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

特許第4632950号
(P4632950)

(45)発行日　平成23年2月16日(2011.2.16)　(24)登録日　平成22年11月26日(2010.11.26)

(51)Int.Cl.			F I		
H04L	9/10	(2006.01)	H04L	9/00	621A
G09C	1/00	(2006.01)	G09C	1/00	650A

請求項の数 10　(全 23 頁)

(21)出願番号	特願2005-504397 (P2005-504397)
(86)(22)出願日	平成15年7月22日 (2003.7.22)
(86)国際出願番号	PCT/JP2003/009284
(87)国際公開番号	WO2005/008955
(87)国際公開日	平成17年1月27日 (2005.1.27)
審査請求日	平成17年9月6日 (2005.9.6)

(73)特許権者　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100105360
弁理士　川上　光治
(74)代理人　100062993
弁理士　田中　浩
(72)発明者　伊藤　孝一
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　伊豆　哲也
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

最終頁に続く

(54)【発明の名称】個人鍵を用いた耐タンパ暗号処理

(57)【特許請求の範囲】

【請求項１】

個人鍵を用いて楕円曲線暗号処理を行う暗号装置であって、

外部からアクセスできない内部で生成された乱数に基づいて楕円曲線上の点Ｒをランダムに生成し、前記楕円曲線上の或る入力点Ａの値を入力し、その際、前記点Ｒに応じて初期化される変数に対して前記楕円曲線上の或る入力点Ａの値に応じた初期化を行わず、前記入力点Ａに応じて初期化される変数に対して前記点Ｒの値に応じた初期化を行わない、初期化手段と、

前記点Ｒを初期点V_0にセットし、前記入力点Ａをセットし、前記楕円曲線暗号処理のための前記個人鍵を表す或るスカラー値ｄのビット・シーケンスの全てのビット値に対して、前記初期点V_0と前記入力点Ａのスカラー倍との和$V_1 = V_0 + dA$の演算を実行する演算手段と、

前記演算によって得られた和V_1から前記初期点V_0を減算する演算$V = V_1 - V_0$を実行するランダム化解除手段と、

前記ランダム化解除手段によって得られた点Ｖを出力として供給する手段と、

を具える、暗号装置。

【請求項２】

前記初期化手段は、前記楕円曲線上の点Ｒの座標Ｘを生成し、前記座標Ｘに従って前記点Ｒの座標Ｙを生成して、前記生成された座標（Ｘ，Ｙ）を前記点Ｒとして生成するものである、請求項１に記載の暗号装置。

【請求項３】
前記初期化手段は、乱数ｓを生成し、ベースポイントＧのｓ倍を求め、前記求められたｓＧを前記点Ｒとして生成するものである、請求項１に記載の暗号装置。
【請求項４】
前記演算手段は、前記ｓＧを求めるために、乱数ｓを前記スカラー値ｄにセットし、前記入力点Ａを前記ベースポイントＧにセットし、無限遠点Ｏを前記点Ｒにセットすることによって、前記演算$Ｖ_{1}＝Ｖ_{0}＋ｄＡ$を用いてｓＧ＝Ｏ＋ｓＧを求めるものである、請求項３に記載の暗号装置。
【請求項５】
最初にランダムに発生された前記楕円曲線上の点$Ｒ_{0}$が前記点Ｒにセットされ、その後前記点Ｒとして関数$Ｒ_{n}＝ＥＣＤＢＬ（Ｒ_{n-1}）$で表される前記楕円曲線上の点$Ｒ_{n}$が用いられ、ここで$Ｒ_{n}$はｎ回目の楕円曲線暗号処理を表すものであり、ＥＣＤＢＬ（Ａ）は楕円曲線上の点Ａの２倍算を表すものである、請求項１に記載の暗号装置。
【請求項６】
最初にランダムに発生された前記楕円曲線上の初期点$Ｒ_{0}$が前記点Ｒにセットされ、その後前記点Ｒとして前記楕円曲線上の点$Ｒ_{n}$が用いられ、$Ｒ_{n}$はｎ回目の楕円曲線暗号処理を表すものであり、前記楕円曲線上の前回の点$Ｒ_{n-1}$が投影座標の点（$Ｒ_{X}$，$Ｒ_{Y}$，$Ｒ_{Z}$）で表されるとき、前記楕円曲線上の今回の点$Ｒ_{n}$が投影座標の点（$ｔ×Ｒ_{X}$，$ｔ×Ｒ_{Y}$，$ｔ×Ｒ_{Z}$）で表され、ｔは定数を表すものである、請求項１に記載の暗号装置。
【請求項７】
最初にランダムに発生された前記楕円曲線上の初期点$Ｒ_{0}$が前記点Ｒにセットされ、その後前記点Ｒとして前記楕円曲線上の点$Ｒ_{n}$が用いられ、$Ｒ_{n}$はｎ回目の楕円曲線暗号処理を表すものであり、前記楕円曲線上の前回の点$Ｒ_{n-1}$がヤコビアン座標の点（$Ｒ_{X}$，$Ｒ_{Y}$，$Ｒ_{Z}$）で表されるとき、前記楕円曲線上の今回の点$Ｒ_{n}$がヤコビアン座標の点（$ｔ^{2}×Ｒ_{X}$，$ｔ^{3}×Ｒ_{Y}$，$ｔ×Ｒ_{Z}$）で表され、ｔは定数を表すものである、請求項１に記載の暗号装置。
【請求項８】
個人鍵を用いてべき乗剰余暗号処理を行う暗号装置であって、
外部からアクセスできない内部で生成された乱数に基づいて整数ｒをランダムに生成し、或る入力値ａを入力し、その際、前記整数ｒに応じて初期化される変数に対して前記入力値ａに応じた初期化を行わず、前記入力値ａに応じて初期化される変数に対して前記整数ｒに応じた初期化を行わない、初期化手段と、
前記整数ｒを初期値$Ｖ_{0}$にセットし、前記入力値ａをセットし、前記べき乗剰余暗号処理のための前記個人鍵を表す或る値ｄのビット・シーケンスの全てのビット値に対して、前記初期値$Ｖ_{0}$と前記入力値ａに対するべき乗剰余演算$Ｖ_{1}＝Ｖ_{0}ａ^{d}（ｍｏｄ　ｎ）＝ｒ×ａ^{d}（ｍｏｄ　ｎ）$の演算を実行する演算手段と、
前記演算によって得られた値$Ｖ_{1}$とｒ（ｍｏｄ　ｎ）の逆元$ｒ^{-1}$（ｍｏｄ　ｎ）との乗算剰余演算$Ｖ＝Ｖ_{1}×ｒ^{-1}$（ｍｏｄ　ｎ）を実行するランダム化解除手段と、
前記ランダム化解除手段によって得られた値Ｖを出力として供給する手段と、
を具える、暗号装置。
【請求項９】
情報処理装置に個人鍵を用いた楕円曲線暗号処理を実行させるためのプログラムであって、
外部からアクセスできない内部で生成された乱数に基づいて楕円曲線上の点Ｒをランダムに生成し、前記楕円曲線上の或る入力点Ａの値を入力するステップと、
前記点Ｒに応じて初期化される変数に対して前記楕円曲線上の或る入力点Ａの値に応じた初期化を行わず、前記入力点Ａに応じて初期化される変数に対して前記点Ｒの値に応じた初期化を行わずに、前記点Ｒを初期点$Ｖ_{0}$にセットし、前記入力点Ａをセットするステップと、
前記楕円曲線暗号処理のための前記個人鍵を表す或るスカラー値ｄのビット・シーケン

スの全てのビット値に対して、前記初期点V_0と前記入力点Aのスカラー倍との和$V_1 = V_0 + dA$の演算を実行するステップと、
前記演算によって得られた和V_1から前記初期点V_0を減算する演算$V = V_1 - V_0$を実行するステップと、
前記減算によって得られた点Vを出力として供給するステップと、
を前記情報処理装置に実行させるためのプログラム。
【請求項１０】
情報処理装置に個人鍵を用いたべき乗剰余暗号処理を実行させるためのプログラムであって、
外部からアクセスできない内部で生成された乱数に基づいて整数ｒをランダムに生成し、或る入力値ａを入力するステップと、
前記整数ｒに応じて初期化される変数に対して前記入力値ａに応じた初期化を行わず、前記入力値ａに応じて初期化される変数に対して前記整数ｒに応じた初期化を行わずに、前記整数ｒを初期値V_0にセットし、前記入力値ａをセットするステップと、
前記べき乗剰余暗号処理のための前記個人鍵を表す或る値ｄのビット・シーケンスの全てのビット値に対して、前記初期値V_0と前記入力値ａに対するべき乗剰余演算$V_1 = V_0 a^d$（ｍｏｄ　ｎ）$= r \times a^d$（ｍｏｄ　ｎ）の演算を実行するステップと、
前記演算によって得られた値V_1とｒ（ｍｏｄ　ｎ）の逆元r^{-1}（ｍｏｄ　ｎ）との乗算剰余演算$V = V_1 \times r^{-1}$（ｍｏｄ　ｎ）を実行するステップと、
前記逆元との乗算剰余演算によって得られた値Vを出力として供給するステップと、
を前記情報処理装置に実行させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、暗号処理の分野に関し、特に、ＲＳＡおよび楕円曲線暗号のような公開鍵暗号のためのプロセッサにおけるＳＰＡ、ＤＰＡおよびＲＰＡのような電力解析攻撃を防止する耐タンパ（tamper-proof）暗号化／復号に関する。
【背景技術】
【０００２】
例えば電子決済および日本の住民基本台帳ネットワークのようなネットワークを利用したサービスが普及しつつある。これらのサービスには情報セキュリティのために暗号が用いられる。
【０００３】
上述のサービスにおけるユーザ側のデバイスとして、ユーザの秘密情報を格納したＩＣチップを含むスマートカードが広く普及すると予想される。スマートカードは、暗号、ディジタル署名および認証の機能を有し、その秘密情報を鍵として用いる。スマートカードでは、秘密情報はＩＣチップ・メモリに格納されるので、第三者による不正なアクセスに対する安全性または耐タンパ性が磁気カードに比べて非常に高い。
【０００４】
図１は、スマートカードのような暗号デバイスにおける秘密鍵を用いた暗号化／復号の構成の例を示している。図１において、暗号デバイスは、その内部の暗号化／復号ユニットにおいて周知の形態で個人鍵を用いて入力平文／暗号文メッセージを処理して出力暗号文／平文メッセージを供給する。暗号方式には一般的に公開鍵暗号方式と共通鍵暗号方式が含まれる。
【０００５】
公開鍵暗号方式は、暗号化と復号に異なる鍵（キー）を用いる。典型的には、公開鍵を用いて平文（plaintext）が暗号化され、個人または秘密鍵を用いて暗号文（ciphertext）が復号され、それによって暗号文が安全に送信できる。または、個人鍵を用いて平文が暗号化され、公開鍵を用いて暗号文が復号され、それによって平文を暗号化したユーザが識別される。公開鍵暗号は、送受信者間で秘密鍵を共有する必要がないが、計算量が共通

