(12) 特許協力条約に基づいて公開された国際出願

(19) 世界知的所有権機関
国際事務局

(43) 国際公開日
2011 年 10 月 6 日(06.10.2011)

PCT

(10) 国際公開番号
WO 2011/121661 A1

(51) 国際特許分類:
G09C 1/00 (2006.01)

(21) 国際出願番号: PCT/JP2010/002363

(22) 国際出願日: 2010 年 3 月 31 日(31.03.2010)

(25) 国際出願の言語: 日本語

(26) 国際公開の言語: 日本語

(71) 出願人 (米国を除く全ての指定国について): 富士通株式会社 (**FUJITSU LIMITED**) [JP/JP]; 〒2118588 神奈川県川崎市中原区上小田中４丁目１番１号 Kanagawa (JP).

(72) 発明者; および

(75) 発明者/出願人 (米国についてのみ): 伊藤孝一 (**ITOH, Kouichi**) [JP/JP]; 〒2118588 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 Kanagawa (JP). 山本大 (**YAMAMOTO, Dai**) [JP/JP]; 〒2118588 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 Kanagawa (JP). 武仲正彦 (**TAKENAKA, Masahiko**) [JP/JP]; 〒2118588 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 Kanagawa (JP).

(74) 代理人: 大菅義之 (**OSUGA, Yoshiyuki**); 〒1020084 東京都千代田区二番町８番地２０　二番町ビル３Ｆ Tokyo (JP).

(81) 指定国 (表示のない限り、全ての種類の国内保護が可能): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

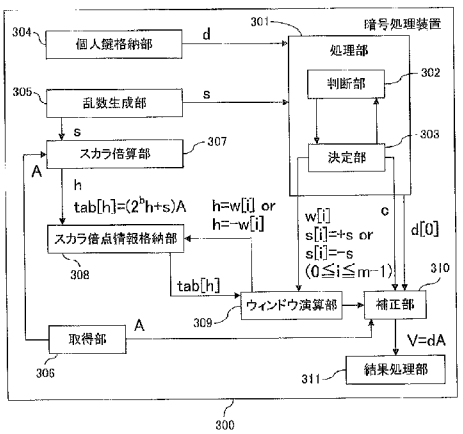
(84) 指定国 (表示のない限り、全ての種類の広域保護が可能): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), ユーラシア

[続葉有]

(54) **Title**: ENCRYPTION PROCESSING DEVICE AND ENCRYPTION PROCESSING METHOD

(54) 発明の名称 : 暗号処理装置および暗号処理方法

[図8]



$$D = g + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i])$$

300 ENCRYPTION PROCESSING DEVICE
301 PROCESSING UNIT
302 ASSESSMENT UNIT
303 DETERMINATION UNIT
304 PRIVATE KEY STORAGE UNIT
305 RANDOM NUMBER GENERATION UNIT
306 ACQUISITION UNIT
307 SCALAR MULTIPLIER UNIT
308 SCALAR MULTIPLE INFORMATION STORAGE UNIT
309 WINDOW CALCULATION UNIT
310 CORRECTION UNIT
311 RESULTS PROCESSING UNIT

(57) **Abstract**: Disclosed is an encryption processing device which includes a private key storage unit which stores a private key (d) in an elliptical curve cryptography system; a random number generator which generates a b-bit random number value (s); and a processing unit. The bit sequence (D) is obtained as a result of processing the private key (d) so that the value of the private key (d) or the most significant bit thereof is zero; and between the length (u) of the bit sequence (D), the window size (k) and the positive integer (m), the relation u = mk + b is satisfied. The processing unit defines, the window value (w[i]) for the signed k-bit value corresponding to each (i) in the range 0 ≤ i ≤ (m - 1); the random number value (s[i]) for the signed b-bit value corresponding to each (i); and the correction value (g). Further, the processing unit defines the abovementioned values under the constraint that the Equation 18 is satisfied while determining each random number value (s[i]) to be +s or -s.

(57) 要約: 暗号処理装置は、楕円曲線暗号における個人鍵ｄを格納する個人鍵格納部と、ｂビットの乱数値ｓを生成する乱数生成部と、処理部を含む。ビット列Ｄは、個人鍵ｄ、または最上位ビットの値が０になるように個人鍵ｄを加工して得られるビット列であり、ビット列Ｄの長さｕとウィンドウサイズｋと正整数ｍの間にｕ＝ｍｋ＋ｂなる関係が成立する。処理部は、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｋビット値のウィンドウ値ｗ［ｉ］と各ｉに対応する符号つきｂビット値の乱数値ｓ［ｉ］と補正値ｇを定める。なお、処理部は、各乱数値ｓ［ｉ］を＋ｓまたは－ｓに決定しながら、以下の式が成立するという制約条件下で、上記の値を定める。

WO 2011/121661 A1

(AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), ヨーロッパ (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

添付公開書類:

— 国際調査報告（条約第 21 条(3)）

明　細　書

発明の名称 ： 暗号処理装置および暗号処理方法

技術分野

[0001] 本発明は楕円曲線暗号に関する。

背景技術

[0002] 近年、情報セキュリティ技術の重要性がますます高まってきている。また、情報セキュリティの基盤技術の１つとして、公開鍵暗号（public-key cryptography）が盛んに研究されている。

[0003] 公開鍵暗号にはいくつか種類があり、指数剰余演算を利用するRivest, Shamir, Adleman（ＲＳＡ）アルゴリズムや、楕円曲線上の点のスカラ倍算を利用する楕円曲線暗号（Elliptical Curve Cryptography；ＥＣＣ）などが知られている。

[0004] 公開鍵暗号の利用においては、セキュリティの維持のために、個人鍵（private key）を秘密に保つことが重要である。ところが、近年では個人鍵を解読する（break）ための、いくつかの攻撃手法が知られている。よって、公開鍵暗号を用いた処理を行う機器が耐タンパ性である（tamper-proof）ためには、少なくとも既知の攻撃手法に対する対策が当該機器において実装されている必要がある。

[0005] 例えば、サイドチャネル攻撃の１種として、電力解析（Power Analysis；ＰＡ）攻撃と呼ばれる攻撃手法が知られている。また、ＰＡには、単純電力解析（Simple Power Analysis；ＳＰＡ）と電力差分解析（Differential Power Analysis；ＤＰＡ）の２種類がある。

[0006] よって、公開鍵暗号を用いた処理を行う機器には、ＳＰＡ攻撃に対する安全性とＤＰＡ攻撃に対する安全性が求められる。例えば、ＳＰＡ攻撃への対策の１つには「ウィンドウ法」と呼ばれる手法があり、ＤＰＡ攻撃への対策の１つにはデータをランダム化する手法がある。さらに、効率的な耐タンパ性の指数剰余演算および点のスカラ倍算を実現する暗号装置や、指数剰余演

算を実行する暗号化方法において、ＰＡ攻撃を用いた個人鍵の推定を困難にし、高い耐タンパ性をもつ暗号プロセッサも提案されている。

先行技術文献

特許文献

[0007] 特許文献1：特開２００３－２３３３０７号公報

特許文献2：国際公開ＷＯ２００９／１２２４６１号公報

発明の概要

発明が解決しようとする課題

[0008] 楕円曲線上の点のスカラ倍算を行う装置において、ＳＰＡ攻撃への対策として、ウィンドウサイズがｋビットのウィンドウ法（あるいはその変種）が採用される場合、メモリには、ｋビットの各インデックスに対応して楕円曲線上の点を示すデータが格納される。したがって、メモリ使用量はウィンドウサイズｋの指数オーダであり、ウィンドウサイズｋが大きいほど、メモリ使用量も多い。

[0009] 他方で、近年では、サーバ・コンピュータやパーソナル・コンピュータなどの汎用的なコンピュータだけではなく、例えば組み込み機器などの他の様々な装置においても、暗号技術の利用が広まりつつある。スカラ倍算を行う装置には、装置の種類によらず、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する対策を実装することが求められる。

[0010] ところが、ある種の装置のメモリ容量は、汎用的なコンピュータのメモリ容量と比べるとかなり少ない。そして、メモリ容量の少ない装置においては、なるべく少ないメモリ使用量で処理を実行することが好ましい。

[0011] そこで本発明は、メモリ使用量を少なく抑えつつ、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性を保って、スカラ倍算を行う暗号処理装置を提供することを目的とする。

課題を解決するための手段

[0012] 一態様による暗号処理装置は、個人鍵格納部と乱数生成部と処理部を備え

る。

前記個人鍵格納部は、楕円曲線暗号における個人鍵ｄを格納する。前記乱数生成部は、符号つきまたは符号なしのｂビットの乱数値ｓを生成する。

[0013] また、前記処理部は、前記個人鍵格納部から前記個人鍵ｄを読み出し、前記個人鍵ｄ、または最上位ビットの値が０になるように前記個人鍵ｄを加工して得られるビット列であるビット列Ｄの長さｕとウィンドウサイズｋとの間にｕ＝ｍｋ＋ｂなる関係が成立する正整数ｍに関して、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｋビット値のウィンドウ値ｗ［ｉ］と０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｂビット値の乱数値ｓ［ｉ］と補正値ｇを、前記ビット列Ｄと前記乱数値ｓとを用いて、各乱数値ｓ［ｉ］を＋ｓまたはーｓに決定しながら、

[数1]

$$D = g + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i])$$

が成立するという制約条件下で定める。

発明の効果

[0014] ウィンドウ値ｗ［ｉ］を用いたウィンドウ演算は、ＳＰＡ攻撃に対して安全なスカラ倍算を実現する。また、乱数値ｓ［ｉ］の導入により、ＤＰＡ攻撃に対する安全性も実現される。

[0015] そして、処理部が、各乱数値ｓ［ｉ］を、一律に乱数値ｓに定めるのではなく、上記制約条件下で適宜＋ｓまたはーｓに定めることで、ウィンドウ値ｗ［ｉ］の絶対値の範囲の縮小が実現される。したがって、上記暗号処理装置によれば、ウィンドウ演算用に各インデックスに対応して楕円曲線上の点を示すデータを格納するためのメモリの量は、2^kのオーダよりも少なくて済む。

[0016] よって、上記暗号処理装置は、少ないメモリ使用量で、ＳＰＡ攻撃とＤＰ

Ａ攻撃の双方に対する安全性を保ってスカラ倍算を行うことを可能にする。

図面の簡単な説明

[0017] [図1]ランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせる試みについて例示する図である。

[図2]図１の試みが失敗する理由を説明する図である。

[図3]第１～第３実施形態に共通するアプローチを説明する図である。

[図4]第１～第３実施形態でのアプローチを別の観点から示す図である。

[図5]ウィンドウ列の値が乱数列の値に依存して決まることを説明する図である。

[図6]第１～第３実施形態の暗号処理装置の第１のハードウェア構成例を示す図である。

[図7]第１～第３実施形態の暗号処理装置の第２のハードウェア構成例を示す図である。

[図8]第１～第３実施形態の暗号処理装置の機能構成を説明する図である。

[図9]第１～第３実施形態の暗号処理装置が、個人鍵と点からスカラ倍点を求める処理のフローチャートである。

[図10]第１～第３実施形態の暗号処理装置が、決定したウィンドウ列および乱数列、ならびに生成したスカラ倍点情報を利用して行う演算のフローチャートである。

[図11]第１実施形態において暗号処理装置がウィンドウ列と乱数列と補正値を決定する処理のフローチャートである。

[図12A]第１実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その１）である。

[図12B]第１実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その２）である。

[図13]ウィンドウ列と乱数列と補正値を示すとともに、スカラ倍点情報格納部がランダム化テーブルデータを保持するテーブルを示す図である。

[図14]ウィンドウサイズが３の場合のテーブルデータのエントリ数に関して

、第１実施形態と第３の比較例と第４の比較例を比べる図である。

[図15]第１実施形態においてスカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。

[図16]第１実施形態においてスカラ倍点情報格納部でインデックスとして使われる値についてまとめた図である。

[図17]第２実施形態においてスカラ倍点情報格納部でインデックスとして使われる値についてまとめた図である。

[図18]第２実施形態においてスカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。

[図19]第２実施形態において暗号処理装置がウィンドウ列と乱数列と補正値を決定する処理のフローチャートである。

[図20A]第２実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その１）である。

[図20B]第２実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その２）である。

[図21]ウィンドウ列と乱数列と補正値を示すとともに、スカラ倍点情報格納部がランダム化テーブルデータを保持するテーブルを示す図である。

[図22]ウィンドウサイズが３の場合のテーブルデータのエントリ数に関して、第２実施形態と第３の比較例と第４の比較例を比べる図である。

[図23]第３実施形態によるメモリ使用量の削減について説明する図である。

[図24]第３実施形態において暗号処理装置がウィンドウ列と乱数列と補正値を決定する処理のフローチャートである。

[図25A]第３実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その１）である。

[図25B]第３実施形態において決定される、ウィンドウ列と乱数列と補正値の具体例を示す図（その２）である。

[図26]ウィンドウ列と乱数列と補正値を示すとともに、スカラ倍点情報格納部がランダム化テーブルデータを保持するテーブルを示す図である。

[図27]ウィンドウサイズが３の場合のテーブルデータのエントリ数に関して、第３実施形態と第３の比較例と第４の比較例を比べる図である。

[図28]０以下の乱数値を生成するように変形された第１実施形態において、スカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。

[図29]０以下の乱数値を生成するように変形された第１実施形態において、インデックスとして使われる値についてまとめた図である。

[図30]０以下の乱数値を生成するように変形された第２実施形態において、スカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。

[図31]０以下の乱数値を生成するように変形された第２実施形態と、負の乱数値を生成するように変形された第３実施形態において、インデックスとして使われる値についてまとめた図である。

[図32]テーブルデータのエントリ数を比較する表である。

発明を実施するための形態

[0018] 以下、実施形態について、図面を参照しながら詳細に説明する。説明の順序は以下のとおりである。

後述の第１～第３実施形態の暗号処理装置は、楕円曲線上の点に対するスカラ倍算を行うためのデータ（具体的には、後述のウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃ）を生成し、生成したデータを用いてスカラ倍算を行う装置である。そこで、第１～第３実施形態についての理解を助けるために、まず、楕円曲線上の演算について説明する。また、第１～第３実施形態についての理解を助けるために、第１～第４の比較例についても説明する。

[0019] その後、本願の発明者が比較例についての検討から得た知見と、当該知見に基づいて発明者が開発した第１～第３実施形態の処理方法の共通点について説明する。また、第１～第３実施形態に共通の装置構成についても説明する。その後、第１～第３実施形態について順に説明し、最後に、その他の実

施形態などを説明する。

[0020] さて、楕円曲線上の演算について説明する。ＥＣＣで使われる楕円曲線のうちで主なものは、素体ＧＦ（ｐ）上で定義される式（１．１）の楕円曲線と、２の拡大体ＧＦ（2^n）上で定義される式（１．２）の楕円曲線である（なお、ＧＦはGalois fieldの略であり、ｐは素数である）。

$$y^2=x^3+\alpha x+\beta \pmod p,\ \text{ただし}\alpha,\ \beta,\ x,\ y\in GF(p) \qquad (1.1)$$

$$y^2+xy=x^3+\alpha x^2+\beta \pmod{F(x)}$$

ただし、α，β，x，$y\in GF(2^n)$かつF(x)は$GF(2^n)$の既約多項式 (1.2)

[0021] 式（１．１）で表される楕円曲線上の点Ｐは、式（１．１）を満たすｘとｙの組によりＰ＝（ｘ，ｙ）と表される。同様に、式（１．２）で表される楕円曲線上の点Ｐは、式（１．２）を満たすｘとｙの組によりＰ＝（ｘ，ｙ）と表される。また、楕円曲線上の特殊な点として無限遠点（point at infinity, or infinite point）が定義される。無限遠点を以下では「Ｏ」（大文字のオー）と表記する。

[0022] なお、本明細書の以下の議論は、式（１．１）の楕円曲線についても式（１．２）の楕円曲線についても同様に成り立つ。よって、以下では式（１．１）と（１．２）の区別について言及せずに、単に「楕円曲線」、「点Ｐ」、「点（ｘ，ｙ）」、「ｘ座標」、「ｙ座標」、「楕円曲線パラメタαおよびβ」などの表記を用いることがある。

[0023] 楕円曲線上の点同士に対して、ある演算（以下「加算」といい、「＋」と表す）を定義すると、楕円曲線上の点の集合は可換群をなすことが知られている。無限遠点Ｏは零元（すなわち加算における単位元）にあたる。また、楕円曲線上の任意の点Ｐ（点Ｐは無限遠点Ｏでもよい）に対して式（１．３）が成り立つ。

$$P+0=0+P=P \qquad (1.3)$$

[0024] 以下では点Ｐの逆元を－Ｐと表記する。点－Ｐのｘ座標とｙ座標は、楕円曲線が定義される体ＧＦ（ｐ）またはＧＦ（2^n）上の加算または減算により計算することができる。具体的には、点Ｐ＝（ｘ，ｙ）の逆元である点－Ｐ

は、楕円曲線が式（１．１）で定義される場合は式（１．４）により表され、楕円曲線が式（１．２）で定義される場合は式（１．５）により表される。

$$-P=(x, -y) \tag{1.4}$$

$$-P=(x, x+y) \tag{1.5}$$

[0025] 式（１．４）と（１．５）から理解されるように、点Ｐから点－Ｐを求めるための計算量（computational complexity）はわずかである。符号つきウィンドウ法や後述の第１～第３実施形態では、逆元の計算容易性を利用することで、メモリ使用量の削減が図られる。上記のような逆元の計算容易性は、ＲＳＡ暗号で利用される素体上の除算の計算困難性と対照的である。

[0026] なお、ある点$Ｐ_1$と点$Ｐ_2$に対して$Ｐ_3＝Ｐ_1＋Ｐ_2$と表される点$Ｐ_3$のｘ座標とｙ座標も、点$Ｐ_1$と$Ｐ_2$のｘ座標とｙ座標を用いて、体ＧＦ（ｐ）またはＧＦ（２^n）上の加減乗除により計算することができる。ここで、$Ｐ_2＝Ｐ_1$である場合には、$Ｐ_3＝Ｐ_1＋Ｐ_1$を$２Ｐ_1$とも表し、点$Ｐ_1$から点$Ｐ_3＝２Ｐ_1$を求める演算を２倍算（doubling）という。２倍算も、体ＧＦ（ｐ）またはＧＦ（２^n）上の加減乗除により実現される。

[0027] また、減算は、式（１．６）のように逆元の加算として定義される。

$$P_1-P_2=P_1+(-P_2) \tag{1.6}$$

[0028] さらに、楕円曲線上の点に対しては、スカラ倍算（scalar multiplication）と呼ばれる演算が定義される。スカラ倍算は、加算、減算、２倍算の組み合わせにより実現される。具体的には、スカラ値ｄと楕円曲線上の点Ｐに対して、Ｑ＝ｄＰと表現される点Ｑは、次の式（１．７）のように定義される。

[数2]

$$Q = dP = \underbrace{P + P + \cdots + P}_{d} = O + \underbrace{P + P + \cdots + P}_{d} \tag{1.7}$$

[0029] なお、任意の整数d_1とd_2と、楕円曲線上の任意の点Ｐに対して、次の式（１．８）と（１．９）が成立する。

$$d_1P+d_2P=(d_1+d_2)P \tag{1.8}$$

$$d_1(d_2P)=d_2(d_1P)=(d_1d_2)P \tag{1.9}$$

[0030] ＥＣＣにおいては、スカラ値が個人鍵として使われ、秘密にされる。逆に、楕円曲線上の「ベースポイント（base point）」と呼ばれるある点Ｇ、楕円曲線のパラメタαおよびβは、公開される情報である。個人鍵ｄに対して、公開鍵はＶ＝ｄＧを満たす点Ｖにより与えられる。

[0031] たとえ点ＧとＶが攻撃者に知られていても、点ＧとＶから個人鍵ｄを計算することは、莫大な計算量を必要とするので非常に困難である。この計算困難性は、離散対数問題の計算困難性として知られている。

[0032] ところで、ＥＣＣは、Diffie-Hellman（ＤＨ）アルゴリズムによる鍵共有（key agreement）や、ディジタル署名アルゴリズム（Digital Signature Algorithm；ＤＳＡ）などに利用可能である。何の目的でＥＣＣが利用されるにしろ、ＥＣＣを利用した処理はスカラ倍算を含む。ＤＨ鍵共有を例に説明すれば次のとおりである。

[0033] 例えば、第１の装置の個人鍵がd_Aであり、第２の装置の個人鍵がd_Bであるとする。すると、第１の装置の公開鍵Q_Aは、ベースポイントＧから$Q_A=d_AG$と計算され、第２の装置の公開鍵Q_Bは、ベースポイントＧから$Q_B=d$

$_{B}$Ｇと計算される。このように、公開鍵の生成のためにスカラ倍算が行われる。

[0034] また、第１の装置は自らの公開鍵Q_Aを第２の装置に送り、第２の装置は自らの公開鍵Q_Bを第１の装置に送る。すると、第１の装置はスカラ倍算によりd_AQ_Bを計算し、第２の装置もスカラ倍算によりd_BQ_Aを計算する。その結果、式（１．１０）に示すように、第１の装置と第２の装置は同じ鍵Ｋを共有することができる。

$$K=d_AQ_B=d_A(d_BG)=d_B(d_AG)=d_BQ_A \quad (1.10)$$

ＥＣＣが、上記に例示したＤＨ鍵共有以外の目的に利用される場合も、やはりスカラ倍算が行われる。

[0035] ところで、ＰＡ攻撃は、スカラ倍算を実行中の装置の消費電力を測定することで、個人鍵として使われるスカラ値ｄを解読しようとする、非破壊攻撃の１種である。よって、何の目的でＥＣＣが利用されるにしろ、個人鍵ｄの漏洩防止の対策としては、スカラ倍算を実行中の装置の消費電力波形が個人鍵ｄの特徴を示さないようにすることが有効である。

[0036] もしＰＡ攻撃に対して何の対策もとられないと、スカラ倍算を実行中の装置の消費電力波形の特徴から、個人鍵ｄが解読されてしまう危険性がある。具体的には、ＳＰＡ攻撃は、スカラ倍算を効率よく行うための演算手順に注目して、ある１回のスカラ倍算の処理中の消費電力波形から個人鍵ｄを解読しようとする攻撃手法である。また、ＤＰＡ攻撃は、複数の異なる点に対するスカラ倍算における電力波形の相違に注目して、個人鍵ｄを解読しようとする攻撃手法である。

[0037] 後述の第１～第３実施形態の暗号処理装置は、ＳＰＡ攻撃に対してもＤＰＡ攻撃に対しても安全であるように、かつ使用メモリ量を抑えるように、設計されている。そこで、第１～第３実施形態の利点についての理解を助けるため、続いて、いくつかの比較例について説明する。

[0038] まず、第１の比較例として、「バイナリ法」について説明する。バイナリ法は、ＳＰＡ攻撃に対してもＤＰＡ攻撃に対しても脆弱である。

例えば個人鍵ｄが１６０ビットである場合、ｄは非常に大きな数（例えば、2^{160}に近い数）である可能性がある。よって、式（１．７）の定義どおりにスカラ倍算を実行することは、非常に多くの回数の点の加算をともなうため、非現実的である。バイナリ法は、スカラ倍算の計算量のオーダを個人鍵ｄのビット数のオーダに抑えるための手法である。

[0039] ここで説明の便宜上、個人鍵ｄのビット長をｕとする。また、個人鍵ｄのｉビット目をｄ［ｉ］と表記する（０≦ｉ≦ｕ－１）。ｄ［０］が最下位ビット（Least Significant Bit；ＬＳＢ）でありｄ［ｕ－１］が最上位ビット（Most Significant Bit；ＭＳＢ）である。すると、式（２．１）より、式（２．２）が得られる。

[数3]

$$d=\sum_{i=0}^{u-1}2^{i}d[i] \tag{2.1}$$

$$dA=2^{u-1}d[u-1]A+\cdots+2^{1}d[1]A+2^{0}d[0]A \tag{2.2}$$

[0040] バイナリ法は式（２．２）を利用した計算手順である。例えば、個人鍵ｄが$(1100101)_2$の場合について具体的に説明すると、バイナリ法は式（２．３）にしたがってスカラ倍算を実現する手法である。

$$dA=2(2(2(2(2(2O+A)+A)))+A))+A=2^{6}A+2^{5}A+2^{2}A+A \tag{2.3}$$

[0041] すなわち、スカラ倍算の結果を変数Ｖで表すことにすると、バイナリ法では式（２．４）のとおり、変数Ｖがまず無限遠点Ｏにより初期化される。

$$V=O \tag{2.4}$$

[0042] その後、ＭＳＢからＬＳＢへと順に、「２倍算により２Ｖを求め、その後、もしｄ［ｉ］＝１ならば点Ａの加算を行い、得られた結果を変数Ｖに代入する」という処理が繰り返される。具体的には、ｄ［６］＝１なので、式（２．５）のとおり、６ビット目に対応して２倍算と加算が行われる。

$$V=2O+A \tag{2.5}$$

[0043] そして、ｄ［５］＝１なので、式（２．６）のとおり、５ビット目に対応して２倍算と加算が行われる。

$$V=2(2O+A)+A \tag{2.6}$$

[0044] また、ｄ［４］＝０なので、式（２．７）のとおり、４ビット目に関しては２倍算のみが行われ、加算は行われない。

$$V=2(2(2O+A)+A) \tag{2.7}$$

[0045] 同様に、ｄ［３］＝０なので、式（２．８）のとおり、３ビット目に関しても２倍算のみが行われ、加算は行われない。

$$V=2(2(2(2O+A)+A)) \tag{2.8}$$

[0046] 次の２ビット目に関しては、ｄ［２］＝１なので、式（２．９）のとおり、２倍算と加算が行われる。

$$V=2(2(2(2(2O+A)+A)))+A \tag{2.9}$$

[0047] また、次の１ビット目に関しては、ｄ［１］＝０なので、式（２．１０）のとおり、２倍算のみが行われ、加算は行われない。

$$V=2(2(2(2(2(2O+A)+A)))+A) \tag{2.10}$$

[0048] そして、最後の０ビット目に関しては、ｄ［０］＝１なので、式（２．１１）のとおり、２倍算と加算が行われる。

$$V=2(2(2(2(2(2(2O+A)+A)))+A))+A \tag{2.11}$$

[0049] 以上のようにしてｄ［ｉ］＝１であるｉビット目に対応して加算された点Ａの係数は、式（２．１１）から理解されるように、2^{i}である。よって、上記の式（２．４）～（２．１１）とともに例示した手順により、確かに式（２．３）にしたがって、Ｖ＝ｄＡが得られる。

[0050] 上記の例から明らかなとおり、バイナリ法によれば、２倍算の回数は個人鍵ｄのビット長ｕに等しく、加算の回数は個人鍵ｄのハミング重み（Hamming weight）に等しい。よって、バイナリ法によるスカラ倍算の計算量は、2^{u}のオーダではなく、ｕのオーダに抑えられる。

[0051] ところで、もし２倍算のときの消費電力波形と加算のときの消費電力波形が区別可能であるとすると、バイナリ法はＳＰＡ攻撃に対して脆弱である。

すなわち、バイナリ法の計算手順から、攻撃者は「２倍算の消費電力波形に続いて加算の消費電力波形が現れれば、ビット値ｄ［ｉ］は１である」と解析することができる。同様に、攻撃者は「２倍算の消費電力波形に続いて加算の消費電力波形が現れなければビット値ｄ［ｉ］は０である」と解析することができる。

[0052] また、バイナリ法はＤＰＡ攻撃に対しても脆弱である。なお、ＤＰＡ攻撃への安全性に関する説明の理解を助けるために、ＤＰＡ攻撃の概要を説明すれば次のとおりである。

すなわち、ＤＰＡ攻撃を行う攻撃者は、楕円曲線上のＬ個（Ｌ≧２）の既知の点A_1，A_2，…，A_Lに対してそれぞれ個人鍵ｄによるスカラ倍算が行われるときの消費電力波形を観察することにより、個人鍵ｄを解読する。以下では、点A_jに対応して観察された消費電力をPow_j（ｔ）と表記し、ＤＰＡ攻撃による個人鍵ｄの解読について説明する。なお、ｔは時刻情報を示す。

[0053] 説明の便宜上、１ビット以上の長さのビット列同士の連結（concatenation）を“||”という記号で表すことにする。すると、個人鍵ｄは式（３．１）のように表される。

$$d=d[u-1]||d[u-2]||\cdots||d[1]||d[0] \qquad (3.1)$$

式（２．３）～（２．１１）に例示したバイナリ法は、まず式（２．５）によりｄ［６］Ａを計算して変数Ｖに代入し、以下順次ビット数を増やしながら変数Ｖの値を更新していく方法である。すなわち、式（２．６）により（ｄ［６］||ｄ［５］）Ａが計算され、式（２．７）により（ｄ［６］||ｄ［５］||ｄ［４］）Ａが計算され、式（２．８）により（ｄ［６］||ｄ［５］||ｄ［４］||ｄ［３］）Ａが計算される。そして、式（２．９）により（ｄ［６］||ｄ［５］||ｄ［４］||ｄ［３］||ｄ［２］）Ａが計算され、式（２．１０）により（ｄ［６］||ｄ［５］||ｄ［４］||ｄ［３］||ｄ［２］||ｄ［１］）Ａが計算される。最後に、式（２０）によりｄＡが得られる。

[0054] 一般に、ハードウェアは、ロードまたはストアされるデータ値のハミング重みに応じた電力を消費する。そこで、攻撃者は、ＭＳＢ（すなわち（ｕ－１）ビット目）からＬＳＢ（すなわち０ビット目）まで順に、次の処理を行うことでビット値ｄ［ｉ］を解読する。

[0055] 攻撃者は、ビット値ｄ［ｉ］を予想する。そして、攻撃者は、既に解読した上位のビット値ｄ［ｕ－１］，…，ｄ［ｉ＋１］と、予想したビット値ｄ［ｉ］を用いて、１≦ｊ≦Ｌなる各ｊについて、式（３．２）で表される点B_jを計算する。

$$B_j=(d[u-1]||\cdots||d[i])A_j \qquad (3.2)$$

[0056] さらに、攻撃者は、消費電力Pow_j（ｔ）を、点B_jを表すデータ内の特定位置のビット値に応じて、２つの集合S_0とS_1のいずれかに分類する。以下、上記特定位置のビット値が０の集合をS_0とし、上記特定位置のビット値が１の集合をS_1とする。攻撃者は、集合S_1に属する消費電力Pow_j（ｔ）の平均から、集合S_0に属する消費電力Pow_j（ｔ）の平均を引いた差分波形Ｄｉｆｆ（ｔ）を求める。

[0057] もし、差分波形Ｄｉｆｆ（ｔ）にスパイクが出現していれば、攻撃者は、予想したビット値ｄ［ｉ］が正しいと判定する。逆に、差分波形Ｄｉｆｆ（ｔ）が平坦であれば、攻撃者は、予測したｄ［ｉ］の値が誤りだったと判定する。その結果、攻撃者は、ビット値ｄ［ｉ］を解読することができる。

[0058] このように、「ＭＳＢから順に、ビット数を増やしながら、（ｄ［ｕ－１］｜｜…｜｜ｄ［ｉ］）Ａの計算を行う」というバイナリ法の特徴を利用して、ＤＰＡ攻撃では、攻撃者がＭＳＢから順にビット値ｄ［ｉ］を解読してゆく。

