-3.	1-43
A	US 5,894,519 A (CLEMOT et al.) 13 April 1999 (13.4.1999) Entire Document	1-43
A	US 6,408,388 B1 (FISCHER) 18 June 2002 (18.6.2002) Entire Document.	1-43
A	US 3,798,605 A (FEISTEL) 19 March 1974 (19.3.1974) Entire Document.	1-43
A	US 5,668,876 A (FALK et al.) 16 September 1997 (16.9.1997) Entire Document.	1-43

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
25 June 2002 (25.06.2002)	31 JUL 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No.	Authorized officer Gilberto Barron For James R. Matthews Telephone No. 703-305-3900

Form PCT/ISA/210 (second sheet) (July 1998)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US02/09812

Continuation of B. FIELDS SEARCHED Item 1:
713/184, 156, 159, 171
380/282,44,30
705/66,67,72,76

フロントページの続き

(81)指定国 AP(GH,GM,KE,LS,MW,MZ,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,CH,CY,DE,DK,ES,FI,FR,GB,GR,IE,IT,LU,MC,NL,PT,SE,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,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,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,NO,NZ,OM,PH,PL,PT,RO,RU,SD,SE,SG,SI,SK,SL,TJ,TM,TN,TR,TT,TZ,UA,UG,UZ,VN,YU,ZA,ZM,ZW

Ｆターム(参考) 5B017 AA03 BA07 BB10 CA16
5J104 EA22 NA37 NA39 NA40 NA42