鍵暗号と比べて非常に大きい。公開鍵暗号にはＲＳＡおよび楕円曲線暗号が含まれる。

【０００６】

ＲＳＡ暗号は、$z = a^x$（ｍｏｄ　ｎ）で表される冪（べき）乗剰余演算（modular exponentiation）に基づくものである。ＲＳＡ暗号に基づく暗号機能は、暗号化、復号、署名生成および署名確認に関係する。復号および署名生成ではユーザの秘密情報を個人鍵として用いる。べき乗剰余演算は、$v = a^d$（ｍｏｄ　ｎ）を満たすは出力データｖを生成する。ここで、ａは入力データを表し、ｍｏｄ　ｎは剰余の法を表し、ｄは個人鍵を表す。

【０００７】

楕円曲線暗号は、点のスカラー倍算に基づくものである。点のスカラー倍算は、スカラー値ｄ、および楕円曲線上の点Ａに対して、Ｖ＝ｄＡを満たす点Ｖを求める。楕円曲線暗号に基づく暗号機能は、ＥＣＥＳ暗号化／復号、ＥＣＤＳＡ署名生成／署名確認およびＥＣＤＨ秘密鍵共有に関係する。ＥＣＥＳ復号、ＥＣＤＳＡ署名生成およびＥＣＤＨ秘密鍵共有のための処理は、ユーザの秘密情報を個人鍵として用いる。例えば、ＥＣＤＨ秘密鍵共有処理は、秘密共有値を表す点Ｖを計算するためにＶ＝ｄＡで表される点のスカラー倍算を行う。ここで、Ａは秘密鍵を共有する公開鍵の点を表し、ｄは個人鍵のスカラー値を表す。

【０００８】

ＲＳＡ暗号における乗算剰余$c = a \times b$（ｍｏｄ　ｎ）、２乗剰余$c = a^2$およびべき乗剰余$c = a^x$（ｍｏｄ　ｎ）は、楕円曲線暗号における点の加算Ｃ＝Ａ＋Ｂ、点の２倍算Ｃ＝２Ａおよび点のスカラー倍算Ｃ＝ｘＡにそれぞれ対応する。

【０００９】

暗号解読（分析）またはタンパ（tamper）は、暗号文のような入手可能な情報から個人鍵を含めた秘密情報を推定する。暗号解読の１つである電力解析攻撃が、１９９８年にPaul Kocher（ポール　コーチャ）によって考案された。

【００１０】

電力解析攻撃は、文献、P. Kocher, J. Jaffe and B. Jun "Differential Power Anaysis", Crypto'99, LNCS 1666, pp.388-397, Springer-Verlag, 1999に記載されている。

【００１１】

この電力解析攻撃は、スマートカードのような暗号デバイスに含まれる暗号プロセッサに相異なる入力データを与え、例えば図１に示されているようにオシロスコープ等を用いてその処理中における時間に対する消費電力の変化を測定し、統計的に充分な数の消費電力変化の曲線を収集しそれを解析し、それによって暗号プロセッサ内部の鍵情報を推定する。これは、共通鍵暗号と公開鍵暗号の双方に適用できる。

【００１２】

電力解析攻撃には、単純電力解析（Simple Power Analysis）（以下、ＳＰＡという）および電力差分攻撃（Differential Power Analysis）（以下、ＤＰＡという）が含まれる。ＳＰＡは暗号プロセッサにおける１つの消費電力変化曲線の特徴から秘密鍵または個人鍵の推定を行う。ＤＰＡは相異なる多数の消費電力変化曲線の差分（以下、電力差分曲線という）を用いて解析することによって秘密鍵の推定を行う。一般的にはＤＰＡの方が強力である。

【００１３】

電力解析攻撃を防止する必要性が各種国際標準に記載されている。例えば、セキュリティの国際標準ＩＳＯ１５４０８によるスマートカード用プロテクション・プロファイル（ＰＰ）では電力解析攻撃への対策が必須である。また、暗号モジュールに関する米国標準ＦＩＰＳ１４０－２には、現在は電力解析攻撃対策の必要性に関するコメントがあるだけであるが、将来的に電力解析攻撃対策が必須になるであろう。

【００１４】

ＲＳＡと楕円曲線暗号に対する電力解析攻撃ＳＰＡおよびＤＰＡを防止するための様々な対策が開発された。しかし、２００３年に、改善された電力解析攻撃（Refined Power

Analysis）（ＲＰＡ）が、L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Crypto-systems", PKC 2003, LNCS 2567, Springer-Verlag, 2003によって発表された。この攻撃は、楕円曲線暗号に対する電力解析攻撃を防止する公開鍵暗号の中の一部の暗号を攻撃するのに有効である。

【００１５】

次に、楕円曲線暗号に用いられる用語を説明する。詳細は文献IEEE P1363/D13 (Draft Version 13, November 12, 1999) main document, Standard Specifications for Public Key Cryptography, http：//grouper.ieee.org/groups/1363/P1363/draft.htmlの標準を参照されたい。

【００１６】

次の式で表される変数ｘおよびｙの関数で表される曲線が楕円曲線と呼ばれる。

楕円曲線（素体）：　$y^2 = x^3 + ax + b$　（ｍｏｄ　ｐ）

ここで、ｐは素数を表し、ａおよびｂは楕円曲線パラメータ（０≦ａ，ｂ＜ｐ）を表す。

【００１７】

楕円曲線（２べき）：　$y^2 + xy = x^3 + ax^2 + b$　（ｍｏｄ　ｆ（ｘ））

ここで、ｆはＧＦ（2^m）の多項式を表し、ａおよびｂは楕円曲線パラメータ（ａ，ｂ，⊆ＧＦ（2^m））を表す。

【００１８】

楕円曲線は、主に素体（prime field）と２べき（binary field）からなる。楕円曲線は楕円曲線パラメータａおよびｂによって一意的に決定される。

【００１９】

楕円曲線上の点（elliptic point）は、楕円曲線で表される式を満たす座標（ｘ，ｙ）であり、素体の場合０≦ｘ，ｙ＜ｐを満たす整数（ｘ，ｙ）の集合であり、２べきの場合は（ｘ，ｙ）⊆ＧＦ（2^m）を満たす要素（ｘ，ｙ）の集合である。

【００２０】

無限遠点（infinite point）は、Ｏと表記され、任意の点Ａに対しＡ＋Ｏ＝Ｏ＋Ａ＝Ａを満たす楕円曲線上の特殊な点である。ここで、＋は点の加算を表す。

【００２１】

ベースポイント（base point）は、楕円曲線上の１点であり、Ｇと表記される。ベースポイントは楕円曲線暗号のユーザ間で共有され、公開鍵／個人鍵のペアの生成、および楕円曲線暗号を用いたその他の処理に用いられる。

【００２２】

２次元ベクトル（ｘ，ｙ）を用いた点の表現は、アファイン座標（Affine coordinates）と呼ばれる。（ｘ，ｙ）＝（Ｘ／Ｚ，Ｙ／Ｚ）を満たす３次元ベクトル（Ｘ，Ｙ，Ｚ）を用いた点の表現は、投影座標（projective coordinates）と呼ばれる。（ｘ，ｙ）＝（X/Z^2，Y/Z^3）を満たす３次元ベクトル（Ｘ，Ｙ，Ｚ）を用いた点の表現は、ヤコビアン座標（Jacobian coordinates）と呼ばれる。３次元ベクトル表現の使用によって、点のスカラー倍算における除算の回数を大幅に減らし、全体の演算を高速化できる。

【００２３】

点ＡおよびＢに対する演算Ａ＋Ｂは、点の加算（elliptic point addition）と呼ばれる。Ａ＋Ｂ＝Ｂ＋Ａが成立する。点ＡおよびＢに対する演算Ａ－Ｂは、点の減算（elliptic point subtraction）と呼ばれる。

【００２４】

点の２倍算（elliptic point doubling）は、楕円曲線上の点ＡからＣ＝２Ａで表される楕円曲線上の点Ｃが定義される。この演算２Ａを点の２倍算と呼ぶ。

【００２５】

点のスカラー倍算（elliptic scalar multiplication）は、楕円曲線上の点Ａおよびスカラー値ｄから、Ｖ＝ｄＡによって定義される楕円曲線上のＶを導出する演算であり、点の加算、点の減算および２倍算の組み合わせからなる。

【００２６】

ベースポイントＧと、個人鍵を表すスカラー値ｄとに対し、公開鍵はＶ＝ｄＧを満たすＶにより与えられる。公開鍵は楕円曲線上の点である。個人鍵はスカラー値である。

【００２７】

次に、電力解析攻撃について説明する。

べき乗剰余または点のスカラー倍算を実現するためのアルゴリズムには、例えばバイナリ法、符号付きバイナリ法およびウィンドウ法等が含まれる。攻撃者はスマートカードにおけるべき乗剰余または点のスカラー倍算のアルゴリズムを知っているものと仮定する。

【００２８】

図２は通常のバイナリ法によるべき乗剰余のアルゴリズムを示している。与えられた基数ａ、指数ｄおよび法ｎに対して、べき乗剰余値ｖ＝a^d（ｍｏｄ　ｎ）が求められる。図３は、通常のバイナリ法による点のスカラー倍算のアルゴリズムを示している。与えられた点Ａおよびスカラー値ｄに対して、点のスカラー倍を表す点Ｖ＝ｄＡが求められる。ここで、ｄはｍビットの２進値を表す。２進値ｄ中のｉ番目のビット値はｄ［ｉ］で表される（ｉ＝０，１，．．．　ｍ－１）。２進値ｄ中のｉ番目からｊ番目のビット値（ｉ≧ｊ≧０）のストリングの値はｄ［ｉ，ｊ］で表される。例えば、ｄ＝１９＝（１００１１）$_2$のとき、ｄ［４］＝１、ｄ［３］＝０、ｄ［２］＝０、ｄ［１］＝１、ｄ［０］＝１であり、ｄ［０，０］＝（１）$_2$＝１、ｄ［１，０］＝（１１）$_2$＝３、ｄ［２，０］＝（０１１）$_2$＝３、ｄ［３，０］＝（００１１）$_2$＝３、ｄ［４，０］＝（１００１１）$_2$＝１９、ｄ［４，１］＝（１００１）$_2$＝９である。ＥＣＡＤＤは点の加算を表し、ＥＣＤＢＬは点の２倍算を表す。

【００２９】

図２を参照すると、ステップ１０２においてワーク変数ｔおよびｖがそれぞれｔ＝ａおよびｖ＝１と初期化される。ステップ１０３～１０６は、ｉに対するループ処理であり、ｉ＝０，１，．．．ｍ－１について順次、ｄ［ｉ］＝０に対してステップ１０５における２乗算が実行され、ｄ［ｉ］＝１に対してステップ１０４における乗算とステップ１０５における２乗算が実行される。ステップ１０６の後、ワーク変数ｖの値はｖ＝$a^{d[i,0]}$（ｍｏｄ　ｎ）で表される。

【００３０】

図３を参照すると、ステップ２０２においてワーク変数ＴおよびＶがＴ＝ＡおよびＶ＝Ｏと初期化される。ステップ２０３～２０６は、ｉに対するループ処理であり、ｉ＝０，１，．．．ｍ－１について順次、ｄ［ｉ］＝０に対してステップ２０５における２倍算が実行され、ｄ［ｉ］＝１に対してステップ２０４における点の加算（ＥＣＡＤＤ）とステップ２０５における２倍算（ＥＣＤＢＬ）とが実行される。ステップ２０６の後、ワーク変数Ｖの値はＶ＝ｄ［ｉ，０］で表される。