[0059] 以上のようにバイナリ法はＳＰＡ攻撃にもＤＰＡ攻撃にも脆弱である。それに対し、次に第２の比較例として挙げる「ウィンドウ法」は、ＳＰＡ攻撃に対して安全である。

バイナリ法では、式（２．４）～（２．１１）に例示したように、個人鍵ｄの１ビットごとに、ビット値に応じて「２倍算と加算」または「２倍算」

という処理が行われる。それに対して、ウィンドウ法では、個人鍵ｄのｋビットごとに、ビット値によらず常に「ｋ回の２倍算と１回の加算」という処理が行われる。よって、たとえ２倍算の消費電力波形と加算の消費電力波形が異なっていたとしても、ウィンドウ法によるスカラ倍算はＳＰＡ攻撃に対して安全である。

[0060] 以下では説明の簡単化のため、個人鍵ｄのビット数ｕがウィンドウサイズｋで割り切れるものとする。すなわち、ｍ＝ｕ／ｋとすると、ｍは整数である。また、０≦ｉ≦（ｍ－１）なる各ｉについて、ｉ番目のウィンドウ値ｗ［i］は式（４．１）のように定義される。

$$w[i]=d[ik+k-1]||\cdots||d[ik] \quad (4.1)$$

[0061] なお、ｉ番目のウィンドウ値を表す“ｗ［ｉ］”という表記法における“［ｉ］”の意味は、個人鍵ｄのｉビット目を表す“ｄ［ｉ］”という表記法における“［ｉ］”の意味とは異なる。しかし、“［ｉ］”の意味は文脈から明らかなので、以下では適宜“ｗ［ｉ］”のような表記法を用いる。

[0062] 例えば、ウィンドウサイズｋが３ビットであり、個人鍵ｄが（０１１１１１０１）$_2$である場合、ウィンドウ値は次の式（４．２）～（４．４）のとおりである。以下では、ｗ［ｍ－１］，…，ｗ［１］，ｗ［０］のようなウィンドウ値の系列を「ウィンドウ列」（window sequence）ともいう。

$$w[2]=(011)_2=3 \quad (4.2)$$

$$w[1]=(111)_2=7 \quad (4.3)$$

$$w[0]=(101)_2=5 \quad (4.4)$$

[0063] また、ウィンドウ法では、スカラ倍算の対象として与えられる点Ａの座標を用いて、０≦ｈ≦2^k-1なる各ｈに対して、予め、スカラ倍点（scalar multiple）ｈＡが計算される。そして、計算されたスカラ倍点ｈＡは、インデックスｈと対応づけられてメモリに格納される。

[0064] 以下では、インデックスｈに対応づけられたスカラ倍点ｈＡをｔａｂ［ｈ］と表記し、ｔａｂ［ｈ］（＝ｈＡ）を「テーブルデータ」ともいう。より詳しくは、テーブルデータｔａｂ［ｈ］は、スカラ倍点ｈＡのｘ座標とｙ座

標の組により表される。

[0065] ウィンドウ法では、点ｄＡの計算は、式（４．５）のようにテーブルデータを用いて行われる。

$$dA=2^3(2^3(2^3(0)+tab[(011)_2])+tab[(111)_2])+tab[(101)_2] \quad (4.5)$$

[0066] より具体的には、スカラ値ｄと点Ａのスカラ倍算の結果を変数Ｖで表すことにすると、ウィンドウ法では、式（４．６）のとおり、変数Ｖがまず無限遠点Ｏにより初期化される。

$$V=0 \quad (4.6)$$

[0067] その後、ｉ＝ｍ－１からｉ＝０へと順に、「２倍算をｋ回（すなわち３回）行い、ｔａｂ［ｗ［ｉ］］を加算し、得られた結果を変数Ｖに代入する」という処理が行われる。すなわち、まずウィンドウ値ｗ［２］に対応して式（４．７）のとおり３回の２倍算と１回の加算が行われる。

$$V=2^3(0)+tab[(011)_2] \quad (4.7)$$

[0068] 次に、ウィンドウ値ｗ［１］に対応して式（４．８）のとおり３回の２倍算と１回の加算が行われる。

$$V=2^3(2^3(0)+tab[(011)_2])+tab[(111)_2] \quad (4.8)$$

[0069] そして、最後にウィンドウ値ｗ［０］に対応して式（４．９）のとおり３回の２倍算と１回の加算が行われる。

$$V=2^3(2^3(2^3(0)+tab[(011)_2])+tab[(111)_2])+tab[(101)_2] \quad (4.9)$$

[0070] 以上のように、ウィンドウ法によれば、個人鍵ｄに含まれるビット値の値によらず、同じ種類の処理が行われる。よって、ウィンドウ法はＳＰＡ攻撃に対して安全である。

[0071] 続いて、第３の比較例として、ＳＰＡ攻撃に対してのみならずＤＰＡ攻撃に対しても安全となるようにウィンドウ法を改良した「ランダム化ウィンドウ法」について説明する。ランダム化ウィンドウ法では、ｂビットの乱数値ｓによってテーブルデータがランダム化される。ランダム化により、データの内容と消費電力の相関関係が隠される。すなわち、ランダム化により、攻撃者によるビット値の予測の正誤に応じて差分波形Ｄｉｆｆ（ｔ）が変化す

ることはなくなる。したがって、ランダム化により、ＤＰＡ攻撃に対する安全性が実現される。

[0072] なお、乱数値ｓのビット数ｂは、例えば３０以下が好ましい。また、ｍを整数とし、個人鍵ｄのビット数をｕとすると、式（５．１）が成り立つものとする。

$$u=b+km \tag{5.1}$$

[0073] 例えば、個人鍵ｄが３７８＝（１０１１１１０１０）$_2$であるとすると、ｕ＝９である。また、ウィンドウサイズｋが２であり、ｂが３であり、乱数値ｓが３＝（０１１）$_2$であるとする。この場合、式（５．１）よりｍ＝３である。ランダム化ウィンドウ法では、０≦ｉ≦（ｍ－１）なる各ｉについてｋビットのウィンドウ値ｗ［ｉ］が計算され、さらに、式（５．２）が成立するようにｂビットの補正値ｃが計算される。

[数4]

$$d = c + \sum_{i=0}^{m-1} 2^{ki}(w[i] \| s) \tag{5.2}$$

[0074] ここで、乱数値ｓはｂビットなので、式（５．２）から式（５．３）が得られる。

[数5]

$$d = c + \sum_{i=0}^{m-1} \left(2^{ki+b} w[i] + 2^{ki} s\right) \tag{5.3}$$

[0075] そして、式（５．３）から移項により式（５．４）が得られる。

[数6]

$$d - \sum_{i=0}^{m-1} 2^{ki} s = c + \sum_{i=0}^{m-1} 2^{ki+b} w[i] \tag{5.4}$$

[0076] 式（５．４）の左辺は、個人鍵ｄと乱数値ｓから計算される値を示す。そして、式（５．４）の右辺は、式（５．４）の左辺により計算される値を最

上位ビットからｋビットずつ区切ることでウィンドウ列が得られることと、式（５．４）の左辺により計算される値の最下位のｂビットが補正値ｃとなることを示す。

[0077] 例えば、上記のとおりｄ＝３７８＝（１０１１１１０１０）$_2$であり、乱数値ｓが３＝（０１１）$_2$である場合、式（５．４）の左辺を計算すると、式（５．５）のとおりである。

$$378-(2^0\times 3+2^2\times 3+2^4\times 3)=378-(3+12+48)=315=(100111011)_2 \quad (5.5)$$

[0078] よって、式（５．５）により得られた値（１００１１１０１１）$_2$を２ビットずつ区切ることでウィンドウ列ｗ［２］、ｗ［１］、ｗ［０］が得られる。また、この値（１００１１１０１１）$_2$の最下位のｂ（＝３）ビットから補正値ｃが得られる。具体的には、式（５．６）～（５．９）のとおりである。

$$w[2]=(10)_2=2 \quad (5.6)$$

$$w[1]=(01)_2=1 \quad (5.7)$$

$$w[0]=(11)_2=3 \quad (5.8)$$

$$c=(011)_2=3 \quad (5.9)$$

[0079] ランダム化ウィンドウ法では、以上のようにしてウィンドウ値ｗ［ｉ］（０≦ｉ≦ｍ－１）と補正値ｃが計算される。また、ランダム化ウィンドウ法ではさらに、スカラ倍算の対象である点Ａの座標を用いて、０≦ｈ≦2^k-1なる各ｈに対して、予め、インデックスｈと対応づけて式（５．１０）のテーブルデータが計算され、メモリに格納される。

$$tab[h]=(2^b h+s)A \quad (5.10)$$

[0080] 例えば、上記のとおりｂ＝３でありｓ＝（０１１）$_2$＝３である場合、メモリにはテーブルデータとして次の式（５．１１）～（５．１４）のデータが格納される。

$$tab[(00)_2]=tab[0]=(2^3\times 0+3)A=3A \quad (5.11)$$

$$tab[(01)_2]=tab[1]=(2^3\times 1+3)A=11A \quad (5.12)$$

$$tab[(10)_2]=tab[2]=(2^3\times 2+3)A=19A \quad (5.13)$$

$$tab[(11)_2]=tab[3]=(2^3\times 3+3)A=27A \quad (5.14)$$

[0081] そして、点ｄＡの計算は、式（５．１５）にしたがって行われる。

$$\begin{aligned} dA &= 2^2(2^2(2^2(0)+tab[w[2]])+tab[w[1]])+tab[w[0]]+cA \\ &= 4(4(4(0)+19A)+11A)+27A+3A \\ &= 4(4(0+19A)+11A)+27A+3A \\ &= 4(4(19A)+11A)+27A+3A \\ &= 4(76A+11A)+27A+3A \\ &= 4(87A)+27A+3A \\ &= 348A+27A+3A \\ &= 378A \end{aligned} \quad (5.15)$$

[0082] 式（５．２）が満たされるように、式（５．５）により式（５．９）の補正値ｃが算出されているので、式（５．１５）の計算により確かにｄＡ（すなわちこの例では３７８Ａ）が得られる。

[0083] また、式（５．１５）は、個人鍵ｄのビット値によらず、「２倍算をｋ回（上記の例ではｋ＝２）行い、加算を１回行う」という処理がｍ回（上記の例ではｍ＝３）繰り返され、補正値ｃを用いた１回のスカラ倍算と点ｃＡの加算が行われることを示す。よって、ランダム化ウィンドウ法は、ウィンドウ法と同様にＳＰＡ攻撃に対して安全である。また、テーブルデータが乱数値ｓによってランダム化されているので、ランダム化ウィンドウ法は、ＤＰＡ攻撃に対しても安全である。

[0084] ところで、ウィンドウ法でもランダム化ウィンドウ法でも、ウィンドウサイズｋに応じて、2^k個のエントリがテーブル内に作成される。他方で、組み込み機器など、ある種の装置では、メモリ容量が少ないため、各種の処理を行うためのメモリ使用量を少なくすることが望ましい。

[0085] 組み込み機器の１つの例はスマートカードである。また、組み込み機器の他の例は、プリンタなどの電子機器により認証される部品である。例えば、偽造品排除のために、二次電池あるいはプリンタカートリッジなどのアクセサリ部品に、「認証チップ」と呼ばれるLarge Scale Integration（ＬＳＩ）

が組み込まれていてもよい。プリンタによるプリンタカートリッジの認証は、例えば、正規品以外のプリンタカートリッジでの印刷を不能にするために行われてもよい。もちろん、組み込み機器にはその他の様々な種類がある。

[0086] 例えば、スマートカードや認証機能つきプリンタカートリッジなどの装置では特に、メモリ搭載量が少ない。よって、スカラ倍算に関しても、メモリ使用量を抑えた処理アルゴリズムが好ましい。

[0087] そこで、第４の比較例として、「符号つき（signed）ウィンドウ法」について次に説明する。符号つきウィンドウ法は、ウィンドウ法と同様にＳＰＡ攻撃に対して安全であり、かつウィンドウ法よりもメモリ使用量が少ない方法である。

[0088] 具体的には、ウィンドウサイズをｋとすると、ウィンドウ法ではテーブルデータのエントリ数が2^k個であるのに対し、符号つきウィンドウ法ではテーブルデータのエントリ数が$2^{k-1}+1$個である。よって、符号つきウィンドウ法のメモリ使用量は、ウィンドウ法のメモリ使用量の約半分である。同様に、符号つきウィンドウ法のメモリ使用量は、ランダム化ウィンドウ法のメモリ使用量の約半分である。

[0089] 符号つきウィンドウ法は、「楕円曲線上の点Ｐから逆元である点－Ｐを求める計算の処理負荷は、比較的軽い（つまり逆元の計算コストが低い）」ということに注目した方法である（式（１．４）および（１．５）、ならびにその説明を参照）。符号つきウィンドウ法におけるテーブルデータのインデックスｈは、具体的には$0 \leqq h \leqq 2^{k-1}$である。

[0090] 例えば、ウィンドウサイズｋが３であるとする。すると、符号つきウィンドウ法で使われるインデックスは０、１、２、３、４の５つである。そして、これらの５つのインデックスに対応する対応するテーブルデータは、０Ａ、１Ａ、２Ａ、３Ａ、４Ａである。符号つきウィンドウ法では、次の式（６．１）～（６．３）の関係を利用することにより、３つのインデックス５、６、７に対応するテーブルデータ５Ａ、６Ａ、７Ａが省略される。

$$5A=2^kA-3A=8A-3A \qquad (6.1)$$

$$6A=2^kA-2A=8A-2A \quad (6.2)$$

$$7A=2^kA-1A=8A-1A \quad (6.3)$$

[0091] 例えば、ウィンドウサイズｋが３ビットであり、個人鍵ｄが２４１＝（０１１１１０００１）$_2$であるとする。符号つきウィンドウ法では、まず、ウィンドウ法と同様にして仮のウィンドウ値が求められる。すなわち、仮のウィンドウ値は次の式（６．４）～（６．６）のとおりである。なお、特に混乱のおそれはないので、以下では、仮のウィンドウ値と確定した実際のウィンドウ値をともにｗ［ｉ］と表記する。

$$w[2]=(011)_2=3 \quad (6.4)$$

$$w[1]=(110)_2=6 \quad (6.5)$$

$$w[0]=(001)_2=1 \quad (6.6)$$

[0092] そして、仮のウィンドウ値は最下位（すなわちｗ［０］）から順にスキャンされ、次のようにして各ウィンドウ値が確定される。すなわち、もしｗ［ｉ］≧2^{k-1}＋１であれば、仮のウィンドウ値ｗ［ｉ］から2^kを引いた値（すなわちｗ［ｉ］－2^k）がウィンドウ値ｗ［ｉ］として設定される。また、ｉ番目の仮のウィンドウ値ｗ［ｉ］から2^kの減算が行われた場合は、減算の影響をキャンセルするため、１つ上位の仮のウィンドウ値ｗ［ｉ＋１］には、１が足される。

[0093] なお、以下では上記の2^kの減算を「ウィンドウ補正」といい、上記の１の加算を「キャリー（carry）補正」という。また、上記のｗ［ｉ］≧2^{k-1}＋１という条件の代わりに、ｗ［ｉ］≧2^{k-1}という条件を用いてもよいが、以下では説明の便宜上、ｗ［ｉ］≧2^{k-1}＋１という条件が用いられるものとする。

[0094] 式（６．４）～（６．６）の仮のウィンドウ値から、実際のウィンドウ値は、次のようにして求められる。

すなわち、式（６．６）よりｗ［０］＜2^{k-1}＋１が成立する。よって、０番目（つまり最下位）のウィンドウ値ｗ［０］は仮のウィンドウ値と同じく１である。

[0095] また、式（６．５）よりｗ［１］≧2^{k-1}＋１が成立する。よって、１番目のウィンドウ値は、仮のウィンドウ値６から８（＝2^{k}）を引くことで得られ、ｗ［１］＝－２と確定する。よって、キャリー補正により、式（６．４）の仮のウィンドウ値ｗ［２］には１が足され、ｗ［２］＝４となる。

[0096] そして、このキャリー補正された仮のウィンドウ値ｗ［２］＝４は、ｗ［２］＜2^{k-1}＋１を満たす。よって、２番目（つまり最上位）のウィンドウ値はｗ［２］＝４と確定する。

[0097] 以上のようにして確定したウィンドウ値ｗ［０］～ｗ［２］を用いて、符号つきウィンドウ法では式（６．７）により点ｄＡが計算される。

$$dA=2^3(2^3(2^3(0)+tab[w[2]])-tab[-w[1]])+tab[w[0]] \quad (6.7)$$

[0098] より具体的には、スカラ倍算の結果を変数Ｖで表すことにすると、式（６．８）のとおり、変数Ｖがまず無限遠点Ｏにより初期化される。

$$V=0 \quad (6.8)$$

[0099] その後、ｉ＝ｍ－１からｉ＝０へと順に、「２倍算をｋ回行い、ウィンドウ値ｗ［ｉ］が０以上ならｔａｂ［ｗ［ｉ］］を加算し、ウィンドウ値ｗ［ｉ］が負ならｔａｂ［－ｗ［ｉ］］を減算し、得られた結果を変数Ｖに代入する」という処理が行われる。なお、ｍは、個人鍵ｄのビット長ｕをウィンドウサイズｋで割った値であり、本例ではｍ＝３である。

[0100] 式（６．８）の初期化に続いて、ｉ＝ｍ－１に対応する処理が行われる。すなわち、ウィンドウ値ｗ［２］（＝４）に対応して、式（６．９）のとおり３回の２倍算と１回の加算が行われる。

$$V=2^3(0)+tab[w[2]] \quad (6.9)$$

[0101] 次に、ウィンドウ値ｗ［１］（＝－２）に対応して、式（６．１０）のとおり３回の２倍算と１回の減算が行われる。

$$V=2^3(2^3(0)+tab[w[2]])-tab[-w[1]] \quad (6.10)$$

[0102] そして、最後にウィンドウ値ｗ［０］（＝１）に対応して、式（６．１１）のとおり３回の２倍算と１回の減算が行われる。

$$V=2^3(2^3(2^3(0)+tab[w[2]])-tab[-w[1]])+tab[w[0]] \quad (6.11)$$

[0103] 式（６．１１）の右辺を展開すると、下記の式（６．１２）のとおりである。また、上記のとおり本例において個人鍵ｄは２４１である。よって、以上説明した符号つきウィンドウ法により点ｄＡが正しく計算されることが理解されるであろう。

$$V=2^3(2^3(2^3(0)+tab[4])-tab[2])+tab[1]$$
$$=2^3(2^3(4A)-2A)+1A$$
$$=8(32A-2A)+1A$$
$$=241A \qquad (6.12)$$

[0104] なお、符号つきウィンドウ法は、ウィンドウ法と同様の理由から、ＳＰＡ攻撃に対しては安全であるが、ＤＰＡ攻撃に対しては脆弱である。

[0105] さて、以上のとおり第１～第４の比較例について説明したが、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対して安全なランダム化ウィンドウ法はメモリ消費量が比較的多く、メモリ消費量の少ない符号つきウィンドウ法はＤＰＡ攻撃に対しては脆弱である。すなわち、上記の４つの比較例の中には、「ＳＰＡ攻撃とＤＰＡ攻撃の双方に対して安全で、かつメモリ使用量が少ない」という特徴を持つ手法は存在しない。

[0106] 他方、サイドチャネル攻撃の１種であるＰＡ攻撃は、組み込み機器に対しても行われる危険性が十分にあり、また、組み込み機器の中には、何らかの理由からメモリ容量が限られたものがある。よって、例えば組み込み機器のようにメモリ容量の小さな装置におけるスカラ倍算の処理は、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対して安全で、かつメモリ使用量が少ないことが好ましい。

[0107] しかしながら、本願発明者が研究したところ、「ＳＰＡ攻撃とＤＰＡ攻撃の双方に対して安全で、かつメモリ使用量が少ない」という特徴は、単純素朴にランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせることでは得られないことが判明した。むしろ、発明者は「ランダム化ウィンドウ法と符号つきウィンドウ法は、単純素朴に組み合わせることは不可能である」という知見を得た。この知見をもう少し詳しく述べると、次のとおりである

。

[0108] 乱数の使用は、符号つきウィンドウ法の単純な適用を阻害する。そのため、ＳＰＡ攻撃への対策としてランダム化ウィンドウ法が採用されると、符号つきウィンドウ法の単純な適用によるメモリ使用量の削減は不可能となる。つまり、「符号つきウィンドウ法とランダム化ウィンドウ法の単純素朴な組み合わせによって、少ないメモリ使用量で、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性を確保しよう」という試みは、うまくいかない。

[0109] 上記知見は、後述の第１～第３実施形態について理解するうえで有益なので、以下、上記知見について詳しく説明する。

図１は、ランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせる試みについて例示する図である。以下に説明するとおり、図１に例示する試みは失敗に終わる。

[0110] 図１の例における個人鍵ｄは、式（７．１）に示す２３ビットの値である。

$$d=(01001010110100011011011)_2 \tag{7.1}$$

また、図１の例における乱数値ｓは、式（７．２）に示す８ビットの値である。

$$s=(10001101)_2 \tag{7.2}$$

[0111] したがって、図１の例ではｕ＝２３でありｂ＝８である。また、ウィンドウサイズｋが３であるとする。よって、式（５．１）の整数ｍ（すなわちウィンドウ列に含まれるウィンドウ値の数）は、式（７．３）より５である。

$$m=(u-b)/k=(23-8)/3=5 \tag{7.3}$$

[0112] 上記のとおりランダム化ウィンドウ法では、式（５．２）が満たされるように、式（５．４）にしたがってウィンドウ値ｗ［ｉ］（０≦ｉ≦ｍ－１）と補正値ｃが決定される。そこで、図１の例でも、式（５．４）にしたがってランダム化ウィンドウ法と同様に、仮にウィンドウ値ｗ［ｉ］と補正値ｃが計算されると仮定する。すると、式（５．４）の左辺の計算結果は式（７．４）のとおりであるから、図１および式（７．５）～（７．１０）に示し

たように仮のウィンドウ値ｗ［ｉ］と補正値ｃが得られる。

$$d-(2^0s+2^3s+2^6s+2^9s+2^{12}s)=(00110110101011010100110)_2 \quad (7.4)$$

$$w[4]=(001)_2=1 \quad (7.5)$$

$$w[3]=(101)_2=5 \quad (7.6)$$

$$w[2]=(101)_2=5 \quad (7.7)$$

$$w[1]=(010)_2=2 \quad (7.8)$$

$$w[0]=(110)_2=6 \quad (7.9)$$

$$c=(10100110)_2 \quad (7.10)$$

[0113] 続いて、上記の式（７．５）～（７．９）のように得られた各ウィンドウ値ｗ［ｉ］に対して、符号つきウィンドウ法と同様にして、ウィンドウ補正とキャリー補正が行われると仮定する。図１の例ではウィンドウサイズｋが３なので、ｗ［ｉ］≧５（＝2^{k-1}＋１）のときにウィンドウ補正により仮のウィンドウ値ｗ［ｉ］から８（＝2^k）が引かれ、ウィンドウ値ｗ［ｉ＋１］に対してキャリー補正が行われる。

[0114] すなわち、最下位のウィンドウ値ｗ［０］がｗ［０］≧2^{k-1}＋１の場合、ウィンドウ補正によりウィンドウ値ｗ［０］からは2^kが引かれ、１つ上位のウィンドウ値ｗ［１］に対してキャリー補正が行われる。最下位以外のウィンドウ値ｗ［ｉ］（１≦ｉ≦ｍ－１）に関しては、仮のウィンドウ値ｗ［ｉ］とキャリー補正値の合計が2^{k-1}＋１以上の場合に、ウィンドウ補正が行われ、１つ上位のウィンドウ値ｗ［ｉ＋１］に対してキャリー補正が行われる。

[0115] 具体的には、図１に示すように、まず最下位の仮のウィンドウ値ｗ［０］＝６（≧５）に対してウィンドウ補正が行われ、ウィンドウ値ｗ［０］は式（７．１１）のように確定する。

$$w[0]=6-8=-2 \quad (7.11)$$

[0116] そして、次のウィンドウ値ｗ［１］に対してキャリー補正により１が足され、ウィンドウ値ｗ［１］は式（７．１２）のように確定する。なお、３＜５なのでウィンドウ値ｗ［１］に対するウィンドウ補正は行われない。

w[1]=2+1=3 (7.12)

[0117] 続いて、次のウィンドウ値ｗ［２］＝５（≧５）に対してウィンドウ補正が行われ、ウィンドウ値ｗ［２］は式（７．１３）のように確定する。

w[2]=5-8=-3 (7.13)

[0118] そして、さらに次のウィンドウ値ｗ［３］に対して、キャリー補正により１が足される。キャリー補正されたウィンドウ値ｗ［３］は、５以上であるから、ウィンドウ補正の対象となる。したがって、ウィンドウ値ｗ［３］は式（７．１４）のように確定する。

w[3]=5+1-8=-2 (7.14)

[0119] そして、次の（すなわち最上位の）ウィンドウ値ｗ［４］に対してはキャリー補正により１が足される。なお、キャリー補正されたウィンドウ値ｗ［４］は５未満なので、ウィンドウ補正の対象ではなく、結局、ウィンドウ値ｗ［４］は式（７．１５）のように確定する。

w[4]=1+1=2 (7.15)

[0120] 図１には、以上のようにして確定した式（７．１１）～（７．１５）のウィンドウ値ｗ［０］～ｗ［４］と、式（７．１０）の補正値ｃが示されている。以上のようにして得られたウィンドウ値ｗ［ｉ］の絶対値は、上記の例からも分かるとおり、$２^{ｋ-１}$以下である。

[0121] しかしながら、第４の比較例として挙げた符号つきウィンドウ法とは異なり、ランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせようとした図１の例では、たとえすべてのウィンドウ値の絶対値が$２^{ｋ-１}$以下でも、テーブルデータの削減ができない。すなわち、「ランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせることで、耐タンパ性とメモリ使用量の抑制を両立させよう」という試みは失敗する。その理由について、図２を参照しながら以下に説明する。

[0122] 図２は、図１の試みが失敗する理由を説明する図である。図２は、図１と同様に、ウィンドウサイズｋが３であり乱数値ｓのビット長ｂが８の場合の例を示す。図２において、テーブル１０１は符号つきウィンドウ法に対応し

、テーブル１０２は図１の試みに対応する。

[0123] 符号つきウィンドウ法によれば、$2^{k-1}+1$以上のウィンドウ値からは、ウィンドウ補正により2^kが引かれる。例えば、ｋ＝３の場合、５以上のウィンドウ値からは８が引かれる。

[0124] よって、必要に応じてキャリー補正されたウィンドウ値が５ならば、ウィンドウ補正されたウィンドウ値は－３となる。そして、式（１．４）と（１．５）に示したように、楕円曲線上のある点からその逆元の点を得るための計算量は少ないので、スカラ倍点－３Ａは、スカラ倍点３Ａから少ない計算量で容易に求められる。

[0125] よって、符号つきウィンドウ法によれば、テーブル１０１は、５というインデックスに対応づけてスカラ倍点５Ａのテーブルデータを保持する必要もないし、５というインデックスに対応づけてスカラ倍点－３Ａのテーブルデータを保持する必要もない。すなわち、符号つきウィンドウ法では、図２のテーブル１０１のように、テーブル１０１においてインデックスが５のエントリは省略可能であり、インデックスが３のエントリがその代わりに使われる。

[0126] 同様に、符号つきウィンドウ法によれば、テーブル１０１に示すようにインデックスが６と７のエントリも省略可能であり、インデックスが２と１のエントリがその代わりに使われる。その結果、符号つきウィンドウ法によれば、テーブル１０１は、2^k個のエントリを保持する必要がない。すなわち、テーブル１０１は、インデックスが2^{k-1}以下のエントリさえ保持すればよい。よって、符号つきウィンドウ法ではウィンドウ法（あるいはランダム化ウィンドウ法）と比べてメモリ使用量が少ない。

[0127] それに対して、図１の試みに対応するテーブル１０２では、インデックスが$2^{k-1}+1$以上のエントリも省略不可能であり、ウィンドウ法（あるいはランダム化ウィンドウ法）と比較してメモリ使用量を少なくすることは不可能である。その理由は次のとおりである。

[0128] テーブル１０２において、インデックスが１のエントリのテーブルデータ

ｔａｂ［１］は、（１×２^{8}＋ｓ）Ａである。よって、テーブルデータｔａｂ［１］の減算は、点（－１×２^{8}－ｓ）Ａの加算を意味する。同様に、テーブル１０２におけるテーブルデータｔａｂ［２］の減算は、点（－２×２^{8}－ｓ）Ａの加算を意味し、テーブル１０２におけるテーブルデータｔａｂ［３］の減算は、点（－３×２^{8}－ｓ）Ａの加算を意味する。

[0129] それに対して、図１の試みでは、ウィンドウ補正やキャリー補正により初期ウィンドウ値が補正されることはあるが、乱数値ｓは一定である。そのため、例えば、必要に応じてキャリー補正が行われた後のウィンドウ値が５の場合、ウィンドウ値は－３に補正されるが、こうして－３に補正されたウィンドウ値に対応するのは点（－３×２^{8}＋ｓ）Ａの加算である。

[0130] そして、点（－３×２^{8}＋ｓ）Ａは、インデックスが３のエントリのテーブルデータが表す点の逆元（－３×２^{8}－ｓ）Ａではない。つまり、式（－３×２^{8}＋ｓ）Ａと（－３×２^{8}－ｓ）Ａでは、乱数値ｓの符号が異なる。

[0131] よって、点（－３×２^{8}＋ｓ）Ａは、インデックスが３のエントリのテーブルデータからわずかな計算量で計算される点ではない。よって、テーブル１０２は、ウィンドウ補正された－３というウィンドウ値に対応するテーブルデータとして、スカラ倍算点（－３×２^{8}＋ｓ）Ａを保持する必要がある。すなわち、インデックス５に対応するエントリをテーブル１０２から省略することはできない。

[0132] 同様に、点（－２×２^{8}＋ｓ）Ａは、インデックスが２のエントリのテーブルデータｔａｂ［２］＝（２×２^{8}＋ｓ）Ａからわずかな計算量で計算可能な点（－２×２^{8}－ｓ）Ａとは異なる。したがって、インデックス６に対応するエントリをテーブル１０２から省略することはできない。

[0133] また、点（－１×２^{8}＋ｓ）Ａは、インデックスが１のエントリのテーブルデータｔａｂ［１］＝（１×２^{8}＋ｓ）Ａからわずかな計算量で計算可能な点（－１×２^{8}－ｓ）Ａとは異なる。したがって、インデックス７に対応するエントリをテーブル１０２から省略することもできない。

[0134] 以上のとおり、結局、テーブル１０２のエントリ数を削減することはでき

ない。したがって、耐タンパ性とメモリ使用量の抑制を両立させようとする図１の試みは失敗する。換言すれば、ランダム化ウィンドウ法における乱数の使用は、符号つきウィンドウ法の単純な適用を阻害する要素であるから、ランダム化ウィンドウ法と符号つきウィンドウ法を単純に組み合わせようという図１の試みは失敗するのである。

