(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2008-252299
(P2008-252299A)
(43)公開日　平成20年10月16日(2008.10.16)

(51)Int.Cl.	F I		テーマコード（参考）
H04L 9/10 (2006.01)	H04L	9/00　621A	5J104

審査請求 未請求　請求項の数 20　ＯＬ　（全 24 頁）

(21)出願番号　特願2007-88812(P2007-88812)
(22)出願日　平成19年3月29日(2007.3.29)

（出願人による申告）国等の委託研究の成果に係る特許出願（平成１８年度　独立行政法人情報通信研究機構「モバイル端末におけるセキュリティ保護技術に関する研究開発」委託研究、産業再生法第３０条の適用を受ける特許出願）

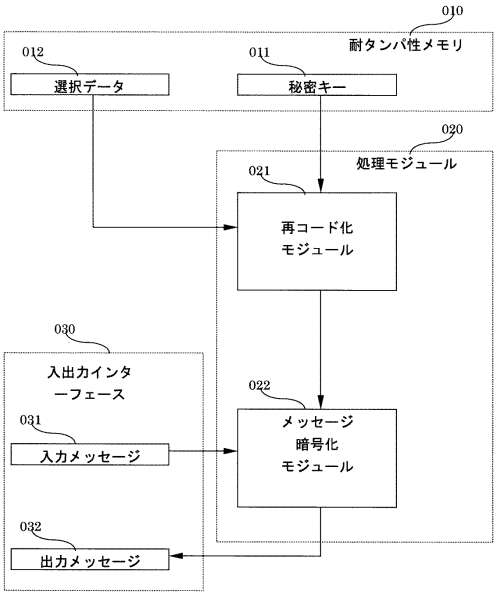
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　110000350
ポレール特許業務法人
(72)発明者　ヴィオム　カミーユ
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内
(72)発明者　桶屋　勝幸
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内

最終頁に続く

(54)【発明の名称】暗号処理システム及び暗号処理方法

(57)【要約】
【課題】　秘密データとサイドチャネル情報との相関関係を取り除き、同一の秘密鍵の重複かつ安全な利用を可能にする暗号化処理システムと方法を提供する。
【解決手段】　暗号化処理動作を安全に実行するための暗号処理システムは、（ａ）暗号化処理すべき入力メッセージを受信するための第１の手段と、（ｂ）秘密鍵と選択データを記憶するメモリと、（ｃ）再コード化された鍵を生成するため、（１）前記秘密鍵を処理して少なくとも少なくとも二つの再コード化した鍵を生成する所定の変換モジュールと（２）前記選択データに従って、再コード化した鍵の中から一の再コード化した鍵を選択する選択モジュールとを備えた再コード化モジュールと、（ｄ）前記入力メッセージを前記再コード化した鍵に従って暗号化する第２の手段とを備え、前記秘密鍵と前記選択データの一対は、一意的に決定される。
【選択図】　図１

図１



【特許請求の範囲】
【請求項１】
暗号化処理動作を安全に実行するための暗号処理システムであって、
（ａ）暗号化処理すべき入力メッセージを受信するための第１の手段と、
（ｂ）秘密鍵と選択データを記憶するメモリと、
（ｃ）再コード化された鍵を生成するため、（１）前記秘密鍵を処理して少なくとも二つの再コード化した鍵を生成するための、少なくとも２個の所定の変換モジュールと、（２）前記選択データに従って、再コード化した鍵の中から一の再コード化した鍵を選択する選択モジュールとを備えた再コード化モジュールと、
（ｄ）前記入力メッセージを前記再コード化した鍵に従って暗号化する第２の手段とを備え、
前記秘密鍵と前記選択データの一対が、一意的に決定されることを特徴とする暗号処理システム。
【請求項２】
前記請求項１に記載した暗号処理システムにおいて、更に、前記秘密鍵を処理すると共に前記選択データを生成し、もって、前記秘密鍵によって前記選択データを一意的に決定する選択データ生成モジュールを備えていることを特徴とする暗号処理システム。
【請求項３】
前記請求項２に記載した暗号処理システムにおいて、前記選択データ生成モジュールは、前記秘密鍵をシードとして利用して乱数を生成する処理を行う乱数生成器を備えており、もって、選択データは乱数であることを特徴とする暗号処理システム。
【請求項４】
前記請求項２に記載した暗号処理システムにおいて、前記選択データ生成モジュールは、ハッシュ関数からなることを特徴とする暗号処理システム。
【請求項５】
前記請求項４に記載した暗号処理システムにおいて、前記ハッシュ関数はSHA-1であることを特徴とする暗号処理システム。
【請求項６】
前記請求項２に記載した暗号処理システムにおいて、前記選択データ生成モジュールは、ブロック暗号からなることを特徴とする暗号処理システム。
【請求項７】
前記請求項１に記載した暗号処理システムにおいて、前記再コード化モジュールは、
（ａ）前記秘密鍵の少なくとも一のビットを抽出するためのビット抽出モジュールと、
（ｂ）前記抽出されたビットを処理すると共に、少なくとも二つのデジット候補を生成するための、少なくとも２個の所定変形モジュールと、
（ｃ）前記選択データに従って、前記デジット候補から再コード化されたデジットを選択する選択モジュールと、そして、
（ｄ）前記ビット抽出モジュール、前記変形モジュール、及び、前記選択モジュールを、対話形式により、前記秘密鍵の全てのビットが抽出されるまで実行する制御モジュールとを備えており、
前記再コード化された秘密鍵は、前記再コード化モジュールにより生成された前記再コード化された複数のデジットを備えていることを特徴とする暗号処理システム。
【請求項８】
入力メッセージのデジタル署名を生成するための、コンピュータシステムにおけるデジタル署名生成システムであって、
（ａ）前記入力メッセージを、秘密鍵に従って暗号化し、かつ、前記デジタル署名を出力とする、前記請求項１に記載した暗号処理システムと、
（ｂ）前記入力メッセージと前記デジタル署名とを、第２のコンピュータシステムへ転送するための手段と
を備えたことを特徴とするデジタル署名生成システム。

【請求項９】
第２のコンピュータによって暗号化された入力メッセージを復号するための、コンピュータシステムにおける復号化システムであって、前記請求項１に記載の暗号処理システムを備えており、もって、前記暗号処理システムが秘密鍵に従って前記入力メッセージを暗号化し、そして、当該復号化されたメッセージが前記暗号処理システムの出力メッセージとなっていることを特徴とする復号化システム。
【請求項１０】
第２のコンピュータシステムとセッション鍵を交換するための、コンピュータシステムにおける鍵交換システムであって、
（ａ）前記第２のコンピュータから受信した入力メッセージを秘密鍵に従って暗号化し、前記セッション鍵を出力メッセージとするための、前記請求項１に記載した暗号処理システムと、
（ｂ）前記セッション鍵に従ってデータを暗号化かつ復号化するための手段と、
（ｃ）前記セッション鍵で暗号化したデータを第２のコンピュータシステムと交換するための手段とを備えており、
前記コンピュータシステム及び前記第２のコンピュータシステムとが同じセッションデータを共有することを特徴とする鍵交換システム。
【請求項１１】
暗号処理システムにおける暗号化演算を安全に実行するための方法であって、
（ａ）入力メッセージを受信するステップと、
（ｂ）秘密鍵と選択データとを保存するためにメモリに保存するステップと、
（ｃ）前記秘密鍵を再コード化するステップであって、
（１）前記秘密鍵を処理して、少なくとも二つの再コード化した秘密鍵を生成し、
（２）前記選択データに従って前記再コード化した秘密鍵から一を選択し、そして、
（ｄ）前記入力メッセージを、前記コード化された秘密鍵に従って、暗号化するステップとを備えており、
前記秘密鍵と前記選択データとの対が、一意的に決定されることを特徴とする暗号処理方法。
【請求項１２】
前記請求項１１に記載された暗号処理方法であって、更に、選択データを、当該選択データが前記密秘鍵によって一意的に決定されるように、前記秘密鍵から生成するステップを備えていることを特徴とする暗号処理方法。
【請求項１３】
前記請求項１２に記載された暗号処理方法であって、前記選択データの生成ステップは、
（ａ）前記秘密鍵を、乱数生成方法のシードとして使用するステップと、
（ｂ）乱数を、前記乱数生成方法によって生成するステップとを備えており、
もって、前記選択データが前記乱数であることを特徴とする暗号処理方法。
【請求項１４】
前記請求項１２に記載された暗号処理方法であって、前記選択データの生成ステップでは、ハッシュ関数によってハッシュ化することを特徴とする暗号処理方法。
【請求項１５】
前記請求項１４に記載された暗号処理方法であって、前記ハッシュ関数はSHA-1であることを特徴とする暗号処理方法。
【請求項１６】
前記請求項１２に記載された暗号処理方法であって、前記選択データの生成ステップは、ブロック暗号による処理を含んでいることを特徴とする暗号処理方法。
【請求項１７】
前記請求項１２に記載された暗号処理方法であって、前記秘密鍵の再コード化のステップは、

（ａ）前記秘密鍵の少なくとも一つのビットを抽出するステップと、
（ｂ）前記抽出したビットを、少なくとも２つのデジット候補を生成するための少なくとも２個の所定変形によって処理するステップと、
（ｃ）前記選択データに従って、前記デジット候補から一の再コード化されたデジットを選択するステップと、
（ｄ）前記ステップ（ａ）から（ｃ）を、対話形式によって、前記秘密鍵の全てのビットが抽出されるまでに要求される回数、繰り返すステップとを備えており、
前記再コード化された秘密鍵は、前記再コード化ステップによって生成された複数の前記再コード化デジットを備えていることを特徴とする暗号処理方法。
【請求項１８】
入力メッセージのデジタル署名を生成するための、コンピュータシステムにおけるデジタル署名の生成方法であって、
（ａ）前記デジタル署名が、前記入力メッセージの処理ステップにより生成された出力メッセージとなるよう、前記入力メッセージを、前記請求項１１に記載した暗号処理方法に従って処理するステップと、
（ｂ）前記入力メッセージと前記デジタル署名とを、第２のコンピュータシステムへ転送するステップと
を備えたことを特徴とするデジタル署名の生成方法。
【請求項１９】
第２のコンピュータによって暗号化された入力メッセージを復号するための、コンピュータシステムにおける復号化の方法であって、前記請求項１１に記載の方法により前記入力メッセージを処理するステップを備えており、もって、復号化されたメッセージが前記入力メッセージの処理ステップによる出力となっていることを特徴とする復号化方法。
【請求項２０】
第２のコンピュータシステムとセッション鍵を交換するための、コンピュータシステムにおける鍵交換方法であって、
（ａ）前記セッション鍵を前記暗号処理方法からの出力メッセージとして、前記請求項１１に記載した暗号処理方法によって、前記第２のコンピュータから受信した入力メッセージを処理するステップと、
（ｂ）前記セッション鍵に従って、データを暗号化及び復号化するステップと、
（ｃ）前記セッション鍵で暗号化したデータを第２のコンピュータシステムと交換するステップとを備えており、
前記コンピュータシステム及び前記第２のコンピュータシステムとが同じセッションデータを共有することを特徴とする鍵交換方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、機密保護の分野における秘密データを安全に処理するための暗号処理方法、及び、そのための装置に関し、特に、スマートカード（ＩＣカード）、携帯電話、パーソナルコンピュータ、ワークステーション、サーバ等における公開鍵暗号化システム（public key cryptosystems）に関する。
【背景技術】
【０００２】
公開鍵暗号化システムは、銀行業（banking applications）や電子取引（electronic commerce）等のため、より一般的には、デジタルの世界における安全のために、必須なものとなってきた。公開鍵暗号化システム（public key cryptosystems）のおかげで、安全とはいえない経路（チャネル：channels）を介して、共有秘密値を安全に決定することが可能である。また、公開鍵暗号化システムによれば、一方の者は、如何なる共有秘密値を事前に交換することなしに、他方の者へのデータを暗号化することが出来る。そして、最終的には、この公開鍵暗号化システムのおかげで、デジタルの署名を生成することができ