【００３１】

図２および３のアルゴリズムと個人鍵ｄの間には、次の（ＣＯＲ．１）および（ＣＯＲ．２）の相関関係がある。

【００３２】

（ＣＯＲ．１）　値ｄのビット値と実行される演算の間に相関関係がある。図３のアルゴリズムにおいて、ｄの各ビット値に従って、ＥＣＤＢＬ、またはＥＣＤＢＬおよびＥＣＡＤＤが行われる。

（ＣＯＲ．２）　値ｄのビット値とワーク変数の値の間に相関関係がある。図３のアルゴリズムにおいて、ステップ１０６の後のＶの値はＶ＝（ｄ［ｉ，０］）Ａ＝（2^iｄ［ｉ］）Ａ＋（ｄ［ｉ－１，０］）Ａで表される。

【００３３】

ＳＰＡは相関関係（ＣＯＲ．１）に基づいて個人鍵を求める。ＤＰＡは相関関係（ＣＯＲ．２）に基づいて個人鍵を推定する。ＲＰＡは相関関係（ＣＯＲ．１）および（ＣＯＲ．２）の任意のいずれかに基づいて個人鍵を推定する。

【００３４】

ＳＰＡは、単一の電力波形を測定し、スマートカード内部で行われている処理を推定す

ることによって個人鍵を推定する。

【００３５】

例えば、図３のアルゴリズムを処理するスマートカードの消費電力を測定し、図４に例示された電力波形が得られたと仮定する。図４のように、得られた消費電力波形ＡおよびＤがＥＣＡＤＤとＥＣＤＢＬを識別できるような形状を有するとき、相関関係（ＣＯＲ．１）に従って、電力波形Ｄとその後続の波形Ａの組はビット値１に対応し、単独の波形Ｄはビット値０に対応する。このようにして、個人鍵のビット列“００１１０”が求まる。

【００３６】

ＳＰＡに対する対策として、個人鍵ｄのビット値に関係なく一定の演算手順を繰り返すアド・アンド・ダブル・オールウェイズ（add-and-double-always、常に加算と２倍算を行う）およびモンゴメリ・ラダー（Montgomery-Ladder）等が知られている。

【００３７】

アド・アンド・ダブル・オールウェイズ法は、文献、J. Coron, "Resistance against differential power analysis for elliptic curve cryptographic cryptosystem", CHES '99, LNCS 1717, pp.292-302, Springer-Verlag, 1999に記載されている。

【００３８】

モンゴメリ・ラダー法は、文献、P. Montgomery, "Speeding the Pollard and elliptic curve methods for factorizations", Math of Comp, vol.48, pp.243-264, 1987に記載されている。

【００３９】

図５は、通常のアド・アンド・ダブル・オールウェイズのアルゴリズムを示している。この場合、演算の手順は、個人鍵ｄの各ビット値に関係なく、ＥＣＡＤＤ，ＥＣＤＢＬ，ＥＣＡＤＤ，ＥＣＤＢＬ，．．．，ＥＣＡＤＤ，ＥＣＤＢＬとなる。

【００４０】

ＤＰＡは、複数の電力波形を測定して解析することによって個人鍵を推定する。次の手順（ＤＰＡ．０１）～（ＤＰＡ．０３）は、図３のアルゴリズムに対するＤＰＡの例を示している。

【００４１】

（ＤＰＡ．０１）　ｋ個のランダムな与えられた点A_0，A_1，A_0，．．．A_{k-1}に対して、各A_jのｄ倍算を行っている期間の消費電力が測定される。A_jに対するスカラー倍算において測定されたｋ個の消費電力データの集合がＣ（A_j，ｔ）で表される（ｊ＝０，１，．．．ｋ－１）。

【００４２】

（ＤＰＡ．０２）　以下の手順に従って、ｄ［０］の値が推定され、その推定が正しいどうかが判定される。

－　推定値ｄ［０］が正しいと仮定する。この仮定に基づいて、ｉ＝０についてステップ２０３～２０６が終了した時点における、スマートカード内部の変数Ｖの座標ｘのデータ値をそれぞれのA_jについて予想する。例えばｄ［０］＝１と仮定した場合、図３のアルゴリズムに従ってＶ＝A_jと予想できる。予想の結果、Ｖのデータ値の最下位ビットが１であるとき、Ｃ（A_j，ｔ）をG_1に分類し、Ｖのデータ値の最下位ビットが０であるとき、Ｃ（A_j，ｔ）をG_0に分類する。分類に用いられるＶのデータ値は、ｘの代わりにｙまたはｚのデータ値であってもよい。

－　上述の分類に基づいて、次の式で表される電力差分曲線を作成する。

Δ（ｔ）＝｛G_1に属するＣ（A_i，ｔ）の平均｝
　　－｛G_0に属するＣ（A_j，ｔ）の平均｝

－　その電力差分曲線が図６Ａに示すようなスパイクを有するときは、ｄ［０］の推定値が正しいと判定する。その電力差分曲線が図６Ｂに示すように平坦なときは、ｄ［０］の推定値が誤りであると判定する。

【００４３】

（ＤＰＡ．０３）　手順（ＤＰＡ．０２）を、ｄ［１］，ｄ［２］，．．．ｄ［ｍ－１］

に対して順に繰り返し実行することによって個人鍵ｄの値を求める。ｄ［ｈ］に対して手順（ＤＰＡ．０２）を実行するとき、手順（ＤＰＡ．０２）における分類の基準となるＶのデータとして、ｉ＝ｈについて終了した時点のデータが用いられる。Ｖのデータ値の予想にあたっては、既に求められたｄ［０］～ｄ［ｈ－１］の値と今回推定したｄ［ｈ］の値が用いられる。その理由は、Ｖの値が、$V=2^{h}d[h]A+d[h-1,0]A$、即ち既に求められたｄ［０］～ｄ［ｈ－１］と今回推定したｄ［ｈ］によって決定される値だからである。

【００４４】

ＤＰＡは、スマートカードの消費電力がデータ値の“１”の数に比例するという性質に基づいている。

【００４５】

ＤＰＡに対する一般的な対策として乱数を用いた方法が知られている。その方法は、スマートカード内部で生成された乱数を用いて変数データを隠蔽またはランダム化することによって、ＤＰＡによるデータ値の推定を防止する。

【００４６】

例えば、ランダム化座標と呼ばれる方法は、投影座標またはヤコビアン座標の点のデータ表現をランダム化する。この場合、通常の暗号処理において、アファイン座標を表すワーク変数（ｘ，ｙ）に対して、ワーク変数が、投影座標（ｒ×ｘ，ｒ×ｙ，ｒ×ｚ）に変換され、またはヤコビアン座標（$r^{2}\times x$，$r^{3}\times y$，ｒ×ｚ）に変換される。ここで、ｒは乱数を表す。乱数ｒによって全ての座標データがランダム化され、ワーク変数の値は予測不可能となり、ＤＰＡが防止できる。それによってワーク変数の値はランダム化されるが、乱数ｒの値に関係なくワーク変数の値はいずれも同一の点を表すので、ｄ［０］～ｄ［ｍ－１］の計算後に、求めた座標をアファイン座標に逆変換することによって、通常の処理と同じ演算結果dA_{j}が一意的に求められる。

【００４７】

次に、ＲＰＡを説明する。ＲＰＡは、個人鍵ｄのビット値を推定し、その値に基づいて特定の入力点を選択してそれを暗号装置のスカラー倍算の入力として供給することによって、個人鍵ｄを推定する。スカラー倍算の特定のタイミングにおけるワーク変数値が０であれば、ＲＰＡはビット値の推定が正しいと判定し、そうでない場合は推定が誤りであると判定する。ワーク変数値が０であるか否かは、ＳＰＡにおける単一の電力波形を用いても、またＤＰＡにおける複数の電力波形の差分を用いても判定できる。

【００４８】

次の手順（ＲＰＡ．０１）～（ＲＰＡ．０３）は、図３のアルゴリズムに対するＲＰＡの例を示している。ＲＰＡにおいて、攻撃者は、既に求められたｄ［０］，ｄ［１］，．．．ｄ［ｈ－１］に基づいて、次のビット値ｄ［ｈ］を求める。

【００４９】

（ＲＰＡ．０１）　既知の値ｄ［０］，ｄ［１］，．．．ｄ［ｈ－１］と、推定値ｄ［ｈ］とから、$A=(d[h,0]^{-1}\ (\mathrm{mod}\ \varphi))Q$を求める。ここで、φは位数（order）の値であり、ＱはＱ＝（ｘ，０）または（０，ｙ）を満たす楕円曲線上の点、即ち座標ｘまたはｙが０となる座標である。

（ＲＰＡ．０２）　スマートカードにｄＡを演算させて、消費電力波形Ｃ（Ａ，ｄ）を測定する。

（ＲＰＡ．０３）　消費電力波形Ｃ（Ａ，ｄ）の曲線のうち、ステップ２０３～２０６のｉ＝ｈ，ｈ＋１に対応する部分波形を観察して、ワーク変数Ｖの座標ｘが０であるかどうかを判定する。Ｖの座標ｘが０かどうかの判定法については、前述のGoubinの文献を参照されたい。

【００５０】

ＲＰＡが可能な理由は、ｄ［ｈ］の推定が正しいとき、ステップ２０３～２０６のループがｉ＝ｈについて完了した後、ワーク変数Ｖの座標ｘまたはｙが０となるからである。Ｖの座標ｘまたはｙが０となると、この値がｉ＝ｈ＋１に対する演算に用いられるときに