[0135] なお、図２を参照して以上説明した事柄をより一般的に述べれば下記のとおりである。

図１の試みでは、ランダム化ウィンドウ法と同様に、式（５．２）を満たすように式（５．４）により初期ウィンドウ値と補正値が求められる。また、上記のとおり式（５．２）は式（５．３）のように書き換えられる。そして、式（５．３）の括弧の中は式（７．１６）のように書き換えられる。式（７．１６）の右辺は、インデックスｈに対応するエントリのテーブルデータｔａｂ［ｈ］が（２^{b}ｈ＋ｓ）Ａであることに対応している。

$$2^{ki+b}w[i]+2^{ki}s=2^{ki}(2^{b}w[i]+s) \qquad (7.16)$$

[0136] また、あるウィンドウ値ｗ［ｉ］がウィンドウ補正の対象となり、ウィンドウ値ｗ［ｉ＋１］に対してキャリー補正が行われる場合に注目して、省略記号を用いて式（５．３）を書き換えると、式（７．１７）が得られる。

$$d=c+\cdots+2^{ki+b}w[i]+2^{ki}s+2^{k(i+1)+b}w[i+1]+2^{k(i+1)}s+\cdots \qquad (7.17)$$

[0137] ここで、互いに打ち消しあう－２$^{k（i＋1）＋b}$という項と＋２$^{k（i＋1）＋b}$という項を式（７．１７）の右辺に付け加えると、式（７．１８）が得られる。

$$d=c+\cdots+2^{ki+b}w[i]-2^{k(i+1)+b}+2^{ki}s$$
$$+2^{k(i+1)+b}w[i+1]+2^{k(i+1)+b}+2^{k(i+1)}s+\cdots \qquad (7.18)$$

そして、式（７．１８）を変形すると、式（７．１９）が得られる。

$$d=c+\cdots+2^{ki}((w[i]-2^{k})2^{b}+s)+2^{k(i+1)}((w[i+1]+1)2^{b}+s)+\cdots \qquad (7.19)$$

[0138] 式（７．１９）において、（ｗ［ｉ］－２^{k}）はウィンドウ補正後のウィンドウ値を表し、（ｗ［ｉ＋１］＋１）はキャリー補正された１つ上位のウィンドウ値を表す。また、式（７．１９）における（（ｗ［ｉ］－２^{k}）２^{b}＋ｓ）は、「図１の試みでは、ウィンドウ補正後のウィンドウ値をインデック

スｈとするテーブルデータとして、（２^{b}（ｈ－２^{k}）＋ｓ）が必要である」ということを示す。

[0139] このように、式（７．１９）における乱数値ｓの符号は、式（７．１６）における乱数値ｓの符号と変わらない。よって、たとえ図１の試みのようにウィンドウ補正とキャリー補正が行われるとしても、乱数値ｓによりテーブルデータがランダム化される限り、テーブルデータは省略することができない。

[0140] つまり、ｈ≧２$^{k-1}$＋１なるインデックスｈに対応する点（２^{b}（ｈ－２^{k}）＋ｓ）Ａは、テーブルデータｔａｂ［－（ｈ－２^{k}）］＝（－２^{b}（ｈ－２^{k}）＋ｓ）Ａの逆元ではない。よって、たとえウィンドウ補正とキャリー補正が行われるとしても、ｈ≧２$^{k-1}$＋１なるインデックスｈに対応するテーブルデータｔａｂ［ｈ］＝（２^{b}（ｈ－２^{k}）＋ｓ）Ａは、テーブルから省略することができない。

[0141] 図２のテーブル１０２の例は、ｋ＝３の場合について、たとえウィンドウ補正とキャリー補正が行われるとしても、ｈ＝５に対応するテーブルデータｔａｂ［５］がテーブルデータｔａｂ［３]の逆元ではないので省略不能であることを示している。同様に、図２のテーブル１０２の例は、テーブルデータｔａｂ［６］がテーブルデータｔａｂ［２］の逆元ではないので省略不能であることを示している。また、図２のテーブル１０２の例は、テーブルデータｔａｂ［７］がテーブルデータｔａｂ［１］の逆元ではないので省略不能であることも示している。

[0142] 以上説明したように、単純素朴にランダム化ウィンドウ法と符号つきウィンドウ法を組み合わせようとする試みは失敗する。そこで、第１～第３実施形態では、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性と、メモリ使用量の抑制とを両立させるために、図１の試みとは別のアプローチが取られる。

[0143] 以下に、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性とメモリ使用量の抑制とを両立するために第１～第３実施形態で取られるアプローチの概要を、図３～５を参照しながら説明する。

[0144] 図３は、第１～第３実施形態に共通するアプローチを説明する図である。図３のテーブル１０３は、ウィンドウサイズｋが３であり乱数値ｓのビット長ｂが８の場合の例である。ＤＰＡ攻撃に対する安全性を確保するため、テーブル１０３が保持するテーブルデータは、乱数値ｓによってランダム化される。

[0145] また、図１の試みが失敗する理由は、ウィンドウ補正の結果によらず常に一定の乱数値ｓを用いることにあった。この理由を考慮して、図３のアプローチでは、ウィンドウ値に応じて乱数値ｓの符号を反転する。すなわち、図３のアプローチでは、ウィンドウ補正の結果としてウィンドウ値が負になった場合には、乱数値ｓの符号も合わせて反転する。

[0146] 例えば、ｋ＝３の場合、必要に応じてキャリー補正されたウィンドウ値は、５（＝2^{k-1}＋１）以上のときにウィンドウ補正の対象となり、８（＝2^{k}）が減算される。ウィンドウ補正されたウィンドウ値は、結果として負になる。テーブル１０３に示されるように、図３のアプローチでは、こうして負に補正されたウィンドウ値には、元の乱数値ｓではなく、－ｓが対応づけられる。

[0147] 例えば、必要に応じてキャリー補正されたウィンドウ値が５のとき、ウィンドウ補正されたウィンドウ値は－３（＝５－８）であり、負である。よって、－３に補正されたウィンドウ値に、＋ｓではなく－ｓによりランダム化された点（－３×2^{8}－ｓ）Ａを対応づけることにより、インデックスが５のエントリを省略することが可能となる。なぜなら、点（－３×2^{8}－ｓ）Ａは点（３×2^{8}＋ｓ）Ａの逆元であり、インデックスが３のエントリのテーブルデータから少ない計算量で計算可能だからである。

[0148] 同様に、必要に応じてキャリー補正されたウィンドウ値が６のとき、ウィンドウ補正されたウィンドウ値は－２（＝６－８）であり、負である。よって、－２に補正されたウィンドウ値に点（－２×2^{8}－ｓ）Ａを対応づけることにより、インデックスが６のエントリを省略することが可能となる。なぜなら、点（－２×2^{8}－ｓ）Ａは点（２×2^{8}＋ｓ）Ａの逆元であり、インデ

ックスが２のエントリのテーブルデータから少ない計算量で計算可能だからである。

[0149] 同様に、必要に応じてキャリー補正されたウィンドウ値が７のとき、ウィンドウ補正されたウィンドウ値は－１（＝７－８）であり、負である。よって、－１に補正されたウィンドウ値に点（－１×2^{8}－ｓ）Ａを対応づけることにより、インデックスが７のエントリを省略することが可能となる。なぜなら、点（－１×2^{8}－ｓ）Ａは点（１×2^{8}＋ｓ）Ａの逆元であり、インデックスが１のエントリのテーブルデータから少ない計算量で計算可能だからである。

[0150] このように、ウィンドウ補正の対象とならない０以上2^{k-1}以下のウィンドウ値には、乱数値ｓが対応づけられるが、ウィンドウ補正の対象となって負に補正されたウィンドウ値には、符号を反転させた乱数値－ｓが対応づけられる。そこで、以下では、あるウィンドウ値ｗ［ｉ］に対応する乱数値をｓ［ｉ］と表記する。乱数値ｓ［ｉ］は、ｉの値に応じて、＋ｓまたは－ｓのいずれかの値をとる。

[0151] 以上のように、ウィンドウ値ｗ［ｉ］に対応する乱数値ｓ［ｉ］を、ウィンドウ値ｗ［ｉ］に応じてｓ［ｉ］＝＋ｓまたはｓ［ｉ］＝－ｓと設定することで、ＤＰＡ攻撃に対する安全性を確保するためのランダム化と、メモリ使用量の抑制が両立可能となる。もちろん、ＳＰＡ攻撃に対する安全性は、ウィンドウの使用により確保される。よって、ある点Ａと個人鍵ｄに対してスカラ倍点ｄＡを求めるのに、図３のテーブル１０３のようなテーブルデータを生成し、生成したテーブルデータに基づいた演算を行う暗号処理装置は、様々な分野に好適である。

[0152] なお、以下では表記の簡略化のため、あるｉに対応するウィンドウ値を「ウィンドウ値ｗ［ｉ］」と表現することもあるし、ウィンドウ列（すなわちウィンドウ値の系列ｗ［ｍ－１］，…，ｗ［１］，ｗ［０］）全体を「ウィンドウ列ｗ［ｉ］」と表現することもある。同様に、乱数値の系列ｓ［ｍ－１］，…，ｓ［１］，ｓ［０］を「乱数列」ともいい、あるｉに対応する乱

数値を「乱数値ｓ［ｉ］」と表現することもあるし、乱数列全体を「乱数列ｓ［ｉ］」と表現することもある。

[0153] 図４は、以上説明した第１～第３実施形態でのアプローチを別の観点から図示したものである。すなわち、ｕ（＝ｂ＋ｋｍ）ビットの個人鍵ｄとウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃの間には式（８．１）が成立する。

[数7]

$$d = c + \sum_{i=0}^{m-1} 2^{ki}(w[i] \| s[i])$$

$$= c + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i]) \qquad (8.1)$$

[0154] そして、０≦ｉ≦ｍ－１なる各ｉについて、式（８．１）が成立するという制約条件のもとで、ウィンドウ値ｗ［ｉ］に応じて乱数値ｓ［ｉ］が＋ｓまたはーｓに設定される。すなわち、必要に応じてキャリー補正とウィンドウ補正が行われたウィンドウ値ｗ［ｉ］が０以上ならば、乱数値ｓ［ｉ］は＋ｓに設定されるのが適切である。また、必要に応じてキャリー補正とウィンドウ補正が行われたウィンドウ値ｗ［ｉ］が負ならば、乱数値ｓ［ｉ］はーｓに設定されるのが適切である。

[0155] 以上のように、式（８．１）が成立するという制約条件のもとで、ウィンドウ値ｗ［ｉ］に応じて乱数値ｓ［ｉ］が＋ｓまたはーｓに設定されると、図３のテーブル１０３のように、ランダム化され、かつエントリ数の少ないテーブルを用いた演算が可能となる。すなわち、ＰＡ攻撃に対する安全性とメモリ使用量の抑制がともに実現される。

[0156] ところが、実際に式（８．１）が成立するという制約条件のもとでウィンドウ値ｗ［ｉ］に応じて乱数値ｓ［ｉ］を＋ｓまたはーｓと設定しようとすると、循環参照の取り扱いに関する困難が生じる。もちろん、第１～第３実施形態の暗号処理装置は、その困難を克服するように設計されている。そこ

で、続いて、第１～第３実施形態についての理解の助けとするために、上記困難について説明する。

[0157] 図３と図４を参照して説明したとおり、各乱数値ｓ［ｉ］は、対応するウィンドウ値ｗ［ｉ］に応じて決まる。換言すれば、乱数列ｓ［ｉ］の値は、ウィンドウ列ｗ［ｉ］の値に依存して決まる。他方で、図５を参照して以下に説明するとおり、ウィンドウ列ｗ［ｉ］の値は、乱数列ｓ［ｉ］の値に依存して決まる。よって、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］は互いに循環参照している。そのため、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］の値を決定する自明な方法はない。

[0158] 図５は、ウィンドウ列ｗ［ｉ］の値が乱数列ｓ［ｉ］の値に依存して決まることを説明する図である。図５においても、ウィンドウサイズはｋであり、乱数値ｓのビット長はｂであり、ｍは正整数であり、個人鍵ｄのビット長はｕ＝ｂ＋ｍｋである。よって、各ウィンドウ値ｗ［ｉ］と各乱数値ｓ［ｉ］は符号つきｋビット値であり、補正値ｃは符号つきｂビット値である。

[0159] なお、本明細書において、Ｎが正整数のとき、「符号つきＮビット値」とは、Ｎビットで表される０以上（2^N-1）以下の整数と、正または負の符号との組み合わせによって表される値のことである。よって、符号つきＮビット値は、－（2^N-1）以上（2^N-1）以下である。

[0160] 符号つきＮビット値は、符号を示す１ビットと上記Ｎビットをあわせた（Ｎ＋１）ビットにより表すこともできるが、（Ｎ＋１）ビットの２の補数表現とは異なる。例えば、－５の２の補数表現は“１０１１”である。他方、５＝$(101)_2$なので、－５を符号つき３ビット値として表すと、－$(101)_2$である。

[0161] また、符号つきＮビット値の最上位または最下位の一部のビットのみを取り出す場合、取り出されたビット値の符号は、元の符号つきＮビット値の符号と同じである。例えば、符号つき４ビット値－$(1011)_2$の最上位２ビットを取り出すと、－$(10)_2$＝－２が得られ、符号つき４ビット値－$(1011)_2$の最下位２ビットを取り出すと、－$(11)_2$＝－３が得られる。

[0162] なお、以下の説明では、符号つきＮビット値についても、ビット列のｉビット目を表す記号“［ｉ］”と連結を表す記号“｜｜”を用いる。符号つきＮビット値に関してこれらの記号が用いられる場合、正負の符号も継承される。例えば、ａ＝－（１０１１）$_2$のとき、（ａ［３］｜｜ａ［２］）＝－（１０）$_2$＝－２であり、（ａ［１］｜｜ａ［０］）＝－（１１）$_2$＝－３である。

[0163] 図５は、式（８．１）から移項により得られる式（８．２）を表している。

[数8]

$$d-\sum_{i=0}^{m-1}2^{ki}s[i]=c+\sum_{i=0}^{m-1}2^{ki+b}w[i] \qquad (8.2)$$

[0164] 式（８．２）の左辺は、各乱数値ｓ［ｉ］をｉｋビット左シフトした値2^{ik}ｓ［ｉ］の合計値を個人鍵ｄから減算した結果を表す。また、式（８．２）の右辺は、左辺が示す減算結果の上位ｍｋビットをｋビットずつ区切ることでウィンドウ列ｗ［ｉ］が得られることと、左辺が示す減算結果の下位ｂビットが補正値ｃとなることを示す。

[0165] 式（８．１）は式（５．２）および（５．３）と類似しており、式（８．２）は、式（５．３）から得られた式（５．４）と類似している。そして、第３の比較例では、式（５．５）の具体例に示したように、式（５．４）の左辺の値を計算することでウィンドウ列と補正値が一意に得られる。よって、式（８．２）は、一見すると、第３の比較例と同様にして、左辺の値を計算することでウィンドウ列と補正値を一意に導出するための式であるかのように見えるかもしれない。

[0166] しかしながら、式（５．４）と式（８．２）には大きな違いがある。すなわち、式（５．４）では、ウィンドウの位置を示す変数ｉによらず一定の乱数値ｓが使われるのに対して、式（８．２）では、変数ｉによって乱数値ｓ［ｉ］が異なり、乱数値ｓ［ｉ］は＋ｓまたは－ｓである。

[0167] そのため、残念ながら式（８．２）から一意にウィンドウ列と補正値を決定することはできない。つまり、各乱数値ｓ［ｉ］が２つの値をとりうるので、式（８．２）の左辺で個人鍵ｄから減じる値（すなわち各乱数値ｓ［ｉ］をｉｋビット左シフトした値2^{ik}ｓ［ｉ］の合計値）には、2^m通りのパターンがある。よって、式（８．２）の左辺で個人鍵ｄから減じる値が2^m通りのパターンのいずれであるのかが決定されないと、式（８．２）の右辺からウィンドウ列ｗ［ｉ］と補正値ｃの値を確定することはできない。

[0168] 例えば、個人鍵ｄのビット長ｕが１６０であり、乱数値のビット長ｂが１０であり、ウィンドウサイズｋが３であるとすると、式（８．３）よりｍ＝５０である。

$$m=(u-b)/k=(160-10)/3=50 \qquad (8.3)$$

この場合、式（８．２）の左辺で個人鍵ｄから減じる値は、2^{50}通りの厖大なパターンのうちの１つである。つまり、この場合、乱数列ｓ［ｉ］は、潜在的に可能な2^{50}通りのパターンのいずれか１つである。

[0169] したがって、式（８．２）によってウィンドウ列ｗ［ｉ］と補正値ｃを計算しようとすると、乱数列ｓ［ｉ］として潜在的に可能な2^{50}通りのパターンのうちの１つを選択しない限り、ウィンドウ列ｗ［ｉ］と補正値ｃは得られない。すなわち、ウィンドウ列ｗ［ｉ］の値は乱数列ｓ［ｉ］の値に依存する。

[0170] 他方、図３と図４を参照して説明したとおり、乱数列ｓ［ｉ］の値はウィンドウ列ｗ［ｉ］の値に依存する。したがって、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］は互いに循環参照している。

[0171] 以上のような循環参照が存在すると、ウィンドウ列ｗ［ｉ］の値と乱数列ｓ［ｉ］の値と補正値ｃを一意に決定する自明な方法は存在しない。このことは、循環参照の存在しない第３の比較例においては式（５．４）に示すごとくウィンドウ列ｗ［ｉ］の値を一意に決定する自明な方法があることと対照的である。第１～第３実施形態では、循環参照に起因する困難を克服するため、後述の方法にしたがって、ウィンドウ列ｗ［ｉ］の値と乱数列ｓ［ｉ

］の値と補正値ｃが決定される。

[0172] 以上説明したとおり、第１～第３実施形態の暗号処理装置は、図３に示したアプローチにしたがいつつ、循環参照に起因する困難を克服してウィンドウ列ｗ［ｉ］の値と乱数列ｓ［ｉ］の値と補正値ｃを決定するという点が共通する。さらに、第１～第３実施形態の暗号処理装置は、他のいくつかの点でも共通する。そこで、以下では第１～第３実施形態の暗号処理装置の共通点について図６～１０を参照して説明し、その後、個々の実施形態について詳細に説明する。

[0173] さて、図６は、第１～第３実施形態の暗号処理装置の第１のハードウェア構成例を示す図である。

図６の暗号処理装置２００は、Central Processing Unit（ＣＰＵ）２０１、Read Only Memory（ＲＯＭ）２０２、Random Access Memory（ＲＡＭ）２０３、通信回路２０４、および通信インタフェイス（Ｉ／Ｆ）２０５を有する。通信回路２０４は、通信Ｉ／Ｆ２０５を介して他の装置との間で通信を行う。

[0174] そして、ＣＰＵ２０１、ＲＯＭ２０２、ＲＡＭ２０３、および通信回路２０４は、バス２０６により互いに接続されている。また、暗号処理装置２００は、電源端子２０７とグランド端子２０８を有し、暗号処理装置２００内の各部へは、不図示の配線と電源端子２０７を介して電源電圧が供給される。暗号処理装置２００内の各部は、不図示の配線を介してグランド端子２０８とも接続されている。

[0175] ＣＰＵ２０１は、ＲＯＭ２０２に予め記憶されたプログラムをＲＡＭ２０３にロードし、ＲＡＭ２０３をワーキングエリアとして用いながらプログラムを実行することで、各種の処理を行う。例えば、ＣＰＵ２０１は、図９の処理を行う。なお、後述するように、図９の処理は、図１０の処理を含み、図１１、１９または２４の処理も含む。

[0176] なお、ＲＯＭ２０２の代わりにフラッシュメモリなどの他の種類の不揮発性記憶装置が使われてもよい。フラッシュメモリなどの書き換え可能な記憶

装置がＲＯＭ２０２の代わりに使われる場合は、プログラムは、通信Ｉ／Ｆ２０５を介して暗号処理装置２００にダウンロードされ、暗号処理装置２００にインストールされてもよい。

[0177] また、暗号処理装置２００は、通信Ｉ／Ｆ２０５を介して他の装置と通信することができる。例えば、暗号処理装置２００は、暗号処理装置２００自身の公開鍵などの情報を、通信Ｉ／Ｆ２０５を介して他の装置に送信してもよいし、他の装置の公開鍵などの情報を、通信Ｉ／Ｆ２０５を介して受信してもよい。

[0178] 通信Ｉ／Ｆ２０５の種類は、暗号処理装置２００の種類に応じた任意の種類でよい。例えば、暗号処理装置２００は、スマートカードでもよいし、プリンタカートリッジなどのアクセサリ部品に組み込まれるＬＳＩチップでもよいし、家電製品に組みこまれるＬＳＩチップでもよい。例えば、暗号処理装置２００が接触式スマートカードの場合は、通信Ｉ／Ｆ２０５は通信用端子を含んでもよいし、暗号処理装置２００が非接触式スマートカードの場合は、通信Ｉ／Ｆ２０５はアンテナを含んでもよい。

[0179] 通信回路２０４は、通信Ｉ／Ｆ２０５の種類と通信プロトコルに応じて、適宜の処理を行う。例えば、通信回路２０４は、ディジタル・アナログ変換、アナログ・ディジタル変換、変調、復調、符号化、復号などの処理を行ってもよい。

[0180] なお、ＰＡ攻撃を行う攻撃者は、通信Ｉ／Ｆ２０５を介して楕円曲線上の点のデータを暗号処理装置２００に入力し、入力された点に関して暗号処理装置２００が処理を行っているときの消費電力を測定することで、暗号処理装置２００の個人鍵を推測する。例えば、攻撃者は、電源端子２０７に抵抗器を接続することで、消費電力の測定を行う。

[0181] 図７は、第１～第３実施形態の暗号処理装置の第２のハードウェア構成例を示す図である。図７の暗号処理装置２１０は、ＣＰＵ２０１とＲＯＭ２０２の代わりにＥＣＣハードウェア回路２１１を有する。

[0182] また、暗号処理装置２１０は、図６の暗号処理装置２００と同様のＲＡＭ

２０３と通信回路２０４と通信Ｉ／Ｆ２０５を有する。そして、暗号処理装置２１０において、ＥＣＣハードウェア回路２１１とＲＡＭ２０３と通信回路２０４は互いにバス２０６で接続されている。また、暗号処理装置２１０にも図６の暗号処理装置２００と同様の電源端子２０７とグランド端子２０８がある。

[0183] 暗号処理装置２１０においては、ＲＯＭ２０２からプログラムを読み出して実行するＣＰＵ２０１の代わりに、ＥＣＣハードウェア回路２１１が図９の処理を行う。ＥＣＣハードウェア回路２１１は、例えば、Application Specific Integrated Circuit（ＡＳＩＣ）でもよいし、ＥＣＣハードウェア回路２１１の少なくとも一部がField Programmable Gate Array（ＦＰＧＡ）により実現されていてもよい。また、ＥＣＣハードウェア回路２１１も、不図示の配線により電源端子２０７およびグランド端子２０８と接続されている。

[0184] なお、実施形態によっては、暗号処理装置が、汎用プロセッサとしての図６のＣＰＵ２０１と、ＣＰＵ２０１が実行するプログラムを格納する図６のＲＯＭ２０２と、コプロセッサとしての図７のＥＣＣハードウェア回路２１１を有していてもよい。そして、図９の処理の一部をＣＰＵ２０１が行い、残りの一部をＥＣＣハードウェア回路２１１が行ってもよい。その場合も、暗号処理装置は、図６および図７と同様に、ＲＡＭ２０３と通信回路２０４と通信Ｉ／Ｆ２０５を有する。

[0185] 図８は、第１～第３実施形態の暗号処理装置の機能構成を説明する図である。図８に示す暗号処理装置３００は、図６または図７に例示したハードウェアにより実現されてもよい。

[0186] 暗号処理装置３００は処理部３０１を有し、処理部３０１は判断部３０２と決定部３０３を含む。暗号処理装置３００はさらに、個人鍵格納部３０４、乱数生成部３０５、取得部３０６、スカラ倍算部３０７、スカラ倍点情報格納部３０８、ウィンドウ演算部３０９、補正部３１０および結果処理部３１１を有する。

[0187] 処理部３０１は、ｕビットの個人鍵ｄとｂビットの乱数値ｓから、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを得るための処理を行う。

処理部３０１内の判断部３０２は、各乱数値ｓ［ｉ］を＋ｓと－ｓのいずれに設定するのが良いかを判断する。判断部３０２の判断基準は、実施形態に応じて異なるので、詳しくは後述する。

[0188] また、処理部３０１内の決定部３０３は、判断部３０２の決定にしたがって、乱数値ｓ［ｉ］とウィンドウ値ｗ［ｉ］と補正値ｃを決定する。そして、決定部３０３は、決定した乱数値ｓ［ｉ］とウィンドウ値ｗ［ｉ］に応じた値（具体的には後述の補正済み差分値ｄｉｆｆ）を、次の乱数値ｓ［ｉ－１］についての判断のために判断部３０２にフィードバックする。また、決定部３０３は、決定したウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、決定した補正値ｃを補正部３１０に出力する。

[0189] また、詳しくは後述するが、第１～第３実施形態では、処理部３０１が個人鍵ｄに対する前処理を図９のステップＳ１０２で行い、補正部３１０が図９のステップＳ１０９～Ｓ１１１の後処理を行う。そこで、処理部３０１は、個人鍵ｄのＬＳＢであるｄ［０］を、後処理のために補正部３１０に出力する。

[0190] なお、判断部３０２と決定部３０３を含む処理部３０１は、図６のＣＰＵ２０１によって実現されてもよいし、図７のＥＣＣハードウェア回路２１１によって実現されてもよいし、ＣＰＵ２０１とＥＣＣハードウェア回路２１１の組み合わせにより実現されてもよい。また、処理部３０１がＣＰＵ２０１により実現される場合、判断部３０２を実現するためのプログラムモジュールと決定部３０３を実現するためのプログラムモジュールは、別個のプログラムモジュールでもよいし、あるいは１つに統合されていてもよい。

[0191] また、個人鍵格納部３０４は、個人鍵ｄを格納しており、例えばＲＯＭ２０２により実現される。処理部３０１はこの個人鍵格納部３０４から個人鍵ｄを読み出す。なお、個人鍵ｄは符号なしの（unsigned）正の値である。

[0192] 乱数生成部３０５は、ｂビットの乱数値ｓを生成し、乱数値ｓを処理部３

０１に出力する。説明の簡単化のため、以下の第１～第２実施形態の説明では、乱数値ｓが０以上と仮定し、第３実施形態の説明では乱数値ｓが正であると仮定するが、乱数値ｓが負の場合の変形例についても、後述する。乱数生成部３０５は、ＣＰＵ２０１またはＥＣＣハードウェア回路２１１により実現される。

[0193] 取得部３０６は、スカラ倍算の対象である楕円曲線上の点Ａのｘｙ座標を取得し、取得した点Ａのｘｙ座標をスカラ倍算部３０７と補正部３１０に出力する。なお、取得部３０６は、暗号処理装置３００の不図示の記憶部から点Ａのｘｙ座標を読み出すことにより点Ａのｘｙ座標を取得してもよいし、外部装置と通信して外部装置から点Ａのｘｙ座標を受信することで点Ａのｘｙ座標を取得してもよい。

[0194] 例えば、点Ａは、暗号処理装置３００自身が予め決定したベースポイントでもよい。この場合、取得部３０６は、ベースポイントのｘｙ座標を記憶するための暗号処理装置３００内の不図示の記憶部を参照することで、点Ａのｘｙ座標を取得する。

[0195] ベースポイントのｘｙ座標を記憶する記憶部は、例えばＲＯＭ２０２により実現されてもよい。そして、取得部３０６は、ＲＯＭ２０２からデータを読み出すＣＰＵ２０１またはＥＣＣハードウェア回路２１１により実現されてもよい。

[0196] あるいは、点Ａは、暗号処理装置３００以外の装置から暗号処理装置３００に与えられる点でもよい。例えば、点Ａは、外部装置の公開鍵でもよい。外部装置の公開鍵は、例えば、ＤＨ鍵共有のために外部装置から暗号処理装置３００へと通知されることもあるし、ＤＳＡによる認証のために外部装置から暗号処理装置３００へと通知されることもある。

[0197] 点Ａが暗号処理装置３００以外の装置から暗号処理装置３００に与えられる点である場合、取得部３０６は、具体的には、通信Ｉ／Ｆ２０５と通信回路２０４により実現される。すなわち、取得部３０６は、外部装置から点Ａのｘｙ座標を受信することで、点Ａのｘｙ座標を取得する。

[0198] また、スカラ倍算部３０７は、実施形態に応じて適宜に決められた範囲内の各インデックスｈに対して、点（２ｂｈ＋ｓ）Ａのｘｙ座標を計算する。点（２ｂｈ＋ｓ）Ａは、点Ａのスカラ倍点なので、以下では点（２ｂｈ＋ｓ）Ａを表す情報（すなわち点（２ｂｈ＋ｓ）Ａのｘｙ座標）を「スカラ倍点情報」ともいう。スカラ倍算部３０７は、インデックスとスカラ倍点情報をスカラ倍点情報格納部３０８に出力する。スカラ倍算部３０７は、ＣＰＵ２０１により実現されてもよいし、ＥＣＣハードウェア回路２１１により実現されてもよいし、両者の組み合わせにより実現されてもよい。

[0199] また、スカラ倍点情報格納部３０８は、スカラ倍算部３０７が生成したスカラ倍点情報をインデックスと対応づけて格納する。スカラ倍点情報格納部３０８は、ＲＡＭ２０３により実現される。

[0200] なお、第１～３実施形態のスカラ倍点情報格納部３０８は、スカラ倍点情報とインデックスをテーブル形式のデータとして格納するので、以下ではインデックスｈに対応づけられたスカラ倍点情報を、「テーブルデータ」ともいい、ｔａｂ［ｈ］とも表記する。また、テーブルデータｔａｂ［ｈ］は乱数値ｓによりランダム化されているので、以下ではテーブルデータｔａｂ［ｈ］を「ランダム化テーブルデータ」ともいう。

[0201] もちろん、スカラ倍点情報のデータ形式は実施形態に応じて任意であり、テーブル形式以外のデータ形式も利用可能である。例えば、スカラ倍点情報格納部３０８は、単に、ベースアドレスとインデックスｈから一意に決定されるメモリアドレスに点（２ｂｈ＋ｓ）Ａのｘｙ座標を記憶するだけでもよい。つまり、スカラ倍点情報格納部３０８は、点（２ｂｈ＋ｓ）Ａのｘｙ座標をインデックスｈと対応づけて記憶してさえいればよく、インデックスｈ自体を明示的に記憶していなくてもよい。

[0202] また、ウィンドウ演算部３０９は、処理部３０１により得られたウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］を用いて、スカラ倍点情報格納部３０８に格納されたスカラ倍点情報ｔａｂ［ｈ］を参照しながら、ウィンドウを利用した演算を行う。そして、ウィンドウ演算部３０９は、演算結果を補正部３１

０に出力する。ウィンドウ演算部３０９は、ＣＰＵ２０１により実現されてもよいし、ＥＣＣハードウェア回路２１１により実現されてもよいし、両者の組み合わせにより実現されてもよい。