る。

【０００３】

暗号化システムは、理論的な見地からは安全ではあるが、実際上、慎重に実行されない場合には、破られ得る。特に、実行時間や電力消費などのサイドチャネル情報（side channel information）を利用することにより、攻撃者（attacker）は、慎重に実行されない場合には、秘密情報を暴くことができる。このサイドチャネル攻撃の概念は、例えば、装置の消費電力など、暗号化システムの物理的なパラメータを観測し、当該物理的パラメータから暗号情報を推測するものである。このような試みは、２つの理由から実行可能である。第１には、機密と暗号化アルゴリズムを実行する装置の挙動（動作）には相関関係があること。第２には、サイドチャネル情報も装置の挙動（動作）に相互に関連しており、例えば、実行されている動作に従属する電力の消費に依存することである。

【０００４】

かかる実行時間、電力消費、又は、電磁放射などの物理的情報のタイプに加え、サイドチャネル攻撃（side channel attack）にとっては、幾つかの方法がある。電力解析攻撃の場合には、攻撃者が単一の電力消費の波形（trace）を解析する単純電力解析（SPA：simple power analysis）と、そして、攻撃者が統計的なツールを用いて幾つかの電力波形を分析する差分電力解析（DPA：differential power analysis）とに区別される。

【０００５】

サイドチャネル攻撃に対する幾つかの対策では、サイドチャネル情報と機密データとの間の相関を取り除くため、機密データの表現（representation of secret data）が変形される（modified）。例えば、機密データの表現内に固定されたパターンを導入することが一般的であり、機密に応じた動作が同じパターンに従って編成され（organized）、SPAタイプの漏洩を阻止する。他のアプローチでは、幾つかの候補の中からランダムに表現を選択する。同様に、機密に依存する動作がランダムに再編成され（re-organized）、DPAタイプの漏洩を阻止する。

【０００６】

以下の特許文献１に記載された、耐（防）SPAの分数窓方式（SPA-resistant fractional window method）は、楕円曲線のためのランダム化されたサイドチャネル対抗策に属しており、暗号化ルーチンが呼び出される度に秘密表現をランダム化する。以下の第２の特許文献２に記載された発明では、ランダムなビット列を生成して、このビット列を利用して符号化演算のための記録領域をランダムに選択する方法が記載されており、しかしながら、秘密表現の変更は行わない。

【特許文献１】特開２００５－０５５４８８号公報

【特許文献２】特表２００４－０５５７５６号公報

【非特許文献１】「暗号化ハンドブック（Handbook of applied cryptography）」Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone: ””, CRC press, ISBN: 0-8493-8523-7

【発明の開示】

【発明が解決しようとする課題】

【０００７】

例えば、上記特許文献１又は２に記載された公開鍵の暗号化の実行には、しばしば、耐タンパ性（tamper-resistance）を確保するための対抗策が含まれる。しかしながら、上記の従来技術は、以下のような問題点があった。

【０００８】

かかる特許文献１の従来技術によれば、安全な表現のために秘密鍵を再利用することが出来ない。即ち、一方において、秘密鍵が単一の暗号化演算（cryptographic operation）に使用された場合には、サイドチャネル攻撃によって秘密情報を引き出すことは出来ないことから、上記特許文献１のようなランダム化技術は安全である。しかしながら、これに対し、秘密鍵が幾度か使用された場合には、新たな暗号化演算が実行される度に、攻撃者には新鮮な情報が提供されることから、攻撃者（attacker）は秘密に関する統計的な情

報を得ることができることとなる。

【０００９】

従って、上記従来技術に記載された発明の目的や効果の他に、本発明によって得られる目的や効果は、以下の通りである。

（１）秘密データとサイドチャネル情報との相関関係を取り除く。

（２）メッセージや交換鍵を復号し、又は、デジタル署名を生成するため、同一の秘密鍵を重複し、かつ、安全に利用することを可能にする。

【課題を解決するための手段】

【００１０】

本発明によれば、秘密データとサイドチャネル情報との相関関係を取り除くために、ランダム化された表現を利用するものである。従来技術における技術では、秘密データがランダム化の対策と関連して使用された場合、暗号化ルーチンにおける各実行において、秘密鍵が攻撃者に新たな情報を提供することから、攻撃者は統計的な情報を収集することが可能である。実際、従来技術では、ランダム性の源（source of randomness）は、ランダムなシードによって初期化される（擬似）乱数生成器、又は、ハードウェアからなる乱数生成器によるものである。本発明では、ランダム性の源は、一意的に、秘密鍵によるものであり、全てのランダムな選択は、秘密鍵の値によって決定される。より詳細には、本発明では、例えば、非可逆なハッシュ関数又はブロック暗号を利用して、秘密鍵から、一意的かつ決定論的に決められた一連のビットを生成する。そして、秘密鍵に対し同時に生成した幾つかの表現を演算して、生成したビット列に従って、その一を選択する。最後に、暗号化演算が、選択された表現に従って実行される。

【発明の効果】

【００１１】

本発明によれば、秘密データのランダム化された表現が、一意的に決定された選択データに従って選択される。それ故、メッセージが変えられても、秘密が同じままである限り、鍵交換、データの暗号化、又は、デジタル署名となる暗号化のアルゴリズムは、同一片（same iece）のサイドチャネル情報を出力する。それ故、攻撃者は、暗号化アルゴリズムを複数回呼び出すことによる利点を享受することが出来ない。或いは、同じ意味では、本発明になるランダム化された表現を基礎とした対応策によれば、同じ秘密鍵やデータを安全に再利用することが可能となる。

【発明を実施するための最良の形態】

【００１２】

＜一般設定：図１＞

本発明による最も一般的な設定では、２つの処理モジュールを含む処理モジュール020と、秘密データを格納するための耐タンパ性（tamper-resistant）のメモリ010と、入出力インターフェース030とを備えている。処理モジュールは、再コード化モジュール021と、暗号化モジュール022とを含んでいる。暗号化モジュール022の役割は、デジタル署名、復号、又は鍵交換プロトコルの枠内において、入力メッセージ031を秘密鍵に従って処理し、出力メッセージ032を出力することである。しかしながら、秘密鍵を単に実行すると情報を漏洩しかねないことから、それ故、本発明によれば、秘密鍵011への直接的なアクセスを避けるため、再コード化モジュール021を使用し、代わりに、再コード化した鍵が暗号化モジュール022によって処理が行なわれる。

【００１３】

再コード化モジュール021は、潜在的に、秘密鍵011について、他とは異なった幾つかの明瞭な再コードを出力しうるものであり、ここでは、秘密鍵の再コード化（recoding）を、一連のデジットによる鍵の表現と呼ぶ。ここでは、例えば、２進、１０進、又は１６進による表現が可能な表現である。しかしながら、より賢明な選択としては、上記特許文献１にあるような安全な表現であることが好ましい。実際、上記特許文献１で導入された表現は、サイドチャネル情報の漏洩と秘密鍵との間の相関関係を取り除く特性を備えている。

【００１４】
加えて、再コード化モジュールは、選択データ012に従って可能な再コード化（recoding）の一つを選択し、一つの秘密鍵011が一つの選択データ012に対して一意的に関連付けられている。例えば、選択データを非可逆的なハッシュ関数で処理することにより、秘密鍵から選択データを導出することが出来る。或いは、秘密鍵と同時に選択データを生成して、当該秘密鍵と選択データとからなる対を、その読出しアクセスを制御すると共に書込みを禁止する耐タンパ性（tamper-resistant）のメモリ011内に格納する。何れの場合でも、秘密鍵と選択データとの対は、一意に規定される。
【００１５】
以下、RSA（リベスト－シャミール－アルドレマン）アルゴリズムのための秘密鍵の安全な多重使用について、詳細に説明する。
【００１６】
＜コンピュータシステムとネットワーク：図２＞
ここで、コンピュータとしては、例えば、ワークステーション、サーバ、銀行端末、スマートカード（ＩＣカード）、携帯電話機、又は、データ格納部、通信部及び処理部を備えた各種の電子装置を指す。コンピュータは処理部112を備え、この処理部は複数の演算部を備えてもよく、少なくとも１つのCPU 114と、場合によっては、コプロセッサ115を備えてもよい。このコプロセッサは、或るタイプのオペレーションの演算、特に、公開鍵暗号化システムの場合における剰余演算に有用である。最も一般的な公開鍵の暗号化システムであるRSAアルゴリズム又は楕円曲線の演算を加速するため、コプロセッサは、CPUが実行出来るよりもオーダーの大きな速度で演算を実行する。
【００１７】
コンピュータは、３つのタイプのメモリ、即ち、電源がオフするとその内容が失われる揮発性のメモリであるRAM 103と、RAMより遅いが、読出しと書込みが可能であり、しかし、電源がオフしてもデータを保存することが出来る不揮発性のメモリであるEEPROM 107と、そして、その内容は電源がオフしても失われないが、しかしながら、読出しだけが可能な読出し専用のROM 108とを備えている。ROMはプログラムを格納し、一方、EEPROMは、プログラムやパッチ、そして、公開鍵やプライベート鍵のような長期保存データを保存可能である。RAMは揮発性であることから、短期保存データや一時保存データを保存することが出来るだけである。
【００１８】
また、コンピュータは、典型的には、ディスプレイ109又はキーボード110などの周辺機器とデータを送受信するための入出力インターフェース111を備えており、しかもネットワーク142に接続されている。
【００１９】
本発明による第１の可能なシナリオは、以下の通りである。即ち、コンピュータ101は、ネットワーク142から、又は場合によってはキーボード110を使用している人間であるユーザから、メッセージを入出力インターフェース111を介して受信する。次に、コンピュータ101は、メッセージのデジタル署名を生成する。かかるデジタル署名を生成する最も一般的な方法は、公開鍵暗号システムを利用することであり、かかる暗号システムは、一方は署名者（コンピュータ101）により保持され、他方は認証者（コンピュータ121）にアクセス可能な、２つの異なる鍵を生成する。加えて、秘密鍵で暗号化されたデータは、公開鍵により再暗号化することによって回復することが出来る。本発明に戻り、コンピュータ101は、デジタル署名を得るために、不揮発性メモリ106に保存された秘密鍵によってメッセージを暗号化する。この暗号化プロセスは、算術ユニット116により、特に、特殊な暗号化処理を汎用のCPU 114よりもより効率的に行なうことが可能なコプロセッサ115によって実現される。
【００２０】
最後に、コンピュータ101は、メッセージとその署名の双方を、ネットワーク142を介して、第２のコンピュータ121に送出し、そして、コンピュータ121は、コンピュータ101の