、例外的処理が発生し、この例外処理は電力波形によって観察できる。
【００５１】
図７Ａおよび７Ｂは、ＲＰＡにおける電力波形を例示している。
【００５２】
ｄ［ｈ］の推定が正しいときにｉ＝ｈに対するループ処理の後にＶの座標値を表す波形が出現する理由は、手順（ＲＰＡ．０１）に従ってＡが与えられるからである。このようなＡを与えることによって、ｄ［ｈ］の推定が正しいときにｉ＝ｈに対するステップ２０３～２０６のループ処理の後のＶの値がＱに一致し、座標ｘまたはｙの値が0となる。
【００５３】
従って、ワーク変数の座標値が攻撃者の意図したタイミングにおいて０となることを回避するようアルゴリズムを作成することによってＲＰＡを防止することができる。
【００５４】
基本的にデータをランダム化するＤＰＡ対策によってＲＰＡを防止できるが、例外がある。例えば、ＤＰＡ対策の１つであるランダム化座標は、ＲＰＡに対して脆弱である。これは、ＤＰＡ対策なしの処理におけるワークデータを（Ｘ，Ｙ，Ｚ）としたとき、ランダム化座標を用いた場合のワークデータは（ｒ×Ｘ，ｒ×Ｙ，ｒ×Ｚ）となるからである。即ち、ＤＰＡ対策なしの処理における座標値が０である場合、ランダム化座標における座標値もｒの値に関係なく０となる。
【００５５】
本発明者によって公表された、伊豆、他の文献、“楕円曲線暗号向けサイドチャネル対策方式の比較評価”、２００３年、暗号と情報セキュリティシンポジウム（ＳＣＩＳ２００３）、８Ｄ－３には、楕円曲線暗号に対応する公開鍵暗号の電力解析攻撃の対策として、ＳＰＡおよびＤＰＡに対して最も安全な対策は、ランダム化座標（Randomized Projective Coordinates）（ＲＰＣ）、ランダム化曲線（Randomized Curve）（ＲＣ）、指数分割法（Exponent Splitting）（ＥＳ）、点の隠蔽（Point Blinding）（ＰＢ）の４つの対策であることが記載されている。これらは、ＤＰＡ対策であるが、ＳＰＡ対策とともに用いることによってＤＰＡとＳＰＡの双方を防止できる。
【先行技術文献】
【特許文献】
【００５６】
【特許文献１】特開２０００－１３７４３６号公報
【非特許文献】
【００５７】
【非特許文献１】P. Kocher, J. Jaffe and B. Jun "Differential Power Anaysis", Crypto'99, LNCS 1666, pp.388-397, Springer-Verlag, 1999
【非特許文献２】L. Goubin, "A Refined Power-Analysis Attack on Elliptic Curve Crypto-systems", PKC 2003, LNCS 2567, Springer-Verlag, 2003
【非特許文献３】IEEE P1363/D13 (Draft Version 13, November 12, 1999) main document, Standard Specifications for Public Key Cryptography
【非特許文献４】J. Coron, "Resistance against differential power analysis for elliptic curve cryptographic cryptosystem", CHES'99, LNCS 1717, pp.292-302, Springer-Verlag, 1999
【非特許文献５】伊豆、他“楕円曲線暗号向けサイドチャネル対策方式の比較評価”、２００３年、暗号と情報セキュリティシンポジウム（ＳＣＩＳ２００３）、８Ｄ－３
【発明の概要】
【００５８】
表１は、アド・アンド・ダブル・オールウェイズ法を併用した場合の、ＳＰＡ、ＤＰＡおよびＲＰＡに対する安全性と、点のスカラー倍算に必要な処理量の比較を示している。
【００５９】

【表１】

公開鍵暗号の安全性と、点のスカラー倍算の処理時間の比較

	ＲＰＡ	ＳＰＡ	ＤＰＡ	処理時間
ＲＰＣ	Ｖ	Ｓ	Ｓ	E+I
ＲＣ	Ｖ	Ｓ	Ｓ	E+I
ＥＳ	Ｓ	Ｓ	Ｓ	2E+A
ＰＢ	Ｓ	Ｓ	Ｓ	2E+3A

【００６０】
ここで、Ｓは電力解析攻撃に対して安全であること、Ｖは電力解析攻撃に対して脆弱であることを表す。Ｅは、図５のアド・アンド・ダブル・オールウェイズ法による点のスカラー倍算の処理時間を表し、Ａは、点の加算、減算および２倍算の処理量を表し、Ｉは逆元計算の処理量を表している。
【００６１】
ＲＰＣおよびＲＣはその処理が高速であるが、ＲＰＡに対して弱い。ＥＳおよびＰＢはＲＰＡに対し安全であるがその処理が遅い。
【００６２】
図８Ａは、投影座標に基づくＲＰＣのアルゴリズムを示している。図８Ｂは、ヤコビアン座標に基づくＲＰＣのアルゴリズムを示している。図８Ｃは、アファイン座標に基づくＲＣのアルゴリズムを示している。
【００６３】
これらのアルゴリズムは、（ｉ）入力点Ａの座標データのランダム化（図８Ａのステップ４０１～４０２；図８Ｂのステップ５０１～５０２；図８Ｃのステップ６０１～６０２）、（ii）アド・アンド・ダブル・オールウェイズを用いたスカラー倍算（図８Ａのステップ４０３～４０８；図８Ｂのステップ５０３～５０８；図８Ｃのステップ６０３～６０８）、および（iii）データのランダム化を解除して元に戻して（de-randomize）（逆ランダム化）、演算結果を出力する（図８Ａのステップ４０９～４１０；図８Ｂのステップ５０９～５１０；図８Ｃの６０９～６１０）の３段階で構成される。
【００６４】
段階（ｉ）はＤＰＡ対策のための処理である。乱数ｒを生成し、図８ＡにおいてはＡ’＝（$r \times A_X$，$r \times A_Y$，$r \times A_Z$）、図８ＢにおいてはＡ’＝（$r^2 \times A_X$，$r^3 \times A_Y$，$r \times A_Z$）、図８ＣにおいてはＡ’＝（$r^2 \times A_X$，$r^3 \times A_Y$）に従って座標データをランダム化し、それによって、これらのデータが攻撃者によって推定されるのを防ぐ。ここで、A_X，A_Y，A_Zはそれぞれ点Ａの座標（Ｚ，Ｙ，Ｚ）を表す。
【００６５】
段階（iii）は、データのランダム化を解除して、通常の暗号処理と同じ処理値を出力するための処理である。図８ＡにおいてはＶ＝（$V[0]_X / V[0]_Z$，$V[0]_Y / V[0]_Z$）、図８ＢにおいてはＶ＝（$V[0]_X / (V[0]_Z)^2$，$V[0]_Y / (V[0]_Z)^3$）、図８ＣにおいてはＶ＝（$V[0]_X / r^2$，$V[0]_Y / r^3$）が処理される。
【００６６】
しかし、これらの対策はＲＰＡには無効である。その理由は、段階（ｉ）のランダム化

において座標値と乱数ｒの積を導入するからである。例えば、図８Ｃにおいて、ランダム化を行わない場合のワーク変数データをＶ＝（$Ｖ_{Ｘ}$，$Ｖ_{Ｙ}$）とすると、段階（ｉ）ステップ６０３～６０８のワーク変数データはＶ’＝（$ｒ^{2}×Ｖ_{Ｘ}$，$ｒ^{3}×Ｖ_{Ｙ}$）で表される。即ち、ランダム化を行わない時のＶの座標Ｘ，Ｙの値いずれかが０のとき、Ｖ’の座標Ｘ，Ｙの値いずれかも乱数ｒに関係なく０となり、ＲＰＡが適用できる条件を満たす。図８Ａおよび８Ｂに示したＲＰＣおよびＲＰＡについても同様である。

【００６７】

処理時間は、ＲＰＣおよびＲＣにおいてアド・アンド・ダブル・オールウェイズ演算が１回、ランダム化の解除のための逆元演算が１回必要であり、合計Ｅ＋Ｉである。

図９ＡはＥＳのアルゴリズムを示している。図９ＢはＰＢのアルゴリズムを示している。

【００６８】

これらのアルゴリズムは、図８Ａ～８Ｃのアルゴリズムと同様に、（ｉ）入力点Ａの座標データのランダム化（図９Ａのステップ７０１～７０２；図９Ｂのステップ８０１～８０２）、（ii）アド・アンド・ダブル・オールウェイズを用いたスカラー倍算（図９Ａのステップ７０３～７０４；図９Ｂのステップ８０３～８０４）、および（iii）データのランダム化を解除して元に戻し、演算結果を出力する（図９Ａのステップ７０５～７０６；図９Ｂの８０５～８０６）の３段階で構成される。

【００６９】

図９Ａにおいて、段階（ｉ）ではスカラー値ｄに対して、$ｄ＝ｄ_{1}＋ｄ_{2}$を満たすランダムなスカラー値$ｄ_{1}$および$ｄ_{2}$が生成される。段階（ii）では、前述の$ｄ_{1}$および$ｄ_{2}$に対して$Ｖ_{1}＝ｄ_{1}Ａ$および$Ｖ_{2}＝ｄ_{2}Ａ$で表されるスカラー倍算を、図５のアド・アンド・ダブル・オールウェイズ演算に従って行う。ランダムな$ｄ_{1}$および$ｄ_{2}$に対するスカラー倍算によってワーク変数をランダム化し、それによってＤＰＡ対策が実現される。段階（ｉ）および（ii）によって攻撃者の意図したタイミングでワーク変数の座標値が０となることが防止されて、ＲＰＡ対策が実現される。段階（iii）では、$Ｖ＝Ｖ_{1}＋Ｖ_{2}＝ｄＡ＋（ｄ－ｒ）Ａ＝ｄＡ$の演算によって、データのランダム化を解除して元に戻し、ｄＡを出力する。

【００７０】

図９Ｂにおいて、段階（ｉ）では入力点Ａに対し、$Ａ＝Ａ_{1}＋Ａ_{2}$を満たすランダムな点$Ａ_{1}$および$Ａ_{2}$を生成している。段階（ii）では前述の点$Ａ_{1}$および$Ａ_{2}$に対して$Ｖ_{1}＝ｄＡ_{1}$および$Ｖ_{2}＝ｄＡ_{2}$で表されるスカラー倍算を、図５のアド・アンド・ダブル・オールウェイズ演算に従って行う。ランダムな点$Ａ_{1}$および$Ａ_{2}$に対するスカラー倍算によってワーク変数をランダム化し、それによってＤＰＡ対策が実現される。段階（ｉ）および（ii）によって攻撃者の意図したタイミングでワーク変数の座標値が０となることが防止されて、ＲＰＡ対策が実現される。段階（iii）では、$Ｖ＝Ｖ_{1}＋Ｖ_{2}＝ｄ（Ａ_{1}＋Ａ_{2}）＝ｄＡ$の演算によって、データのランダム化を解除して元に戻し、ｄＡを出力する。

【００７１】

ＥＳの場合、アド・アンド・ダブル・オールウェイズ演算を２回と、ランダム化の解除の際に点の加算を１回を必要とするので、処理時間は合計で２Ｅ＋Ａである。ＰＢの場合、アド・アンド・ダブル・オールウェイズ演算を２回と、データのランダム化とその解除の際に点の加算または減算を３回を必要とするので、処理時間は合計で２Ｅ＋３Ａである。

【００７２】

従って、点のスカラー倍算の処理量が少ない公知の公開鍵暗号処理法ではＲＰＣに対して弱く、ＲＰＣに対して安全な公知の公開鍵暗号処理法ではスカラー倍算の処理量が多くなる。

【００７３】

発明者たちは、スカラー倍算の処理量が少なく、かつＳＰＡ、ＤＰＡおよびＲＰＣに対

して安全な公開鍵暗号の処理法を実現することの必要性を認識した。

【００７４】

本発明の目的は、スカラー倍算の処理量が少なく、かつＳＰＡ、ＤＰＡおよびＲＰＣに対して安全な公開鍵暗号の処理法を実現することである。

【００７５】

発明の概要

本発明の特徴によれば、個人鍵を用いて楕円曲線暗号処理を行う暗号装置は、外部からアクセスできない内部で生成された乱数に基づいて楕円曲線上の点Ｒをランダムに生成し、その楕円曲線上の或る入力点Ａの値を入力し、その際、その点Ｒに応じて初期化される変数に対してその楕円曲線上の或る入力点Ａの値に応じた初期化を行わず、その入力点Ａに応じて初期化される変数に対してその点Ｒの値に応じた初期化を行わない、初期化手段と、その点Ｒを初期点$Ｖ_0$にセットし、その入力点Ａをセットし、その楕円曲線暗号処理のための或るスカラー値ｄのビット・シーケンスの全てのビット値に対して、その初期点$Ｖ_0$とその入力点Ａのスカラー倍との和$Ｖ_1＝Ｖ_0＋ｄＡ$の演算を実行する演算手段と、その演算によって得られた和$Ｖ_1$からその初期点$Ｖ_0$を減算する演算$Ｖ＝Ｖ_1－Ｖ_0$を実行するランダム化解除手段と、そのランダム化解除手段によって得られた点Ｖを出力として供給する手段と、を具えている。