[0203] 具体的には、ウィンドウ演算部３０９は、処理部３０１により得られたウィンドウ値ｗ［ｉ］またはその符号を反転させた値－ｗ［ｉ］を、インデックスｈとして用いて、スカラ倍点情報格納部３０８を参照する。そして、ウィンドウ演算部３０９は、インデックスｈに対応するスカラ倍点情報ｔａｂ［ｈ］をスカラ倍点情報格納部３０８から読み出し、読み出したスカラ倍点情報ｔａｂ［ｈ］を用いた演算を実行する。ウィンドウ演算部３０９の詳しい動作は図１０とともに後述する。

[0204] 補正部３１０は、決定部３０３から入力される補正値ｃと取得部３０６から入力される点Ａのｘｙ座標から、点ｃＡを計算する。そして、補正部３１０は、ウィンドウ演算部３０９からの入力である楕円曲線上の点に点ｃＡを加算し、処理部３０１から通知された個人鍵ｄのＬＳＢであるｄ［０］の値を参照して後処理を行うことで、ウィンドウ演算部３０９の演算結果を補正する。結果として得られる点Ｖは、Ｖ＝ｄＡなる点である。補正部３１０は、点Ｖのｘｙ座標を結果処理部３１１に出力する。

[0205] 補正部３１０は、ＣＰＵ２０１により実現されてもよいし、ＥＣＣハードウェア回路２１１により実現されてもよいし、両者の組み合わせにより実現されてもよい。また、詳しくは図９のステップＳ１０７とともに後述するが、補正部３１０は、ローカルなテーブルデータを持つので、補正部３１０を実現するにはＲＡＭ２０３も用いられる。

[0206] 結果処理部３１１は、点Ｖのｘｙ座標を用いて適宜の処理を行う。例えば、結果処理部３１１は、点Ｖを他の装置に送信してもよいし、ＤＳＡによる認証のための処理を行ってもよいし、ＤＨ鍵共有のための処理を行ってもよい。結果処理部３１１は、ＣＰＵ２０１により実現されてもよいし、ＥＣＣハードウェア回路２１１により実現されてもよいし、両者の組み合わせにより実現されてもよい。また、処理の内容によっては、結果処理部３１１を実

現するために、さらに、通信回路２０４と通信Ｉ／Ｆ２０５などが利用されてもよい。

[0207] 例えば、暗号処理装置３００が、ホスト（例えばプリンタなど）により認証されるアクセサリ部品（例えばプリンタカートリッジなど）に含まれる場合、結果処理部３１１は、ホストと通信するための通信回路２０４と通信Ｉ／Ｆ２０５を含む。

[0208] 続いて、図９と１０を参照して、第１～第３実施形態の暗号処理装置３００が共通して行う処理について説明する。

図９は、第１～第３実施形態の暗号処理装置３００が、個人鍵ｄと点Ａからスカラ倍点Ｖ＝ｄＡを求める処理のフローチャートである。なお、前述のとおり、暗号処理装置３００がスカラ倍点Ｖ＝ｄＡを求める目的は任意であり、換言すれば、結果処理部３１１がスカラ倍点Ｖをどのように利用するかということは任意である。

[0209] ステップＳ１０１で処理部３０１は、個人鍵格納部３０４から個人鍵ｄを読み出す。なお、個人鍵ｄのビット長ｕと乱数値ｓのビット長ｂとウィンドウサイズｋの関係は、正整数ｍを用いて、式（８．４）のように表される。

$$u=b+km \tag{8.4}$$

[0210] したがって、個人鍵ｄは式（８．５）のとおりである。

$$\begin{aligned} d&=d[u-1]||d[u-2]||\cdots||d[1]||d[0] \\ &=d[b+km-1]||d[b+km-2]||\cdots||d[1]||d[0] \end{aligned} \tag{8.5}$$

[0211] 次のステップＳ１０２で処理部３０１は前処理を行う。ステップＳ１０２における前処理は、後述のステップＳ１０９～Ｓ１１１における後処理と対になっている。

[0212] 具体的には、処理部３０１は、式（８．６）のように、ＭＳＢが０になるように個人鍵ｄを１ビット右シフトした値ｅを求める。式（８．６）を別の形式で表せば式（８．７）のとおりである。以下では式（８．６）と（８．７）により表される値ｅを、説明の便宜上「ダミー鍵」という。

$$e=0||d[u-1]||d[u-2]||\cdots||d[1]$$

=0||d[b+km-1]||d[b+km-2]||…||d[1] (8.6)

[数9]

$$e=\left\lfloor \frac{d}{2} \right\rfloor \qquad (8.7)$$

[0213] なお、詳しくは後述するが、第１～第３実施形態では、キャリー補正が行われる。最上位のウィンドウ値ｗ［ｍ－１］に対応して生じるキャリー補正値を適切に処理するためのテクニックはいくつかあり、ステップＳ１０２の前処理とステップＳ１０９～Ｓ１１１の後処理の組み合わせは、それらのテクニックのうちの一つである。他のテクニックについては後述する。

[0214] 式（８．６）の定義より、ダミー鍵ｅは個人鍵ｄと同じくｕビットの長さであり、ダミー鍵ｅのＭＳＢであるｅ［ｕ－１］は必ず０である。ＭＳＢが０であるという性質が、後述のステップＳ１０６の処理の単純化に資するので、第１～第３実施形態ではステップＳ１０２の前処理が行われる。

[0215] なお、処理部３０１は、ステップＳ１０２においてさらに、個人鍵ｄのＬＳＢであるｄ［０］を補正部３１０に出力する。

[0216] 次のステップＳ１０３では、乱数生成部３０５が乱数値ｓを生成し、生成した乱数値ｓを処理部３０１とスカラ倍算部３０７に出力する。そして、処理部３０１が、乱数値ｓを用いて、ダミー鍵ｅからウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを生成する。ステップＳ１０３の詳細は、実施形態に応じて異なり、具体的には図１１、１９、２４とともに後述する。

[0217] 第１～第３実施形態のいずれにおいても、処理部３０１はステップＳ１０３で、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを、乱数値ｓ［ｉ］を＋ｓまたはーｓに決定しながら、式（８．８）が成立するという制約条件下で定める。

[数10]

$$d = 2\left(c + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i]) \right) + d[0] \qquad (8.8)$$

[0218] なお、式（８．８）は、式（８．１）と似ているが式（８．１）そのものとは異なる。式（８．８）は、ステップＳ１０２での前処理とステップＳ１０９～Ｓ１１１の後処理の内容を反映している。

[0219] すなわち、第１～第３実施形態では、式（８．１）の個人鍵ｄをダミー鍵ｅに置き換えた式（８．９）が成立するという制約条件下で、処理部３０１がウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを定める。

[数11]

$$e = c + \sum_{i=0}^{m-1} \left(2^{ki+b} w[i] + 2^{ki} s[i] \right) \qquad (8.9)$$

[0220] ここで、式（８．６）の定義より式（８．１０）が成立する。また、式（８．８）は、式（８．１０）に式（８．９）を代入して得られる。したがって、「式（８．８）と（８．１０）が成立する」という制約条件は、「式（８．９）が成立する」という制約条件と同値である。

$$d=2e+d[0] \qquad (8.10)$$

[0221] そして、処理部３０１（より詳しくは決定部３０３）は、決定したウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、決定した補正値ｃを補正部３１０に出力する。

[0222] すると、次のステップＳ１０４で取得部３０６は、点Ａのｘｙ座標を取得し、スカラ倍算部３０７と補正部３１０に点Ａのｘｙ座標を出力する。

そして、次のステップＳ１０５でスカラ倍算部３０７は、実施形態に応じて適宜決められた所定範囲内の各インデックスｈについて、乱数値ｓに応じたスカラ倍点情報（具体的には、ランダム化テーブルデータｔａｂ［ｈ］＝（２ｂｈ＋ｓ）Ａ）を生成する。なお、乱数値ｓは、上記のとおり、ステップ

Ｓ１０３で乱数生成部３０５により生成されたときに、乱数生成部３０５からスカラ倍算部３０７にも出力されている。

[0223] また、上記所定範囲とは、具体的には、第１実施形態では－２≦ｈ≦２^{k-1}＋１であり、第２実施形態では－１≦ｈ≦２^{k-1}＋１であり、第３実施形態では－１≦ｈ≦２^{k-1}である。

[0224] さらに、ステップＳ１０５でスカラ倍算部３０７は、インデックスｈと対応づけてランダム化テーブルデータｔａｂ［ｈ］をスカラ倍点情報格納部３０８に出力する。すると、スカラ倍点情報格納部３０８は、インデックスｈと対応づけてランダム化テーブルデータｔａｂ［ｈ］を格納する。

[0225] そして、次のステップＳ１０６でウィンドウ演算部３０９は、ウィンドウ列［ｉ］と乱数列ｓ［ｉ］とランダム化テーブルデータｔａｂ［ｈ］を用いて、図１０に示す演算を実行し、計算結果を変数Ｖに格納する。以下、特に混乱のおそれはないので、変数Ｖが表す点を、点Ｖという。ウィンドウ演算部３０９は、図１０の演算の終了後、変数Ｖの内容（すなわち点Ｖのｘｙ座標）を補正部３１０に通知する。

[0226] また、次のステップＳ１０７では、補正部３１０が、補正値ｃと点Ａのｘｙ座標から、点ｃＡのｘｙ座標を計算する。ステップＳ１０７における計算は、第２比較例として説明した単純なウィンドウ法または第４比較例として説明した符号つきウィンドウ法によって行われる。

[0227] ステップＳ１０７におけるウィンドウサイズをｑとすると、補正部３１０は、具体的には、２^{q}個のエントリを有するローカルなテーブルを作成し、作成したテーブルのデータを使って、ウィンドウ法によって点ｃＡのｘｙ座標を計算する。あるいは、補正部３１０は、２^{q-1}＋１個のエントリを有するローカルなテーブルを作成し、作成したテーブルのデータを使って、符号つきウィンドウ法によって点ｃＡのｘｙ座標を計算する。ウィンドウサイズｑは、式（８．４）に“ｋ”として示したウィンドウサイズと同じでもよいし、ウィンドウサイズｋと異なっていてもよい。

[0228] なお、ステップＳ１０７で単純なウィンドウ法または符号つきウィンドウ

法が利用される理由は次のとおりである。

たとえ暗号処理装置３００が同じ個人鍵ｄを用いて何度も図９の処理を行うとしても、補正値ｃは、図９の処理が実行されるたびに、乱数値ｓに応じてランダムに変化する。よって、点ｃＡの計算は、ＤＰＡ攻撃に対して安全である。また、上述のとおり、ウィンドウ法と符号つきウィンドウ法はＳＰＡ攻撃に対して安全である。よって、補正部３１０はウィンドウ法または符号つきウィンドウ法によって点ｃＡを計算することで、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性を確保することができる。

[0229] 以上のようにして点ｃＡを計算すると、補正部３１０は、次のステップＳ１０８において、ウィンドウ演算部３０９から通知された点Ｖに点ｃＡを加算し、加算の結果を変数Ｖに格納する。ステップＳ１０３で得られたウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃは式（８．９）を満たすので、ステップＳ１０８の処理の結果として得られる点Ｖは、点ｅＡである。

[0230] 続くステップＳ１０９～Ｓ１１１は、ステップＳ１０２の前処理に対応する後処理である。具体的には、ステップＳ１０９で補正部３１０は、点Ｖに対して２倍算を行い、結果として得られた点２Ｖを新たに変数Ｖに格納する。

[0231] そして、次のステップＳ１１０で補正部３１０は、処理部３０１から通知されたビット値ｄ［０］が１であるか否かを判断する。もし、ｄ［０］＝１であれば、処理はステップＳ１１１に移行する。逆に、ｄ［０］＝０であれば、処理はステップＳ１１２に移行する。

[0232] ステップＳ１１１で補正部３１０は、点Ｖに点Ａを加算し、加算の結果を変数Ｖに格納する。

なお、ステップＳ１０９～Ｓ１１１の後処理の意味を補足すると、次のとおりである。式（８．１０）より式（８．１１）が成り立つ。

$$dA=(2e+d[0])A=2(eA)+d[0]A \qquad (8.11)$$

[0233] そして、上記のとおりステップＳ１０８が終了した時点での点Ｖは、Ｖ＝ｅＡである。よって、ステップＳ１０９の２倍算の結果として得られる点Ｖ

は、Ｖ＝２（ｅＡ）である。

[0234] また、式（１．７）のスカラ倍算の定義より、ｄ［０］＝０のとき、点ｄ［０］Ａは無限遠点Ｏであり、式（１．３）に示すように無限遠点Ｏは零元である。よって、ｄ［０］＝０のときは、式（８．１１）より、ステップＳ１０９で得られた点Ｖ＝２（ｅＡ）がすなわち点ｄＡである。そのため、上記のステップＳ１１０でｄ［０］＝０と判断された場合、ステップＳ１１１の加算は不要であるから処理はステップＳ１１０からステップＳ１１２へと進む。

[0235] 他方、ｄ［０］＝１のとき、点ｄ［０］Ａとは点Ａ自身である。よって、ｄ［０］＝１のときは、式（８．１１）より、ステップＳ１０９で得られた点Ｖ＝２（ｅＡ）に点Ａを加算した点が、点ｄＡである。そのため、上記のステップＳ１１０でｄ［０］＝１と判断された場合、ステップＳ１１１で点Ａの加算が行われ、その後、処理がステップＳ１１２へ進む。

[0236] 最後に、ステップＳ１１２で補正部３１０は、点Ｖのｘｙ座標を結果処理部３１１に出力する。こうして出力される点Ｖは、Ｖ＝ｄＡなる点である。

[0237] 以上で図９の処理は終了するが、図９のステップの実行順序は実施形態に応じて、適宜変更されてもよい。例えば、ステップＳ１０４での点Ａの取得は、ステップＳ１０５の前でありさえすれば、いつでもよい。また、ステップＳ１０６とＳ１０７の実行順序は逆でもよいし、ステップＳ１０６とＳ１０７が並行して実行されてもよい。

[0238] また、前述のとおりステップＳ１０３の処理は乱数値ｓの生成を含むが、乱数値ｓが生成されてスカラ倍算部３０７に出力された後では、ステップＳ１０３の残りの部分とステップＳ１０５が並行して実行されてもよい。さらに、もし、乱数値ｓの生成後、ステップＳ１０５が完了した後もまだステップＳ１０３の処理が続く場合には、ステップＳ１０３と並行してステップＳ１０６が実行されてもよい。例えば、ウィンドウ演算部３０９がステップＳ１０６でウィンドウ値ｗ［ｉ］と乱数値ｓ［ｉ］を用いて処理を行うのと並行して、処理部３０１がステップＳ１０３でウィンドウ値ｗ［ｉ－Ｍ］と乱

数値ｓ［ｉ－Ｍ］を求めてもよい（ただしＭ≧２である）。

[0239] さて、図１０は、第１～第３実施形態の暗号処理装置が、決定したウィンドウ列ｗ［ｉ］および乱数列ｓ［ｉ］、ならびに生成したスカラ倍点情報を利用して行う演算のフローチャートである。すなわち、図１０は、図９のステップＳ１０６の処理のフローチャートである。なお、具体的なデータが図１０の処理によってどのように処理されるのかについては、図１３、２１、２６とともに後述する。

[0240] さて、ステップＳ２０１でウィンドウ演算部３０９は、変数Ｖを無限遠点Ｏに初期化する。

そして、次のステップＳ２０２でウィンドウ演算部３０９は、ループ変数ｉを（ｍ－１）に初期化する。すなわち、ウィンドウ演算部３０９は、最上位のウィンドウ値ｗ［ｍ－１］に注目する。

[0241] 続いて、ステップＳ２０３でウィンドウ演算部３０９は、２倍算の回数を数えるためのループ変数ｊを１に初期化する。

そして、次のステップＳ２０４でウィンドウ演算部３０９は、変数Ｖの示す点Ｖに対して２倍算を行い、２倍算の結果を変数Ｖに格納する。

[0242] 次のステップＳ２０５でウィンドウ演算部３０９は、変数ｊの値がウィンドウサイズｋと等しいか否かを判断する。そして、ｊ≠ｋの場合（すなわちｊ＜ｋの場合）、処理はステップＳ２０６に移行し、ｊ＝ｋの場合、処理はステップＳ２０７に移行する。

[0243] ステップＳ２０６でウィンドウ演算部３０９は、変数ｊの値を１だけインクリメントする。そして、処理はステップＳ２０４に戻る。以上のステップＳ２０３～Ｓ２０６の処理は、２倍算をｋ回連続して行う処理である。

[0244] また、ステップＳ２０７でウィンドウ演算部３０９は、乱数値ｓ［ｉ］が乱数値ｓと等しいか否かを判断する。なお、図９のステップＳ１０３に関して説明したように、各乱数値ｓ［ｉ］は＋ｓまたは－ｓであるから、ステップＳ２０７の判断は、換言すれば、「ｓ［ｉ］＝＋ｓかそれともｓ［ｉ］＝－ｓか」という判断である。

[0245] 乱数値ｓが０以上に限定されている実施形態では、ウィンドウ演算部３０９は、乱数値ｓ［ｉ］の符号を参照して「符号が正を示していればｓ［ｉ］＝＋ｓであり、符号が負を示していればｓ［ｉ］＝－ｓである」と判断してもよい。また、負の乱数値ｓを利用する実施形態でも、ウィンドウ演算部３０９は、乱数値ｓの符号と乱数値ｓ［ｉ］の符号を比較するだけで、乱数値ｓ［ｉ］が乱数値ｓと等しいか否かを判断することができる。

[0246] そして、ｓ［ｉ］＝＋ｓの場合、処理はステップＳ２０８に移行し、ｓ［ｉ］＝－ｓの場合、処理はステップＳ２０９に移行する。

ステップＳ２０８でウィンドウ演算部３０９は、ウィンドウ値ｗ［ｉ］をインデックスとして用いてスカラ倍点情報格納部３０８を参照し、ウィンドウ値ｗ［ｉ］に対応するテーブルデータｔａｂ［ｗ［ｉ］］を得る。そして、ウィンドウ演算部３０９は、点Ｖにテーブルデータｔａｂ［ｗ［ｉ］］が表す点を加算し、加算結果を新たな点Ｖとして記憶する。

[0247] すなわち、ステップＳ２０８でウィンドウ演算部３０９は、式（８．１２）の演算を行う。そして、処理はステップＳ２１０へ移行する。

$$V=V+tab[w[i]] \tag{8.12}$$

[0248] また、ステップＳ２０９でウィンドウ演算部３０９はウィンドウ値ｗ［ｉ］の符号を反転させた値－ｗ［ｉ］をインデックスとして用いてスカラ倍点情報格納部３０８を参照し、テーブルデータｔａｂ［－ｗ［ｉ］］を得る。そして、ウィンドウ演算部３０９は、テーブルデータｔａｂ［－ｗ［ｉ］］が表す点を点Ｖから減算し、減算結果を新たな点Ｖとして記憶する。換言すれば、ウィンドウ演算部３０９は、テーブルデータｔａｂ［－ｗ［ｉ］］が表す点の逆元を計算し、計算した逆元を点Ｖに加算し、加算結果を新たな点Ｖとして記憶する。

[0249] すなわち、ステップＳ２０９でウィンドウ演算部３０９は、式（８．１３）の演算を行う。そして、処理はステップＳ２１０へ移行する。

$$V=V-tab[-w[i]] \tag{8.13}$$

ステップＳ２１０でウィンドウ演算部３０９は、ループ変数ｉの値を１だ

けデクリメントする。そして、処理はステップＳ２１１に移行する。

[0250] ステップＳ２１１でウィンドウ演算部３０９は、ループ変数ｉの値が０以上であるか否かを判断する。そして、ｉ≧０の場合、まだ最下位のウィンドウ値ｗ［ｉ］と乱数値ｓ［ｉ］の組まで注目し終わっていないので、処理はステップＳ２０３に戻る。他方、ｉ＜０の場合（つまりｉ＝－１の場合）、既にすべてのウィンドウ値ｗ［ｉ］と乱数値ｓ［ｉ］の組に注目し終わっているので、処理はステップＳ２１２に移行する。

[0251] ステップＳ２１２でウィンドウ演算部３０９は、点Ｖを返り値として補正部３１０に出力する。そして、図１０の処理は終了する。

続いて、第１実施形態について説明する。第１実施形態に関しては、まず、図９のステップＳ１０３に相当する図１１の処理を説明する。続いて、個人鍵ｄと乱数値ｓの数値例を含む図１２Ａ～１３を参照して、図９～１１の処理の具体例を説明する。図１１の各ステップの意味などについては、図１４～１６を参照して後述する。

[0252] 図１１は、第１実施形態において暗号処理装置３００がウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを決定する処理のフローチャートである。すなわち、図１１は、第１実施形態における、図９のステップＳ１０３のフローチャートである。

[0253] ステップＳ３０１で判断部３０２は、式（９．１）に示すように、符号つき（ｋ＋ｂ）ビット値d_Hを初期化する。すなわち、判断部３０２は、処理部３０１が得たダミー鍵ｅの先頭の（ｋ＋ｂ）ビットを、符号つき（ｋ＋ｂ）ビット値d_Hとして設定する。

$$d_H=e[b+km-1]||\cdots||e[k(m-1)] \qquad (9.1)$$

[0254] 次のステップＳ３０２で乱数生成部３０５は、ｂビットの乱数値ｓを生成し、乱数値ｓを処理部３０１とスカラ倍算部３０７に出力する。なお、説明の簡単化のため、第１実施形態では０≦ｓとする。乱数値ｓが負の場合については、第１実施形態の変形例として後述する。よって、ステップＳ３０２で生成される乱数値ｓは式（９．２）を満たす。

$$0 \leqq s \leqq 2^{b}-1 \qquad (9.2)$$

[0255] 次のステップＳ３０３で処理部３０１は、ループ変数ｉを（ｍ－１）に初期化する。そして、処理はステップＳ３０４に移行する。

ステップＳ３０４で処理部３０１内の判断部３０２は、判断基準値（$d_H - s$）の値を計算し、計算した値が下記の範囲Ｒ１～Ｒ４のいずれに含まれるかを判断し、判断結果を決定部３０３に通知する。

・範囲Ｒ１：-2^{k+b-1}以下

・範囲Ｒ２：（$-2^{k+b-1}+1$）以上－１以下

・範囲Ｒ３：０以上（$2^{k+b-1}-1$）以下

・範囲Ｒ４：2^{k+b-1}以上

[0256] そして、判断基準値（$d_H - s$）が範囲Ｒ１に含まれるとき、処理はステップＳ３０５に移行し、判断基準値（$d_H - s$）が範囲Ｒ２に含まれるとき、処理はステップＳ３０８に移行する。また、判断基準値（$d_H - s$）が範囲Ｒ３に含まれるとき、処理はステップＳ３１０に移行し、判断基準値（$d_H - s$）が範囲Ｒ４に含まれるとき、処理はステップＳ３１２に移行する。ステップＳ３０４における判断の意味については、図１５～１６とともに後述する。

[0257] ここで、範囲Ｒ１とＲ２の境界値-2^{k+b-1}を範囲Ｒ１とＲ２のいずれに含めるか、範囲Ｒ２とＲ３の境界値０を範囲Ｒ２とＲ３のいずれに含めるか、範囲Ｒ３とＲ４の境界値2^{k+b-1}を範囲Ｒ３とＲ４のいずれに含めるかは、実施形態に応じて任意である。しかし、好ましい定義は、上記のとおりである。なぜなら、上記のように範囲Ｒ１～Ｒ４を定義することで、以下のような処理の簡略化が可能となり、処理が簡略化される分、判断部３０２がステップＳ３０４を実行するのにかかる時間が短縮化されるという効果が得られるからである。

[0258] すなわち、ステップＳ３０４の処理の簡略化のため、判断部３０２は、判断基準値（$d_H - s$）の符号と（ｋ＋ｂ－１）番目のビット（すなわちＭＳＢ）の値をチェックすることで、判断基準値（$d_H - s$）がどの範囲に含まれるかを判断してもよい。

[0259] 具体的には、判断基準値（$d_H - s$）は、符号が負でＭＳＢの値が１ならば、範囲Ｒ１に含まれる。また、判断基準値（$d_H - s$）は、符号が負でＭＳＢの値が０ならば、範囲Ｒ２に含まれる。そして、判断基準値（$d_H - s$）は、符号が正でＭＳＢの値が０ならば、範囲Ｒ３に含まれる。また、判断基準値（$d_H - s$）は、符号が正でＭＳＢの値が１ならば、範囲Ｒ４に含まれる。

[0260] なお、第１実施形態では、個人鍵ｄではなくダミー鍵ｅに対して図１１の処理が行われる。そして、ダミー鍵ｅは、符号なしの正の値である個人鍵ｄから式（８．６）のようにして得られた値なので、$0 \leqq e[b+km-1]||\cdots||e[k(m-1)]=0||\cdots||e[k(m-1)] \leqq 2^{k+b-1}-1$である。よって、判断基準値（$d_H - s$）の定義から、ｉ＝ｍ－１の場合、判断基準値（$d_H - s$）は、範囲Ｒ２またはＲ３に含まれる。よって、後述のステップＳ３０７またはＳ３１４におけるキャリー補正は、ｉ＝ｍ－１の場合には生じない。

[0261] 第１実施形態は、以上のようにｉ＝ｍ－１の場合のキャリー補正が生じないように保証することで、図９のステップＳ１０６に相当する図１０の処理を簡略化するように設計されている。すなわち、ｉ＝ｍ－１の場合のキャリー補正が生じないと保証されていると、図１０のステップＳ２０１の初期化が、変数Ｖを無限遠点Ｏに設定するという単純な処理で済む。

[0262] 換言すれば、第１実施形態では、ダミー鍵ｅを使うことで、最上位のウィンドウ値ｗ［ｍ－１］に対応するキャリー補正値が０になることが保証される。そして、その保証のもとでは、最上位のキャリー補正値に関する処理として妥当なのは、「何もしない」ということである。こうして、第１実施形態では、ダミー鍵ｅの導入にともなう図９のステップＳ１０２の前処理とステップＳ１０９～Ｓ１１１の後処理により、キャリー補正値の適切な取り扱いが実現される。

[0263] さて、ステップＳ３０５で決定部３０３は、乱数値ｓ［ｉ］を＋ｓと決定する。また、次のステップＳ３０６で決定部３０３は、ウィンドウ補正値ｔ［ｉ］を$+2^{k+b}$と決定する。

[0264] そして、続くステップＳ３０７で決定部３０３は、ウィンドウ補正値ｔ［

ｉ］＝２$^{k+b}$の影響を相殺するために、１つ上位のウィンドウ値ｗ［ｉ＋１］に対するキャリー補正を行う。すなわち、決定部３０３は、ウィンドウ値ｗ［ｉ＋１］を１だけデクリメントする。

[0265] 以上のステップＳ３０５～Ｓ３０７の実行順序は任意に入れ換えられてもよく、ステップＳ３０５～Ｓ３０７は並列に実行されてもよい。ステップＳ３０５～Ｓ３０７の実行後、処理はステップＳ３１５に移行する。

[0266] また、ステップＳ３０８で決定部３０３は、乱数値ｓ［ｉ］を－ｓと決定する。次のステップＳ３０９で決定部３０３は、ウィンドウ補正値ｔ［ｉ］を０と決定する。

なお、ステップＳ３０８とＳ３０９の実行順序は入れ換えられてもよく、ステップＳ３０８とＳ３０９は並列に実行されてもよい。ステップＳ３０８とＳ３０９の実行後、処理はステップＳ３１５に移行する。

[0267] また、ステップＳ３１０で決定部３０３は、乱数値ｓ［ｉ］を＋ｓと決定する。次のステップＳ３１１で決定部３０３は、ウィンドウ補正値ｔ［ｉ］を０と決定する。

なお、ステップＳ３１０とＳ３１１の実行順序は入れ換えられてもよく、ステップＳ３１０とＳ３１１は並列に実行されてもよい。ステップＳ３１０とＳ３１１の実行後、処理はステップＳ３１５に移行する。

[0268] また、ステップＳ３１２で決定部３０３は、乱数値ｓ［ｉ］を－ｓと決定する。次のステップＳ３１３で決定部３０３は、ウィンドウ補正値ｔ［ｉ］を－２$^{k+b}$と決定する。

そして、続くステップＳ３１４で決定部３０３は、ウィンドウ補正値ｔ［ｉ］＝－２$^{k+b}$の影響を相殺するために、１つ上位のウィンドウ値ｗ［ｉ＋１］に対するキャリー補正を行う。すなわち、決定部３０３は、ウィンドウ値ｗ［ｉ＋１］を１だけインクリメントする。

[0269] 以上のステップＳ３１２～Ｓ３１４の実行順序は任意に入れ換えられてもよく、ステップＳ３１２～Ｓ３１４は並列に実行されてもよい。ステップＳ３１２～Ｓ３１４の実行後、処理はステップＳ３１５に移行する。

[0270] そして、ステップＳ３１５では、決定部３０３が、式（９．３）に示す符号つき（ｋ＋ｂ）ビット値である補正済み差分値ｄｉｆｆを計算する。また、決定部３０３は、補正済み差分値ｄｉｆｆを判断部３０２にフィードバックする。

$$diff=d_H-s[i]+t[i] \quad (9.3)$$

[0271] 続くステップＳ３１６で決定部３０３は、補正済み差分値ｄｉｆｆの上位ｋビットをウィンドウ値ｗ［ｉ］として設定する。すなわち、ウィンドウ値ｗ［ｉ］は式（９．４）のとおりである。なお、符号つきＮビット値の定義より、ウィンドウ値ｗ［ｉ］の符号は、補正済み差分値ｄｉｆｆの符号に等しい。

$$w[i]=diff[k+b-1]||\cdots||diff[b] \quad (9.4)$$

[0272] その後、ステップＳ３１７で処理部３０１は、ループ変数ｉの値が０か否かを判断する。ループ変数ｉの値が０でない場合（すなわちｉ＞０の場合）、処理はステップＳ３１８に移行する。他方、ループ変数ｉの値が０の場合、処理はステップＳ３１９に移行する。

[0273] ステップＳ３１８で判断部３０２は、補正済み差分値ｄｉｆｆの下位ｂビットを取り出し、取り出した符号つきｂビット値の2^k倍にダミー鍵ｅの（ｋｉ－１）ビット目からｋ（ｉ－１）ビット目までのｋビットを足した値を計算する。そして、判断部３０２は、計算した値を新たに符号つき（ｋ＋ｂ）ビット値d_Hとして記憶する。すなわち、判断部３０２は、次の判断に備えて、式（９．５）にしたがって符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=(diff[b-1]||\cdots||diff[0])2^k+(e[ki-1]||\cdots||e[k(i-1)]) \quad (9.5)$$

[0274] また、ステップＳ３１９で判断部３０２は、補正済み差分値ｄｉｆｆの下位ｂビットを新たに符号つき（ｋ＋ｂ）ビット値d_Hとして記憶する。すなわち、判断部３０２は、式（９．６）にしたがって符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=diff[b-1]||\cdots||diff[0] \quad (9.6)$$

[0275] ステップＳ３１８またはステップＳ３１９での符号つき（ｋ＋ｂ）ビット

値d_Hの更新後、処理はステップＳ３２０に移行する。そして、ステップＳ３２０で処理部３０１は、ループ変数ｉを１だけデクリメントする。

[0276] また、次のステップＳ３２１で処理部３０１は、ループ変数ｉが０以上か否かを判断する。そして、ｉ≧０の場合、処理はステップＳ３０４に戻り、ｉ＜０の場合、処理はステップＳ３２２に移行する。