公開鍵を利用して署名を暗号化する。公開鍵暗号システムの特性によれば、署名が本当にコンピュータ101によって生成されたものであれば、コンピュータ121は、暗号化プロセスの中で、最初のメッセージを回復しなければならない。それ故、受信したメッセージを公開鍵で暗号化した署名と比較することにより、コンピュータ121は、当該メッセージMがコンピュータ101によって正しく書き込まれて署名されたことを確認することができる。

【００２１】

本発明の第２の可能なシナリオは、以下の通りである。即ち、コンピュータ101は、ネットワーク及び入出力インターフェース111を介して、第２のコンピュータ１２１から暗号化された入力メッセージを受信する。そして、この入力メッセージを公開鍵暗号システムにより暗号化する。又は、より詳細には、コンピュータ101の公開鍵により、コンピュータ121によって暗号化する。次に、コンピュータ101は、その秘密鍵を利用して入力メッセージを復号し、算術ユニット116と共に、特に、コプロセッサ115を利用して、元のメッセージを回復する。最後に、復号化したメッセージは、入出力インターフェースを介してディスプレイ109へ送出され、もって、人間であるユーザにより確認することが出来る。

【００２２】

本発明の第３のシナリオは、以下の通りである。即ち、２つのコンピュータ101と121とがネットワーク142を介して、公開鍵暗号システムを利用して共通のセッション鍵を交換する。最初に、コンピュータ101と121とが、公衆に知られた共通メッセージに同意する。それから、コンピュータ101は、その秘密鍵を利用して共通のメッセージを暗号化し、暗号化したメッセージをコンピュータ121へ送る。コンピュータ121は、同様のことを行なう。次に、コンピュータ101は、コンピュータ121によって暗号化されたメッセージを受信して、メッセージをその秘密鍵によって再暗号化する。コンピュータ121も同様のことを行なう。さて、コンピュータ101と121とは、それらにだけが知る共通のセッション鍵を共有しており、即ち、最初のメッセージはそれらの秘密鍵の双方によって暗号化されたこととなる。その後、コンピュータ101と121は、ネットワーク141を介して、それらに共通のセッション鍵で暗号化したメッセージを交換することが出来ることとなる。

【００２３】

＜タイムチャートとデータフロー：図３＞

図３は、コンピュータ101により実行される暗号化演算のタイムチャートである。この暗号化演算（処理）では、出力メッセージが、デジタル署名、公開鍵の復号、又は鍵交換のために入出力インターフェース111へ提供されるよう、出力メッセージを演算するため、入力メッセージを、秘密鍵をテーブルサイズに従って処理する。

【００２４】

秘密暗号化演算の入力211には、入力メッセージ216、秘密鍵212、そして、テーブルサイズ214が含まれる。入力メッセージは、例えばキーボード110のような周辺機器によって生成してもよく、又は、入出力インターフェース111を介してネットワークから受信したメッセージとしてもよい。或いは、メッセージは、メモリ102内に保存したデータであってもよい。この秘密鍵は、不揮発性メモリであるEEPROM 106又はROM 107内に保存されている。最後に、テーブルサイズは、例えばキーボード110などの周辺機器によって選択することが出来る。これに替えて、テーブルサイズは、利用可能なRAM 103に従って、CPU 114によって動的に選択することもでき、或いは、不揮発性メモリ（ROM又はEEPROM）から取得されるデータサイズに従ってもよく、その場合には、そのデータサイズは固定のシステムパラメータである。

【００２５】

最初に、秘密鍵212から、選択データp 221及びq 222がモジュール202の内部で生成される。この選択データの生成は、CPU 114やコプロセッサ115を含む処理部によって行なわれる。場合により、選択データは、以前のセッション又はカード情報から不揮発性メモリ106又は107内に保存することも出来る。将来の使用のため、選択データは、より速いアクセスを可能にするよう、RAM 103内に保存される。

【００２６】

次に、選択データをテーブルサイズを利用して、幅w 231及びインデックステーブルB[1],…,B[2^w] 232を含め、システムパラメータがモジュール203内で選択される。このモジュールは、選択データ221 pとテーブルサイズ214の双方を要求し、CPU 114によってシステムパラメータを計算して、それらをRAM 103内に保存する。場合によっては、システムパラメータは、以前のセッション又は初期化の段階から、不揮発性メモリによって取り出すことも出来るが、しかしながら、より速いアクセスを可能にするためにRAM 103へ転送される。

【００２７】

その後、再コード化（recoding）モジュール204内において、この時点でRAM内に格納されている幅231、インデックステーブル232、及び、選択データq 222を利用して、秘密鍵212の表現が変更される。この再コード化モジュールは、秘密鍵を検索し、そして、テーブルサイズ、選択データ、及び、システムパラメータに従って、CPU 114により、秘密鍵のための新たな表現を算出する。最後に、再コード化された秘密鍵がRAM 103内に保存される。先にも述べたと同様に、以前のセッションやカードの初期化から秘密鍵を取り出すことも可能である。

【００２８】

最後に、入力メッセージ212、幅231、インデックステーブル232、及び、再コード化された秘密鍵241から、メッセージ暗号モジュール205内のCPU 114とコプロセッサ115によって、出力メッセージ251が算出される。この出力メッセージ251は、入出力インターフェース111へ提供され、そしてネットワーク142へ送られる。

【００２９】

＜算術モジュール：図４＞

算術モジュールは、４つのカテゴリーに分類することができる。即ち、短演算モジュール310、長剰余演算モジュール320、乱数生成器303、そして、ハッシュ関数302、である。

【００３０】

短算術モジュールには、短演算モジュール310と長剰余演算モジュール320とを含んでいる。短演算とは、小さなオペランドによる演算であり、例えば、32ビットまでのサイズのものである。本発明の好適な実施の形態では、比較モジュール311、ビット操作（bit manipulation）モジュール312、算術モジュール313を含むこれらの命令は、CPU 114の命令セットにおいてサポートされている。

【００３１】

比較モジュール311は、例えば、「０」又は「１」など、RAM 103又はEEPROM 106からの変数、或いは、ＲROM 107又はEEPROM 106からの定数である２個のデータを比較することが可能である。比較の範囲（種類）としては、「同一」=、「相違」<>、「より小」<、「より大」>、「より小又は同一」<=、「より大又は同一」>=である。ビット操作モジュール312のビットワイズ（bit wise）演算は、変数又は定数である演算のビットを操作し、以下の演算を含んでいる：ビット方式の（ビットワイズ：bitwise）XOR、ビットワイズ（bitwise）AND、ビットワイズ（bitwise）OR、ビットワイズ（bitwise）の否定（negation）であるNOT、左シフト（left shift）<<、右シフト（left shift）>>、そして、循環シフト（cyclic shift）。また、本好適な実施の形態では、CPU 114の命令のセットには、例えば、３２ビットの短変数や定数の加算「+」、減算「-」、乗算「*」、除算「/」など、算術演算（arithmetic operations）313を含んでいる。また、CPU 114により、増加（インクリメント：increment）x=x+1、減少（デクレメント：decrement）x=x-1がサポートされている。最後に、例えば、「０」、「１」又は「0x6ed9eba1」などの短定数は、ROM 107又はEEPROM 106内において入手可能である。ここでは、表記（notation）「0x…」は１６進の表記を指す。

【００３２】

長剰余演算モジュール320は、RSAの場合、例えば、1024ビットの、より長いオペランドを操作する。本発明の好適な実施の形態では、剰余乗算モジュール323は、任意のA、B及びNに対してNを法とするA*B（A*B modulo N）を計算するコプロセッサ115に実装されてい

る。剰余加算321及び剰余減算322は、乗算に比較して計算コストが大きなことから、それらは、ROM 107内に保存され、CPU 114によって実行されるプログラムとして実装されている。最後に、素整数Ｐに対する剰余逆元算「$A^{-1} \bmod P = A^{P-2} \bmod P$」は、冪算（exponentiation）「$A^{P-2} \bmod P$」における乗算のためのコプロセッサ１１５にサポートされ、CPU 114により実行されるプログラムとして実装されている。

【００３３】

長剰余乗算の役割は、デジタル署名、特に、公開鍵暗号システムのRSAアルゴリズムにとっては、非常に重要である。より詳細には、RSA署名が、以下のようにして生成され、認証される。例えば、メモリ内に整数として保存されているメッセージのビット列（bit sequence）を解釈することによって、入力メッセージＭが整数としてコード化される。加えて、鍵の一対が生成され、秘密の整数ｄからなるこの鍵の一対が不揮発性のメモリ内に保存され、そして、これ以降、秘密鍵と呼ばれ、二つの秘密の素整数PとQ、そして、一つの公開冪指数ｅと一つの公開された法（public modulus）Nは、以下の式を満足する：
N=P*Q and e*d = 1 mod (P-1)*(Q-1)

次に、メッセージMの署名は、例えば、A*B mod Nなどの剰余乗算により演算された秘密鍵を利用した冪算$C=M^d \bmod N$の結果である。メッセージMとその署名Cとを受信し、$M'=C^e \bmod N$を計算することによってメッセージの真正を確認することができ、ここではM'=M.の場合、このメッセージは本物であると認証される。

【００３４】

最大で512ビットのシード（seed）を入力として任意長のランダムなデータを返す乱数生成器303は、ハッシュ関数302と共に、比較演算311、ビットワイズ演算312、算術演算313、そして、短定数314を含め、短演算モジュールを利用する。本発明の好適な実施の形態では、乱数生成器は、FIPSによって規格化され、かつ、ハッシュ関数SHA-1 302に基づくDSA乱数生成器に基づいており、これらの両者は非特許文献１に記載されている。乱数生成器303及びハッシュ関数302は、ROM 107内に保存されたプログラムとして搭載されており、CPU 114により実行される。しかしながら、本発明では、選択データの生成方法を特定の生成方法に限定するものではない。他の決定方法としては、例えば、ハッシュ関数やブロック暗号を純粋に使用すること、又は、異なる乱数生成器を使用することも可能である。

【００３５】

＜選択データ生成モジュール：図５＞

選択データ生成モジュールの目的は、２個のデータp及びqを演算することであり、ここでは、ｐは160ビット、そして、ｑは秘密鍵ｄと同じビット長を有している。本発明の実施の形態では、選択データは、所謂、FIPSにより規格化され、非特許文献１に記載されたDSA乱数生成器である乱数生成器により、専用的に生成される。しかしながら、これでは、乱数生成器の典型的な使用に比較し、一つの主要な違いがある。即ち、シード（seed）ｓは乱数ではなく、実際には、秘密鍵ｄから生成されるということである。即ち、同じ秘密鍵は、常に、同一の選択データを生成しており、選択データが攻撃者（attacker）によって知られたとしても、秘密鍵を回復することは不可能である。