【００７６】

本発明の別の特徴によれば、個人鍵を用いてべき乗剰余暗号処理を行う暗号装置は、外部からアクセスできない内部で生成された乱数に基づいて整数ｒをランダムに生成し、或る入力値ａを入力し、その際、その整数ｒに応じて初期化される変数に対してその入力値ａに応じた初期化を行わず、その入力値ａに応じて初期化される変数に対してその整数ｒに応じた初期化を行わない、初期化手段と、その整数ｒを初期値$Ｖ_0$にセットし、その入力値ａをセットし、そのべき乗剰余暗号処理のための或る値ｄを表すｍビット・シーケンスの全てのビット値に対して、その初期値$Ｖ_0$と或る入力値ａに対するべき乗剰余演算$Ｖ_1＝Ｖ_0×Πａ^（２^ｉｄ［ｉ］）（ｍｏｄ　ｎ）＝ｒ×ａ^ｄ$（ｍｏｄ　ｎ）の演算（ｉ＝０，１，．．．ｍ－１）を実行する演算手段と、その演算によって得られた値$Ｖ_1$とｒ（ｍｏｄ　ｎ）の逆元$ｒ^{－1}$（ｍｏｄ　ｎ）との乗算剰余演算$Ｖ＝Ｖ_1×ｒ^{－1}$（ｍｏｄ　ｎ）を実行するランダム化解除手段と、そのランダム化解除手段によって得られた値Ｖを出力として供給する手段と、を具える。ここで、$Πａ_ｉ$は積$ａ_0×ａ_1×．．．×ａ_{ｍ－1}$を表し、ａ^ｄは$ａ^ｄ$を表す。

【００７７】

本発明は、上述の暗号装置を実現するプログラムに関する。

本発明は、上述の暗号装置を実現する方法に関する。

【００７８】

本発明によれば、スカラー倍算の処理量が少なく、かつＳＰＡ、ＤＰＡおよびＲＰＣに対して安全な公開鍵暗号の処理法を実現できる。

【図面の簡単な説明】

【００７９】

図面において同じ要素には同じ参照番号が付されている。

【００８０】

【図１】図１は、スマートカードのような暗号デバイスにおける秘密鍵を用いた暗号化／復号の構成の例を示している。

【図２】図２は、通常のバイナリ法によるべき乗剰余のアルゴリズムを示している。

【図３】図３は、通常のバイナリ法による点のスカラー倍算のアルゴリズムを示している。

【図４】図４は、スマートカードの消費電力を測定して得られた電力波形を示している。

【図５】図５は、通常アド・アンド・ダブル・オールウェイズのアルゴリズムを示している。

【図６】図６Ａは、スパイクを有する、時間に対する消費電力の曲線を示している。図６

Ｂは、平坦な、時間に対する電力差分曲線を示している。
【図７】図７Ａおよび７Ｂは、ＲＰＡにおける電力波形を例示している。
【図８Ａ】図８Ａは、投影座標に基づくＲＰＣのアルゴリズムを示している。
【図８Ｂ】図８Ｂは、ヤコビアン座標に基づくＲＰＣのアルゴリズムを示している。
【図８Ｃ】図８Ｃは、アファイン座標に基づくＲＣのアルゴリズムを示している。
【図９】図９Ａは、ＥＳ暗号のアルゴリズムを示している。図９Ｂは、ＰＢ暗号のアルゴリズムを示している。
【図１０】図１０は、本発明による暗号処理装置の概略的構成を示している。
【図１１】図１１は、図１０の暗号化装置によって実行される、本発明の電力解析攻撃対策が施された公開鍵暗号処理のためのアルゴリズムを示している。
【図１２】図１２は、図１０の暗号化装置によって実行される、本発明による電力解析攻撃対策を楕円曲線暗号に適用した基本的アルゴリズムを示す。
【図１３】図１３は、図１０の暗号化装置によって実行される、本発明による電力解析攻撃対策をＲＳＡ暗号に適用した基本的アルゴリズムを示す。
【図１４】図１４～１６は、図１２におけるランダムな点Ｒの生成のアルゴリズムを示している。
【図１５】(図14で説明)
【図１６】(図14で説明)
【図１７】図１７は、図１２における投影座標における点Ｒの生成のアルゴリズムを示している。
【図１８】図１８は、図１２におけるヤコビアン座標における点Ｒの生成のアルゴリズムを示している。
【図１９】図１９は、図１２におけるワーク変数Ｖ［２］に対する点の２倍算の実施形態を示している。
【図２０】図２０は、図１６、２０および１９のアルゴリズムを組み合わせた、図１２に示された本発明による基本的アルゴリズムの実施形態を示している。
【発明を実施するための形態】
【００８１】
図１０は、本発明による暗号処理装置１０の概略的構成を示している。暗号処理装置１０は、値ａ、ｄおよびｎまたは値Ａおよびｄを入力して供給する入力部１２と、乱数Ｒまたはｒを発生する乱数生成部１４と、乱数生成部１４からの乱数Ｒまたはｒに従ってワーク変数をＶ’［０］：＝Ｒ(またはｒ)、Ｖ［２］＝Ａ（またはａ）として初期化するワーク変数初期化部１６と、ワーク変数Ｖ’［０］、Ｖ’［１］およびＶ［２］を格納するメモリ１８と、を具えている。暗号処理装置１０は、さらに、値ｄの各ビット値を検出しながら初期化部１６のワーク変数に対して公開鍵加算（ＰＵＢＡＤＤ）および公開鍵２倍算（ＰＵＢＤＢＬ）を繰り返し実行して、Ｖ’［０］＝ｄＡ＋Ｒまたは$a^d \times r$（ｍｏｄｎ）を計算するスカラー倍算処理部２０を具えている。ここで、ＰＵＢＡＤＤは点の加算または法ｎの乗算剰余を表し、ＰＵＢＤＢＬは点の２倍算または法ｎの２乗剰余を表す。ＰＵＢＡＤＤとＰＵＢＤＢＬの演算手順はｄビット値に依存しない。暗号処理装置１０は、さらに、Ｖ’［０］からＶ＝ＰＵＢＳＵＢ（Ｖ’［０］，Ｒまたはｒ）を計算してＲまたはｒの影響を取り除いたＶ＝ｄＡを計算するランダム・データ除去部を具えている。ここで、公開鍵減算ＰＵＢＳＵＢは点の減算またはｒの逆元を用いた法ｎにおける乗算剰余を表す。暗号処理装置１０は、さらに、演算結果の値Ｖ＝ｄＡまたは$v = a^d$（ｍｏｄｎ）を出力する出力部２４を含んでいる。Ｒまたはｒは、暗号装置１０内部で生成され、外部からはアクセスできない値である。
【００８２】
暗号化装置１０は、さらに、プロセッサ３２と、ＲＯＭのようなプログラム・メモリ３４とを含んでいる。プロセッサ３２は、メモリ３４に格納されているプログラムに従ってこれらの要素１２～２４を制御する。代替構成として、プロセッサ３２は、要素１２～２４に対応する機能を実現するメモリ３４中のプログラムを実行することによって要素１２

～２４を実現してもよい。この場合、図１０はフロー図として見ることができる。

【００８３】

図１１は、図１０の暗号化装置１０によって実行される、本発明による電力解析攻撃への対策が施された公開鍵暗号処理のための基本的アルゴリズムを示している。このアルゴリズムは楕円曲線暗号とＲＳＡ暗号の双方に適用できる。

【００８４】

図１１のアルゴリズムは、図８Ａ～８Ｃのアルゴリズムと同様に、（ｉ）データのランダム化（ステップ１００４～１００６）、（ii）アド・アンド・ダブル・オールウェイズを用いたスカラー倍算（ステップ１００８）、および（iii）データのランダム化を解除して元に戻し、演算結果Ｖを出力する（ステップ１０１０）の３段階で構成される。

【００８５】

図１１を参照すると、ステップ１００２において、入力部１２は、値ａ、ｄおよびｎまたは値ＡおよびｄをＩＮＰＵＴする。ステップ１００４において、生成部１４は、ランダムなデータＲまたはｒを生成する。Ｒは、楕円曲線暗号において楕円曲線上のランダムな点のデータであり、ＲＳＡにおいてランダムな整数値である。

【００８６】

ステップ１００６において、初期化部１６は、アド・アンド・ダブル・オールウェイズ法で用いられるワーク変数を、Ｖ’［０］＝ＲまたはＲ、Ｖ［２］＝Ａまたはａと、初期化する。

【００８７】

ステップ１００８において、点のスカラー倍算実行部２０は、ワーク変数Ｖ’［０］、Ｖ’［１］、Ｖ［２］を用いてアド・アンド・ダブル・オールウェイズを実行する。即ち、ｉ＝０，１，．．．，ｍ－１に対して、楕円曲線暗号においてＶ’［１］＝ＥＣＡＤＤ（Ｖ’［０］，Ｖ［２］），Ｖ［２］＝ＥＣＤＢＬ（Ｖ［２］）、およびＶ’［０］＝Ｖ’［ｄ［ｉ］］の演算を繰り返し実行し、またはＲＳＡ暗号においてＶ’［１］＝Ｖ’［０］×Ｖ［２］（ｍｏｄ　ｎ）、Ｖ［２］＝Ｖ［２］^{2}（ｍｏｄ　ｎ）、およびＶ’［０］＝Ｖ’［ｄ［ｉ］］の演算を繰り返し実行する。個人鍵ｄのビット値に関係なく、アド・アンド・ダブル・オールウェイズの所定の演算パターンを繰り返すことによってＳＰＡ対策を実現する。ステップ１００４においてＶ’［０］＝ＲまたはＲとセットすることによって、ワーク変数Ｖ’［０］およびＶ’［１］の値がランダム化されて、ＤＰＡを防止できる。Ｖ［２］の値はランダム化されない。その理由は、アド・アンド・ダブル・オールウェイズ法においてはＶ［２］の値を利用したＤＰＡおよびＳＰＡが不可能なので、Ｖ［２］をランダム化する必要がないからである。Ｖ［２］の値は、個人鍵ｄのビット値と関係なく、Ａの２倍算または２乗算を繰り返し実行することによって求められる値なので、この値を利用してＤＰＡを実行することはできない。

【００８８】

ステップ１０１０において、ランダム・データ除去部２２はランダム化されたデータを元に戻す。そのために、ステップ１００４で生成されＲに対して、楕円曲線暗号においてはＶ＝Ｖ’［０］－Ｒが計算され、ＲＳＡにおいてはＶ＝Ｖ’［０］×r^{-1}が求められる。ステップ１０１２において、出力部２４は演算結果Ｖを出力する。