[0277] ステップＳ３２２で判断部３０２は、式（９．６）により得た符号つき（ｋ＋ｂ）ビット値d_Hを決定部３０３に通知する。すると、決定部３０３は、式（９．７）に示すように、符号つき（ｋ＋ｂ）ビット値d_Hを補正値ｃとして決定する。

$$c=d_H \tag{9.7}$$

[0278] そして、次のステップＳ３２３で決定部３０３は、決定したウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、決定した補正値ｃを補正部３１０に出力する。以上で図１１の処理は終了する。

[0279] 続いて、具体的な数値を例として図１２Ａ～１２Ｂを参照して、第１実施形態についてさらに詳しく説明する。

図９のステップＳ１０２で処理部３０１が得たダミー鍵ｅが、式（９．８）に示す１５ビットの値であるとする。なお、式（９．８）に対応する個人鍵ｄの具体例は、図１３とともに後述する。

$$e=(010110010010101)_2=11413 \tag{9.8}$$

[0280] 続いて、図９のステップＳ１０３に相当する図１１の処理が開始される。なお、図１２Ａ～１２Ｂの例では、ウィンドウサイズｋは３であり、乱数値ｓのビット長ｂは６であるとする。したがって、式（９．９）のとおりｍ＝３である。

$$m=(u-b)/k=(15-6)/3=3 \tag{9.9}$$

[0281] 図１１の処理が開始されると、ステップＳ３０１で判断部３０２は、式（９．１０）のように符号つき（ｋ＋ｂ）ビット値d_Hを初期化する。

$$d_H=e[14]||\cdots||e[6]=(010110010)_2=178 \tag{9.10}$$

[0282] また、ステップＳ３０２で乱数生成部３０５が、ｂ（＝６）ビットの乱数

値ｓとして式（９．１１）の値を生成したとする。

$$s=(001101)_2=13 \tag{9.11}$$

すると、続くステップＳ３０３では、処理部３０１が式（９．９）よりループ変数ｉを２（＝ｍ－１）に初期化する。そして、ステップＳ３０４で判断部３０２は、式（９．１０）と（９．１１）より、判断基準値（d_H－ｓ）を式（９．１２）のとおり計算する。

$$d_H-s=178-13=165=(010100101)_2=(010 \,||\, 100101)_2 \tag{9.12}$$

[0283] 式（９．１２）の判断基準値（d_H－ｓ）は正である。また、判断基準値（d_H－ｓ）のＭＳＢは０である。よって、判断基準値（d_H－ｓ）は範囲Ｒ３に属する。

[0284] よって、決定部３０３は、ステップＳ３１０で乱数値ｓ［２］を式（９．１３）のとおり＋ｓに決定し、ステップＳ３１１でウィンドウ補正値ｔ［２］を式（９．１４）のとおり０に決定する。また、ウィンドウ補正値ｔ［２］が０なので、キャリー補正は行われない。

$$s[2]=+s=13=(001101)_2 \tag{9.13}$$

$$t[2]=0=(0000000000)_2 \tag{9.14}$$

[0285] そして、ステップＳ３１５で決定部３０３は、式（９．３）にしたがい、具体的には式（９．１５）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[2]+t[2] \\ &=178-13+0 \\ &=165 \\ &=2\times 2^6+37 \\ &=(010100101)_2 \\ &=(010 \,||\, 100101)_2 \end{aligned} \tag{9.15}$$

[0286] さらに、ステップＳ３１６で決定部３０３は、式（９．１６）のようにウィンドウ値ｗ［２］を計算する。なお、ここで得られるウィンドウ値ｗ［２］は、まだ確定していない。なぜなら、後にループ変数ｉが１となった段階

で、キャリー補正によってウィンドウ値ｗ［２］がインクリメントまたはデクリメントされる潜在的可能性があるからである。

$$w[2]=diff[8]||diff[7]||diff[6]=(010)_2=2 \quad (9.16)$$

[0287] また、ｉ＝２なので、処理がステップＳ３１８に進む。ステップＳ３１８で判断部３０２は、式（９．５）にしたがい、具体的には式（９．１７）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$\begin{aligned} d_H&=(diff[5]||\cdots||diff[0])2^3+(e[5]||\cdots||e[3]) \\ &=37\times 2^3+(010)_2 \\ &=296+2 \\ &=298 \\ &=(100101010)_2 \end{aligned} \quad (9.17)$$

[0288] その後、ステップＳ３２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝１となる。そして、ｉ≧０なので処理はステップＳ３２１からステップＳ３０４に戻る。

[0289] ステップＳ３０４では、判断部３０２が、式（９．１７）のように更新された符号つき（ｋ＋ｂ）ビット値d_Hを用いて、判断基準値（d_H－ｓ）を式（９．１８）のとおり計算する。

$$d_H-s=298-13=285=(100\ ||\ 011101)_2 \quad (9.18)$$

式（９．１８）の判断基準値（d_H－ｓ）は、正であり、ＭＳＢが１である。よって、判断基準値（d_H－ｓ）は範囲Ｒ４に属する。

[0290] よって、決定部３０３は、ステップＳ３１２で乱数値ｓ［１］を式（９．１９）のとおり－ｓに決定し、ステップＳ３１３でウィンドウ補正値ｔ［１］を式（９．２０）のとおり-2^{k+b}に決定する。

$$s[1]=-s=-13=-(001101)_2 \quad (9.19)$$

$$t[1]=-2^{3+6}=-512=-(1000000000)_2 \quad (9.20)$$

[0291] そして、ウィンドウ補正値ｔ［１］が非ゼロなので、ステップＳ３１４で決定部３０３はキャリー補正を行う。すなわち、決定部３０３は、式（９．１６）で求めたウィンドウ値ｗ［２］に１を足す。その結果、ウィンドウ値

ｗ［２］は式（９．２１）の値に確定する。

$$w[2]=2+1=3 \tag{9.21}$$

[0292] さらに、ステップＳ３１５で決定部３０３は、式（９．３）にしたがい、具体的には式（９．２２）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[1]+t[1] \\ &=298+13-512 \\ &=-201 \\ &=-3\times 2^6-9 \\ &=-(011001001)_2 \\ &=-(011\ ||\ 001001)_2 \end{aligned} \tag{9.22}$$

[0293] さらに、ステップＳ３１６で決定部３０３は、式（９．２３）のようにウィンドウ値ｗ［１］を計算する。なお、ここで得られるウィンドウ値ｗ［１］は、後にキャリー補正される潜在的可能性があるので、まだ確定していない。

$$w[1]=diff[8]||diff[7]||diff[6]=-(011)_2=-3 \tag{9.23}$$

[0294] また、ｉ＝１なので処理がステップＳ３１８に進む。そして、ステップＳ３１８で判断部３０２は、式（９．５）にしたがい、具体的には式（９．２４）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$\begin{aligned} d_H&=(diff[5]||\cdots||diff[0])2^3+(e[2]||\cdots||e[0]) \\ &=-9\times 2^3+(101)_2 \\ &=-72+5 \\ &=-67 \\ &=-(001000011)_2 \end{aligned} \tag{9.24}$$

[0295] その後、ステップＳ３２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝０となる。そして、ｉ≧０なので処理はステップＳ３２１からステップＳ３０４に戻る。

[0296] ステップＳ３０４では、判断部３０２が、式（９．２４）のように更新さ

れた符号つき（ｋ＋ｂ）ビット値d_Hを用いて、判断基準値（d_H－ｓ）を式（９．２５）のとおり計算する。

$$d_H-s=-67-13=-80=-(001\ ||\ 010000)_2 \qquad (9.25)$$

式（９．２５）の判断基準値（d_H－ｓ）は、負であり、ＭＳＢが０である。よって、符号つき（ｋ＋ｂ）ビット値d_Hは範囲Ｒ２に属する。

[0297] よって、決定部３０３は、ステップＳ３０８で乱数値ｓ［０］を式（９．２６）のとおり－ｓに決定し、ステップＳ３０９でウィンドウ補正値ｔ［０］を式（９．２７）のとおり０に決定する。

$$s[0]=-s=-13=-(001101)_2 \qquad (9.26)$$

$$t[0]=0=(0000000000)_2 \qquad (9.27)$$

[0298] なお、ウィンドウ補正値ｔ［０］が０なので当然キャリー補正も行われない。換言すれば、決定部３０３は、キャリー補正値を暗黙的に０に設定している。よって、１つ上位のウィンドウ値ｗ［１］は、式（９．２３）に示した－３という値に確定する。

[0299] さらに、ステップＳ３１５で決定部３０３は、式（９．３）にしたがい、具体的には式（９．２８）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[0]+t[0] \\ &=-67+13+0 \\ &=-54 \\ &=-(000\ ||\ 110110)_2 \end{aligned} \qquad (9.28)$$

[0300] さらに、ステップＳ３１６で決定部３０３は、式（９．２９）のようにウィンドウ値ｗ［０］を計算する。なお、ウィンドウ値ｗ［０］は、最下位のウィンドウ値なのでキャリー補正されることはなく、ここで値が確定する。

$$w[0]=diff[8]||diff[7]||diff[6]=-(000)_2=(000)_2=0 \qquad (9.29)$$

[0301] また、ｉ＝０なので処理がステップＳ３１９に進む。そして、ステップＳ３１９で判断部３０２は、式（９．６）にしたがい、具体的には式（９．３０）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=(diff[5]||\cdots||diff[0])=-(110110)_2=-54 \quad (9.30)$$

[0302] その後、ステップＳ３２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝－１となる。よって、ｉ＜０なので処理はステップＳ３２１からステップＳ３２２に進む。したがって、ステップＳ３２２で決定部３０３は、式（９．３１）のとおり補正値ｃを得る。

$$c=d_H=-54 \quad (9.31)$$

[0303] 最後に、ステップＳ３２３で決定部３０３は、式（９．３２）に示すウィンドウ列ｗ［ｉ］と、式（９．３３）に示す乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、式（９．３１）に示す補正値ｃを補正部３１０に出力する。式（９．３２）は、式（９．２１）、（９．２３）、（９．２９）から明らかであり、式（９．３３）は、式（９．１３）、（９．１９）、（９．２６）から明らかである。

$$w[2]=3,\ w[1]=-3,\ w[0]=0 \quad (9.32)$$

$$s[2]=13,\ s[1]=-13,\ s[0]=-13 \quad (9.33)$$

[0304] 以上のようにして、図９のステップＳ１０３に相当する図１１の処理が終了すると、図９のステップＳ１０４で取得部３０６が点Ａのｘｙ座標を取得する。そして、ステップＳ１０５でスカラ倍算部３０７が、$-2 \leqq h \leqq 2^{k-1}+1=5$なる範囲の各インデックスｈについて、乱数ｓ＝１３（式（９．１１）参照）に応じたスカラ倍点情報として、式（９．３４）のランダム化テーブルデータｔａｂ［ｈ］を生成する。

$$tab[h]=(2^b h+s)A=(2^6 h+13)A=(64h+13)A \quad (9.34)$$

[0305] そして、スカラ倍点情報格納部３０８は、生成されたランダム化テーブルデータｔａｂ［ｈ］を、インデックスｈと対応づけて格納する。

[0306] 図１３は、以上のようにして得られた、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを示すとともに、スカラ倍点情報格納部３０８がランダム化テーブルデータｔａｂ［ｈ］を保持するテーブル１０４を示す図である。

[0307] 図１１～１２Ｂを参照して上記で説明した処理によれば、図１３に示すとおり、式（８．９）が成立する。具体的には、図１２Ａ～１２Ｂの例におけ

るｋ＝３、ｂ＝６、ｍ＝３、ｅ＝１１４１３という値および式（９．３１）～（９．３３）に示した値を式（８．９）の各変数に代入すると、下記の式（９．３５）が得られる。

$$c+2^6w[0]+2^0s[0]+2^9w[1]+2^3s[1]+2^{12}w[2]+2^6s[2]$$
$$=-54+64\times0+1\times(-13)+512\times(-3)+8\times(-13)+4096\times3+64\times13$$
$$=11413 \qquad (9.35)$$

[0308] また、式（９．３４）に示したように、図１３のテーブル１０４には、－２≦ｈ≦５なる範囲の各インデックスｈに対応して、それぞれ、スカラ倍点－１１５Ａ、－５１Ａ、１３Ａ、７７Ａ、１４１Ａ、２０５Ａ、２６９Ａ、３３３Ａのｘｙ座標が記憶されている。

[0309] よって、図９のステップＳ１０６に相当する図１０の処理と、図９のステップＳ１０７～Ｓ１１１により、式（９．３６）のようにスカラ倍点ｄＡのｘｙ座標が計算される。

$$dA=2(2^3(2^3(2^3(0)+tab[w[2]])-tab[-w[1]])-tab[-w[0]]+cA)+d[0]A \quad (9.36)$$

[0310] すなわち、式（９．３６）における無限遠点Ｏは、図１０のステップＳ２０１の初期化に対応する。また、式（９．３６）における“2^3”は、図１０のステップＳ２０３～Ｓ２０６で行われるｋ（＝３）回の２倍算に対応する。

[0311] そして、式（９．３３）よりｓ［２］＝＋ｓなので、ｉ＝２のときステップＳ２０８が実行され、このことが式（９．３６）における“＋ｔａｂ［ｗ［２］］”に対応する。同様に、式（９．３３）よりｓ［１］＝－ｓなので、ｉ＝１のときステップＳ２０９が実行され、このことが式（９．３６）における“－ｔａｂ［－ｗ［１］］”に対応する。そして、式（９．３３）よりｓ［０］＝－ｓなので、ｉ＝０のときステップＳ２０９が実行され、このことが式（９．３６）における“－ｔａｂ［－ｗ［０］］”に対応する。

[0312] そして、式（９．３６）における“＋ｃＡ”は、図９のステップＳ１０７～Ｓ１０８に対応する。そして、式（９．３６）の右辺の最初の“２”は、図９のステップＳ１０９の２倍算に対応し、式（９．３６）の“＋ｄ［０］

Ａ”は、図９のステップＳ１１０～Ｓ１１１に対応する。

[0313] なお、この式（９．３６）の右辺において、ステップＳ１０９の２倍算の対象を示す部分を変形すると、次の式（９．３７）が得られる。式（９．３７）も、第１実施形態において式（８．９）が成立することを示している。

$$\begin{aligned}&2^3(2^3(2^3(0)+tab[3])-tab[3])-tab[0]+cA\\&=8(8(0+205A)-205A)-13A-54A\\&=8(1640A-205A)-67A\\&=8(1435A)-67A\\&=11413A\end{aligned} \tag{9.37}$$

[0314] そして、図１３に例示するように、式（９．８）のダミー鍵ｅは、式（９．３８）または（９．３９）に示す個人鍵ｄから得られる。図１３には、例として、式（９．３８）の個人鍵ｄを示してある。

$$d=(101100100101010)_2=22826 \tag{9.38}$$

$$d=(101100100101011)_2=22827 \tag{9.39}$$

[0315] 個人鍵ｄが式（９．３８）で示される場合、式（９．３８）よりｄ［０］＝０なので、式（９．３７）を式（９．３６）の右辺に代入すると、式（９．４０）が得られる。

$$2(11413A)+d[0]A=22826A+0=22826A=dA \tag{9.40}$$

[0316] また、個人鍵ｄが式（９．３９）で示される場合、式（９．３９）よりｄ［０］＝１なので、式（９．３７）を式（９．３６）の右辺に代入すると、式（９．４１）が得られる。

$$2(11413A)+d[0]A=22826A+A=22827A=dA \tag{9.41}$$

[0317] 以上から、第１実施形態によれば、図９～１１に示した処理により、確かに所望の点ｄＡが得られる。

[0318] さて、第１実施形態では、図９のステップＳ１０５に関して説明したとおり、スカラ倍点情報格納部３０８に格納されるスカラ倍点情報ｔａｂ［ｈ］に対応するインデックスｈの範囲は、$-2^{k-1}+1 \leqq h \leqq 2^{k-1}+1$である。よって、第１実施形態において、スカラ倍点情報格納部３０８が有するテーブルのエ

ントリ数は、（$2^{k-1}+4$）個である。

[0319] 図１４は、ウィンドウサイズｋが３の場合のテーブルデータのエントリ数に関して、第１実施形態と第３の比較例と第４の比較例を比べる図である。

図１４において、テーブル１０５は、第１実施形態においてスカラ倍点情報格納部３０８がスカラ倍点情報を記憶するためのテーブルである。ｋ＝３なので、テーブル１０５には、$-2 \leqq h \leqq 2^{3-1}+1=5$なる各インデックスｈについて、テーブルデータｔａｂ［ｈ］として点（$h \times 2^{b}+s$）Ａのｘｙ座標が格納されている。つまり、テーブル１０５のエントリ数は８個である。

[0320] また、テーブル１０６は、第４の比較例として説明した符号つきウィンドウ法においてウィンドウサイズｋが３の場合のテーブルである。テーブル１０６には、$0 \leqq h \leqq 2^{3-1}=4$なる各インデックスｈについて、テーブルデータｔａｂ［ｈ］として点ｈＡのｘｙ座標が格納されている。つまり、テーブル１０６のエントリ数は５個である。

[0321] また、テーブル１０７は、第３の比較例として説明したランダム化ウィンドウ法においてウィンドウサイズｋが３の場合のテーブルである。テーブル１０７には、$0 \leqq h \leqq 2^{3}-1=7$なる各インデックスｈについて、テーブルデータｔａｂ［ｈ］として点（$h \times 2^{b}+s$）Ａのｘｙ座標が格納されている。つまり、テーブル１０７のエントリ数は８個である。

[0322] 以上のテーブル１０５～１０７を比較すると、符号つきウィンドウ法のテーブル１０６は、エントリ数がテーブル１０５より少ない点で有利だが、符号つきウィンドウ法には、ＤＰＡ攻撃に対して脆弱であるという欠点がある。それに対し、第１実施形態によるテーブル１０５は、テーブル１０６と比較すると、ｈ＝－２、ｈ＝－１、$h=2^{k-1}+1$という３つのインデックスに対応する３つのエントリが多い。しかし、第１実施形態は、ＤＰＡ攻撃に対しても安全な方法を提供するという点で、符号つきウィンドウ法よりも優れている。また、エントリ数が2^{k-1}のオーダであるという点では、第１実施形態によるテーブル１０５は、テーブル１０６と同じである。

[0323] また、ランダム化ウィンドウ法のテーブル１０７のエントリ数は2^kのオーダである。上記のようにｋ＝３の場合にはテーブル１０６と１０７のエントリ数は同じだが、ｋ≧４の場合は、オーダの違いの影響で、第１実施形態のテーブルのエントリ数はランダム化ウィンドウ法のテーブルのエントリ数よりも小さい。よって、第１実施形態は、ＰＡ攻撃に対する安全性とメモリ使用量の抑制をともに実現するという優れた効果を有する。

[0324] ところで、テーブル１０６と比べてテーブル１０５で増加している３つのエントリ、すなわち、ｈ＝－２、ｈ＝－１、ｈ＝2^{k-1}＋１という３つのインデックスに対応するエントリは、乱数により生じる誤差を吸収する役割を果たす。以下では、なぜスカラ倍点情報のインデックスｈの範囲が－２≦ｈ≦2^{k-1}＋１であるのかという理由を、図１１のステップＳ３０４の判断の意味とともに、図１５～１６を参照しながら説明する。

[0325] 上記のとおり、図１１のステップＳ３０４で決定部３０３は、判断基準値（d_H－ｓ）を計算する。この判断基準値（d_H－ｓ）の計算は、意味的には、「乱数値ｓ［ｉ］が＋ｓである」という仮定の下でウィンドウ値ｗ［ｉ］を見積もることに相当する。その理由は次のとおりである。

[0326] 前処理を考慮して、図４における個人鍵ｄをダミー鍵ｅに置き換えたうえで図４を解釈すると、符号つき（ｋ＋ｂ）ビット値d_Hは、図４の（ｗ［ｉ］||ｓ［ｉ］）に相当する。よって、ウィンドウ値ｗ［ｉ］は、符号つき（ｋ＋ｂ）ビット値d_Hから乱数値ｓ［ｉ］を引いた値の上位ｋビットである。したがって、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、符号つき（ｋ＋ｂ）ビット値d_Hから乱数値ｓを引いた判断基準値（d_H－ｓ）の上位ｋビットである。

[0327] そして、第１実施形態では、図３に示すように、処理部３０１はウィンドウ値ｗ［ｉ］の絶対値を2^{k-1}以下にしようとし、それにより、スカラ倍点情報格納部３０８のメモリ消費量の抑制が図られる。そこで、第１実施形態の処理部３０１は、具体的には、「乱数値ｓ［ｉ］が＋ｓである」という仮定の下でウィンドウ値ｗ［ｉ］を見積もり、見積もったウィンドウ値ｗ［ｉ］

に応じて、仮定が妥当であるか否かを判断する。そして、処理部３０１は、判断の結果に応じて、上記仮定を正式に採用して乱数値ｓ［ｉ］を＋ｓに決定するか、または乱数値ｓ［ｉ］を－ｓと決定する。

[0328] 以上説明したようなｓ［ｉ］＝＋ｓという仮定の下での見積もりと、見積もりの結果に応じたウィンドウ値ｗ［ｉ］と乱数値ｓ［ｉ］の決定と、スカラ倍点情報のインデックスｈの範囲の関係について、続いて、図１５～１６を参照して説明する。

[0329] 図１５は、第１実施形態においてスカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。図１５は、ウィンドウサイズｋが２であり、かつ乱数値ｓの長さｂが３である場合の例である。

[0330] 図１１のステップＳ３０４に関して説明した範囲Ｒ１～Ｒ４の定義より、符号つき５（＝ｋ＋ｂ）ビット値である判断基準値（d_H－ｓ）は、式（９．４２）を満たす場合（より具体的には、式（９．４３）を満たす場合）に、範囲Ｒ１に属する。

$$-2^{k+b}+1 \leqq d_H-s \leqq -2^{k+b-1} \quad (9.42)$$

$$-2^5+1=-(100000)_2+1 \leqq d_H-s \leqq -2^4=-(10000)_2 \quad (9.43)$$

[0331] 図１５には、範囲Ｒ１に属する場合の判断基準値（d_H－ｓ）を、矩形Ｅ１０１により示してある。なお、矩形Ｅ１０１の左端の白丸は、－３２（＝－２^5＝－２$^{k+b}$）が範囲Ｒ１に含まれないことを示し、矩形Ｅ１０１の右端の黒丸は、－１６（＝－２^4＝－２$^{k+b-1}$）が範囲Ｒ１に含まれることを示す。他の矩形の左端と右端の白丸または黒丸も、同様の意味を示す。

[0332] また、判断基準値（d_H－ｓ）は、式（９．４４）を満たす場合（より具体的には、式（９．４５）を満たす場合）に、範囲Ｒ２に属する。図１５には、範囲Ｒ２に属する場合の判断基準値（d_H－ｓ）を、矩形Ｅ１０２により示してある。

$$-2^{k+b-1}+1 \leqq d_H-s \leqq -1 \quad (9.44)$$

$$-2^4+1=-(01111)_2 \leqq d_H-s \leqq -1=-(00001)_2 \quad (9.45)$$

[0333] また、判断基準値（$d_H - s$）は、式（９．４６）を満たす場合（より具体的には、式（９．４７）を満たす場合）に、範囲Ｒ３に属する。図１５には、範囲Ｒ３に属する場合の判断基準値（$d_H - s$）を、矩形Ｅ１０３により示してある。

$$0 \leqq d_H - s \leqq 2^{k+b-1} - 1 \quad (9.46)$$

$$0 \leqq d_H - s \leqq 2^4 - 1 = (01111)_2 \quad (9.47)$$

[0334] また、判断基準値（$d_H - s$）は、式（９．４８）を満たす場合（より具体的には、式（９．４９）を満たす場合）に、範囲Ｒ４に属する。図１５には、範囲Ｒ４に属する場合の判断基準値（$d_H - s$）を、矩形Ｅ１０４により示してある。

$$2^{k+b-1} \leqq d_H - s \leqq 2^{k+b} - 1 \quad (9.48)$$

$$2^4 = (10000)_2 \leqq d_H - s \leqq 2^5 - 1 = (11111)_2 \quad (9.49)$$

[0335] 続いて、上記の矩形Ｅ１０１～Ｅ１０４が示す判断基準値（$d_H - s$）の範囲とウィンドウ値ｗ［ｉ］の範囲の関係について、範囲Ｒ３、Ｒ１、Ｒ２、Ｒ４の順で、説明していく。

[0336] 図１５の矩形Ｅ１０３に示すように、判断基準値（$d_H - s$）が範囲Ｒ３に属する場合、判断基準値（$d_H - s$）の符号は正であり、判断基準値（$d_H - s$）のＭＳＢの値は０である。また、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は判断基準値（$d_H - s$）の上位ｋビットである。よって、判断基準値（$d_H - s$）が範囲Ｒ３に属する場合、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、符号が正であり、絶対値が０以上2^{k-1}未満である。

[0337] つまり、判断基準値（$d_H - s$）が範囲Ｒ３に属する場合、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、ウィンドウ補正の必要がない。したがって、当然キャリー補正値も０である。そして、このように望ましいウィンドウ値ｗ［ｉ］が得られたのだから、ｓ［ｉ］＝＋ｓという仮定は妥当である。そのため、決定部３０３は、判断基準値（$d_H - s$）が範囲Ｒ３に属する場合、図１１のステップＳ３１０～Ｓ３１１の処理を行

う。

[0338] その結果、決定部３０３がステップＳ３１５で計算する補正済み差分値ｄｉｆｆの範囲は、図１５に矩形Ｅ１１３として示すとおり、矩形Ｅ１０３に対応する範囲Ｒ３とまったく同じである。したがって、ステップＳ３１６で補正済み差分値ｄｉｆｆの上位ｋビットとして得られるウィンドウ値ｗ［ｉ］は、０以上$2^{k-1}-1$以下（図１５のようにｋ＝２の場合、０以上１以下）である。

[0339] また、図１５の矩形Ｅ１０１に示すように、判断基準値（d_H-s）が範囲Ｒ１に属する場合、判断基準値（d_H-s）の絶対値は2^{k+b-1}以上である。つまり、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］の絶対値は、2^{k-1}以上である。よって、この場合、スカラ倍点情報格納部３０８のメモリ使用量を抑えるためには、ウィンドウ補正を行うことが適切である。そして、判断基準値（d_H-s）が範囲Ｒ１に属する場合、判断基準値（d_H-s）の符号は負なので、この場合のウィンドウ補正は、具体的には、正の数を足す補正である。

[0340] そして、符号つきｋビットのウィンドウ値ｗ［ｉ］に対して2^kを足す補正は、図４、１２Ａ、１２Ｂから理解されるように、符号つき（ｋ＋ｂ）ビット値（ｗ［ｉ］｜｜ｓ［ｉ］）に対して2^{k+b}を足す補正と同じことである。よって、判断基準値（d_H-s）が範囲Ｒ３に属する場合、図１１のステップＳ３０６に示すように、決定部３０３はウィンドウ補正値ｔ［ｉ］を2^{k+b}と決定する。なお、この正のウィンドウ補正値ｔ［ｉ］＝2^{k+b}は、１つ上位のウィンドウ値ｗ［ｉ＋１］への負のキャリー補正値（つまり－１）と相殺する。

[0341] そして、判断基準値（d_H-s）にウィンドウ補正値ｔ［ｉ］＝2^{k+b}を足した値は、式（９．４２）より式（９．５０）の範囲に含まれる。

$$-2^{k+b}+1+2^{k+b}=1 \leqq d_H-s+t[i] \leqq -2^{k+b-1}+2^{k+b}=2^{k+b-1} \qquad (9.50)$$

[0342] 式（９．５０）が示す１以上2^{k+b-1}以下という範囲は、図１５において矩形Ｅ１１１により示されている。この矩形Ｅ１１１が示す範囲は、両端点を

除いて、上述の矩形Ｅ１１３が示す範囲と同じである。つまり、判断基準値（d_H-s）が範囲Ｒ１に属する場合、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、ウィンドウ補正値ｔ［ｉ］＝2^{k+b}によって補正されることで、符号が正であり絶対値が2^{k-1}以下という好ましい値になる。

[0343] よって、判断基準値（d_H-s）が範囲Ｒ１に属する場合、ｓ［ｉ］＝＋ｓという仮定は妥当である。よって、決定部３０３は、図１１のステップＳ３０５に示すように、正式にｓ［ｉ］＝＋ｓと決定する。また、上記のとおり決定部３０３は、ステップＳ３０６でウィンドウ補正値ｔ［ｉ］を2^{k+b}に決定し、ステップＳ３０７では、ウィンドウ補正値ｔ［ｉ］と相殺するキャリー補正値－１を１つ上位のウィンドウ値ｗ［ｉ＋１］に足す。

[0344] その結果、決定部３０３がステップＳ３１５で計算する補正済み差分値ｄｉｆｆの範囲は、図１５の矩形Ｅ１１１が示す範囲（すなわち上記式（９．５０）の範囲である。よって、ステップＳ３１６で補正済み差分値ｄｉｆｆの上位ｋビットとして得られるウィンドウ値ｗ［ｉ］は、０以上2^{k-1}以下（図１５のようにｋ＝２の場合、０以上２以下）である。

[0345] また、図１５の矩形Ｅ１０２に示すように、判断基準値（d_H-s）が範囲Ｒ２に属する場合、判断基準値（d_H-s）の符号は負である。そして、上記のとおり、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、判断基準値（d_H-s）の上位ｋビットであるから、やはり負である。

[0346] ここで、第１実施形態で採用される図３のアプローチによれば、処理部３０１は、ウィンドウ値ｗ［ｉ］と乱数値ｓ［ｉ］の符号を揃えようとする。

つまり、判断基準値（d_H-s）が範囲Ｒ２に属する場合、判断部３０２は、「ｓ［ｉ］＝＋ｓという仮定は、負のウィンドウ値ｗ［ｉ］と適合しない」と判断する。すなわち、判断部３０２は、「ｓ［ｉ］＝＋ｓという仮定が妥当でなかった」と判断する。そこで、判断基準値（d_H-s）が範囲Ｒ２に属する場合、決定部３０３は、図１１のステップＳ３０８に示すように、乱

数値ｓ［ｉ］をーｓに決定する。

[0347] また、ｓ［ｉ］＝ーｓのとき、式（９．５１）が成立する。そして、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合、式（９．４４）と（９．５１）から、式（９．５２）が得られる。

$$d_H-s[i]=d_H+s=(d_H-s)+2s \quad (9.51)$$

$$-2^{k+b-1}+1+2s \leqq d_H-s[i] \leqq -1+2s \quad (9.52)$$

[0348] 図１５において、矩形Ｅ１０２を＋２ｓ移動させた矩形Ｅ１１２は、式（９．５１）の（d_H－ｓ［ｉ］）の範囲（すなわち式（９．５２）が示す範囲）を図示したものである。