【００３６】

より詳細には、秘密鍵ｄの最初の最下位512ビットを抽出することにより、ステップ502において、シード（seed）ｓが演算される。次に、160ビットの数ｔが、例えばEEPROM 106などの不揮発性メモリから読み出される。本発明の実施の形態では、ｔは$t = t_0||t_1||t_2||t_3||t_4$として定義されており、32ビットの数$t_0, \ldots, t_4$が連鎖されている。本実施の形態では、$t_0$=0x98BADCFE, t_1=0x10325476, t_2=0xC3D2E1F0, t_3=0x67452301, t_4=0xEFCDAB89と定義するが、しかしながら、これに替る実施の形態では、ｔとして任意の値を使用することが出来る。その後、ステップ504では、FIPSで規格化され、非特許文献１に記載されたハッシュ関数SHA-1に基づいて一方向性関数（one-way function）Gを利用することにより、pがG(t,s)として演算され、そして、sはSHA-1により処理された入力メッセージである。SHA-1はROM 107内にプログラムとして搭載することができ、CPU 114により、或

いは回路的に、コプロセッサ115の一部によって実行可能である。その後、シード（seed）ｓは、ステップ504において、s=(1+s+p) mod 2^{512}としてアップデートされる。換言すれば、その何れかが長整数の加算をサポートしている場合には、加算1+s+pがCPU 114又はコプロセッサによって演算され、そして、最初の最下位512ビットがその結果から抽出される。

【００３７】

選択データの第２片 $(q_{160}\cdots q_0)_2$ の最初の160ビットを得るため、同じ演算がステップ505で繰り返される。次に、ステップ511と512では、選択データqのnビット以上を得るために、同じ演算が繰り返される。最後に、ステップ521において、qの最初のn個の最下位ビットが抽出され、そして、将来の使用のためにp と qがRAM内に保存される。

【００３８】

本発明の範囲は、特定の一方向性関数（one-way function）Gを利用することや、特定の一方向性関数（one-way function）を搭載することに限定されるものではない。代替的な実施の形態として、例えば、異なるハッシュ関数であるRIPEMD-160を基に、DES、 triple DES又はAESなどの異なる一方向性関数Gを利用することも可能である。加えて、一方向性関数は、同様に、CPU 114によって実行されるプログラムとして、又は、回路的に、或いは、その他の如何なる形態によっても実装することが出来る。最後に、乱数生成器に替えて、例えば、乱数生成器によらずに、ハッシュ関数やブロック暗号によって、秘密鍵から選択データを派生するという異なるアプローチを行なうことも可能である。

【００３９】

＜システムパラメータ生成モジュール：図６＞

秘密鍵213、テーブルサイズ214、そして選択データから、システムパラメータ生成モジュールが上記の幅wとインデックステーブルB[1], B[2], …, B[2^w] を演算する。このモジュールには、B[1],…,B[2^{w-1}] を演算する最下位インデックステーブル生成610と、B[2^{w-1}+1],…,B[2^w] を演算する上位インデックステーブル生成620の、２つのステージを有している。モジュール620においては、選択データpは選択指数（indices）をランダムに選択するために使用され、即ち、当該目的のためには、ランダムな指数がpから抽出されなければならず、そして、それからpは新たな値にアップデートされなければならない。

【００４０】

最初に、ステップ602では、上記の幅wが以下のように演算され：

$w = CEIL(\log_2(k))$,

ここで、$\log_2$は基本２のアルゴリズム関数（base 2 logarithm function）を示し、そして、CEIL($\log_2$(k)) は$\log_2$(k)以上の最小の整数である。この幅wとしての可能な値は、本実施の形態では、EEPROM 106又はROM 107内の、幾つかの小さな値のための、テーブルサイズkのルックアップテーブル内に保存されている。これに替えて、このステップをEEPROM 106又はROM 107内に保存したプログラムとして搭載してCPU 114によって実行し、又は、コプロセッサ115内で回路的に実行することも可能である。

【００４１】

その後、インデックステーブルB[1], B[2], …, B[2^w] が演算される。テーブルB[i]は整数である。より詳細には、1<=i<=2^{w-1}に対しては、B[i] は常に非零（non-zero）であり、そして、2^{w-1}+1<=i<=2^wに対しては、B[i] は非零（non-zero）又は零（zero）である。トータルとして、インデックステーブルには、丁度、k個の非零（non-zero）のエントリ（加入：entries）が存在しており、即ち、kはテーブルサイズであり、2^{w-1}個のエントリがインデックステーブルの下半分に存在し、そして、ランダムに選択されたk-2^{w-1}個のエントリがインデックステーブルの上半分に存在する。

【００４２】

ステップ611、612及び613では、インデックステーブルの下半分が以下のようにして初期化される：

B[1]=1, B[2]=2, …, B[2^{w-1}]=2^{w-1}.

これらのステップは単純なメモリ割り当て（memory assignments）であり、整数B[1] ～

$B[2^{w-1}]$ がRAM 103内に保存される。ステップ621では、インデックステーブルの上半分の部分が零で初期化される。この時点では、インデックステーブル内において、2^{w-1}個の非零（non-zero）のエントリが利用可能であるが、$k-2^{w-1}$個の非零エントリはなお欠落しており、上半分のインデックステーブルにおいて、以下のステップ622、623、624、625及び626のようにしてランダムに選択される。

【００４３】

ステップ623では、w-1-ビット（w-1-bit）の値Pが、$P=p \bmod 2^{w-1}$として抽出される。実際には、CPU 114が、Pを算出するために、160ビットデータ（160-bit data）pの最下位ｗ－１ビット（w-1 lower bits）を抽出する。その後、ステップ６２４では、pは、コプロセッサ115を利用して、$p=3*p \bmod 2^{160}$としてアップデートされる。それから、ランダム指数（random index）$2^{w-1}+1 <= P+2^{w-1}+1 <= 2^w$ がインデックステーブルの上半分のために取得される。もしも$B[P+2^{w-1}+1] <> 0$であれば、インデックスは過去において既に非零（non-zero）エントリとして選択されており、そして、Pのために新たな値が得られるまで、ステップ623と624とが繰り返される。かかる後、新たなインデックスが選択データから抽出され、ステップ626において、$B[P+2^{w-1}+1]$ がインデックス値iによりアップデートされ、そして、i がインクリメントされる。ステップ622～626は、インデックステーブルが丁度、k個の非零（non-zero）エントリを含むまで繰り返される。

【００４４】

最後に、上記の幅wとインデックステーブルB[1], B[2]～ $B[2^w]$が、将来の使用のためにRAM内に保存される。本発明の範囲は、ステップ623で選択データpからランダムな指標（インデックス：indices）を抽出し、又は、ステップ624でアップデートするための特定の方法に限定されるものではなく、それに替わる実施の形態では、インデックステーブルBを異なる仕方で構成することも可能であろう。一つの可能性として、pを$p/2^{w-1}$ 又は SHA-1(p) でアップデートすることも可能である。

【００４５】

＜再コード化モジュール：図７＞

再コード化モジュールは、ステップ701では、nビットの秘密鍵（n-bit secret key）d 213、選択データq 221、そして、システムパラメータw, B[1], B[2],…,$B[2^w]$ を入力とし、そして、再コード化された秘密鍵 $(v_{n-1}\cdots v_0)$ をステップ763において出力する。以下においては、秘密鍵の最上位ビットd_{n-1}は1であるとする。再コード化モジュールは秘密鍵の新たな表現をデジット毎に演算する。より詳細には、モジュール720は、dから抽出されたw ビット（w bits）でxを演算し、他方、モジュール730は、dから抽出されたw-1 ビット（w-1 bits）でyを演算し、これらの両モジュールは、同時に実行される。それから、モジュール740は、システムパラメータと選択データqとに依存しており、x又はyを選択する。最後に、ステップ751において、選択され再コード化されたデジットがRAM 103内に保存され、そして、秘密鍵 d の次のデジットによってアルゴリズムが進行される。

【００４６】

次に、デジット演算モジュール720及び730について詳細に述べる。モジュール720が秘密鍵 d からw ビット（w bits）をスキャンすること、他方では、モジュール730がw-1ビット（w-1 bits）だけをスキャンすることを除いて、これらは全く同一である。換言すれば、d のi番目のビット（i-th bit）から開始して、ステップ721において、値$(d_{i+w-1}\cdots d_i)_2$-cがx に対して関連付けられ、他方、cは、ステップ702において零へのキャリーの初期化が行なわれ、そして、ステップ731において、値$(d_{i+w-1}\cdots d_i)_2$-c がy対して関連付けられる。ステップ722と732において、ＣＰＵは x と yとが負又は零であるか否かをチェックする。もしも、x が負又は零であれば、ＣＰＵ１１４により2^wがxに加算され、そして、ステップ732において、一時キャリー（temporary carry）c_x が1に設定される。もし、そうでなければ、ステップ724において、値零が一時キャリーc_x に割り当てられ、即ち、定数１（constant 1）がc_xに対応してRAM領域へ移動される。もしもyが負又は零であれば、CPU 114により2^wがyに加算され、ステップ733において、一時キャリー（temporary carry）c_x が1に設定される。もしそうでなければ、値零が一時キャリーc_y に割り当てら

れる。

【００４７】

デジット選択モジュール740は、以下の処理を行なう。まず、ステップ741において、CPUは、x が2^{w-1}よりも小さいか否かをチェックする。xが2^{w-1}よりも小さい場合には、x は確率（probability）$k/2^{w-1}$ 1によって、再コード化されたデジットとして選択され、y は確率$2-k/2^{w-1}$によって選択される。より詳細には、ステップ742においてQ=q mod 2^{w-1}を演算することにより、CPU 114によって選択データq からwビットが抽出され、そして、q が$q-Q/2^{w-1}$によりアップデートされ、即ち、CPUが選択データqに対して右(w-1)ビット移動（right (w-1)-bit shift）を実行する。もしもQ が$k-2^{w-1}$よりも大きい場合には、ステップ746においてx が選択され、他方、ステップ744においてy が選択される。Qはw-1のランダムなビット（w-1 random bits）から構成されることから、Q は2^{w-1} 個の異なる値をとり得、そして、Q が$k-2^{w-1}$よりも大きくなり、それ故、x が選択される確率は、実際、$k/2^{w-1}-1$となる。さて、$x>2^{w-1}$の場合には、２つの可能性がある：即ち、インデックステーブルの上半分のエントリ（entry）B[x] が零の場合、或いは、零でない（non-zero）場合である。もしも零でない場合には、ステップ746において、x が選択され、零である場合には、ステップ744においてy が選択される。ステップ744では、RAM内のy の値をu に対応するRAM領域に移動することにより、y を再コード化されたデジットuに割り当て、一時キャリー c_y の値を、次の繰り返しのためのキャリー c へ割り当て、そして、選択された幅 r を w-1に設定する。同様に、ステップ746では、x をu へ割り当て、c_x をc へ割り当て、そして、w をr へ割り当る。