【００８９】

このようにして、図１１に示されたアルゴリズムによってＳＰＡおよびＤＰＡが防止される。次に、図１１のアルゴリズムを楕円曲線暗号に適用したとき、ＲＰＡが防止されることを説明する。

【００９０】

図１２は、図１０の暗号化装置１０によって実行される、本発明による電力解析攻撃対策を楕円曲線暗号に適用した基本的アルゴリズムを示す。

【００９１】

図１２を参照すると、ステップ１１０１において、乱数生成部１４は楕円曲線上のランダムな点Ｒを生成または選択する。初期化部１６は、ステップ１１０２において初期値と

してＲをワーク変数Ｖ’［０］にセットし、ステップ１１０３において初期値としてＡをワーク変数Ｖ［２］にセットする。

【００９２】

ステップ１１０４～１１０８を含むループは、ワーク変数Ｖ’［０］、Ｖ’［１］およびＶ［２］に対してアド・アンド・ダブル・オールウェイズ法による点のスカラー倍算を実行する。このループは、図５のステップ３０３～３０７のＲＰＡ対策無しのアド・アンド・ダブル・オールウェイズ法と同じ処理を行っているが、Ｖ’［０］の初期値としてランダム点Ｒがセットされることによって、ＲＰＡ対策が実現される。

【００９３】

そのような初期値によって、図１２のワーク変数Ｖ’［０］およびＶ’［１］と図５のＶ［０］およびＶ［１］の間の関係は、Ｖ’［０］＝Ｖ［０］＋ＲおよびＶ’［１］＝Ｖ［１］＋Ｒで表される。即ち、図５で計算されるワーク変数のＶ［０］およびＶ［１］の値に対して、図１２のアルゴリズムは、Ｖ’［０］＝Ｖ［０］＋ＲおよびＶ’［１］＝Ｖ［１］＋Ｒを求め、これらの値はＲに応じてランダムに変化する。即ち、ＲＰＣおよびＲＣのように、ランダム値Ｒに関係なくＶ’［０］およびＶ’［１］の座標データが０となることはない。従って、攻撃者は意図的にワーク変数の座標データ値を０とすることが不可能となり、ＲＰＡが防止される。

【００９４】

ステップ１１０４～１１０８のループが終了した後、ステップ１１０９において出力部２４はＶ＝Ｖ’［０］－Ｒを計算することによって、Ｒによってランダム化されたデータを元に戻して、その値Ｖを出力する。

【００９５】

このようにして、本発明によるアルゴリズムはＲＰＡに対して安全な暗号処理を実現する。ＲＰＡ対策なしの図５のアルゴリズムと本発明による図１２のアルゴリズムを比較すると、ＲＰＡ対策のための追加的処理はステップ１１０１に示す点Ｒの生成と、ステップ１１０９における点Ｒの減算だけである。即ち、ＲＰＡ対策に関係する速度低下は、これら２つの処理分だけに限定されるので、図５の２倍の処理時間を必要とするＥＳおよびＰＢと比較すると高速な処理が実現される。

【００９６】

図１３は、図１０の暗号化装置１０によって実行される、本発明による電力解析攻撃対策をＲＳＡ暗号に適用した基本的アルゴリズムを示す。

【００９７】

図１３を参照すると、ステップ１２０１において、乱数生成部１４は乱数ｒを生成する。初期化部１６は、ステップ１２０２において初期値としてｒをワーク変数Ｖ’［０］にセットし、ステップ１２０３において初期値としてａをワーク変数Ｖ［２］にセットする。

【００９８】

ステップ１２０４～１２０８を含むループは、ワーク変数Ｖ’［０］、Ｖ’［１］およびＶ［２］に対してアド・アンド・ダブル・オールウェイズ法によりべき剰余演算を実行してワーク変数を更新する。

【００９９】

ステップ１２０４～１２０８のループが終了した後、ステップ１２０９において出力部２４はｒの逆元r^{-1}（ｍｏｄ　ｎ）とＶ’［０］の乗算剰余を計算することによって、ｒによってランダム化されたデータを元に戻して、その値Ｖを出力する。

【０１００】

ステップ１２０４～１２０８におけるアド・アンド・ダブル・オールウェイズ法によるループによってＳＰＡを防ぎ、ワーク変数Ｖ’［０］の初期値に乱数ｒがセットされることによって、そのループにおけるワーク変数をランダムに変化させＤＰＡを防ぐ。

【０１０１】

処理のオーバーヘッドについて、図５をＲＳＡに適用する場合と比較すると、ＲＰＡ対

策のための追加的処理はステップ１２０１に示す追加処理はｒの生成と、ステップ１２０９に示すｒの逆元の乗算剰余だけである。

【０１０２】

次に、図１２の楕円曲線暗号用の基本的アルゴリズムの実施形態について説明する。

【０１０３】

図１２におけるステップ１１０１における点Ｒの生成、ステップ１１０７におけるＶ［２］の２倍算として、必要な計算量が異なる複数の実施形態が存在する。

【０１０４】

図１４～１８は、図１２におけるステップ１１０１を実現する相異なるランダムな点Ｒの生成のアルゴリズムを示している。

【０１０５】

図１４を参照すると、ステップ１３０１において、乱数点生成部１４は、取り得る任意の座標Ｘの値をランダムに生成し、値ＸをR_Xにセットする。即ち、素体の楕円曲線パラメータを用いた楕円曲線暗号の場合、$0 \leqq R_X < p$を満たす整数を、２べきの楕円曲線パラメータの場合ＧＦ（2^m）のランダムな値を与えることによって生成する。ステップ１３０２において所与の座標Ｘに対応する座標Ｙを求めR_Yを生成する。ステップ１３０３および１３０４において、Ｒ＝（R_X，R_Y）とセットする。座標Ｘから座標Ｙを求めるためのアルゴリズムは、前記文献ＩＥＥＥ　Ｐ１３６３／Ｄ１３に記載されている。

【０１０６】

図１５を参照すると、生成部１４は、ステップ１４０１において乱数ｓを生成し、ステップ１４０２において図５のアド・アンド・ダブル・オールウェイズによってベースポイントＧをｓ倍したｓＧをＲにセットする。あるいは、図５のアルゴリズムの代わりに、図１２のアルゴリズムをＲ＝Ｏ、ｄ＝ｓおよびＡ＝Ｇについて実行してもよい。このように同一アルゴリズムを共用することによって、リソースの利用を効率化できる。乱数ｓは任意の値であるが、それがスカラー値ｄのビット長を超える場合、ステップ１４０２のスカラー倍算による計算オーバーヘッドが大きい。例えば１６０ビットの楕円曲線暗号においてｓが１６０ビットの乱数であるとき、ステップ１４０２の計算量が大きくなる。その結果、本発明によるアルゴリズム全体の計算量が図５のアルゴリズムの概ね２倍となって処理速度が低下する。従って、ｓは２０～３０ビットの値であることが好ましい。

【０１０７】

図１６は、スマートカードの初期化等において点Ｒの初期値をランダムに与えた後、本発明による電力解析攻撃から保護された暗号処理が呼び出される度にその点Ｒの値を更新するアルゴリズムを示している。その更新を簡単な演算で実現することによって、図１４および１５に示されたアルゴリズムと比べて、その処理時間を短くできる。

【０１０８】

ステップ１５０１において、生成部１４は、通常の電力解析攻撃対策において使用されるＲの値をメモリ上の特定の領域から読み込む。代替構成として、例えば図１４および１５に示した方法を用いて点Ｒの初期値を与えてもよい。ステップ１５０２において、生成部１４は、点Ｒに対して２倍算を行うことによってその値を更新する。ステップ１５０３において、生成部１４はその更新した点Ｒの値を次回の処理のために格納する。

【０１０９】

図１７は、図１２における投影座標における点Ｒの生成のアルゴリズムを示している。図１８は、図１２におけるヤコビアン座標における点Ｒの生成のアルゴリズムを示している。

【０１１０】

これらのアルゴリズムにおいては、生成部１４は、図１６のアルゴリズムと同様にスマートカードが初期化されるとき等において点Ｒの初期値をランダムに生成し、その後、本発明による暗号プログラムが呼び出される度に、生成部１４は点Ｒ初期値を更新する。

【０１１１】

図１６のアルゴリズムと同様に、生成部１４は、ステップ１６０１においてメモリ１８

に格納されている前回のＲ読み込み、ステップ１６０３において更新されたＲを格納する。図１７において、生成部１４は、ステップ１６０２において定数ｔを座標値Ｒ＝（R_X，R_Y，R_Z）に乗算することによって初期値Ｒを更新する。図１８において、生成部１４は、ステップ１６０２において定数ｔ、t^2およびt^3を座標値Ｒ（R_X，R_Y，R_Z）の各成分に乗算することによって初期値Ｒを更新する。ｔの値として、例えばｔ＝３のような小さな値を選択することによって、Ｒの更新に必要な計算量を小さくできる。

【０１１２】

図１９は、図１２におけるステップ１１０７におけるワーク変数Ｖ［２］に対する点の２倍算の実施形態を示している。

【０１１３】

ステップ１１０７における点の２倍算は、同一ワーク変数Ｖ［２］に対して繰り返し実行され、Ｖ［２］の値はループ変数ｉを用いてＶ［２］＝2^{i+1}Ａと表される。ヤコビアン座標および素体の場合について、同一の点に対して繰り返し実行される２倍算（点の2^k倍算）によって計算を高速化する手法が、本発明者の中の２人、武仲および伊藤によって２０００年５月１６日付けで公開された特開２０００－１３７４３６号公報（Ａ）に開示されており、ここでこの文献を引用して組み込む。図１９では、その公報における図２に示されている点の2^k倍算の手法が用いられる。

【０１１４】

図１９におけるｉは、図１７におけるループ変数ｉの値と同じである。このアルゴリズムは、変数ＰとしてのＡ［２］＝（X_i，Y_i，Z_i）＝2^iＡの入力からＡ［２］＝（X_{i+1}，Y_{i+1}，Z_{i+1}）＝2^{i+1}Ａの出力までの演算を表している。ｉ＝０のときは、ｉｆ文によってｉ＝０に対する処理に分岐することによって、通常の素体ヤコビアン座標における２倍算と同じ処理を行う。ｉ≧１のときは、ｉｆ文によってｉ≧１に対する処理に分岐する。このとき、点の２倍算のためのワーク変数R_iを計算するために、ｉ－１におけるワーク変数T_{i-1}の値を再利用することによって計算量が削減される。前記文献ＩＥＥＥ　Ｐ１３６３に記載されている標準的な点の２倍算を用いたとき、１回当たり１０回の乗算が必要であるが、図１９のアルゴリズムを用いたとき１回当たり８回の乗算のみを必要とする。即ち、図１９に示されたアルゴリズムを用いることによって、標準的な方法を用いるのと比較して点の２倍算に必要な計算量を２割だけ削減できる。

【０１１５】

図２０は、図１５（Ｒ初期化）、１６（ステップ１１０１、Ｒの生成）および１９（ステップ１１０６、点の2^k倍算）のアルゴリズムを組み合わせた、図１２に示された本発明による基本的アルゴリズムの実施形態を示している。