[0349] ここで、乱数値ｓは０≦ｓ≦2^b－１である。また、ウィンドウサイズｋは２以上の任意の整数である。よって、式（９．５３）が成立する。

$$0 \leqq 2s \leqq 2^{b+1}-2 \leqq 2^{b+k-1}-2 \quad (9.53)$$

[0350] 式（９．５２）と（９．５３）より、乱数値ｓがいかに大きくても、矩形Ｅ１１２が示す範囲の最大値は2^{k+b-1}に満たない。また、式（９．５２）と（９．５３）より、乱数値ｓがいかに小さくても、矩形Ｅ１１２が示す範囲の最小値はー2^{k+b-1}より大きい。したがって、矩形Ｅ１１２が示す範囲に含まれる値のＭＳＢの値は０である。つまり、（d_H－ｓ［ｉ］）の上位ｋビットの絶対値は2^{k-1}未満である。

[0351] よって、（d_H－ｓ［ｉ］）の上位ｋビットが示す値は、ウィンドウ補正をしなくても、絶対値が2^{k-1}未満という、ウィンドウ値ｗ［ｉ］として好適な値である。よって、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合、決定部３０３は図１１のステップＳ３０９のとおり、ウィンドウ補正値ｔ［ｉ］を０に設定する。

[0352] そして、ｔ［ｉ］＝０なので、決定部３０３がステップＳ３１５で計算する補正済み差分値ｄｉｆｆの範囲は、図１５の矩形Ｅ１１２が表す範囲と同じである。

ところで、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合、上記のように決定部３０３はｓ［ｉ］＝ーｓと決定した。よって、図１０のステップＳ２

０７とＳ２０９に示すとおり、ウィンドウ演算部３０９は、ウィンドウ値ｗ［ｉ］ではなく、ウィンドウ値ｗ［ｉ］の符号を反転させた値－ｗ［ｉ］をインデックスとして用いて、スカラ倍点情報格納部３０８を参照する。

[0353] つまり、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合に決定部３０３が決定するウィンドウ値ｗ［ｉ］の符号を反転させた値－ｗ［ｉ］に対応するエントリが、スカラ倍点情報格納部３０８のテーブルには存在しなくてはならない。そして、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合のウィンドウ値ｗ［ｉ］は、図１５の矩形Ｅ１１２が示す範囲に含まれる値の上位ｋビットである。よって、矩形Ｅ１１２が示す範囲の符号を反転させた範囲に含まれる値の上位ｋビットが、テーブルのインデックスとして使われる。

[0354] 図１５では、矩形Ｅ１１２が示す範囲の符号を反転させた範囲を、矩形Ｅ１２２により示している。式（９．５２）とｔ［ｉ］＝０より、式（９．５４）が得られるので、矩形Ｅ１２２が示す範囲は、式（９．５５）のとおりである。

$$-2^{k+b-1}+1+2s \leqq d_H-s[i]+t[i]=diff \leqq -1+2s \quad (9.54)$$

$$-2s+1 \leqq -diff \leqq 2^{k+b-1}-1-2s \quad (9.55)$$

[0355] つまり、式（９．５５）と図１５の矩形Ｅ１２２が示すように、負数の上位ｋビットがインデックスとして使われうる。具体的には、－１がインデックスとして使われうる。その理由は次のとおりである。

[0356] 乱数値ｓは０≦ｓ≦2^{b}－１なので、－2^{b+1}＋２≦－２ｓ≦０である。よって、式（９．５５）における（－２ｓ＋１）が－2^{b+1}未満になることはないが、（－２ｓ＋１）が－2^{b}未満になることはありうる。そして、－2^{b}を符号つき（ｋ＋ｂ）ビット値として表現したビット列の上位ｋビットは、（ｋ－１）ビットの０の後に１というビットが続く、負の符号を持つビット列である。

[0357] したがって、矩形Ｅ１２２が示す範囲に含まれる値の上位ｋビット（すなわちインデックスとして使われる符号つきｋビット値）は、乱数値ｓの値によっては、－１（つまり、符号が負で、（ｋ－１）ビットの０の後に１ビッ

トの１が続くビット列）のこともある。そのため、第１実施形態では、スカラ倍点情報格納部３０８のテーブルには、－１というインデックスに対応づけられたテーブルデータｔａｂ［－１］＝（－１×２^{b}＋ｓ）Ａのエントリが設けられる。

[0358] なお、－０は＋０と等しいので、ウィンドウ値ｗ［ｉ］が０の場合は、乱数値ｓ［ｉ］が－ｓであろうと＋ｓであろうと、ウィンドウ演算部３０９がスカラ倍点情報格納部３０８を参照するのに用いるインデックスは０である。つまり、－ｄｉｆｆ＜０であっても、－ｄｉｆｆの上位ｋビットがすべて０ならば、インデックスは０である（正確には、キャリー補正の影響がない場合には、補正済み差分値ｄｉｆｆの上位ｋビットがすべて０ならば、補正済み差分値ｄｉｆｆの正負によらず、インデックスは０である）。

[0359] また、図１５の矩形Ｅ１０４に示すように、判断基準値（ｄ$_{H}$－ｓ）が範囲Ｒ４に属する場合、判断基準値（ｄ$_{H}$－ｓ）は２$^{k+b-1}$以上である。つまり、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］は、２$^{k-1}$以上である。

[0360] よって、この場合、スカラ倍点情報格納部３０８のメモリ使用量を抑えるためには、ウィンドウ補正を行うことが適切である。そして、判断基準値（ｄ$_{H}$－ｓ）が範囲Ｒ４に属する場合、判断基準値（ｄ$_{H}$－ｓ）が正なので、この場合のウィンドウ補正は、具体的には、負の数を足す補正である。

[0361] より具体的には、決定部３０３は、図１１のステップＳ３１３に示すように、ウィンドウ補正値ｔ［ｉ］を－２$^{k+b}$と決定する。この負のウィンドウ補正値ｔ［ｉ］＝－２$^{k+b}$は、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値（つまり＋１）と相殺する。

[0362] そして、判断基準値（ｄ$_{H}$－ｓ）にウィンドウ補正値ｔ［ｉ］＝－２$^{k+b}$を足した値は、式（９．４８）より式（９．５６）の範囲に含まれる。

$$2^{k+b-1}-2^{k+b}=-2^{k+b-1}\leqq d_H-s+t[i]\leqq 2^{k+b}-1-2^{k+b}=-1 \qquad (9.56)$$

[0363] 式（９．５６）の範囲に含まれるのは負数のみである。そして、ｓ［ｉ］＝＋ｓという仮定の下で見積もられるウィンドウ値ｗ［ｉ］を、絶対値が小

さくなるように補正した値は、判断基準値（d_H－ｓ）をウィンドウ補正値ｔ［ｉ］で補正した値の上位ｋビットである。したがって、式（９．５６）が成り立つとき、見積もったウィンドウ値ｗ［ｉ］を補正した値も負である。

[0364] よって、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合と同様に、判断部３０２は、「ｓ［ｉ］＝＋ｓという仮定は、補正されて負となったウィンドウ値ｗ［ｉ］と適合しない」と判断する。すなわち、判断部３０２は、「ｓ［ｉ］＝＋ｓという仮定が妥当でなかった」と判断する。

[0365] そこで、判断基準値（d_H－ｓ）が範囲Ｒ４に属する場合、決定部３０３は、図１１のステップＳ３１２に示すように、乱数値ｓ［ｉ］を－ｓに決定する。

また、ｓ［ｉ］＝－ｓのとき、上述の式（９．５１）が成立する。そして、判断基準値（d_H－ｓ）が範囲Ｒ４に属する場合、式（９．５１）と（９．５６）から、式（９．５７）が得られる。

$$-2^{k+b-1}+2s \leqq d_H-s[i]+t[i] \leqq -1+2s \qquad (9.57)$$

[0366] 図１５において、矩形Ｅ１０４を（$-2^{k+b}+2s$）移動させた矩形Ｅ１１４は、式（９．５７）の範囲を図示したものである。矩形Ｅ１１４が示す範囲は、左端点以外は矩形Ｅ１１２が示す範囲と同じである。よって、範囲Ｒ２に関する上記の議論と同様の議論が成り立つ。

[0367] すなわち、図１５の矩形Ｅ１２４は、矩形Ｅ１１４が示す範囲の符号を逆転した範囲を示す。そして、矩形Ｅ１２４が示す範囲の上位ｋビットが、テーブルのインデックスとして使われる。具体的には、インデックスとして使われる符号つきｋビット値は、乱数値ｓの値によっては、－１のこともある。

[0368] 以上、図１５を参照して説明したとおり、第１実施形態によれば、キャリー補正の影響を考慮しなければ、スカラ倍点情報格納部３０８でインデックスとして使われる値は、－１以上2^{k-1}以下である。したがって、キャリー補正の影響を考慮すれば、スカラ倍点情報格納部３０８でインデックスとして使われる値は、－２以上（$2^{k-1}+1$）以下である。そして、図１６は、第１

実施形態においてスカラ倍点情報格納部３０８でインデックスとして使われる値についてまとめた図である。なお、図１６も、図１５と同様に、ウィンドウサイズｋが２の場合の例を示す。

[0369] 図１６には、図１５と同じく、範囲Ｒ１に対応する矩形Ｅ１１１、範囲Ｒ２に対応する矩形Ｅ１１２とその符号を反転させた矩形Ｅ１２２、範囲Ｒ３に対応する矩形Ｅ１１３、範囲Ｒ４に対応する矩形Ｅ１１４とその符号を反転させた矩形Ｅ１２４を示した。図１５に関して説明したとおり、スカラ倍点情報格納部３０８でインデックスとして使われる値は、矩形Ｅ１１１、Ｅ１２２、Ｅ１１３およびＥ１２４がそれぞれ示す範囲に含まれる値の上位ｋビットである。また、図１６には、乱数値ｓの最大値である（2^b-1）の大きさを両矢印で示した。図１６に示すとおり、ウィンドウ値ｗ［ｉ］における大きさ１が、2^bの大きさに相当するので、乱数値ｓの最大値は、ウィンドウ値ｗ［ｉ］における大きさ１に満たない。

[0370] 矩形Ｅ１１１に対応するウィンドウ値ｗ［ｉ］は、そのままインデックスとして使われる。矩形Ｅ１１１に対応するウィンドウ値ｗ［ｉ］は、０以上2^{k-1}以下（ｋ＝２の場合、０以上２以下）である。

[0371] 矩形Ｅ１１２に対応するウィンドウ値ｗ［ｉ］は、そのままではインデックスとしては使われず、ウィンドウ値ｗ［ｉ］の符号を反転した値がインデックスとして使われる。すなわち、インデックスとして使われる値は、矩形Ｅ１２２が表す範囲に含まれる値の上位ｋビットである。

[0372] したがって、図１５に関して説明したように、矩形Ｅ１２２に対応してインデックスとして使われる値の最小値は－１である。また、矩形Ｅ１２２に対応してインデックスとして使われる値の最大値は、乱数値ｓに応じて異なるが、（$2^{k-1}-1$）か（$2^{k-1}-2$）である。図１６には、ウィンドウサイズｋが２の場合での具体例として、矩形Ｅ１２２に対応してインデックスとして使われる値が－１以上０以下の例が示されている。

[0373] 矩形Ｅ１１３に対応するウィンドウ値ｗ［ｉ］は、そのままインデックスとして使われる。矩形Ｅ１１３に対応するウィンドウ値ｗ［ｉ］は、０以上

（$2^{k-1}-1$）以下（ｋ＝２の場合、０以上１以下）である。

[0374] 矩形Ｅ１１４に対応するウィンドウ値ｗ［ｉ］は、そのままではインデックスとしては使われず、ウィンドウ値ｗ［ｉ］の符号を反転した値がインデックスとして使われる。すなわち、インデックスとして使われる値は、矩形Ｅ１２４が表す範囲に含まれる値の上位ｋビットである。

[0375] したがって、図１５に関して説明したように、矩形Ｅ１２４に対応してインデックスとして使われる値の最小値は－１である。また、矩形Ｅ１２４に対応してインデックスとして使われる値の最大値は、乱数値ｓに応じて異なるが、ｓ＝０の場合に最も大きく、すなわち2^{k-1}である。図１６には、ウィンドウサイズｋが２の場合での具体例として、矩形Ｅ１２４に対応してインデックスとして使われる値が－１以上０以下の例が示されている。

[0376] 以上から、補正済み差分値ｄｉｆｆの上位ｋビットを取り出すことで得られるウィンドウ値ｗ［ｉ］（すなわち、まだキャリー補正されていないウィンドウ値ｗ［ｉ］）に対応するインデックスの範囲は、図１６に示す－１以上2^{k-1}以下の範囲Ｕ１のとおりである。より具体的には、ｋ＝２の場合、範囲Ｕ１は、－１以上２以下の範囲である。

[0377] ところが、補正済み差分値ｄｉｆｆから得られたウィンドウ値ｗ［ｉ］は、後に＋１または－１のキャリー補正値により補正されることがある。よって、実際にスカラ倍点情報格納部３０８のインデックスとして使われる可能性がある値の範囲は、－２以上（$2^{k-1}+1$）以下である。

[0378] 図１６には、ウィンドウサイズｋが２で乱数値ｓのビット数ｂが３の場合のスカラ倍点情報格納部３０８のテーブルデータを図示してある。すなわち、図１６には、$-2 \leqq h \leqq 2^{k-1}+1=3$なる各インデックスｈと、インデックスｈに対応する値（$h \times 2^3+s$）Ａが示されている。

[0379] さて、続いて第２実施形態について説明する。第２実施形態は、スカラ倍点情報格納部３０８のインデックスとして使われる可能性がある値の範囲を第１実施形態よりも狭めることで、さらにメモリの節約を図る実施形態である。

[0380] そこで、第２実施形態に関しては、まず、どのようにしたらインデックスの範囲を狭めることが可能なのかについて、図１７～１８を参照して説明する。その後、図１９～２２を参照して第２実施形態における暗号処理装置３００の動作の詳細を説明する。

[0381] 図１７は、第２実施形態においてスカラ倍点情報格納部３０８でインデックスとして使われる値についてまとめた図である。図１７の形式は第１実施形態に関する図１６と同様である。以下、図１７と図１６を比較しながら説明する。

[0382] 図１６のとおり、第１実施形態においてキャリー補正される前のウィンドウ値ｗ［ｉ］が－１となる可能性がある理由は、判断基準値（d_H－ｓ）が範囲Ｒ２またはＲ４に属するとき、矩形Ｅ１２２またはＥ１２４が示す範囲の最小値が（－２ｓ＋１）だからである。具体的には、乱数値ｓが（2^b－１）のときに、補正済み差分値ｄｉｆｆの符号を反転させた－ｄｉｆｆの最小値が最も小さくなる（つまり、矩形Ｅ１２２またはＥ１２４の左端の位置が最も左になる）。すなわち、乱数値ｓが（2^b－１）のときに、－ｄｉｆｆの最小値（－２ｓ＋１）が（-2^{b+1}－１）となる。

[0383] そして、図１６と１７に示すように、ウィンドウ値ｗ［ｉ］における大きさ１は、補正済み差分値ｄｉｆｆにおける2^bの大きさに相当する。そこで、第１実施形態では、－ｄｉｆｆの最小値が（-2^{b+1}－１）の場合に対処するため、－１というインデックスに対応するエントリがスカラ倍点情報格納部３０８に作られる。さらに、キャリー補正が行われる場合に備えて、第１実施形態では、－２というインデックスに対応するエントリも、スカラ倍点情報格納部３０８に作られる。

[0384] つまり、キャリー補正がされる前の段階で、ｓ［ｉ］＝－ｓという乱数値に対応してｗ［ｉ］＝１というウィンドウ値が設定され、さらにキャリー補正値が＋１である場合、ｓ［ｉ］＝－ｓという乱数値とｗ［ｉ］＝２というウィンドウ値がペアになる。よって、ウィンドウ演算部３０９は、－ｗ［ｉ］＝－２というインデックスを用いてスカラ倍点情報格納部３０８を参照し

、ｔａｂ［－２］＝（－２×２^{b}＋ｓ）Ａなる点を得る。第１実施形態では、このような場合に備えて、スカラ倍算部３０７が、－２なるインデックスに対応するテーブルデータｔａｂ［－２］を生成し、スカラ倍点情報格納部３０８に格納する。

[0385] ところで、図１７に示すように、ウィンドウ値ｗ［ｉ］における大きさ１が、補正済み差分値ｄｉｆｆにおける２^{b}の大きさに相当するので、乱数値ｓの最大値は、ウィンドウ値ｗ［ｉ］における大きさ１に満たない。つまり、図１６の矩形Ｅ１２２とＥ１２４は０より左側の部分の大きさが２ｓだが、この０より左側の部分の大きさをｓにすることができれば、１つのエントリ（つまり－２というインデックスに対応するエントリ）の必要性をなくすことができる。

[0386] そこで、第２実施形態では、第１実施形態の図１６の矩形Ｅ１１１、Ｅ１１２、Ｅ１１３、Ｅ１１４が、それぞれ－ｓだけずれた図１７の矩形Ｅ２１１、Ｅ２１２、Ｅ２１３、Ｅ２１４に置き換えられる。

[0387] 上記のとおり、乱数値ｓの最大値は、ウィンドウ値ｗ［ｉ］における大きさ１に満たない。よって、０以上２$^{k-1}$以下のウィンドウ値ｗ［ｉ］に対応する図１６の矩形Ｅ１１１を－ｓずらした図１７の矩形Ｅ２１１に対応するウィンドウ値ｗ［ｉ］の範囲は、ｓ＞０ならば０以上（２$^{k-1}$－１）以下であり、ｓ＝０ならば０以上２$^{k-1}$以下である。なお、－０＝＋０＝０なので、補正済み差分値ｄｉｆｆが正でも負でも、その上位ｋビットがすべて０であれば、キャリー補正前のウィンドウ値ｗ［ｉ］は０である。

[0388] そして、図１６の矩形Ｅ１１２を－ｓずらした図１７の矩形Ｅ２１２が示す範囲の符号を反転させた範囲は、図１７において矩形Ｅ２２２により示される。そして、矩形Ｅ２２２が示す範囲の最小値は（－ｓ＋１）である。よって、たとえ乱数値ｓが最大の（２^{b}－１）であったとしても、矩形Ｅ２２２が示す範囲の最小値の上位ｋビットはすべて０である。つまり、矩形Ｅ２２２が－１なるインデックスに対応することはない。

[0389] また、０以上（２$^{k-1}$－１）以下のウィンドウ値ｗ［ｉ］に対応する図１６

の矩形Ｅ１１３をーｓずらした図１７の矩形Ｅ２１３に対応するウィンドウ値ｗ［ｉ］の範囲は、０以上（$2^{k-1}-1$）以下である。

[0390] そして、図１６の矩形Ｅ１１４をーｓずらした図１７の矩形Ｅ２１４が示す範囲の符号を反転させた範囲は、図１７において矩形Ｅ２２４により示される。そして、矩形Ｅ２２４が示す範囲の最小値は（－ｓ＋１）である。よって、たとえ乱数値ｓが最大の（$2^{b}-1$）であったとしても、矩形Ｅ２２４が示す範囲の最小値の上位ｋビットはすべて０である。つまり、矩形Ｅ２２４がー１なるインデックスに対応することはない。

[0391] 以上から、第２実施形態では、補正済み差分値ｄｉｆｆの上位ｋビットを取り出すことで得られるウィンドウ値ｗ［ｉ］（すなわち、まだキャリー補正されていないウィンドウ値ｗ［ｉ］）に対応するインデックスの範囲は、図１７の範囲Ｕ２のとおりである。つまり、範囲Ｕ２は、０以上2^{k-1}以下であり、ｋ＝２の場合、０以上２以下である。

[0392] したがって、キャリー補正を考慮すると、第２実施形態で使われるインデックスはー１以上（$2^{k-1}+1$）以下であり、ｋ＝２の場合、－１以上３以下である。図１７には、ウィンドウサイズｋが２で乱数値ｓのビット数ｂが３の場合のスカラ倍点情報格納部３０８のテーブルデータを図示してある。すなわち、図１７には、$-1 \leqq h \leqq 2^{k-1}+1=3$なる各インデックスｈと、インデックスｈに対応する値（$h \times 2^{3}+s$）Ａが示されている。

[0393] さて、続いて、図１７の矩形Ｅ２１１、Ｅ２１２、Ｅ２１３、およびＥ２１４に対応して決定部３０３がウィンドウ値ｗ［ｉ］を決定するためには、具体的には判断部３０２が何を基準にして判断を行えばよいのかを、図１８を参照して説明する。図１８は、第２実施形態においてスカラ倍点情報格納部３０８のインデックスとして使われる値の範囲について模式的に説明する図である。図１８の形式は第１実施形態に関する図１５と同様である。

[0394] 結論から述べれば、第２実施形態では、判断部３０２は符号つき（ｋ＋ｂ）ビット値d_Hそのものを、判断の基準として用いる。そして、符号つき（ｋ＋ｂ）ビット値d_Hの値が範囲Ｒ１～Ｒ４のいずれに属するのかに応じて、決

定部３０３は異なる処理を行う。

[0395] すなわち、図１８の矩形Ｅ２０１は、範囲Ｒ１に属する場合の符号つき（ｋ＋ｂ）ビット値d_Hを示す。矩形Ｅ２０１が示す範囲Ｒ１は、式（１０．１）のとおりであり、図１８のようにウィンドウサイズｋが２で乱数値ｓの長さｂが３の場合は、具体的には式（１０．２）のとおりである。

$$-2^{k+b}+1 \leqq d_H \leqq -2^{k+b-1} \quad (10.1)$$

$$-2^5+1=-(11111)_2 \leqq d_H \leqq -2^4=-(10000)_2 \quad (10.2)$$

[0396] また、矩形Ｅ２０２は、範囲Ｒ２に属する場合の符号つき（ｋ＋ｂ）ビット値d_Hを示す。矩形Ｅ２０２が示す範囲Ｒ２は、式（１０．３）のとおりであり、図１８の例では具体的には式（１０．４）のとおりである。

$$-2^{k+b-1}+1 \leqq d_H \leqq -1 \quad (10.3)$$

$$-2^4+1=-(01111)_2 \leqq d_H \leqq -1=-(00001)_2 \quad (10.4)$$

[0397] そして、矩形Ｅ２０３は、範囲Ｒ３に属する場合の符号つき（ｋ＋ｂ）ビット値d_Hを示す。矩形Ｅ２０３が示す範囲Ｒ３は、式（１０．５）のとおりであり、図１８の例では具体的には式（１０．６）のとおりである。

$$0 \leqq d_H \leqq 2^{k+b-1}-1 \quad (10.5)$$

$$0 \leqq d_H \leqq 2^4-1=(01111)_2 \quad (10.6)$$

[0398] また、矩形Ｅ２０４は、範囲Ｒ４に属する場合の符号つき（ｋ＋ｂ）ビット値d_Hを示す。矩形Ｅ２０４が示す範囲Ｒ４は、式（１０．７）のとおりであり、図１８の例では具体的には式（１０．８）のとおりである。

$$2^{k+b-1} \leqq d_H \leqq 2^{k+b}-1 \quad (10.7)$$

$$2^4=(10000)_2 \leqq d_H \leqq 2^5-1=(11111)_2 \quad (10.8)$$

[0399] 続いて、矩形Ｅ２０１～Ｅ２０４が示す符号つき（ｋ＋ｂ）ビット値d_Hの範囲とウィンドウ値ｗ［ｉ］の範囲の関係について、範囲Ｒ３、Ｒ１、Ｒ２、Ｒ４の順で、図１５と比較しながら説明していく。

[0400] 第１実施形態において、決定部３０３は、判断基準値（d_H－ｓ）が範囲Ｒ３に属する場合に、乱数値ｓ［ｉ］を＋ｓに決定し、ウィンドウ補正値ｔ［ｉ］を０に決定する。第２実施形態においては、決定部３０３は、符号つき

（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に属する場合に、同様に乱数値ｓ［ｉ］を＋ｓに決定し、ウィンドウ補正値ｔ［ｉ］を０に決定する。

[0401] ここで、補正済み差分値ｄｉｆｆは、第２実施形態においても第１実施形態の式（９．３）と同様に定義される。よって、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に属する場合、式（１０．５）、ｓ［ｉ］＝＋ｓ、およびｔ［ｉ］＝０から、式（１０．９）が得られる。

$$0-s+0 \leqq diff=d_H-s[i]+t[i] \leqq 2^{k+b-1}-1-s+0 \quad (10.9)$$

[0402] そして、式（１０．９）が表す補正済み差分値ｄｉｆｆの範囲は、図１８の矩形Ｅ２１３のとおりである。図１８の矩形Ｅ２１３は図１７の矩形Ｅ２１３と同じであるから、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に属する場合、キャリー補正の影響がなければ、インデックスは図１７に示す所望の範囲Ｕ２に収まる。

[0403] 続いて範囲Ｒ１について説明する。第１実施形態では、判断基準値（d_H－ｓ）が範囲Ｒ１に属する場合に、決定部３０３は、ウィンドウ補正値ｔ［ｉ］を2^{k+b}と決定し、１つ上位のウィンドウ値ｗ［ｉ＋１］へのキャリー補正値を－１と決定し、乱数値ｓ［ｉ］を＋ｓに決定する。第２実施形態では、決定部３０３は、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に属する場合に、ウィンドウ補正値ｔ［ｉ］とキャリー補正値と乱数値ｓ［ｉ］を同様に決定する。

[0404] すると、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に属する場合、式（９．３）の補正済み差分値ｄｉｆｆの定義、式（１０．１）、ｓ［ｉ］＝＋ｓ、およびｔ［ｉ］＝2^{k+b}から、式（１０．１０）が得られる。

$$-2^{k+b}+1-s+2^{k+b}=-s+1 \leqq diff=d_H-s[i]+t[i] \leqq -2^{k+b-1}-s+2^{k+b}=2^{k+b-1}-s \quad (10.10)$$

[0405] そして、式（１０．１０）が表す補正済み差分値ｄｉｆｆの範囲は、図１８の矩形Ｅ２１１のとおりである。図１８の矩形Ｅ２１１は図１７の矩形Ｅ２１１と同じであるから、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に属する場合も、キャリー補正の影響がなければ、インデックスは図１７に示す所望の範囲Ｕ２に収まる。

[0406] 続いて範囲Ｒ２について説明する。第１実施形態では、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合に、決定部３０３は、乱数値ｓ［ｉ］を－ｓに決定し、ウィンドウ補正値ｔ［ｉ］を０に設定する。第２実施形態では、判断基準値（d_H－ｓ）が範囲Ｒ２に属する場合に、決定部３０３は、同様に乱数値ｓ［ｉ］を－ｓに決定し、ウィンドウ補正値ｔ［ｉ］を０に設定する。

[0407] すると、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ２に属する場合、式（９．３）の補正済み差分値ｄｉｆｆの定義、式（１０．３）、ｓ［ｉ］＝－ｓ、およびｔ［ｉ］＝０から、式（１０．１１）が得られる。

$$-2^{k+b-1}+1+s+0 \leqq diff=d_H-s[i]+t[i] \leqq -1+s+0 \qquad (10.11)$$

[0408] そして、式（１０．１１）が表す補正済み差分値ｄｉｆｆの範囲は、図１８の矩形２１２のとおりである。図１８の矩形Ｅ２１２は図１７の矩形Ｅ２１２と同じである。よって、矩形Ｅ２１２が表す式（１０．１１）の範囲の符号を反転させた範囲（すなわち矩形Ｅ２２２が表す範囲）は、式（１０．１２）のとおりである。

$$-s+1 \leqq -diff \leqq 2^{k+b-1}-1-s \qquad (10.12)$$

[0409] よって、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ２に属する場合も、キャリー補正の影響がなければ、インデックスは図１７に示す所望の範囲Ｕ２に収まる。

[0410] 続いて範囲Ｒ４について説明する。第１実施形態では、判断基準値（d_H－ｓ）が範囲Ｒ４に属する場合に、決定部３０３は、ウィンドウ補正値ｔ［ｉ］を-2^{k+b}と決定し、１つ上位のウィンドウ値ｗ［ｉ＋１］へのキャリー補正値を＋１と決定し、乱数値ｓ［ｉ］を－ｓと決定する。第２実施形態では、決定部３０３は、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ４に属する場合に、ウィンドウ補正値ｔ［ｉ］とキャリー補正値と乱数値ｓ［ｉ］を同様に決定する。

[0411] すると、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ４に属する場合、式（９．３）の補正済み差分値ｄｉｆｆの定義、式（１０．７）、ｓ［ｉ］＝－ｓ、およびｔ［ｉ］＝-2^{k+b}から、式（１０．１３）が得られる。

$$2^{k+b-1}+s-2^{k+b}=-2^{k+b-1}+s\leqq diff=d_H-s[i]+t[i]\leqq 2^{k+b-1}+s-2^{k+b}=s-1 \quad (10.13)$$

[0412] そして、式（１０．１３）が表す補正済み差分値ｄｉｆｆの範囲は、図１８の矩形Ｅ２１４のとおりである。図１８の矩形Ｅ２１４は図１７の矩形Ｅ２１４と同じである。よって、矩形Ｅ２１４が表す式（１０．１３）の範囲の符号を反転させた範囲（すなわち矩形Ｅ２２４が表す範囲）は、式（１０．１４）のとおりである。

$$-s+1\leqq -diff\leqq 2^{k+b-1}-s \quad (10.14)$$

[0413] よって、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ４に属する場合も、キャリー補正の影響がなければ、インデックスは図１７に示す所望の範囲Ｕ２に収まる。

[0414] 以上図１６と１７を参照して説明したことをまとめれば、第２実施形態では、判断部３０２が、符号つき（ｋ＋ｂ）ビット値d_Hから乱数値ｓを引いた値（d_H－ｓ）ではなく、符号つき（ｋ＋ｂ）ビット値d_Hそのものの値を判断基準として用いる。それにより、第２実施形態では、「－ｓなる乱数値ｓ［ｉ］に対応するウィンドウ値ｗ［ｉ］が、１に設定され、さらにキャリー補正によって２に決定される」という事態が起こらなくなり、スカラ倍点情報格納部３０８のインデックスとして－２は不要となる。つまり、第２実施形態によれば、判断部３０２の判断基準を変えるだけで、スカラ倍点情報格納部３０８のメモリ消費量のさらなる削減が達成される。

[0415] 以下では、以上説明した第２実施形態について、さらに具体的に説明する。

図１９は、第２実施形態において暗号処理装置３００がウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを決定する処理のフローチャートである。すなわち、図１９は、第２実施形態における、図９のステップＳ１０３のフローチャートである。

[0416] 図１９のステップＳ４０１～Ｓ４０３は、第１実施形態に関する図１１のステップＳ３０１～Ｓ３０３と同様であり、ステップＳ４０５～Ｓ４２３は、図１１のステップＳ４０５～Ｓ４２３と同様である。よって、これらのス

テップについては説明を割愛する。

[0417] 図１９において、第１実施形態に関する図１１と異なるのは、ステップＳ４０４である。ステップＳ４０４において判断部３０２は、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１～Ｒ４のいずれに含まれるかを判断する。範囲Ｒ１～Ｒ４は、第１実施形態と同様に、以下のように定義される。

・範囲Ｒ１：-2^{k+b-1}以下

・範囲Ｒ２：（$-2^{k+b-1}+1$）以上－１以下

・範囲Ｒ３：０以上（$2^{k+b-1}-1$）以下

・範囲Ｒ４：2^{k+b-1}以上

[0418] そして、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に含まれるとき、処理はステップＳ４０５に移行し、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ２に含まれるとき、処理はステップＳ４０８に移行する。また、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に含まれるとき、処理はステップＳ４１０に移行し、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ４に含まれるとき、処理はステップＳ４１２に移行する。