【００４８】

さて、デジット u、次のキャリー c、そして幅 rは、モジュール740によって選択され、ステップ703では、デジット u の値を i 番目の再コード化されたデジットv_iとして蓄積し、そして、ステップ751において、隣接する (r-1) 番目までのデジットv_{i+1}, v_{i+2}～v_{i+r-1} に０を挿入する。最後に、処理がステップ711から繰り返され、そして、秘密鍵 d の次のビットのスキャンが i+r 番目のビット（bit i+r）から開始される。n-w番目のビット（bit n-w）までの全てのビットをスキャンすると、ステップ761と762では、再コード化により最後のデジットを出力する。d_{n-1}=1であることから、最後のキャリーは常に中和され（neutralized）ており、そして、アルゴリズムが、正又は零のデジットによって、正しく終了する。最後に、ステップ761において、再コード化アルゴリズムが再コード化された秘密鍵（$v_{n-1}\cdots v_0$）を出力し、そして、将来の利用のためにそれらをRAM 103内に保存する。

【００４９】

本発明の範囲は、特定の再コード化アルゴリズムに限定されるものではない。例えば、代替的な実施の形態では、二つ以上の再コード化されたデジットxとyを同時に演算してもよく、そして、なお選択データpによって再コード化されたデジットのどれかを決定することもできる。

【００５０】

＜メッセージ暗号化モジュール：図８＞

メッセージ暗号化モジュール205は、再コード化された秘密鍵($v_{n-1}\cdots v_0$) 241、システムパラメータ231、メッセージM、及び、法（modulus）N 212を入力とし、出力メッセージC 251を出力する。実際、出力メッセージは$C=M^d$ mod Nであり、即ち、法Nを法とする（modulo the modulus N）、秘密鍵 dを冪指数とするメッセージMの冪算である。しかしながら、計算のために秘密鍵 dを利用する代わりに、演算Cのために再コード化された秘密鍵($v_{n-1}\cdots v_0$) 241が利用される。本好適な実施の形態では、メッセージ暗号化モジュールは、ROM 107又はEEPROM 106内に保存されてCPUによって実行されるプログラムとして実装されており、しかしながら、本発明の他の可能な実施の形態では、専用の演算ユニットとしてのハードウェアであってもよい。メッセージ暗号化モジュール205は、前処理モジュール（pre-computation module）810と演算モジュールとの、二つの主モジュールを含んでいる。

【００５１】
前処理モジュール（pre-computation module）810は、前処理された値をテーブル t[1], t[2],…, t[2^w] のエントリ（entry）に割り当てる。ステップ811、812及び813では、前処理されたテーブルの下半分のエントリが評価（evaluated）され、RAM 103内に保存される。ステップ811では、メッセージの値が、テーブルのエントリt[1] に対応したRAM領域内に移動される。次に、ステップ813では、t[2]がコプロセッサ115によりt[1]*M mod Nとして演算され、t[2^{w-1}] まで演算される。なお、本発明では、乗算には長オペランドが含まれており、CPU 114で演算すると多くの時間がかかることから、コプロセッサ115が乗算を演算するために使用されている。

【００５２】
ステップ821～825では、前処理されたテーブルの上半分のエントリが評価され、RAM 103内に保存される。テーブルサイズは kであり、その下半分において既に2^{w-1}個のエントリを有していることから、このフェーズ（phase）では、k-2^{w-1}個だけのエントリが演算される。より詳細には、その対応するインデックスエントリB[i] が零でない場合においてのみ、新たなエントリが演算される。例えば、或るインデックス2^{w-1}+1<=i<=k に対しては、コプロセッサにより、テーブルエントリt[B[i]] がt[i-2^{w-1}]*t[2^{w-1}] mod Nとして演算される。ここでも、再び、長オペランドを有しており、CPU 114で演算すると多くの時間がかかる乗算を加速するために、コプロセッサ１１５が利用されている。テーブルt[1],…,t[k] におけるk個のエントリの演算が終了すると、演算モジュール830が作動される。

【００５３】
演算モジュール830は、前処理されたテーブルt[1],…,t[k]、インデックステーブルB[1], …,B[2^w] 、及び、再コード化されたデジット（$v_{n-1}\cdots v_0$）を使用する。モジュールは再コード化されたデジットを左から右へスキャンし、即ち、インデックスi=n-1 から始まって0までスキャンする。累算（accumulator）Cが定数１によって初期化され、ステップ831において、Cに対応するRAM領域に移動される。各繰り返しでは、ステップ832において累算が二乗され、コプロセッサ115がC*C mod Nを演算して、その結果をCに対応するRAM領域内に保存する。加えて、ステップ834において、CPUは、i番目に再コード化されたデジットv_iが非零（non-zero）であるか否かをチェックする。もしもそうであれば、ステップ８３５において、累算は前処理されたエントリt[B[v_i]] によって乗算され、その場合、コプロセッサ115がC*C mod Nを演算して、その結果をRAM 103内に保存する。この処理手順は、全ての再コード化されたデジットがスキャンされるまで繰り返され、その後、ステップ841では、累算がメッセージ暗号化モジュールの出力として送出される。

【実施例１】

【００５４】
例えば、秘密のべき指数d = 65 = $(1000001)_2$とテーブルサイズk=3によるRSA表現M^d mod Nについて考える。最初に、選択データ (p,q) が、s=65 及びt=0x98badcfe10325476 c3d2e1f067452301efcdab89によって計算される。

G(t,s) = 0xf66a29cc54a9b116ee864c6f4db496d59279bb69 = p,

それ故、シード（seed）は以下のようになる。

s=s+p+1 mod 2^{160}=0xf66a29cc54a9b116ee864c6f4db496d59279bbab

その後、qが以下のように計算される。

G(t,s)=0xd3020de628c235fb19d961513937233dba489915 and q = $(0010101)_2$.

次に、システムパラメータが生成される。k=3であることから、上記の幅wはw=CEIL($\log_2$(k))=2である。さて、インデックステーブルは、B[1]=1, B[2]=2, B[3]=0, B[4]=0のように用意することができる。上半分のインデックステーブルでは、１つのインデックスは、pに従って、３と４との間で、ランダムに選択され、即ち、p mod 2 = 1であることから、B[4]=3と設定する。換言すれば、メッセージ暗号化ステージにおける前処理されたテーブルは、m^1、 m^2 及び m^4から構成される。その後、秘密のべき指数d = 65が再コード化される。

【００５５】
第１ステップ(i = 0):
x = $(d_1d_0)_2$ = 1及びy = $(d_0)_2$ = 1。x <= 2であることから、両再コード化は可能である。それ故、選択ビットq_0=1を使用し、そして、y: v_0 = y=1を選択する。
【００５６】
第２ステップ(i = 1):
x = $(d_2d_1)_2$ = 0 及び y = $(d_2)_2$ = 0であり;しかしながら、零の値は禁止されており、４をxに加え、そして、次のデジットのためのキャリー c_x = 1を保持する。同様に、２をyに加えて、carry c_y = 1を保持する。即ち、x=4 及び y=2である。インデックステーブル(B[4]=3<>0)では、４が非零のインデックスとして選択されたことから、xが再コード化されたデジットとして選択される。それ故、v_1 = 4, v_2=0 and c = c_x = 1となる。
【００５７】
第３ステップ(i = 3):
x = $(d_4d_3)_2$ c = -1 及び y = $(d_3)_2$ c = -1であり;しかしながら、零の値は禁止されており、４をxに加え、そして、次のデジットのためのキャリー carry c_x = 1を保持する。同様に、２をyに加えて、carry c_y = 1を保持する。即ち、x=3 及び y=1である。B[3]=0であることから、y は再コード化されたデジットとして選択される。それ故、v_3 = 1, そして c = c_y = 1となる。
【００５８】
第４ステップ(i = 4):
x = $(d_5d_4)_2$ c = -1 及び y = $(d_4)_2$ c = -1であり; しかしながら、零の値は禁止されており、４をxに加え、そして、次のデジットのためのキャリー carry c_x = 1を保持する。同様に、２をyに加えて、carry c_y = 1を保持する。再度、x=3 及び y=1である。B[3]=0であることから、y は再コード化されたデジットとして選択される。それ故、v_4 = 1, そして c = c_y = 1となる。
【００５９】
第５ステップ(i = 5):
x = $(d_6d_5)_2$ c = 1 及び c_x = 0である。また、 y = $(d_4)_2$ c = -1である。Yは負であることから、y = y + 2 = 1 であり、そして、キャリー c_y = 1を保持する。x <= 2であることから、両パターンが許容可能である。決定するためにq_1が利用され、即ち、q_1 = 0であることから、再コード化xが選択される。それ故、v_5 = 1、 v_6=0 、そして、c = c_x = 0となる。
【００６０】
最後の再コード化として、d = 65 = $(1000001)_2$ = $(0111041)_{k=3}$ が得られる。その後、前処理されたテーブルが用意される。T[1] = M、及びT[2] = t[1]*M = M^2 mod Nである。前処理されたテーブルの最後のエントリはT[B[4]]=t[3] = T[2]*T[2] = M^4 mod Nである。最後に、冪算が計算される。
Step i=6: C=1
Step i=5: C = C * T[B[v_5]] = 1*T[1] = M mod N
Step i=4: C = C^2 * T[B[v_4]] =M^2 * T[1] = M^3 mod N
Step i=3: C = C^2 * T[B[v_3]] =M^6 * T[1] = M^7 mod N
Step i=2: C = C^2 = M^{14} mod N
Step i=1: C = C^2 * T[B[v_1]] = M^{28}*T[3] = M^{32} mod N
Step i=0: C = C^2 * T[B[v_0]] = M^{64}*T[1] = M^{65}, 出力 C=M^{65} mod N.
【００６１】
＜拡張＞
本発明の範囲は、選択データ生成ステップと、再コード化ステップと、暗号化ステップとを結合するよう、容易に変更することができ、オンザフライ演算（on-the-fly computations）を実現する後者の実施の形態に、限定されるものではない。再コード化ステップは右から左へ行なわれるが、本発明の範囲は、かかる例に限定されるものではなく、即ち

、再コード化は、異なる戦略（strategy）で、異なる端子（反対方向の端子であり、上記の例に対しては、左から右）の場合でも行なうこともでき、より一般的には、秘密の値の表現をランダム化することに基づく如何なる再コード化であってもよい。小さい変形例としては、後者の実施の形態を、例えば、Diffie-Hellman の鍵交換（Diffie-Hellman key exchange）、ElGamal 暗号（ElGamal encryption ）又は DSAなど、他の暗号化ツールにおいて適用することも出来る。加えて、選択データ生成モジュールは、本発明の単に一実施の可能性に過ぎない。なお、その他の可能性としては、しかしながら、シード（seed）としての秘密データを使用した、異なる乱数生成器を利用すること、異なるハッシュ関数を使用すること、ブロック暗号を使用すること、全てに対して選択データを一回だけ演算して保存することに限定される。最後に、上記に開示した実施の形態では、再コード化のアルゴリズムは、可能性x と yの間で一つの再コード化されたデジットを選択するが、本発明の範囲はかかる場合に限定されることはない。再コード化のアルゴリズムは、２つだけでなく、任意の数の可能な選択肢の中から、一つの再コード化されたデジットを選択してもよい。

【実施例２】

【００６２】

ECCのための秘密鍵の安全な多重使用

本発明の第１の実施の形態では、RSA冪算は、乱数生成器のおかげで、同じ秘密鍵によって安全に演算することができた。第２の実施の形態では、ハッシュ関数によって生成した選択データを利用することにより、如何にして楕円曲線演算を安全に演算するかについて示す。