【０１１６】

代替構成として、図１２のステップ１１０１として図１４～１８のいずれかと、ステップ１１０６として図１９および前記文献ＩＥＥＥ　Ｐ１３６３のいずれか、の組み合わせであってもよい。

【０１１７】

次に、図１３のＲＳＡ暗号用の基本的アルゴリズムの実施形態について説明する。ＲＳＡ暗号用の電力解析攻撃対策の場合、図１８がそのまま実施例となる。その理由は、ステップ１２０１における乱数ｒの生成、ステップ１２０６における２乗算の実現例には変形形態がないからである。

【０１１８】

以上説明したように、本発明の実施形態によれば、ＳＰＡ、ＤＰＡおよびＲＰＡの全ての解析に対する対策が実現でき、かつ通常安全なものとして知られていたＥＳおよびＰＢと比較して計算量が小さくなる。

【０１１９】

表２は、電力解析攻撃対策の通常の公開鍵暗号と本発明による楕円曲線暗号の比較を示している。表３は、電力解析攻撃対策の通常の公開鍵暗号と本発明によるＲＳＡ暗号の比較を示している。表３においてＲＰＣ、ＲＣおよびＰＢ暗号は楕円曲線暗号に適用できな

いので除外されている。

【０１２０】

【表２】

楕円曲線用の電力解析攻撃への対策の比較

	ＲＰＡ	ＳＰＡ	ＤＰＡ	処理時間
RPC(図８Ａ，８Ｂ)	V	S	S	E+I
ＲＣ(図８Ｃ)	V	S	S	E+I
ＥＳ(図９Ａ)	S	S	S	2E+1A
ＰＢ(図９Ｂ)	S	S	S	2E+3A
本発明(図２０)	S	S	S	E+2A

【０１２１】

【表３】

ＲＳＡ用の電力解析攻撃への対策の比較

	ＳＰＡ	ＤＰＡ	処理時間
ＥＳ(図９Ａ)	S	S	2E
本発明(図１３)	S	S	E+I

【０１２２】

ここで、Ｓは解析に対して安全であること、Ｖは解析に対して脆弱であることを表す。電力解析攻撃対策なしのアド・アンド・ダブル・オールウェイズにおいて、スカラー算の処理時間がＥで表され、逆元計算の処理時間がＩで表され、点の加算、減算および２倍算がＡで表されるものとする。ＡはＥに比べて無視できるほど小さい。

【０１２３】

通常の暗号化による処理量は表１のものである。本発明による図２０のアルゴリズムの処理量は、ステップ１７０２における点の２倍算とステップ１７１１における点の減算以外は、図５におけるアド・アンド・ダブル・オールウェイズと同じ演算を行うので、合計Ｅ＋２Ａと表される。

【０１２４】

表２から分かるように、ＳＰＡ、ＤＰＡおよびＲＰＡの全てに対して安全な暗号は、ＥＳ暗号、ＰＢ暗号および本発明の暗号である。処理時間を比較すると、ＡはＥに比べて無視できるほど小さいので、本発明の暗号の処理時間は、安全なＥＳ暗号およびＰＢ暗号の処理時間のほぼ半分である。

【０１２５】

ＡがＥと比べて無視できるほど小さいのは、アド・アンド・ダブル・オールウェイズに

おいて点の加算と２倍算がそれぞれｍ回実行され、合計２ｍ回実行されることから明らかである。例えば、１６０ビット楕円曲線暗号においてｍ＝１６０としたとき、Ｅ＝３２０Ａとなる。

【０１２６】

ＥＳに必要な計算は、ＲＳＡにおけるＥＳの演算手順が次の通りなので、２Ｅとなる。

【数１】

$$v_1 = a^{d_1} \pmod{n},$$

$$v_2 = a^{d_2} \pmod{n},$$

$$v = v_1 v_2 \pmod{n} = a^{d} \pmod{n}$$

【０１２７】

ここで、v_1およびv_2の乗算剰余はべき乗剰余演算と比べて非常に小さく無視できると仮定している。本発明による図１３の処理は、ステップ１２０１～１２０８に示すアド・アンド・ダブル・オールウェイズと、ステップ１２０９～１２１０における逆元r^{-1}（ｍｏｄ　ｎ）の演算で構成されるので、処理量は合計Ｅ＋Ｉと表される。即ち、Ｅ＞Ｉに対して、本発明の方がＥＳよりも高速な処理を実現できる。ＥとＩの大小関係は、処理環境によって異なるが、一般的にＥは演算ビット長の３乗、Ｉは演算ビット長の２乗に比例するので、Ｅ＞Ｉという仮定は妥当である。このように、本発明は電力解析攻撃に対する安全性を損なうことなく、処理時間が小さい。

【０１２８】

以上説明した実施形態は典型例として挙げたに過ぎず、その変形およびバリエーションは当業者にとって明らかであり、当業者であれば本発明の原理および請求の範囲に記載した発明の範囲を逸脱することなく上述の実施形態の種々の変形を行えることは明らかである。

【図１】

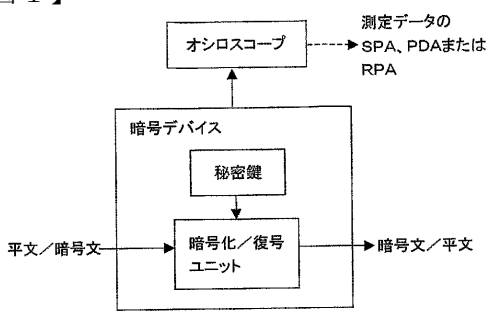


図1

【図２】

```
PowerMod_Bin(a,d,n) {
101: /* d[i]はdのi番目のビット値, i=0,1,...,m-1 */
102: t = a; v = 1;
103: for (i=0; i ≤ m-1; i++) {
104:   if (d[i] == 1) v = v*t (mod n) // =a^(2^i*d[i])*a^(d[i-1,0])= a^d[i,0] (mod n)
105:   t = t^2 (mod n)                // = a^(2^(i+1)) (mod n)
106: }
107: return v;                       // = a^d
}
```

図2

【図３】

```
ScalarMult_Bin(A,d) {
201: /* d[i]はdのi番目のビット値, i=0,1,...,m-1 */
202: T = A; V = O; //Vの初期値は無限遠点
203: for (i=0; i ≤ m-1; i++) {
204:   if (d[i] == 1) V = ECADD(V,T) // = (2^i*d[i])A + (d[i-1,0])A= d[i,0]A
205:   T = ECDBL(T)                  // = (2^(i+1)) A
206: }
207: return V;                       // = dA
}
```

図3

【図４】

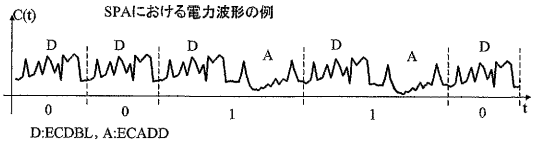


図4

【図５】

```
ScalarMult_AddandDoubleAlways(A,d) {
301: /* d[i]はdのi番目のビット値, i=0,1,...,m-1 */
302: V[0]=O; V[2]=A;
303: for (i=0; i≤m-1; i++) {
304:   V[1] = ECADD(V[0],V[2])   // = 2^iV + (d[i-1,0])V
305:   V[2] = ECDBL(V[2])        // = (2^(i+1))V (mod n)
306:   V[0] = V[d[i]]            // = (2^i*d[i])V + (d[i-1,0])V = (d[i,0])V
307: }
308: return V[0];                // = dA
}
```

図5

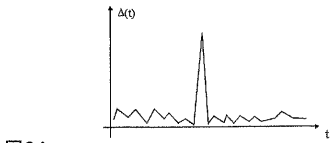


図6A

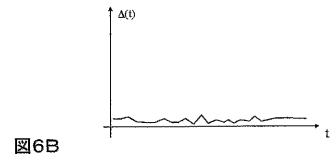


図6B

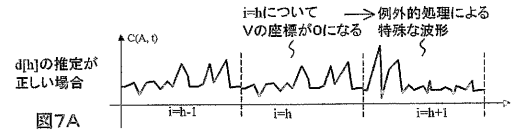


図7A

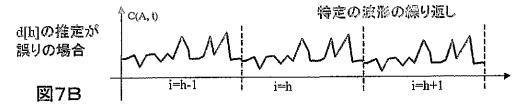


図7B

```
ScalarMult_RPC_Projective(A,d) {   //Aは投影座標          座標Aデータのランダム化
401: r = random number();          // 乱数rを生成
402: A'=(r * A_X, r * A_Y, r * A_Z);   // rを用いてAをランダム化しA'とする
403: V'[0]=O; V'[2]=A';                                    Add-and-Double-Always
404: for (i=0; i ≤ m-1; i++) {
405:   V'[1] = ECADD(V'[0],V'[2]); // = 2^iA' + (d[i-1,0])A'
406:   V'[2] = ECDBL(V'[2]);       // = (2^(i+1))A'
407:   V'[0] = V'[d[i]];           // = (2^i*d[i])A' + (d[i-1,0])A' = (d[i,0])A'
408: }
409: V=(V'[0]_X/V'[0]_Z, V'[0]_Y/V'[0]_Z); // V'[0]をaffine座標に変換する   ランダム化の解除
410: return V;                     // V = dAを出力
}
```

図8A

```
ScalarMult_RPC_Jacboian (A,d) {   //Aはヤコビアン座標
501: r = random number();          // 乱数rを生成          座標Aデータのランダム化
502: A'=(r^2*A_X, r^3*A_Y, r*A_Z);   // rを用いてAをランダム化しA'とする
503: V'[0]=O; V'[2]=A';                                    Add-and-Double-Always
504: for (i=0; i ≤m-1; i++) {
505:   V'[1] = ECADD(V'[0],V'[2]); // = 2^iA' + (d[i-1,0])A'
506:   V'[2] = ECDBL(V'[2]);       // = (2^(i+1))A'
507:   V'[0] = V'[d[i]];           // = (2^i*d[i])A' + (d[i-1,0])A' = (d[i,0])A'
508: }
509: V=(V'[0]_X/(V'[0]_Z)^2, V'[0]_Y/(V'[0]_Z)^3); // V'[0]をaffine座標へ変換する
510: return V;                     // V = dAを出力
}                                                          ランダム化の解除
```

図8B

```
ScalarMult_RC(A,d) {      //Aはaffine座標
601: r = random number();        // 乱数rを生成                    } 座標Aデータのランダム化
602: A'=(r²*Ax, r³*Ay);          // rを用いてAをランダム化しA'とする }
603: V'[0]=O; V'[2]=A';                                            } Add-and-Double-Always
604: for (i=0; i ≤ m-1; i++) {
605:    V'[1] = ECADD(V'[0],V'[2]); // = 2^i A' + (d[i-1,0])A'
606:    V'[2] = ECDBL(V'[2]);       // = (2^(i+1))A'
607:    V'[0] = V'[d[i]];           // = (2^i*d[i])A' + (d[i-1,0])A' = (d[i,0])A'
608: }
609: V=(V'[0]x/r², V'[0]y/r³);     // V'[0]の乱数を取除きVとする      } ランダム化の解除
610: return V;                    // V = dAを出力                   }
}
```