[0419] なお、ステップＳ４０４の処理の簡略化のため、判断部３０２は、符号つき（ｋ＋ｂ）ビット値d_Hの符号とＭＳＢの値をチェックすることで、符号つき（ｋ＋ｂ）ビット値d_Hがどの範囲に含まれるかを判断してもよい。

[0420] すなわち、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が負でＭＳＢの値が１ならば、範囲Ｒ１に含まれる。また、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が負でＭＳＢの値が０ならば、範囲Ｒ２に含まれる。そして、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が正でＭＳＢの値が０ならば、範囲Ｒ３に含まれる。また、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が正でＭＳＢの値が１ならば、範囲Ｒ４に含まれる。

[0421] 続いて、個人鍵ｄと乱数値ｓの数値例を含む図２０Ａ～２１を参照して、第２実施形態における図９、１０、１９の処理の具体例を説明する。

図９のステップＳ１０２で処理部３０１が得たダミー鍵ｅが、式（１０．１５）に示す１５ビットの値であるとする。なお、式（１０．１５）に対応

する個人鍵ｄの具体例は、図２１とともに後述する。

$$e=(010101011101100)_2=10988 \quad (10.15)$$

[0422] 続いて、図９のステップＳ１０３に相当する図１９の処理が開始される。なお、図２０Ａ～２０Ｂの例では、ウィンドウサイズｋは３であり、乱数値ｓのビット長ｂは６であるとする。したがって、式（９．９）と同じく、ｍ＝３である。

[0423] 図１９の処理が開始されると、ステップＳ４０１で判断部３０２は、式（１０．１６）のように符号つき（ｋ＋ｂ）ビット値d_Hを初期化する。

$$d_H=e[14]||\cdots||e[6]=(010101011)_2=171 \quad (10.16)$$

[0424] また、ステップＳ４０２で乱数生成部３０５が、ｂ（＝６）ビットの乱数値ｓとして式（１０．１７）の値を生成したとする。

$$s=(010010)_2=18 \quad (10.17)$$

[0425] すると、続くステップＳ４０３では、処理部３０１がループ変数ｉを２（＝ｍ－１）に初期化する。そして、ステップＳ４０４で判断部３０２は、式（１０．１６）の符号つき（ｋ＋ｂ）ビット値d_Hがどの範囲に属するかを判断する。式（１０．１６）より、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が正でＭＳＢの値が０なので、範囲Ｒ３に属する。

[0426] よって、決定部３０３は、ステップＳ４１０で乱数値ｓ［２］を式（１０．１８）のとおり＋ｓに決定し、ステップＳ４１１でウィンドウ補正値ｔ［２］を式（１０．１９）のとおり０に決定する。また、ウィンドウ補正値ｔ［２］が０なので、キャリー補正は行われない。

$$s[2]=+s=18=(010010)_2 \quad (10.18)$$

$$t[2]=0=(0000000000)_2 \quad (10.19)$$

[0427] そして、ステップＳ４１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１０．２０）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$diff=d_H-s[2]+t[2]$$

$$=171-18+0$$

$=153$

$=2\times 2^{6}+25$

$=(010011001)_{2}$

$=(010\ ||\ 011001)_{2}$ (10.20)

[0428] さらに、ステップＳ４１６で決定部３０３は、式（１０．２１）のようにウィンドウ値ｗ［２］を計算する。なお、ここで得られるウィンドウ値ｗ［２］は、後にキャリー補正によってインクリメントまたはデクリメントされる潜在的可能性があるので、まだ確定していない。

$w[2]=diff[8]||diff[7]||diff[6]=(010)_{2}=2$ (10.21)

[0429] また、ｉ＝２なので、処理がステップＳ４１８に進む。ステップＳ４１８で判断部３０２は、式（９．５）にしたがい、具体的には式（１０．２２）のようにして、符号つき（ｋ＋ｂ）ビット値$ｄ_{H}$を更新する。

$d_{H}=(diff[5]||\cdots||diff[0])2^{3}+(e[5]||\cdots||e[3])$

$=25\times 2^{3}+(101)_{2}$

$=200+5$

$=205$

$=(011001101)_{2}$ (10.22)

[0430] その後、ステップＳ４２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝１となる。そして、ｉ≧０なので処理はステップＳ４２１からステップＳ４０４に戻る。

[0431] ステップＳ４０４では、判断部３０２が、式（１０．２２）のように更新された符号つき（ｋ＋ｂ）ビット値$ｄ_{H}$がどの範囲に属するかを判断する。符号つき（ｋ＋ｂ）ビット値$ｄ_{H}$は、符号が正でありＭＳＢの値が０なので、範囲Ｒ３に属する。

[0432] よって、決定部３０３は、ステップＳ４１０で乱数値ｓ［１］を式（１０．２３）のとおり＋ｓに決定し、ステップＳ４１１でウィンドウ補正値ｔ［１］を式（１０．２４）のとおり０に決定する。また、ウィンドウ補正値ｔ［１］が０なので、ウィンドウ値ｗ［２］へのキャリー補正は行われず、ウ

ィンドウ値ｗ［２］は式（１０．２１）の値に確定する。

$$s[1]=+s=18=(010010)_2 \quad (10.23)$$

$$t[1]=0=(0000000000)_2 \quad (10.24)$$

[0433] そして、ステップＳ４１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１０．２５）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[1]+t[1] \\ &=205-18+0 \\ &=187 \\ &=2\times 2^6+59 \\ &=(010111011)_2 \\ &=(010 \,||\, 111011)_2 \end{aligned} \quad (10.25)$$

[0434] さらに、ステップＳ４１６で決定部３０３は、式（１０．２６）のようにウィンドウ値ｗ［１］を計算する。なお、ここで得られるウィンドウ値ｗ［１］は、後にキャリー補正によってインクリメントまたはデクリメントされる潜在的可能性があるので、まだ確定していない。

$$w[1]=diff[8]||diff[7]||diff[6]=(010)_2=2 \quad (10.26)$$

[0435] また、ｉ＝１なので、処理がステップＳ４１８に進む。ステップＳ４１８で判断部３０２は、式（９．５）にしたがい、具体的には式（１０．２７）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$\begin{aligned} d_H&=(diff[5]||\cdots||diff[0])2^3+(e[2]||\cdots||e[0]) \\ &=59\times 2^3+(100)_2 \\ &=472+4 \\ &=476 \\ &=(111011100)_2 \end{aligned} \quad (10.27)$$

[0436] その後、ステップＳ４２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝０となる。そして、ｉ≧０なので処理はステップＳ４２１からステップＳ４０４に戻る。

[0437] ステップＳ４０４では、判断部３０２が、式（１０．２７）のように更新された符号つき（ｋ＋ｂ）ビット値d_Hがどの範囲に属するかを判断する。符号つき（ｋ＋ｂ）ビット値d_Hは、符号が正でありＭＳＢの値が１なので、範囲Ｒ４に属する。

[0438] よって、決定部３０３は、ステップＳ４１２で乱数値ｓ［０］を式（１０．２８）のとおり－ｓに決定し、ステップＳ４１３でウィンドウ補正値ｔ［０］を式（１０．２９）のとおり-2^{k+b}に決定する。

$$s[0]=-s=-18=-(010010)_2 \tag{10.28}$$

$$t[0]=-2^{k+b}=-2^9=-(1000000000)_2 \tag{10.29}$$

[0439] そして、ウィンドウ補正値ｔ［０］が非ゼロなので、ステップＳ４１４で決定部３０３はキャリー補正を行う。すなわち、決定部３０３は、式（１０．２６）で求めたウィンドウ値ｗ［１］に１を足す。その結果、ウィンドウ値ｗ［１］は式（１０．３０）の値に確定する。

$$w[1]=2+1=3 \tag{10.30}$$

[0440] さらに、ステップＳ４１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１０．３１）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[0]+t[0] \\ &=476+18-512 \\ &=-18 \\ &=-(000010010)_2 \\ &=-(000\ ||\ 010010)_2 \end{aligned} \tag{10.31}$$

[0441] さらに、ステップＳ４１６で決定部３０３は、式（１０．３２）のようにウィンドウ値ｗ［０］を計算する。なお、ウィンドウ値ｗ［０］は、最下位のウィンドウ値なのでキャリー補正されることはなく、ここで値が確定する。

$$w[0]=diff[8]||diff[7]||diff[6]=-(000)_2=(000)_2=0 \tag{10.32}$$

[0442] また、ｉ＝０なので処理がステップＳ４１９に進む。そして、ステップＳ

４１９で判断部３０２は、式（９．６）にしたがい、具体的には式（１０．３３）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=(diff[5]||\cdots||diff[0])=-(010010)_2=-18 \quad (10.33)$$

[0443] その後、ステップＳ４２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝－１となる。よって、ｉ＜０なので処理はステップＳ４２１からステップＳ４２２に進む。したがって、ステップＳ４２２で決定部３０３は、式（１０．３４）のとおり補正値ｃを得る。

$$c=d_H=-18 \quad (10.34)$$

[0444] 最後に、ステップＳ４２３で決定部３０３は、式（１０．３５）に示すウィンドウ列ｗ［ｉ］と、式（１０．３６）に示す乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、式（１０．３４）に示す補正値ｃを補正部３１０に出力する。式（１０．３５）は、式（１０．２１）、式（１０．３０）、式（１０．３２）から明らかであり、式（１０．３６）は、式（１０．１８）、式（１０．２３）、式（１０．２８）から明らかである。

$$w[2]=2,\ w[1]=3,\ w[0]=0 \quad (10.35)$$

$$s[2]=18,\ s[1]=18,\ s[0]=-18 \quad (10.36)$$

[0445] 以上のようにして、図９のステップＳ１０３に相当する図１１の処理が終了すると、図９のステップＳ１０４で取得部３０６が点Ａのｘｙ座標を取得する。そして、ステップＳ１０５でスカラ倍算部３０７が、$-1 \leqq h \leqq 2^{k-1}+1=5$なる範囲の各インデックスｈについて、乱数ｓ＝１８（式（１０．１７）参照）に応じたスカラ倍点情報として、式（１０．３７）のランダム化テーブルデータｔａｂ［ｈ］を生成する。

$$tab[h]=(2^b h+s)A=(2^6 h+18)A=(64h+18)A \quad (10.37)$$

[0446] そして、スカラ倍点情報格納部３０８は、生成されたランダム化テーブルデータｔａｂ［ｈ］を、インデックスｈと対応づけて格納する。

[0447] 図２１は、以上のようにして得られた、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを示すとともに、スカラ倍点情報格納部３０８がランダム化テーブルデータｔａｂ［ｈ］を保持するテーブル１０８を示す図である。

[0448] 図１９～２０Ｂを参照して上記で説明した処理によれば、図２１に示すとおり、式（８．９）が成立する。具体的には、図１９～２０Ｂの例における$k=3$、$b=6$、$m=3$、$e=10988$という値および式（１０．３４）～（１０．３６）に示した値を式（８．９）の各変数に代入すると、下記の式（１０．３８）が得られる。

$$\begin{aligned} & c+2^{6}w[0]+2^{0}s[0]+2^{9}w[1]+2^{3}s[1]+2^{12}w[2]+2^{6}s[2] \\ &=-18+64\times 0+1\times(-18)+512\times 3+8\times 18+4096\times 2+64\times 18 \\ &=10988 \end{aligned} \quad (10.38)$$

[0449] また、式（１０．３７）に示したように、図２１のテーブル１０８には、$-1\leqq h\leqq 5$なる範囲の各インデックスhに対応して、それぞれ、スカラ倍点－４６Ａ、１８Ａ、８２Ａ、１４６Ａ、２１０Ａ、２７４Ａ、３３８Ａのｘｙ座標が記憶されている。

[0450] よって、図９のステップＳ１０６に相当する図１０の処理と、図９のステップＳ１０７～Ｓ１１１により、最終的には、式（１０．３９）のようにスカラ倍点ｄＡのｘｙ座標が計算される。なお、この式（１０．３９）と図９および図１０との対応関係は、第１実施形態の式（９．３６）に関する説明から明らかなので、説明を省略する。

$$dA=2(2^{3}(2^{3}(2^{3}(0)+tab[w[2]])+tab[w[1]])-tab[-w[0]]+cA)+d[0]A \quad (10.39)$$

[0451] そして、この式（１０．３９）の右辺において、ステップＳ１０９の２倍算の対象を示す部分を変形すると、次の式（１０．４０）が得られる。式（１０．４０）も、第２実施形態において式（８．９）が成立することを示している。

$$\begin{aligned} & 2^{3}(2^{3}(2^{3}(0)+tab[2])+tab[3])-tab[0]+cA \\ &=8(8(0+146A)+210A)-18A-18A \\ &=8(1168A+210A)-36A \\ &=8(1378A)-36A \\ &=10988A \end{aligned} \quad (10.40)$$

[0452] そして、図２１に例示するように、式（１０．１５）のダミー鍵ｅは、式

（１０．４１）または（１０．４２）に示す個人鍵ｄから得られる。図２１には、例として、式（１０．４２）の個人鍵ｄを示してある。

$$d=(101010111011000)_2=21976 \quad (10.41)$$

$$d=(101010111011001)_2=21977 \quad (10.42)$$

[0453] 個人鍵ｄが式（１０．４１）で示される場合、式（１０．４１）よりｄ［０］＝０なので、式（１０．４０）を式（１０．３９）の右辺に代入すると、式（１０．４３）が得られる。

$$2(10988A)+d[0]A=21976A+0=21976A=dA \quad (10.43)$$

[0454] また、個人鍵ｄが式（１０．４２）で示される場合、式（１０．４２）よりｄ［０］＝１なので、式（１０．４０）を式（１０．３９）の右辺に代入すると、式（１０．４４）が得られる。

$$2(10988A)+d[0]A=21976A+A=21977A=dA \quad (10.44)$$

[0455] 以上から、第２実施形態によれば、図９、１０、１９に示した処理により、確かに所望の点ｄＡが得られる。

[0456] さて、第２実施形態では、スカラ倍点情報格納部３０８に格納されるスカラ倍点情報ｔａｂ［ｈ］に対応するインデックスｈの範囲は、$-1 \leqq h \leqq 2^{k-1}+1$である。よって、第２実施形態において、スカラ倍点情報格納部３０８が有するテーブルのエントリ数は、$(2^{k-1}+3)$個である。

[0457] 図２２は、ウィンドウサイズｋが３の場合のテーブルデータのエントリ数に関して、第２実施形態と第３の比較例と第４の比較例を比べる図である。

図２２において、テーブル１０６と１０７は図１４と同じである。つまり、テーブル１０６は、第４の比較例として説明した符号つきウィンドウ法のテーブルであり、テーブル１０７は第３の比較例として説明したランダム化ウィンドウ法のテーブルである。

[0458] そして、テーブル１０９が、第２実施形態においてスカラ倍点情報格納部３０８がスカラ倍点情報を記憶するためのテーブルである。ｋ＝３なので、テーブル１０９には、$-1 \leqq h \leqq 2^{3-1}+1=5$なる各インデックスｈについて、テーブルデータｔａｂ［ｈ］として点$(h \times 2^b+s)$Ａのｘｙ座標が格

納されている。つまり、テーブル１０９のエントリ数は７個である。

[0459] 以上のテーブル１０６、１０７、１０９を比較すると、第２実施形態のテーブル１０９は、ＰＡ攻撃に対して安全な方法を提供しつつ、テーブル１０６と同様にエントリ数が2^{k-1}のオーダに抑えられているという利点を持つ。つまり、第２実施形態のテーブル１０９は、テーブル１０６と比べて安全性の点で優位であり、テーブル１０７と比べてメモリ消費量が少ない点で優位である。

[0460] さて、続いて第３実施形態について説明する。第３実施形態は、スカラ倍点情報格納部３０８のインデックスとして使われる可能性がある値の範囲を第２実施形態よりも狭めることで、さらにメモリの節約を図る実施形態である。

[0461] そこで、第３実施形態に関しては、まず、どのようにしたらインデックスの範囲を狭めることが可能なのかについて、図２３を参照して説明する。その後、図２４～２７を参照して第３実施形態における暗号処理装置３００の動作の詳細を説明する。

[0462] 図２３は、第３実施形態によるメモリ使用量の削減について説明する図である。以下、図２３を、第２実施形態に関する図１７および１８と比較しながら説明する。

具体的には第３実施形態では、乱数値ｓを非ゼロの値に限定することで、インデックスの範囲が狭まり、スカラ倍点情報格納部３０８のメモリ使用量が減る。乱数値ｓを非ゼロの値に限定することでメモリ使用量が減る理由は、第２実施形態についてｓ＝０の場合とｓ＞０の場合に分けて検討すると、明らかである。以下、図１７、１８、２３を参照して、その理由について説明する。

[0463] 図１７と１８において、矩形Ｅ２１１が示す範囲は、式（１０．１０）のとおりであり、矩形Ｅ２１３が示す範囲は、式（１０．９）のとおりである。このように、図１７と１８に示した矩形Ｅ２１１とＥ２１３がそれぞれ表す範囲は、境界値が範囲内に含まれるか否かという点以外は同じである。

[0464] そこで、図示の便宜上、図２３では矩形Ｅ２１１とＥ２１３の代わりに、矩形Ｅ２１１が表す範囲と矩形Ｅ２１３が表す範囲の和（union）を示す矩形を用いることにする。矩形Ｅ２１１が表す範囲と矩形Ｅ２１３が表す範囲の和は、式（１０．９）と（１０．１０）より、式（１１．１）のとおりである。

$$-s \leqq diff \leqq 2^{k+b-1}-s \qquad (11.1)$$

[0465] よって、ｓ＝０の場合は、矩形Ｅ２１１が表す範囲と矩形Ｅ２１３が表す範囲の和は、式（１１．１）より式（１１．２）のとおりであり、式（１１．２）の範囲が図２３では矩形Ｅ２３１により表されている。

$$0 \leqq diff \leqq 2^{k+b-1} \qquad (11.2)$$

[0466] また、ｓ＞０の場合は、矩形Ｅ２１１が表す範囲と矩形Ｅ２１３が表す範囲の和は、式（１１．１）より式（１１．３）のとおりであり、式（１１．３）の範囲が図２３では矩形Ｅ２５１により表されている。

$$-s \leqq diff \leqq 2^{k+b-1}-s < 2^{k+b-1} \qquad (11.3)$$

[0467] また、図１７と１８において、矩形Ｅ２１２が示す範囲は、式（１０．１１）のとおりであり、矩形Ｅ２１４が示す範囲は、式（１０．１３）のとおりである。このように、図１７と１８に示した矩形Ｅ２１２とＥ２１４がそれぞれ表す範囲は、境界値が範囲内に含まれるか否かという点以外は同じである。

[0468] そこで、図示の便宜上、図２３では矩形Ｅ２１２とＥ２１４の代わりに、矩形Ｅ２１２が表す範囲と矩形Ｅ２１４が表す範囲の和を示す矩形を用いることにする。矩形Ｅ２１２が表す範囲と矩形Ｅ２１４が表す範囲の和は、式（１０．１１）と（１０．１３）より、式（１１．４）のとおりである。

$$-2^{k+b-1}+s \leqq diff \leqq s-1 \qquad (11.4)$$

[0469] よって、ｓ＝０の場合は、矩形Ｅ２１２が示す範囲と矩形Ｅ２１４が表す範囲の和は、式（１１．４）より式（１１．５）のとおりであり、式（１１．５）の範囲が図２３では矩形Ｅ２３２により表されている。

$$-2^{k+b-1} \leqq diff \leqq -1 \qquad (11.5)$$

[0470] したがって、矩形Ｅ２３２が表す式（１１．５）の範囲の符号を反転すると、式（１１．６）のとおりである。式（１１．６）の範囲は、図２３では矩形Ｅ２４２により表されている。

$$1 \leqq -diff \leqq 2^{k+b-1} \qquad (11.6)$$

[0471] また、ｓ＞０の場合は、矩形Ｅ２１２が示す範囲と矩形Ｅ２１４が表す範囲の和は、図２３では矩形Ｅ２５２により表されている。そして、矩形Ｅ２５２が表す範囲の符号を反転すると、式（１１．７）のとおりである。式（１１．７）の範囲は、図２３では矩形Ｅ２６２により表されている。

$$-2^{b} < -s+1 \leqq -diff \leqq 2^{k+b-1}-s < 2^{k+b-1} \qquad (11.7)$$

[0472] したがって、キャリー補正の影響がないと仮定すると、ｓ＝０のときのインデックスの範囲Ｕ３は、式（１１．２）と式（１１．６）より、０以上2^{k+b-1}以下の（ｋ＋ｂ）ビット値の上位ｋビットがとりうる範囲である。すなわち、範囲Ｕ３は、０以上2^{k-1}以下であり、図２３のようにｋ＝２、ｂ＝３の場合は、０以上２以下である。

[0473] それに対して、キャリー補正の影響がないと仮定した場合におけるｓ＞０のときのインデックスの範囲Ｕ４は、式（１１．３）と式（１１．７）より、－ｓ以上（2^{k+b-1}－ｓ）以下の（ｋ＋ｂ）ビット値の上位ｋビットがとりうる範囲である。そして、０＜ｓ＜2^{b}であり、－０＝＋０＝０なので、範囲Ｕ４は、０以上（2^{k-1}－１）以下であり、図２３のようにｋ＝２、ｂ＝３の場合は、０以上１以下である。

[0474] つまり、乱数値ｓを非ゼロの値に限定することで、キャリー補正前のウィンドウ値に対応するインデックスの範囲から、2^{k-1}が除外される。したがって、キャリー補正を考慮すると、乱数値ｓを非ゼロの値に限定することで（2^{k-1}＋１）というインデックスの必要性をなくせる。よって、乱数値ｓを非ゼロの値に限定することで、スカラ倍点情報格納部３０８のエントリ数が、第２実施形態よりも１つ減る。

[0475] 例えば、図２３のようにｋ＝２、ｂ＝３の場合は、ｓ＝０ならば、インデックスの範囲は－１から３（＝2^{k-1}＋１）であり、スカラ倍点情報格納部３

０８のテーブルには５つのインデックスに対応する５つのエントリが含まれる。それに対して、ｓ＞０ならば、インデックスの範囲は－１から２（＝2^{k-1}）であり、スカラ倍点情報格納部３０８のテーブルには４つのインデックスに対応する４つのエントリが含まれる。

[0476] このように、第３実施形態によれば、乱数生成部３０５が生成する乱数値ｓを非ゼロの値に限定するだけで、スカラ倍点情報格納部３０８のメモリ消費量のさらなる削減が達成される。

[0477] 以下では、以上説明した第３実施形態について、さらに具体的に説明する。

図２４は、第３実施形態において暗号処理装置３００がウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを決定する処理のフローチャートである。すなわち、図２４は、第３実施形態における、図９のステップＳ１０３のフローチャートである。

[0478] 図２４のステップＳ５０１は、第２実施形態に関する図１９のステップＳ４０１と同様であり、図２４のステップＳ５０３～Ｓ５２３は、図１９のステップＳ４０３～Ｓ４２３と同様である。よって、これらのステップについては説明を割愛する。

[0479] 図２４において、第２実施形態に関する図１９と異なるのは、ステップＳ５０２である。ステップＳ５０２において乱数生成部３０５は、ｂビットで、かつ非ゼロの乱数値ｓを生成する。なお、説明の簡単化のため、乱数値ｓは正とする。乱数値ｓが負の場合については、第３実施形態の変形例として後述する。よって、ステップＳ５０２で生成される乱数値ｓは式（１１．８）を満たす。

$$0<s\leqq 2^{b}-1 \qquad (11.8)$$

[0480] 続いて、図２５Ａ～２６を参照して、第３実施形態における図９、１０、２４の処理の具体例を説明する。

[0481] 図９のステップＳ１０２で処理部３０１が得たダミー鍵ｅが、式（１１．９）に示す１５ビットの値であるとする。なお、式（１１．９）に対応する

個人鍵ｄの具体例は、図２６とともに後述する。

$$e=(011001000111110)_2=12862 \quad (11.9)$$

[0482] 続いて、図９のステップＳ１０３に相当する図２４の処理が開始される。なお、図２５Ａ～２６の例では、ウィンドウサイズｋは３であり、乱数値ｓのビット長ｂは６であるとする。したがって、式（９．９）と同じく、ｍ＝３である。

[0483] 図２４の処理が開始されると、ステップＳ５０１で判断部３０２は、式（１１．１０）のように符号つき（ｋ＋ｂ）ビット値d_Hを初期化する。

$$d_H=e[14]||\cdots||e[6]=(011001000)_2=200 \quad (11.10)$$

[0484] また、ステップＳ５０２で乱数生成部３０５が、ｂ（＝６）ビットの非ゼロの乱数値ｓとして式（１１．１１）の値を生成したとする。

$$s=(100110)_2=38 \quad (11.11)$$

[0485] すると、続くステップＳ５０３では、処理部３０１がループ変数ｉを２（＝ｍ－１）に初期化する。そして、ステップＳ５０４で判断部３０２は、式（１１．１０）の符号つき（ｋ＋ｂ）ビット値d_Hがどの範囲に属するかを判断する。式（１１．１０）より、符号つき（ｋ＋ｂ）ビット値d_Hは、符号が正でＭＳＢの値が０なので、範囲Ｒ３に属する。

[0486] よって、決定部３０３は、ステップＳ５１０で乱数値ｓ［２］を式（１１．１２）のとおり＋ｓに決定し、ステップＳ５１１でウィンドウ補正値ｔ［２］を式（１１．１３）のとおり０に決定する。また、ウィンドウ補正値ｔ［２］が０なので、キャリー補正は行われない。

$$s[2]=+s=38=(100110)_2 \quad (11.12)$$

$$t[2]=0=(0000000000)_2 \quad (11.13)$$

[0487] そして、ステップＳ５１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１１．１４）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$diff=d_H-s[2]+t[2]$$

$$=200-38+0$$

$=162$

$=2\times 2^{6}+34$

$=(010100010)_{2}$

$=(010\ ||\ 100010)_{2}$ (11.14)

[0488] さらに、ステップＳ５１６で決定部３０３は、式（１１．１５）のようにウィンドウ値ｗ［２］を計算する。なお、ここで得られるウィンドウ値ｗ［２］は、後にキャリー補正によってインクリメントまたはデクリメントされる潜在的可能性があるので、まだ確定していない。

$w[2]=diff[8]||diff[7]||diff[6]=(010)_{2}=2$ (11.15)

[0489] また、ｉ＝２なので、処理がステップＳ５１８に進む。ステップＳ５１８で判断部３０２は、式（９．５）にしたがい、具体的には式（１１．１６）のようにして、符号つき（ｋ＋ｂ）ビット値d_{H}を更新する。

$d_{H}=(diff[5]||\cdots||diff[0])2^{3}+(e[5]||\cdots||e[3])$

$=34\times 2^{3}+(111)_{2}$

$=272+7$

$=279$

$=(100010111)_{2}$ (11.16)

[0490] その後、ステップＳ５２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝１となる。そして、ｉ≧０なので処理はステップＳ５２１からステップＳ５０４に戻る。

[0491] ステップＳ５０４では、判断部３０２が、式（１１．１６）のように更新された符号つき（ｋ＋ｂ）ビット値d_{H}がどの範囲に属するかを判断する。符号つき（ｋ＋ｂ）ビット値d_{H}は、符号が正でありＭＳＢの値が１なので、範囲Ｒ４に属する。

[0492] よって、決定部３０３は、ステップＳ５１２で乱数値ｓ［１］を式（１１．１７）のとおり－ｓに決定し、ステップＳ５１３でウィンドウ補正値ｔ［１］を式（１１．１８）のとおり－2^{k+b}に決定する。

$s[1]=-s=-38=-(100110)_{2}$ (11.17)

$$t[1]=-2^{k+b}=-2^{9}=-(1000000000)_2 \quad (11.18)$$

[0493] そして、ウィンドウ補正値ｔ［１］が非ゼロなので、ステップＳ５１４で決定部３０３はキャリー補正を行う。すなわち、決定部３０３は、式（１１．１５）で求めたウィンドウ値ｗ［２］に１を足す。その結果、ウィンドウ値ｗ［２］は式（１１．１９）の値に確定する。

$$w[2]=2+1=3 \quad (11.19)$$

[0494] さらに、ステップＳ５１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１１．２０）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[1]+t[1] \\ &=279+38-512 \\ &=-195 \\ &=-3\times 2^{6}-3 \\ &=-(011000011)_2 \\ &=-(011\ ||\ 000011)_2 \end{aligned} \quad (11.20)$$

[0495] そして、ステップＳ５１６で決定部３０３は、式（１１．２１）のようにウィンドウ値ｗ［１］を計算する。なお、ここで得られるウィンドウ値ｗ［１］は、後にキャリー補正によってインクリメントまたはデクリメントされる潜在的可能性があるので、まだ確定していない。

$$w[1]=diff[8]||diff[7]||diff[6]=-(011)_2=-3 \quad (11.21)$$

[0496] また、ｉ＝１なので、処理がステップＳ５１８に進む。ステップＳ５１８で判断部３０２は、式（９．５）にしたがい、具体的には式（１１．２２）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$\begin{aligned} d_H&=(diff[5]||\cdots||diff[0])2^{3}+(e[2]||\cdots||e[0]) \\ &=-3\times 2^{3}+(110)_2 \\ &=-24+6 \\ &=-18 \\ &=-(000010010)_2 \end{aligned} \quad (11.22)$$

[0497] その後、ステップＳ５２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝０となる。そして、ｉ≧０なので処理はステップＳ５２１からステップＳ５０４に戻る。

[0498] ステップＳ５０４では、判断部３０２が、式（１１．２２）のように更新された符号つき（ｋ＋ｂ）ビット値d_Hがどの範囲に属するかを判断する。符号つき（ｋ＋ｂ）ビット値d_Hは、符号が負でありＭＳＢの値が０なので、範囲Ｒ２に属する。

[0499] よって、決定部３０３は、ステップＳ５０８で乱数値ｓ［０］を式（１１．２３）のとおり－ｓに決定し、ステップＳ５０９でウィンドウ補正値ｔ［０］を式（１１．２４）のとおり０に決定する。また、ウィンドウ補正値ｔ［０］が０なので、ウィンドウ値ｗ［１］へのキャリー補正は行われず、ウィンドウ値ｗ［１］は式（１１．２１）の値に確定する。

$$s[0]=-s=-38=-(100110)_2 \tag{11.23}$$

$$t[0]=0=(0000000000)_2 \tag{11.24}$$

[0500] そして、ステップＳ５１５で決定部３０３は、式（９．３）にしたがい、具体的には式（１１．２５）のようにして、補正済み差分値ｄｉｆｆを計算する。

$$\begin{aligned} diff&=d_H-s[0]+t[0] \\ &=-18+38+0 \\ &=20 \\ &=(000010100)_2 \\ &=(000 \ || \ 010100)_2 \end{aligned} \tag{11.25}$$