【００６３】

＜タイムチャートとデータフロー：図９＞

第２の実施の形態では、システムパラメータの生成ステップ及びメッセージの暗号化ステップにおいて、選択データはオンザフライ（on-the-fly）演算される。加えて、システムパラメータの生成ステップでは、前処理されたテーブルが、同時に、インデックステーブルとして演算され、そして、メッセージの暗号化ステップにおいては、再コード化ステップが埋め込まれる。要約すれば、異なるステージ間での一時的データの保存を避けるため、幾つかのステップが統合されている。

【００６４】

第１のステップは、システムパラメータの生成903であり、秘密鍵913とテーブルサイズ914から、上記の幅w、インデックステーブルB[1],…,B[2^w-1]、及び、前処理されたテーブルT[1], …, T[k] を算出する。インデックステーブルにとって必要な選択データpは、このステージにおいて高速に生成される。

【００６５】

第２と最終のステップはメッセージの暗号化905である。メッセージの暗号化ステップでは、選択データp 921、幅931、インデックステーブル932、そして、前処理されたテーブル933を入力として、出力メッセージ951を算出する。秘密データの再コード化は、メッセージの暗号化と交互に行なわれ、ステップ905では、選択データqが高速で算出される。

【００６６】

＜算術モジュール：図１０＞

本発明では、算術モジュールは、上記第１の実施の形態と同様であり、即ち、短演算モジュール310がCPU 114の命令セットによりサポートされており、それに対し、長剰余演算モジュールである、少なくとも剰余乗算モジュール323は、コプロセッサ115から恩恵を受けることができる。ハッシュ関数モジュールSHA-1 302も、また、利用することができる。それに加えて、本実施の形態では、楕円演算モジュールを備えている。

【００６７】

楕円演算モジュール1004は、楕円加算（point addition）1041、楕円二倍算（doubling）1042、そして、楕円逆元（negation）1043の、３つのタイプの演算と、そして、一の特別な一定値である無限遠点1044とを含んでいる。かかる楕円演算は、２つのｎビット座標

P=(x,y) を含む楕円点を操作する。ビット長nは、典型的には、160 又は 256ビットであり、そして、楕円演算は、剰余乗算の演算のためのコプロセッサによるサポートから恩恵を受けることができる。本実施の形態では、楕円演算モジュール1004は、コプロセッサ115により直接的にサポートされており、しかしながら、代替的な実施の形態では、ROM 107 内に保存され、ことによると、剰余乗算のためのコプロセッサによるサポート、又は、他の如何なる均等な方法を伴って、CPU 114によって実行されるプログラムとして実装することもできる。

【００６８】

本第２の実施の形態では、楕円加算（elliptic point addition）ECADD 1031は、以下の一連の演算を実行するコプロセッサ115によりサポートされている。

compute k = (y2-y1)*(x2-x1)$^{-1}$ mod m
compute x3 = k*k x1 x2 mod m
compute y3 = k*(x1 x3) y1 mod m
return R=ECADD(P,Q)=(x3,y3)

なお、ECADDでは、ステップ1、2 及び3においては、コプロセッサ115により実行される剰余乗算323を、ステップ1におおては、剰余逆元算324を、そして、ステップ2 及び3においては、剰余加算321と減算322を使用する。

【００６９】

楕円二重化（elliptic point doubling）ECDBL 1032 は、以下の一連の演算を実行するコプロセッサ115によってサポートされる。

P=(x1,y1)、曲線パラメータをa、そして、法（modulus）をmとして：
演算 k = (3*x1*x1 + a) * (2y1)$^{-1}$ mod m
演算x3 = k*k 2*x1 mod m
演算y3 = k*(x1 x3) y1 mod m
リターン R=ECDBL(P)=(x3,y3)

ステップ1、2及び3における剰余乗算と共に、ステップ1、2及び3における剰余加算及び減算、そして、ステップ1における逆元算も、コプロセッサ115により演算される。

【００７０】

楕円逆元（Point negation）1033は、単純な剰余減算322であり、コプロセッサ115によって次のように演算される：点（point）をP=(x1,y1)、法（modulus）をmとして、負の点（negative point）は、P=(x1,-y1 mod mである。最後に、初期化のために、しばしば「無限遠点（point at infinity）」inf 1034と呼ばれる常数が必要とされる。この無限遠点は、整数の場合における零と同様の役割を果たし：ECADD(P,inf)=ECADD(inf,P)=P 及び ECDBL(inf)=infである。単純化のために、無限遠点は、零座標inf=(0,0)による点としてメモリ102内に保存することができる。

【００７１】

本第２の実施の形態では、楕円演算はコプロセッサ115によって十分にサポートされているが、本発明の範囲はかかる場合に限定されるものではなく、代わりに、楕円演算をプログラムとしてROM 107内に保存し、CPU 114によって、ことによっては、例えば剰余乗算などの演算はコプロセッサ115の助けによって、実行することも可能である

【００７２】

＜システムパラメータの生成：図１１＞

システムパラメータの生成ステップの入力には、入力メッセージM 912、秘密鍵d 913、及びテーブルサイズk 914.を含み、そして、その出力は、幅w 931、選択データp 921、インデックステーブルB[1], B[3], B[5],.., B[2^w-1] 932、及び前処理されたテーブルT[1],…, T[k] 933である。

【００７３】

ステップ1102では、幅w 931がCEIL($\log_2$(2k))として演算される。実際には、w はROM 107内に保存されたプログラムから、CPU 114によって演算することもでき、或いは、EEPROM 106 又はROM 107内に保存されたルックアップテーブルであるメモリから単純に割り当

てることもできる。その後、ステップ1103において、選択データpがSHA-1(d)として演算され、ここでSHA-1は上記非特許文献１に記載された標準的な一方向性のハッシュ関数である。

【００７４】

ステップ1111～1113では、下半分のインデックステーブルB[1], B[3], B[5],…, B[2^{w-1}-1]と前処理されたテーブルT[1], …, T[2^{w-2}] とが演算され、RAM 103内に保存される。より詳細には、B[1]=1, B[3]=2, B[5]=3, B[7]=4 そしてB[2^{w-1}-1]=2^{w-2}までが、及び、T[1]=M, T[2]=3M, T[3]=5M, T[4]=7MそしてT[2^{w-2}]= (2^{w-1}-1)*Mまでが演算されて保存される。ここで、2M=ECDBL(M)はステップ1111で演算されてRAM 103内に保存され、即ち、T[i+1] = (2i+1)*M = ECADD(T[i-1],2M) = (2i+1)*M+2Mがステップ1113において正しく演算される。ここで、処理ECDBLとECADDとは、それぞれ、楕円二倍算と楕円加算を指す。

【００７５】

次に、上半分のインデックステーブルB[2^{w-1}+1],…,B[2^{w}-1] 及び前処理されたテーブルT[2^{w-2}+1], …, T[k] が、ステップ1021～1026において演算される。ステップ1021では、上部インデックステーブルB[2^{w-1}+1],…,B[2^{w}-1] が零により初期化され、そして、楕円点2^{w-1}M が以下のように演算され、

ECADD(T[2^{w-2}],M)= (2^{w-1}-1)*M+M,

RAM 103内に保存される。この初期化を行うことにより、上部インデックステーブルを演算することができる。最初に、ステップ1123において、2^{w-1}+1と2^{w}-1の間の任意の奇数インデックスが、選択データpを利用して選択される。より詳細には、ランダムインデックスは、P=p mod 2^{w-2}を利用して、2^{w-1}+2P+1であり、そして、pが標準的一方向性のハッシュ関数SHA-1を利用してwith p=SHA-1(d) にアップデートされる。もしもインデックスエントリB[2^{w-1}+2P+1] が非零（non-zero）、即ち、エントリが既に選択されていれば、ステップ1123において、Pのための新たな値、即ちpがp=SHA-1(p)によって再度アップデートされる。ここで、P=p mod 2^{w-2}を計算する演算は、CPU 114によるp のw-2の最下位ビットの抽出から成り、そしてSHA-1(p)はCPU 114あるいはコプロセッサ115によって簡単に演算される。結局は、B[2^{w-1}+2P+1]=0 などの値Pが見出され、インデックスiが1により増加され、インデックスエントリB[2^{w-1}+2P+1] が、RAM 103内に保存されたiの値をインデックステーブルに対応するRAM領域に移動することにより、iに設定され、そして、前処理されたエントリT[i] が以下のように演算される。

ECADD(2^{w-1}M,T[2P+1])=2^{w-1}M+(2P+1)*M

ここで、2^{w-1}Mはステップ1021において既に演算されており、T[2P+1]もまた既に前処理された下部テーブルから入手可能であり、それ故、両者はRAM 103内に存在しており、CPU 114及びコプロセッサ115によって処理することができる。ステップ1122～1126は、丁度、k個の前処理されたエントリが計算されるまで繰り返される。最後に、幅w 931、選択データp 921、インデックステーブルB[1], B[3], B[5],…, B[2^{w}-1] 932、前処理されたテーブルT[1], T[2],…, T[k] 933 が、将来の使用のためにRAM 103内に保存される。

【００７６】

本発明の範囲は、特定の一方向性機能の利用や実行に限定されることなく、その代替案としては、例えば、RIPEMD-160などのハッシュ関数、あるいは、DES、トリプル（triple）DES又はAESなどのブロック暗号を利用することもできる。加えて、本発明の範囲は、インデックステーブルＢのランダムな指標（random indices）の演算する特定の方法に限定されることはない。例えば、ステップ1123において、p=p/2^{w-1}のような異なる方法によって選択データpをアップデートすることも可能であろう。

【００７７】

＜メッセージの暗号化：図１２＞

秘密鍵d=(d_{n-1}…$d_1$1)$_2$ 913からは、選択データp 921、幅w 931、インデックステーブルB[1], B[3],…, B[2^{w}-1] 932、前処理されたエントリT[1], T[2],…, T[k] 933とメッセージ912から、メッセージ暗号化モジュールは楕円加算（elliptic additions）d により、C=d*M = M+M+…+Mを演算する。加えて、演算期間に高速で行われたランダム化されたd

のおかげで、演算d*Mが安全な方法で演算される。なお、ここでdは奇数であり、それ以外の場合には、このdは常にd+1に設定されて奇数となる。

【００７８】

ステップ1202では、ビットカウンターiがn-1によって、選択データqがSHA-1(p)によって初期化される。加えて、累算（accumulator）C、そしてX と Yがn-列である楕円点C=(X,Y)とが、無限遠点（point at infinity）である値infによって初期化される。この無限遠点は、いかなる楕円点、M、ECADD(inf,M)=M、更にはECDBL(inf)=infに対しても、整数や加算に対する零と同様の役割を果たす。ステップ1230では、二つの再コード化されたデジットx と yとが同時に、以下のように演算される。

$x = (d_i \cdots d_{i-w+1}1)_2 - 2^w$ and $y = (d_i \cdots d_{i-w+2}1)_2 - 2^{w-1}$.