図8C

```
ScalarMult_ES(A,d) {
701: r = random number(); // 乱数rを生成                                  } 座標Aデータのランダム化
702: d1=r; d2=d-r;  // rを用いてd=d1+d2を満たすランダムなd1,d2を生成       }

703: V1= ScalarMult_AddandDoubleAlways(A,d1) // V1 = d1Aを計算   } Add-and-Double-Always
704: V2= ScalarMult_AddandDoubleAlways(A,d2) // V2 = d2Aを計算   }

705: V = ECADD(V1, V2)          // V = V1+V2 = dA+(d-r)A=dA       } ランダム化の解除
706: return V;                  // V = dAを出力                   }
}
```

図9A

```
ScalarMult_PB(A,d) {
801: R = random_point();        // 楕円曲線上のランダムな点Rを生成          } 座標Aデータのランダム化
802: A1= ECSUB(A,R); A2=ECADD(A,R);                                       }
          // RからA=A1+A2を満たすランダムな点A1,A2を生成                   }

803: V1= ScalarMult_AddandDoubleAlways(A1, d)  // V1 = dA1を計算  } Add-and-Double-Always
804: V2= ScalarMult_AddandDoubleAlways(A2, d)  // V2 = dA2を計算  }

805: V = ECADD(V1, V2)          // V = V1+V2 = d(A1+A2)=dA        } ランダム化の解除
806: return V;                  // V = dAを出力                   }
}
```

図9B

【図１０】

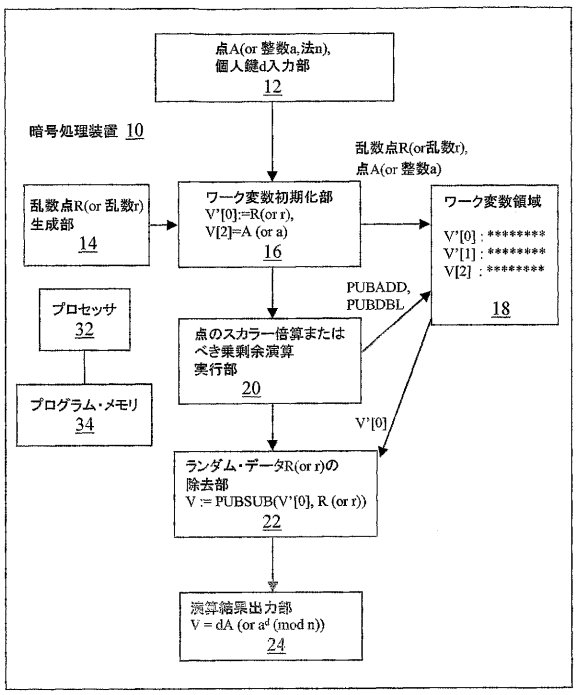


図10

【図１１】

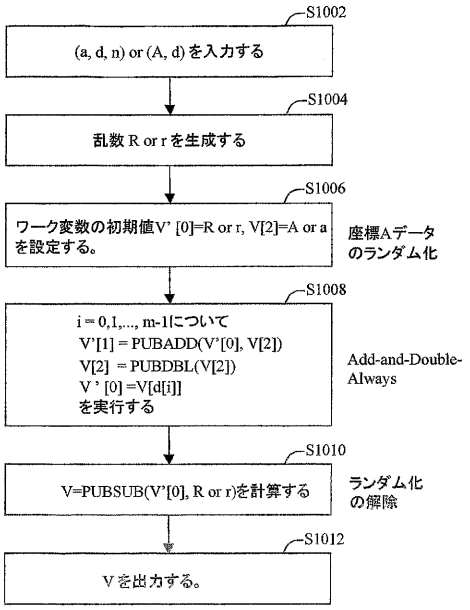


図11

【図１２】

```
ScalarMult_Randomized_Initvalue(A,d) {
1101: R = random_point();              // ランダムな点Rを生成          } V[0]の初期値
1102: V'[0]=R;                         // V'[0]の初期値をRとする。     } のランダム化
1103: V[2]=A;                          // V[2]の初期値をAとする。      } Add-and-
1104: for (i=0; i≦m−1; i++) {                                          } Double-
1105:   V'[1] = ECADD(V'[0],V[2]); // = 2^i A + (d[i−1,0])A+R          } Always
1106:   V[2] = ECDBL(V[2]);      // = (2^(i+1))A+R                     }
1107:   V'[0] = V'[d[i]];        // = (2^i*d[i])A+(d[i−1,0])A+R=(d[i,0])A+R }
1108: }
1109: V=ECSUB(V'[0], R);              // V=V'[0]−R=dAを計算する       } ランダム化
1110: return V;                       // V = dAを出力                 } の解除
}
```

図12

【図１３】

```
PowerMod_Randomized_Initvalue(a,d,n) {
1201: r = random_value();              // ランダムなrを生成            } V[0]の初期値
1202: V'[0]=r;                         // V'[0]の初期値をrとする。     } のランダム化
1203: V[2]=a;                          // V[2]の初期値をaとする。      } Add-and-
1204: for (i=0; i≦m−1; i++) {                                          } Double-
1205:   V'[1] = V'[0]*V[2] (mod n); // = a^(2^i) * a^(d[i−1,0]) * r (mod n) } Always
1206:   V[2] = V[2]^2 (mod n);       // = a^(2^(i+1)) * r (mod n)      }
1207:   V'[0] = V'[d[i]];    // = a^(2^i*d[i]) * a^(d[i−1,0]) * r
                             // = a^(d[i,0]) * r (mod n)               }
1208: }
1209: V=V'[0] * r^(−1) (mod n);      // V'[0]にrの逆元をかける       } ランダム化
1210: return V;                       // V =a^d (mod n)を出力         } の解除
}
```

図13

【図１４】

```
random_point() {
1301: R_X=random_Xcoordinate()    // ランダムなX座標R_Xを生成
1302: R_Y=XtoY(R_X)               // R_Xに対応するR_Yを計算
1303: R=(R_X,R_Y)                 // R=(R_X, R_Y)とする
1304: return R
}
```

図14

【図１５】

```
random_point() {
1401: s=random_small ()           // sは乱数
1402: R=ScalarMult_AddandDoubleAlways(s,G)
                                  // ベースポイントをs倍しRを生成
1403: return R
}
```

図15

【図１６】

```
random_point() {
1501: load R          // 以前使用したRを読み込む
                        Rの初期値は、カード初期化処理時に生成
1502: R=ECDBL(R)      // 以前のRを2倍し、新しいRを生成
1503: store R         // Rを格納する
}
```

図16

【図１７】

```
random_point() {
1601: load R=(R_X,R_Y,R_Z)  // 以前使用したRを読込む
                              Rの初期値は、カード初期化処理時に生成
1602: R=(t*R_X, t*R_Y, t*R_Z)  // 以前のRの座標値をt倍し、
                                  新しいRを生成 (tは定数)
1603: store R               // Rを格納する
}
```

図17

【図１８】

```
random_point() {
1601: load R=(R_X,R_Y,R_Z)  // 以前使用したRを読込む
                              Rの初期値は、カード初期化処理時に生成
1602: R=(t^2*R_X, t^3*R_Y, t*R_Z)  // 以前のRの座標値をt倍して、
                                      新しいRを生成。(tは定数)
1603: store R               // Rを格納する
}
```

図18

【図１９】

```
ECDBL(P) {      // V[2]=P=(X_i,Y_i,Z_i)=2^i A
  if (i==0) {   // d[i]のiと同じ値
    R_i := Z_0^2
    R_i := R_i^2
    R_i := aR_i     // aは楕円曲線パラメータ
  } else {
    R_i := R_i T_i-1   // i ≧ 1ならば、i−1の時の中間データT_i-1を利用
  }
  M_i := 3X_i^2 + R_i
  Q_i := Y_i^2
  S_i := 4X_i Q_i
  T_i := 8Q_i^2
  Z_i+1 := 2Y_i Z_i
  X_i+1 := M_i^2 − 2(S_i)
  Y_i+1 := (M_i)(S_i − X_i) − T_i
  P=(X_i+1, Y_i+1, Z_i+1)   // V[2]=P=2^(i+1)Aを出力
  return P;
}
```

図19

【図２０】

```
ScalarMult_Randomized_Initvalue(A,d) {
1701: load R          // 以前使用したRを読込み          } V[0]の初期値
1702: R=ECDBL(R) //  Rを2倍して更新                    } のランダム化
1703: store R         //  更新したRを格納               }
1704: V'[0]=R                    // V'[0]の初期値をRとする }
1705: V[2]=A;                    // V[2]の初期値をAとする  }
1706: for (i=0; i≦m−1; i++) {                            } Add-and-
1707:   V'[1] = ECADD(V'[0],V[2]); // = 2^i A + (d[i−1,0])A+R } Double-
1708:   V[2] = ECDBL(V[2]);      // V[2]=2^(i+1)を、2^k倍算法を用いて計算 } Always
1709:   V'[0] = V'[d[i]];        // = (2^i*d[i])A+(d[i−1,0])A+R=(d[i,0])A+R }
1710: }
1711: V=ECSUB(V'[0], R);          // V=V'[0]−R=dAを計算する       } ランダム化
1712: return V;                   // V = dAを出力                 } の解除
}
card.init() {                    //乱数点Rの初期値生成(カード初期化時に呼出し)
 1713: s = random()              //乱数sを生成
 1714: R = ScalarMult_AddandDoubleAlways(s,G) //ベースポイントGのs倍点を計算
 1715: store R                   // Rの値を格納
}
```

図20

フロントページの続き

(72)発明者　武仲　正彦
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　鳥居　直哉
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

審査官　金沢　史明

(56)参考文献　特表２００２－５４０４８４（ＪＰ，Ａ）
特開２０００－１８７４３８（ＪＰ，Ａ）
特開２０００－１３７４３６（ＪＰ，Ａ）
特開２００３－１７７６６８（ＪＰ，Ａ）
特開２００１－１９５５５５（ＪＰ，Ａ）
特開２００１－３３７５９９（ＪＰ，Ａ）
特表２００２－５４０４８３（ＪＰ，Ａ）
国際公開第０１／０２４４３９（WO，Ａ１）
特開２００３－０９８９６２（ＪＰ，Ａ）
伊藤孝一，伊豆哲也，武仲正彦，ランダム化初期点を用いた電力解析対策法，2004年暗号と情報セキュリティシンポジウム(SCIS2004)予稿集，電子情報通信学会，２００４年　1月２７日，2A5-2
伊藤孝一，伊豆哲也，武仲正彦，A Randomized Linearly-transformed Coordinate Countermeasure against Power Analysis，2004年暗号と情報セキュリティシンポジウム(SCIS2004)予稿集，電子情報通信学会，２００４年　1月２７日，2A5-3
伊豆哲也、伊藤孝一、武仲正彦、鳥居直哉，楕円曲線暗号向けサイドチャネル対策方式の比較評価，2003年暗号と情報セキュリティシンポジウム講演論文集，日本，電子情報通信学会，２００３年　1月２６日，8D-3

(58)調査した分野(Int.Cl.，ＤＢ名)
H04L　9/10
G09C　1/00
CiNii