即ち、x と yとは奇数であり、$-2^w<x<2^w$ and $-2^{w-1}<y<2^{w-1}$である。

【００７９】

モジュール1240では、xの値、インデックステーブル、及び選択データqに従って、再コード化されたデジットuが、x と yから選択される。より詳細には、$x<2^{w-1}$であれば、xが確率$k/2^{w-2}-1$で、そして y が確率 $2-k/2^{w-2}$で選択される。このランダムな選択は、選択データqのおかげで行われる。即ち、qのw-2の最下位ビットは、ステップ1242においては、$Q=q \bmod 2^{w-2}$のように、そして、qはq=(q-Q)/2^{w-2}、即ち、(w-2)ビットの右移動（(w-2)-bit right shift）でアップデートされる。そして、Qが$k-2^{w-2}$と比較され、$Q>k-2^{w-2}$の場合には、ステップ1244において、y と w-1とが再コード化されたビットと幅として選択され、その他の場合には、ステップ1246において、xとwとが選択される。x>2w-1の場合には、二つの可能性がある。即ち、B[x]<>0の場合であり、xがシステムパラメータ生成モジュール９０３内においてインデックスエントリとして選択されたことを意味し、又は、B[x]=0の場合である。もしもB[x]0であれば、y と w-1 とが選択され、その他の場合には、x と wとが選択される。

【００８０】

ステップ1251～1254では、楕円演算が行われる。ステップ1252と1253は、rが選択ステップ1240によってw-1又はwとなりえるr楕円二倍算ECDBL（r elliptic point doublings ECDBL）を演算するためのである。即ち、累算（accumulator）Cは、全ての繰り返しが実行されたとき、2^rCとなる。その後、楕円加算ECADD（elliptic point addition ECADD）が演算され、uが正の場合には、ステップ1255において、前処理されたエントリT[B[-u]] がC に加算され、そして、uが負の場合には、ステップ1255-Tにおいて、-T[B[-u]] がC に加算される。いずれの場合でも、ビットインデックスiはrだけ増加される。ここで、T[B[-u]]=(x,y) の場合には、T[B[-u]]=(x, -y)となる。

【００８１】

かかる処理手順は、ビットインデックスiがwよりも小さくなるまで繰り返される。iがwよりも小さくなると、最後のi+1ビットが、d_iからd_0=1まで処理される。ステップ1261、1262及び1263では、楕円二倍算が、2^iCにアップデートされた累算（accumulator）に対し、i回実行される。最後の非零のデジット、即ち、ステップ1264では、$u=(d_i \cdots d_1 1)_2 \ 2^i$が演算される。u<0の場合には、ステップ1267において、前処理されたエントリT[B[-u]]が累算（accumulator）Cに加算され、他の場合には、ステップ1266において、T[B[u]] がQ に加算される。最後に、ステップ1268において、累算（accumulator）がモジュールの出力として、即ち、暗号化演算の結果C=dMとして転送される。

【００８２】

＜拡張＞

本発明の範囲は、上記第１の実施の形態に適合するように、容易に変更することができる後者の実施の形態、即ち、選択データと共に、個別に、かつ、非高速で実行される再コード化によるものに限定されるものではない。高速な演算を可能にするため、再コード化のステップは左から右へ行われるが、本発明の範囲は、この例に限定されるものではない。即ち、再コード化は、異なる戦略（strategy）で、異なる端子（反対方向の端子であり、上記の例に対しては、左から右）の場合でも行なうこともでき、そして、より一般的に

は、秘密値の表現（the representation of the secret value）をランダム化することに基づく如何なる再コード化であってもよい。なお、小さい変形例として、後者の実施の形態を、例えば、Diffie-Hellman の鍵交換（Diffie-Hellman key exchange）、ElGamal 暗号（ElGamal encryption ）又は ECDSAなどの他の暗号化ツールにおいて採用することも出来る。加えて、選択データ生成モジュールは、本発明の単に一の可能な実施の形態に過ぎない。しかしながら、その他の可能な例としては、シード（seed）としての秘密データを使用した、異なる乱数生成器を利用すること、異なるハッシュ関数を利用すること、ブロック暗号を利用すること、全てに対して選択データを一回だけ演算して保存することなどに限定される。最後に、上記に開示した実施の形態では、再コード化のアルゴリズムが、二つの可能な再コード化されたデジットから、一の再コード化ジットを選択するが、本発明の範囲はこの場合に限定されることはない。再コード化のアルゴリズムは、任意のzに対して可能なzの選択肢の中から一つを選択してもよい。

【図面の簡単な説明】

【００８３】

【図１】本発明になる暗号化処理方法を実行するための一般的な設定を示すためのシステム構成図である。

【図２】本発明になる暗号化処理方法を実行するための、実施例１になるコンピュータシステムとネットワークのハードウェア構成図である。

【図３】上記システムにおけるコンピュータにより実行される暗号化演算のタイムチャートとデータフローを示す図である。

【図４】上記システムにおける算術モジュールの詳細構成を示す図である。

【図５】上記システムにおける選択データ生成モジュールの動作を示すフローチャート図である。

【図６】上記システムにおけるシステムパラメータ生成モジュールの動作を示すフローチャート図である。

【図７】上記システムにおける再コード化モジュールの動作を示すフローチャート図である。

【図８】上記システムにおけるメッセージ暗号化モジュールの動作を示すフローチャート図である。

【図９】本発明の実施例２になるシステムにおける暗号化演算の動作を示すタイムチャートとデータフロー図である。

【図１０】上記実施例２のシステムにおける算術モジュールの詳細構成を示す図である。

【図１１】上記実施例２のシステムにおけるシステムパラメータの生成動作を説明するフローチャート図である。

【図１２】上記実施例２のシステムにおけるメッセージの暗号化を説明するフローチャート図である。

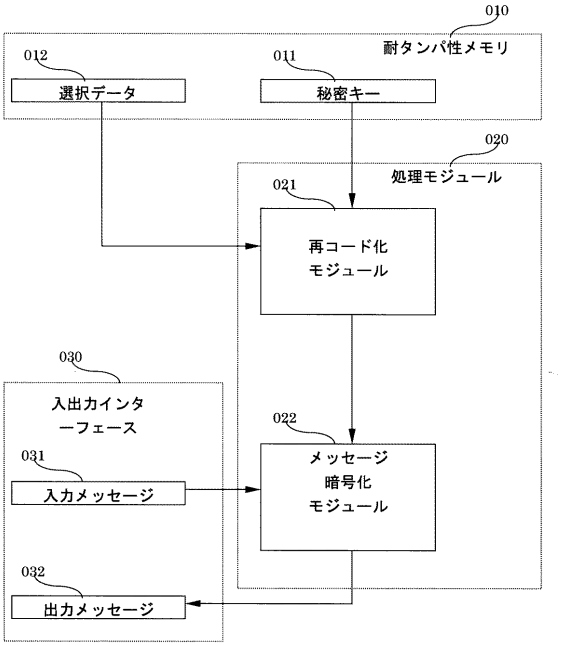
【符号の説明】

【００８４】

010…耐タンパ性メモリ
011…秘密鍵
012…選択データ
020…処理モジュール
021…再コード化モジュール
022…暗号化モジュール

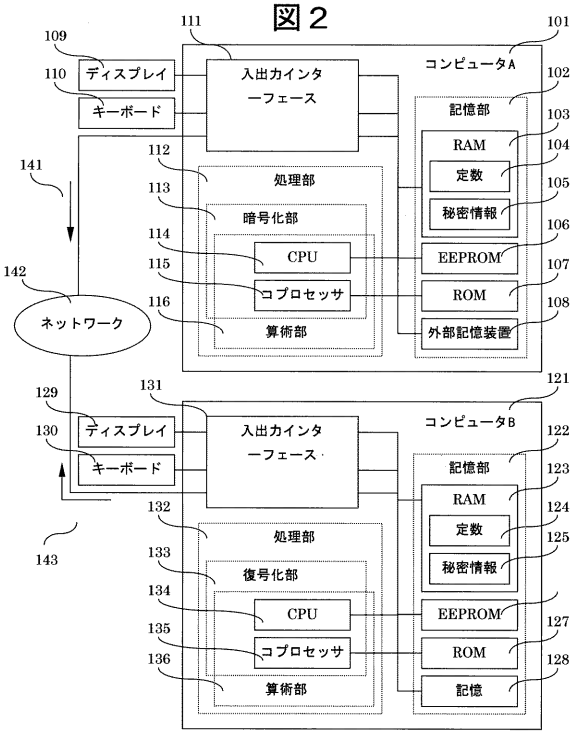
【図１】

図１



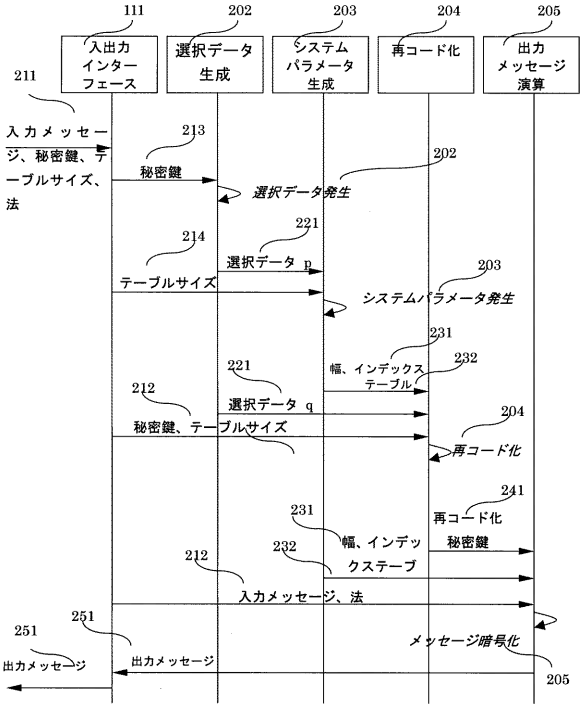
【図２】

図２



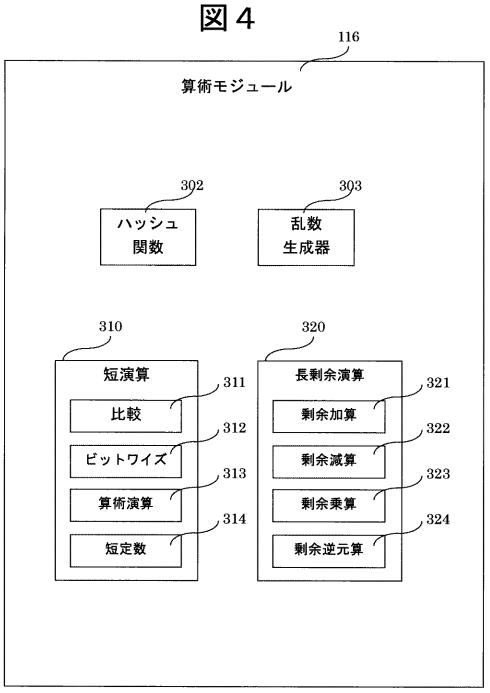
【図３】

図３



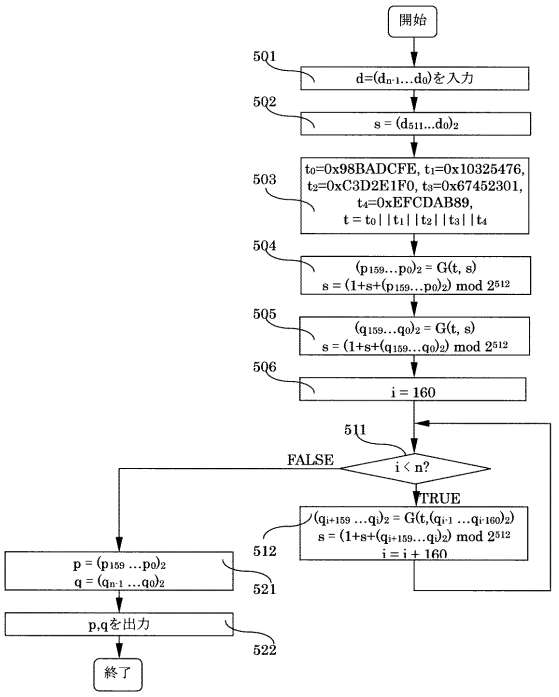
【図４】

図４



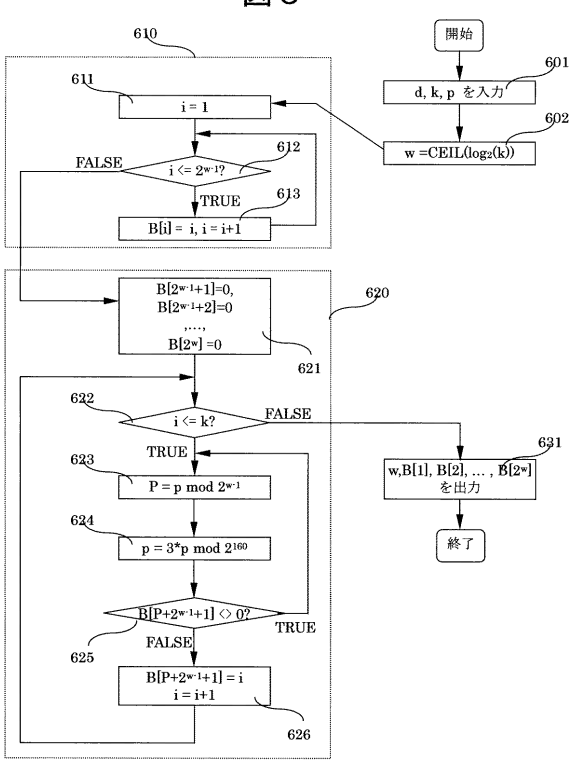
【図５】

図５



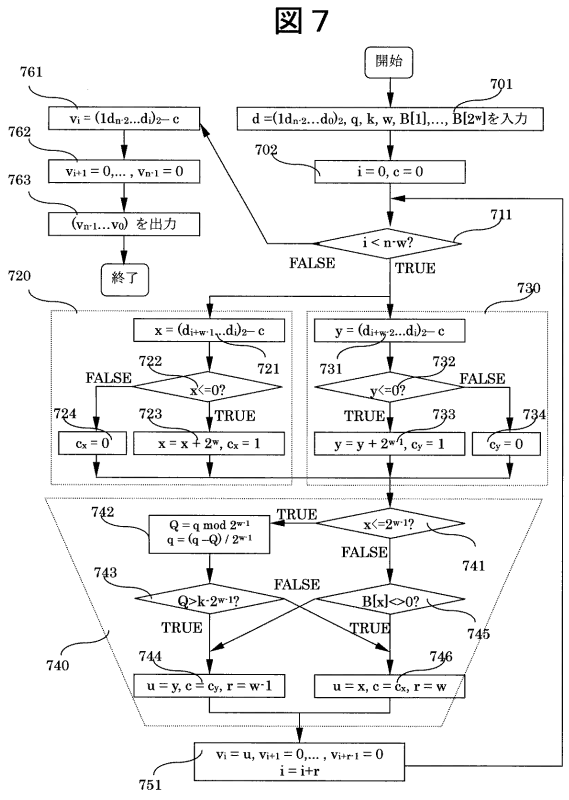
【図６】

図６



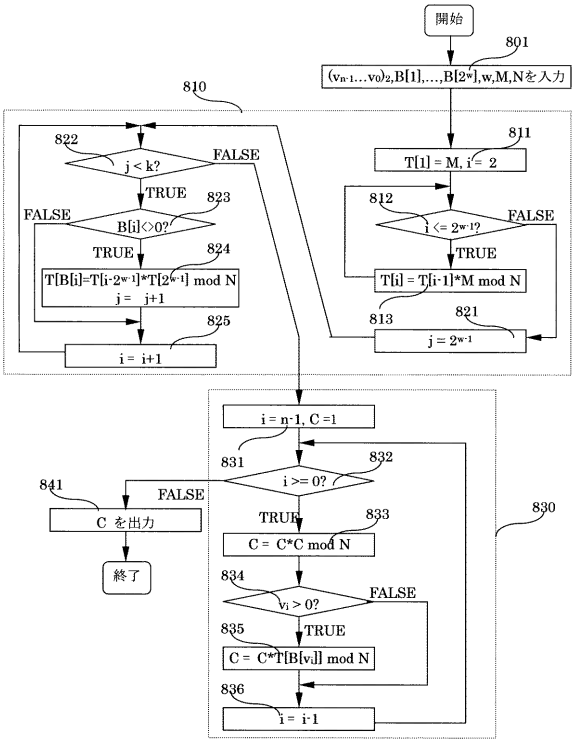
【図７】

図７



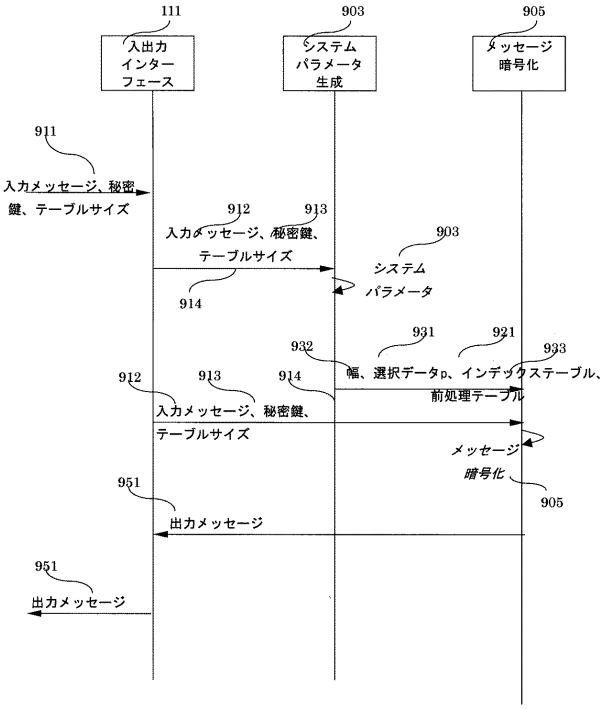
【図８】

図８



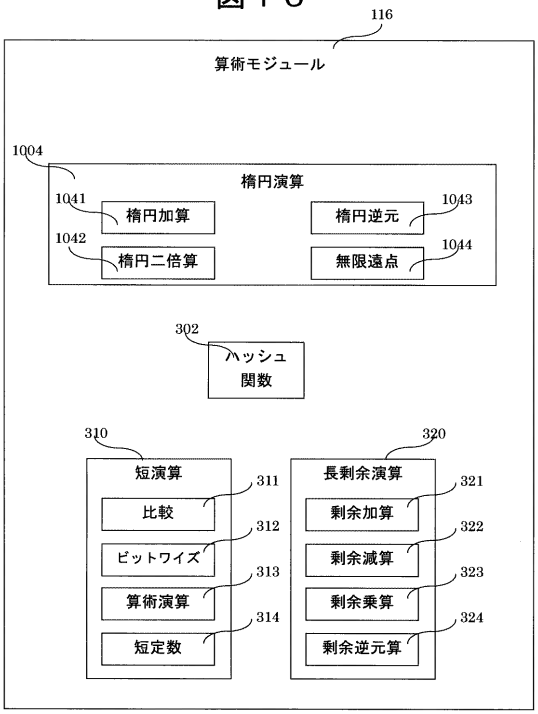
【図９】

図９



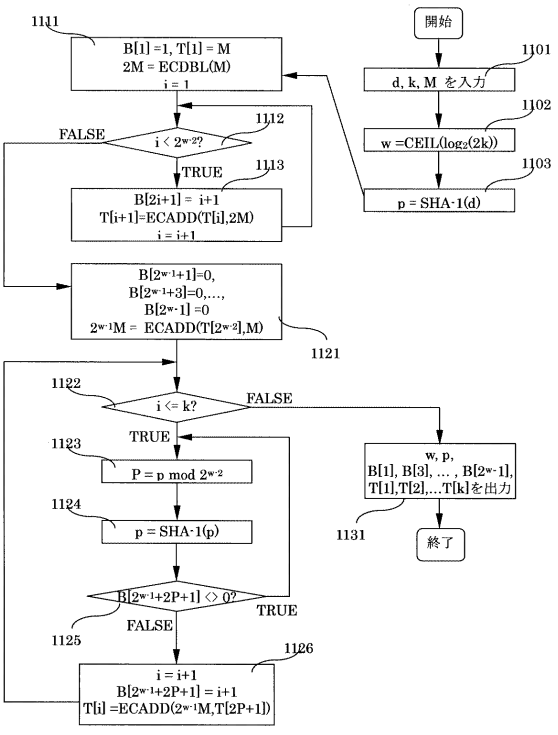
【図１０】

図１０



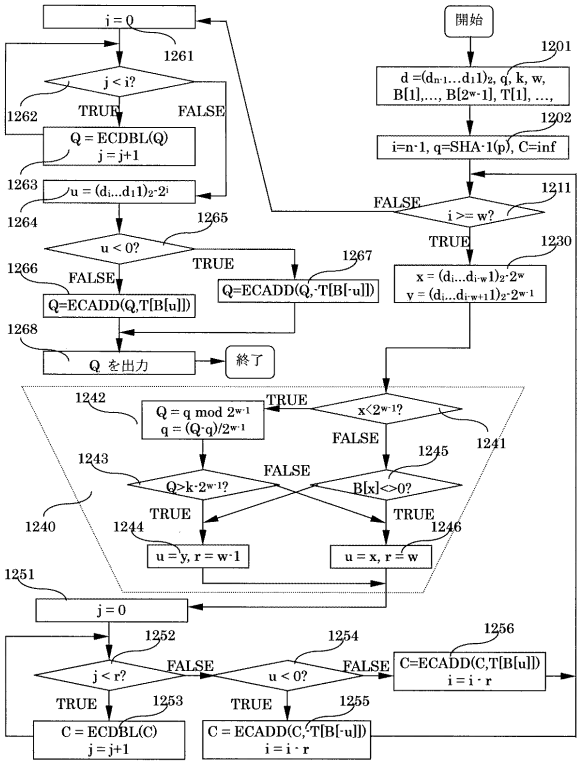
【図１１】

図１１



【図１２】

図１２



フロントページの続き

(72)発明者　吉野　雅之
神奈川県川崎市麻生区王禅寺１０９９番地　株式会社日立製作所システム開発研究所内

Ｆターム(参考) 5J104 AA16 AA32 AA43 AA46 EA04 EA15 EA16 EA17 JA03 JA21
NA02 NA27 NA37 NA39 NA42