[0501] さらに、ステップＳ５１６で決定部３０３は、式（１１．２６）のようにウィンドウ値ｗ［０］を計算する。なお、ウィンドウ値ｗ［０］は、最下位のウィンドウ値なのでキャリー補正されることはなく、ここで値が確定する。

$$w[0]=diff[8]||diff[7]||diff[6]=(000)_2=0 \tag{11.26}$$

[0502] また、ｉ＝０なので処理がステップＳ５１９に進む。そして、ステップＳ

５１９で判断部３０２は、式（９．６）にしたがい、具体的には式（１１．２７）のようにして、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=(diff[5]||\cdots||diff[0])=(010100)_2=20 \quad (11.27)$$

[0503] その後、ステップＳ５２０で処理部３０１がループ変数ｉをデクリメントするのでｉ＝－１となる。よって、ｉ＜０なので処理はステップＳ５２１からステップＳ５２２に進む。したがって、ステップＳ５２２で決定部３０３は、式（１１．２８）のとおり補正値ｃを得る。

$$c=d_H=20 \quad (11.28)$$

[0504] 最後に、ステップＳ５２３で決定部３０３は、式（１１．２９）に示すウィンドウ列ｗ［ｉ］と、式（１１．３０）に示す乱数列ｓ［ｉ］をウィンドウ演算部３０９に出力し、式（１１．２８）に示す補正値ｃを補正部３１０に出力する。式（１１．２９）は、式（１１．１９）、式（１１．２１）、式（１１．２６）から明らかであり、式（１１．３０）は、式（１１．１２）、式（１１．１７）、式（１１．２３）から明らかである。

$$w[2]= 3,\ w[1]= -3,\ w[0]= 0 \quad (11.29)$$

$$s[2]=38,\ s[1]=-38,\ s[0]=-38 \quad (11.30)$$

[0505] 以上のようにして、図９のステップＳ１０３に相当する図１１の処理が終了すると、図９のステップＳ１０４で取得部３０６が点Ａのｘｙ座標を取得する。そして、ステップＳ１０５でスカラ倍算部３０７が、$-1 \leqq h \leqq 2^{k-1}=4$なる範囲の各インデックスｈについて、乱数ｓ＝３８（式（１１．１１）参照）に応じたスカラ倍点情報として、式（１１．３１）のランダム化テーブルデータｔａｂ［ｈ］を生成する。

$$tab[h]=(2^b h+s)A=(2^6 h+38)A=(64h+38)A \quad (11.31)$$

[0506] そして、スカラ倍点情報格納部３０８は、生成されたランダム化テーブルデータｔａｂ［ｈ］を、インデックスｈと対応づけて格納する。

[0507] 図２６は、以上のようにして得られた、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを示すとともに、スカラ倍点情報格納部３０８がランダム化テーブルデータｔａｂ［ｈ］を保持するテーブル１１０を示す図である。

[0508] 図２４～２５Ｂを参照して上記で説明した処理によれば、図２６に示すとおり、式（８．９）が成立する。具体的には、図２４～２５Ｂの例における$k=3$、$b=6$、$m=3$、$e=12862$という値および式（１１．２８）～（１１．３０）に示した値を式（８．９）の各変数に代入すると、下記の式（１１．３２）が得られる。

$$\begin{aligned} & c+2^6w[0]+2^0s[0]+2^9w[1]+2^3s[1]+2^{12}w[2]+2^6s[2] \\ &=20+64\times 0+1\times(-38)+512\times(-3)+8\times(-38)+4096\times 3+64\times 38 \\ &=12862 \end{aligned} \tag{11.32}$$

[0509] また、式（１１．３１）に示したように、図２６のテーブル１１０には、$-1\leqq h\leqq 4$なる範囲の各インデックスhに対応して、それぞれ、スカラ倍点－２６Ａ、３８Ａ、１０２Ａ、１６６Ａ、２３０Ａ、２９４Ａの$x$$y$座標が記憶されている。

[0510] よって、図９のステップＳ１０６に相当する図１０の処理と、図９のステップＳ１０７～Ｓ１１１により、最終的には、式（１１．３３）のようにスカラ倍点ｄＡの$x$$y$座標が計算される。なお、この式（１１．３３）と図９および図１０との対応関係は、第１実施形態の式（９．３６）に関する説明から明らかなので、説明を省略する。

$$dA=2(2^3(2^3(2^3(0)+tab[w[2]])-tab[-w[1]])-tab[-w[0]]+cA)+d[0]A \tag{11.33}$$

[0511] なお、この式（１１．３３）の右辺において、ステップＳ１０９の２倍算の対象を示す部分を変形すると、次の式（１１．３４）が得られる。式（１１．３４）も、第３実施形態において式（８．９）が成立することを示している。

$$\begin{aligned} & 2^3(2^3(2^3(0)+tab[3])-tab[3])-tab[0]+20A \\ &=8(8(0+230A)-230A)-38A+20A \\ &=8(1840A-230A)-18A \\ &=8(1610A)-18A \\ &=12862A \end{aligned} \tag{11.34}$$

[0512] そして、図２６に例示するように、式（１１．９）のダミー鍵ｅは、式（

１１．３５）または（１１．３６）に示す個人鍵ｄから得られる。図２１には、例として、式（１１．３５）の個人鍵ｄを示してある。

$$d=(110010001111100)_2=25724 \quad (11.35)$$

$$d=(110010001111101)_2=25725 \quad (11.36)$$

[0513] 個人鍵ｄが式（１１．３５）で示される場合、式（１１．３５）よりｄ［０］＝０なので、式（１１．３４）を式（１１．３３）の右辺に代入すると、式（１１．３７）が得られる。

$$2(12862A)+d[0]A=25724A+0=25724A=dA \quad (11.37)$$

[0514] また、個人鍵ｄが式（１１．３６）で示される場合、式（１１．３６）よりｄ［０］＝１なので、式（１１．３４）を式（１１．３３）の右辺に代入すると、式（１１．３８）が得られる。

$$2(12862A)+d[0]A=25724A+A=25725A=dA \quad (11.38)$$

[0515] 以上から、第３実施形態によれば、図９、１０、２４に示した処理により、確かに所望の点ｄＡが得られる。

[0516] さて、第３実施形態では、スカラ倍点情報格納部３０８に格納されるスカラ倍点情報ｔａｂ［ｈ］に対応するインデックスｈの範囲は、$-1 \leqq h \leqq 2^{k-1}$である。よって、第３実施形態においてスカラ倍点情報格納部３０８が有するテーブルのエントリ数は、（$2^{k-1}+2$）個である。

[0517] 図２７は、ウィンドウサイズｋが３の場合のテーブルデータのエントリ数に関して、第３実施形態と第３の比較例と第４の比較例を比べる図である。

図２７において、テーブル１０６と１０７は図１４と同じである。つまり、テーブル１０６は、第４の比較例として説明した符号つきウィンドウ法のテーブルであり、テーブル１０７は第３の比較例として説明したランダム化ウィンドウ法のテーブルである。

[0518] そして、テーブル１１１が、第３実施形態においてスカラ倍点情報格納部３０８がスカラ倍点情報を記憶するためのテーブルである。ｋ＝３なので、テーブル１１１には、$-1 \leqq h \leqq 2^{3-1}=4$なる各インデックスｈについて、テーブルデータｔａｂ［ｈ］として点（ｈ×2^{b}＋ｓ）Ａのｘｙ座標が格納さ

れている。つまり、テーブル１１１のエントリ数は６個である。

[0519] 以上のテーブル１０６、１０７、１１１を比較すると、第３実施形態のテーブル１１１は、ＰＡ攻撃に対して安全な方法を提供しつつ、テーブル１０６と同様にエントリ数が2^{k-1}のオーダに抑えられているという利点を持つ。つまり、第３実施形態のテーブル１１１は、テーブル１０６と比べて安全性の点で優位であり、テーブル１０７と比べてメモリ消費量が少ない点で優位である。

[0520] ところで、本発明は上記の第１～第３実施形態に限られるものではない。

例えば、個人鍵ｄの長さｕ、ウィンドウサイズｋ、および乱数値ｓの長さｂに関して、上記の説明では便宜上、具体的な数値をいくつか例示した。しかし、ｕ、ｋ、ｂの具体的な値は実施形態に応じて任意である。また、ｋやｂの具体的な値は、暗号処理装置３００を含む暗号通信システムにおけるシステムパラメタとして固定的に予め決められていてもよいし、暗号処理装置３００によって決められる可変値であってもよい。

[0521] また、第１実施形態では、処理部３０１は、ｓ［ｉ］＝＋ｓと仮定してウィンドウ値ｗ［ｉ］を見積もる。しかし、実施形態によっては、処理部３０１は、ｓ［ｉ］＝－ｓと仮定してウィンドウ値ｗ［ｉ］を見積もってもよい。

[0522] また、第１～第３実施形態では、決定部３０３がウィンドウ演算部３０９に乱数列ｓ［ｉ］を出力している。しかし、図１０のステップＳ２０７に示すように、ウィンドウ演算部３０９は、「乱数値ｓ［ｉ］が＋ｓなのか、それとも－ｓなのか」ということさえ認識することさえできればよい。

[0523] したがって、実施形態によっては、決定部３０３は、乱数列ｓ［ｉ］自体をウィンドウ演算部３０９に出力する代わりに、各ｉについて「乱数値ｓ［ｉ］が＋ｓなのか、それとも－ｓなのか」ということを示す情報をウィンドウ演算部３０９に出力してもよい。例えば、決定部３０３は、乱数列ｓ［ｉ］の代わりに、各ｉに対応する１ビットのフラグの列（すなわち合計ｍビットのフラグ列）を出力してもよい。

[0524] また、第１～第３実施形態では、最上位のキャリー補正値を適切に扱うためのテクニックとして、ダミー鍵ｅが導入されている。しかし、実施形態に応じて、最上位のキャリー補正値を適切に扱うための別のテクニックが使われてもよい。以下に、２つのテクニックを例示する。

[0525] １つ目のテクニックは、ダミー鍵ｅを使わない方法である。すなわち、この１つ目のテクニックによれば、図９のステップＳ１０２が省略され、ステップＳ１０３に相当する図１１、１９、または２４の処理では、ダミー鍵ｅの代わりに個人鍵ｄそのものが使われる。例えば、図１１のステップＳ３０１（または、図１９のステップＳ４０１もしくは図２４のステップＳ５０１）で判断部３０２は、式（９．１）の代わりに式（１２．１）により、符号つき（ｋ＋ｂ）ビット値d_Hを初期化する。

$$d_H=d[b+km-1]||\cdots||d[k(m-1)] \quad (12.1)$$

[0526] また、図１１のステップＳ３１８（または、図１９のステップＳ４１８もしくは図２４のステップＳ５１８）で判断部３０２は、式（９．５）の代わりに式（１２．２）により、符号つき（ｋ＋ｂ）ビット値d_Hを更新する。

$$d_H=(diff[b-1]||\cdots||diff[0])2^k+(d[ki-1]||\cdots||d[k(i-1)]) \quad (12.2)$$

[0527] このようにダミー鍵ｅが使われない場合、個人鍵ｄのＭＳＢであるｄ［ｕ－１］の値が１である可能性がある。よって、たとえｉ＝ｍ－１であっても、図１１のステップＳ３０４で判断基準値（d_H－ｓ）が範囲Ｒ４に属すると判断されて、キャリー補正が生じる可能性がある。あるいは、たとえｉ＝ｍ－１であっても、図１９のステップＳ４０４または図２４のステップＳ５０４で符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ４に属すると判断されて、キャリー補正が生じる可能性がある。

[0528] そこで、この１つ目のテクニックでは、ｉ＝ｍ－１のときに図１１のステップＳ３１４（または、図１９のステップＳ４１４もしくは図２４のステップＳ５１４）を実行する場合、決定部３０３は、ウィンドウ値ｗ［ｍ］を１と設定する。つまり、決定部３０３は、ウィンドウ値ｗ［ｍ］の初期値を０とみなしてキャリー補正を行う。

[0529] こうして設定されるウィンドウ値ｗ［ｍ］は、ｕビットの個人鍵ｄの範囲を超えた上位の桁に対応する。すなわち、ウィンドウ値ｗ［ｍ］は、説明の便宜上、ここでは“ｗ［ｍ］”という記号を用いて「ウィンドウ値」と呼んでいるが、キャリー補正値を表すだけであり、テーブルデータを引くためのインデックスとしては利用されない。

[0530] そして、この１つ目のテクニックでは、図９のステップＳ１０６に相当する図１０の処理のステップＳ２０１の初期化が、点Ｖを式（１２．３）のように設定する処理に置き換えられる。

$$V=2^{b}w[m]A \qquad (12.3)$$

[0531] 式（１２．３）のように点Ｖを設定するには、ウィンドウ演算部３０９は、具体的には次のように動作すればよい。すなわち、ウィンドウ演算部３０９は、最上位のキャリー補正値を表すウィンドウ値ｗ［ｍ］が０か１かを判断する。そして、ウィンドウ演算部３０９は、ｗ［ｍ］＝０ならば変数Ｖに無限遠点Ｏを格納し、ｗ［ｍ］＝１ならば変数Ｖに点Ａを格納する。その後、ウィンドウ演算部３０９は、「点Ｖに対して２倍算を行い、２倍算の結果を新たに変数Ｖに格納する」という処理をｂ回繰り返す。

[0532] なお、図１０のステップＳ２０２以降の処理と、図９のステップＳ１０７～Ｓ１０８には変更はない。また、この１つ目のテクニックではダミー鍵ｅを使わないので、ダミー鍵ｅを使うための後処理である図９のステップＳ１０９～Ｓ１１１は省略される。

[0533] すると、例えばｍ＝４の場合、ステップＳ１１２で補正部３１０が出力する点は、式（１２．４）のとおり、所望の点ｄＡである。なお、式（１２．４）における“±ｔａｂ［±ｗ［ｉ］］”なる記法は、「ｓ［ｉ］＝＋ｓならば＋ｔａｂ［ｗ［ｉ］］、ｓ［ｉ］＝－ｓならば－ｔａｂ［－ｗ［ｉ］］」ということを示すための省略記法である。

$$dA=2^{k}(2^{k}(2^{k+b}(w[4]A)\pm tab[\pm w[3]])\pm tab[\pm w[2]])\pm tab[\pm w[1]])$$
$$\pm tab[\pm w[0]]+cA \qquad (12.4)$$

[0534] すなわち、この１つ目のテクニックによれば、処理部３０１は、式（１２

．５）が成立するという制約条件下で、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを決定する。

[数12]

$$d = c + 2^{km+b} w[m] + \sum_{i=0}^{m-1} \left(2^{ki+b} w[i] + 2^{ki} s[i] \right) \quad (12.5)$$

[0535] ２つ目のテクニックは、上記の第１～第３実施形態におけるのとは異なるダミー鍵を使う方法である。説明の便宜上、全ビットの値が０で長さｚのビット列を“ｚｅｒｏ（ｚ）”と表すことにすると、この２つ目のテクニックでは、図９のステップＳ１０２で処理部３０１は、式（８．６）のダミー鍵ｅの代わりに、式（１２．６）のダミー鍵ｒを求める。以下、混乱を避けるため、「ダミー鍵ｒ」のことを、「ダミー鍵ｅ」とは別の「ゼロ埋めビット列」という名前で呼ぶことにする。

r=zero(z)||d (12.6)

[0536] なお、この２つ目のテクニックでは、個人鍵ｄ自体の長さではなく、ゼロ埋めビット列ｒの長さが記号“ｕ”により表されるものとする。すなわち、個人鍵ｄの長さは（ｕ－ｚ）である。

[0537] そして、ゼロ埋めビット列ｒの長さｕに関して式（８．４）が成立するように、整数ｍ、乱数値ｓのビット長ｂ、およびウィンドウサイズｋの値が定められる。逆の観点から説明すれば、ゼロ埋めビット列ｒの生成のために個人鍵ｄに足されるビットの数ｚは、式（８．４）を満たす正整数ｍが存在するように、選ばれる。

[0538] なお、乱数値ｓが０以上に限定されている場合は、ｚ＝１でもよい。後述の変形例のように負の乱数値ｓを許す場合は、ｚ＞１である。また、例えば個人鍵ｄのビット長（ｕ－ｚ）と乱数値ｓのビット長ｂとウィンドウサイズｋが、システムパラメタとして固定的に決められている環境などにおいては、好適な設定の一例は、ｚ＝ｋという設定である。

[0539] 式（１２．６）によれば、ゼロ埋めビット列ｒのＭＳＢは０と保証される。したがって、図１１のステップＳ３０４に関して説明したのと同じ理由か

ら、ｉ＝ｍ－１のときにキャリー補正が生じないことが保証される。よって、図９のステップＳ１０６に対応する図１０の処理は、第１～第３実施形態とまったく同じである。

[0540] また、式（１２．６）によれば、個人鍵ｄとゼロ埋めビット列ｒは、ビット長が異なるだけで、表す数値は等しい。よって、この２つ目のテクニックを用いる場合、ｄＡ＝ｒＡであるから、図９のステップＳ１０９～Ｓ１１１の後処理は不要であり、省略される。

[0541] 以上説明した２つ目のテクニックによれば、処理部３０１は、式（８．１）が成立するという制約条件下で、ウィンドウ列ｗ［ｉ］と乱数列ｓ［ｉ］と補正値ｃを決定する。

なお、図９～１１、１９、２４に示した第１～第３実施形態でのダミー鍵ｅを用いるテクニックと、上記２つのテクニックの共通点は次のとおりである。すなわち、いずれの場合においても、処理部３０１は、個人鍵ｄに基づくビット列Ｄと乱数値ｓを用いて、ビット列Ｄの長さｕとウィンドウサイズｋとの間にｕ＝ｍｋ＋ｂなる関係が成立する正整数ｍに関して、下記の値を定める。

・０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｋビット値のウィンドウ値ｗ［ｉ］

・０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｂビット値の乱数値ｓ［ｉ］

・補正値ｇ

[0542] 具体的には、処理部３０１は、各乱数値ｓ［ｉ］を＋ｓまたはーｓに決定しながら、式（１２．７）が成立するという制約条件下で、上記の値を定める。

[数13]

$$D = g + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i]) \qquad (12.7)$$

[0543] つまり、第１～第３実施形態は、式（１２．７）におけるビット列Ｄと補

正値ｇが、具体的には式（１２．８）の場合に相当し、より具体的には、式（１２．８）において右シフト量ｆが１の場合に相当する。後述のように乱数生成部３０５が負の乱数値ｓを生成する場合は、式（１２．８）の右シフト量ｆは、２以上が適切である。

D=zero(f)||d[u-1]||…||d[f] かつ g=c (12.8)

[0544] なお、式（１２．８）の場合、式（１２．７）の制約条件は、任意のｆに対して、式（１２．９）の制約条件と同値である。

[数14]

$$d = 2^f \left(c + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i]) \right) + \sum_{i=0}^{f-1} 2^i d[i] \quad (12.9)$$

[0545] この式（１２．９）の制約条件は、右シフト量ｆが１の場合の式（８．８）を一般化した制約条件である。また、任意のｆに対してダミー鍵ｅは式（１２．１０）のとおりである。すなわち、式（１２．１０）は、右シフト量ｆが１の場合の式（８．７）を一般化した式である。

[数15]

$$e = \left\lfloor \frac{d}{2^f} \right\rfloor \quad (12.10)$$

[0546] また、上記の１つ目のテクニックは、式（１２．７）におけるビット列Ｄと補正値ｇが具体的には式（１２．１１）の場合に相当する。

D=d かつ $g=2^{km+b}w[m]+c$ (12.11)

[0547] そして、上記の２つ目のテクニックは、式（１２．７）におけるビット列Ｄと補正値ｇが具体的には式（１２．１２）の場合に相当する。

D=zero(z)||d かつ g=c (12.12)

[0548] ところで、上記の説明では、説明の簡単化のため乱数値ｓを非負と仮定したが、ｓ≦０であってもよい。以下に、第１～第３実施形態それぞれに関して、変形例として、ｓ≦０あるいはｓ＜０の場合を説明する。

[0549] 図２８は、０以下の乱数値を生成するように変形された第１実施形態において、スカラ倍点情報格納部３０８のインデックスとして使われる値の範囲について模式的に説明する図である。

[0550] 図２８において、矩形Ｅ３０１、Ｅ３０２、Ｅ３０３、Ｅ３０４は、それぞれ、第１実施形態に関する図１５の矩形Ｅ１０１、Ｅ１０２、Ｅ１０３、Ｅ１０４と同じである。よって、乱数値ｓ［ｉ］が＋ｓであるという仮定が正式に採用される場合（すなわち、判断基準値（d_H-s）が範囲Ｒ１またはＲ３に属する場合）の補正済み差分値ｄｉｆｆの範囲を示す矩形Ｅ３１１とＥ３１３も、図１５の矩形Ｅ１１１とＥ１１３と同じである。

[0551] 図１５と２８で異なるのは、乱数値ｓ［ｉ］が＋ｓであるという仮定が妥当ではないという判断にしたがい、決定部３０３が乱数値ｓ［ｉ］を－ｓに決定する場合（すなわち、判断基準値（d_H-s）が範囲Ｒ２またはＲ４に属する場合）である。

[0552] 判断基準値（d_H-s）が範囲Ｒ２に属する場合、図１５の第１実施形態では、式（９．５１）に示すように、矩形Ｅ１０２を＋２ｓ移動させた矩形Ｅ１１２が、補正済み差分値ｄｉｆｆの範囲を示す。図２８の変形例においても、矩形Ｅ３０２を＋２ｓ移動させた矩形Ｅ３１２が補正済み差分値ｄｉｆｆの範囲を示す点は図１５と同様である。ただし、図２８の変形例では、ｓ≦０であるから、＋２ｓの移動とは、負の方向への移動であり、図１５のような正の方向への移動ではない。

[0553] よって、矩形Ｅ３１２が表す範囲の符号を反転させた範囲を表す矩形Ｅ３２２の左端は｜２ｓ｜の位置にあり、矩形Ｅ３２２の右端は（$2^{k+b-1}+|2s|$）の位置にある。そして、乱数値ｓによっては、｜２ｓ｜＞2^{b}となることがあるので、矩形Ｅ３２２が示す範囲の上位ｋビットは、最大で、（$2^{k-1}+1$）という値をとる可能性がある。

[0554] したがって、キャリー補正の影響を考慮に入れると、スカラ倍点情報格納部３０８のインデックスとして、（$2^{k-1}+2$）なる値が使われる可能性がある。つまり、図１５の第１実施形態と比べると、図２８の変形例では、スカ

ラ倍点情報格納部３０８において、（$2^{k-1}+2$）なるインデックスに対応するエントリが増える。

[0555] 判断基準値（d_H-s）が範囲Ｒ４に属する場合についても同様の議論が成り立つ。すなわち、判断基準値（d_H-s）が範囲Ｒ４に属する場合、図２８においては、ウィンドウ補正を含めて（$-2^{k+b}+2s$）だけ矩形Ｅ３０４を移動させた矩形Ｅ３１４が、補正済み差分値ｄｉｆｆの範囲を示す。よって、矩形Ｅ３１４が表す範囲の符号を反転させた範囲を表す矩形Ｅ３２４の左端は｜２ｓ｜の位置にあり、矩形Ｅ３２４の右端は（$2^{k+b-1}+|2s|$）の位置にある。

[0556] したがって、矩形Ｅ３２４が示す範囲の上位ｋビットは、最大で、（$2^{k-1}+1$）という値をとる可能性がある。つまり、キャリー補正の影響を考慮に入れると、スカラ倍点情報格納部３０８のインデックスとして、（$2^{k-1}+2$）なる値が使われる可能性がある。

[0557] しかし、図１５の第１実施形態と比べると、図２８の変形例で不要となるエントリもある。すなわち、図１５の第１実施形態では、矩形Ｅ１２２とＥ１２４の左端の位置が-2^b以下の値に対応する可能性があるので、スカラ倍点情報格納部３０８は、キャリー補正の影響を考慮に入れて、－２（＝－１－１）なるインデックスに対応するエントリを有する。ところが、図２８の変形例では、この－２なるインデックスに対応するエントリは不要である。

[0558] なぜなら、図２８に示すように、矩形Ｅ３２２とＥ３２４が表す範囲は、どちらも正の値のみを含むからである。つまり、図２８の変形例では、補正済み差分値ｄｉｆｆの符号を反転した－ｄｉｆｆの上位ｋビットが－１になることはない。したがって、スカラ倍点情報格納部３０８には、－２なるインデックスに対応するエントリは不要である。

[0559] よって、乱数生成部３０５が０以下の乱数値ｓを生成するように変形された図２８の変形例では、使われるインデックスの範囲が第１実施形態とは異なるが、エントリの数は同じである。

[0560] つまり、第１実施形態に関する図１６と同様の形式で表現した図２９にま

とめたように、図２８の変形例では、－１から（2^{k-1}＋２）までの（2^{k-1}＋４）個のインデックスが使われる。

[0561] 図２９には、図２８の矩形Ｅ３１１、Ｅ３１２、Ｅ３２２、Ｅ３１３、Ｅ３１４、およびＥ３２４を示した。また、図２８～２９は、ウィンドウサイズｋが２であり、乱数値ｓが符号つき３ビット値である（すなわちｂ＝３である）場合の例である。よって、図２９に示すように、図２８の変形例では、－１から（2^{k-1}＋２）まで（すなわち、－１から４まで）の各インデックスｈに対応して、スカラ倍点情報格納部３０８は、テーブルデータｔａｂ［ｈ］＝（ｈ×2^3＋ｓ）Ａを保持する。

[0562] 続いて、第２実施形態の変形例について説明する。図３０は、０以下の乱数値を生成するように変形された第２実施形態において、スカラ倍点情報格納部のインデックスとして使われる値の範囲について模式的に説明する図である。

[0563] 図３０において、矩形Ｅ４０１、Ｅ４０２、Ｅ４０３、Ｅ４０４は、それぞれ、第２実施形態に関する図１８の矩形Ｅ２０１、Ｅ２０２、Ｅ２０３、Ｅ２０４と同じである。

そして、判断基準として使われる符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に属する場合の補正済み差分値ｄｉｆｆの範囲は、図１８では、右端が（2^{k+b-1}－ｓ）の位置にある矩形Ｅ２１１により示される。同様に、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に属する場合の補正済み差分値ｄｉｆｆの範囲は、図１８では、右端が（2^{k+b-1}－ｓ）の位置にある矩形Ｅ２１３により示される。

[0564] それに対して、図３０の変形例では、ｓ≦０なので、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ１に属する場合の補正済み差分値ｄｉｆｆの範囲は、右端が（2^{k+b-1}－ｓ）＝（2^{k+b-1}＋｜ｓ｜）の位置にある矩形Ｅ４１１により示される。同様に、符号つき（ｋ＋ｂ）ビット値d_Hが範囲Ｒ３に属する場合の補正済み差分値ｄｉｆｆの範囲は、右端が（2^{k+b-1}＋｜ｓ｜）の位置にある矩形Ｅ４１３により示される。

[0565] ただし、乱数値ｓは符号つきｂビット値なので、｜ｓ｜＜２^{b}である。よって、矩形Ｅ４１１または矩形Ｅ４１３が示す範囲の右端に対応する値の上位ｋビットは、２$^{k-1}$より大きくはない。つまり、矩形Ｅ４１１と矩形Ｅ４１３の右端が２$^{k+b-1}$の位置より右にあっても、インデックスの増加にはつながらない。

[0566] また、判断基準として使われる符号つき（ｋ＋ｂ）ビット値ｄ$_{H}$が範囲Ｒ２に属する場合の補正済み差分値ｄｉｆｆの範囲は、図１８では、右端がｓの位置にある矩形Ｅ２１２により示される。したがって、図１８では、インデックスの範囲を示す矩形Ｅ２２２の左端の位置は、－ｓの位置である。同様に、符号つき（ｋ＋ｂ）ビット値ｄ$_{H}$が範囲Ｒ４の補正済み差分値ｄｉｆｆの範囲は、図１８では、右端がｓの位置にある矩形Ｅ２１４により示される。したがって、図１８ではインデックスの範囲を示す矩形Ｅ２２４の左端の位置は、－ｓの位置である。

[0567] それに対して、図３０の変形例では、ｓ≦０なので、符号つき（ｋ＋ｂ）ビット値ｄ$_{H}$が範囲Ｒ２に属する場合の補正済み差分値ｄｉｆｆの範囲は、右端がｓ＝－｜ｓ｜の位置にある矩形Ｅ４１２により示される。したがって、図３０では、インデックスの範囲を示す矩形Ｅ４２２の左端の位置は、｜ｓ｜の位置であり、０よりも右側である。また、矩形Ｅ４２２の右端の位置は、（２$^{k+b-1}$＋｜ｓ｜）の位置であり、２$^{k+b-1}$の位置より右である。しかし、｜ｓ｜＜２^{b}なので、矩形Ｅ４２２の右端が２$^{k+b-1}$の位置より右にあっても、インデックスの増加にはつながらない。

[0568] 同様のことは符号つき（ｋ＋ｂ）ビット値ｄ$_{H}$が範囲Ｒ４に属する場合の補正済み差分値ｄｉｆｆの範囲を示す矩形Ｅ４１４と、矩形Ｅ４１４が示す範囲の符号を反転させた範囲を示す矩形Ｅ４２４についても成り立つ。すなわち、インデックスの範囲を示す矩形Ｅ４２４は、右端が２$^{k+b-1}$の位置より右にあるが、インデックスの増加をもたらすことはない。

[0569] 以上から、第２実施形態は、たとえ乱数生成部３０５が０以下の乱数値を生成するように変形されても、スカラ倍点情報格納部３０８で使われるイン

デックスの範囲は変わらない。

[0570] ただし、乱数値ｓが非ゼロに限定される第３実施形態は、乱数生成部３０５が負の乱数値を生成するように変形されると、図３１を参照して以下に説明するとおり、インデックスの個数も変わってしまう。

[0571] 図３１は、０以下の乱数値を生成するように変形された第２実施形態と、負の乱数値を生成するように変形された第３実施形態において、インデックスとして使われる値についてまとめた図である。図３１は、第３実施形態によるメモリ使用量の削減について説明する図２３と同様の形式の図である。すなわち、図３１では、乱数値ｓが負の場合と、乱数値ｓが０の場合が別々に図示されている。

[0572] 乱数値ｓが負の場合に関して、図３１には、図３０の矩形Ｅ４１１が表す範囲と矩形Ｅ４１３が表す範囲の和を表す矩形Ｅ４３１と、図３０の矩形Ｅ４１２が表す範囲と矩形Ｅ４１４が表す範囲の和を表す矩形Ｅ４３２が示されている。また、図３１には、矩形Ｅ４３２が表す範囲の符号を反転させた範囲を表す矩形Ｅ４４２も示されている。

[0573] 図３１に示すとおり、ｓ＜０のとき、０＜｜ｓ｜なので、矩形Ｅ４３１と矩形Ｅ４４２が表す範囲は正の値のみを含む。また、０＜｜ｓ｜なので、矩形Ｅ４３１と矩形Ｅ４４２の右端は、2^{k-1}なるインデックスに対応する位置よりも右にある。したがって、乱数値ｓを非ゼロに限定したｓ＜０の場合であっても、キャリー補正の影響がない段階で、2^{k-1}なるインデックスが使われる可能性があり、キャリー補正を考慮に入れると、（2^{k-1}＋１）なるインデックスが使われる可能性がある。

[0574] また、乱数値ｓが０の場合に関して、図３１には、図３０の矩形Ｅ４１１が表す範囲と矩形Ｅ４１３が表す範囲の和を表す矩形Ｅ４５１と、図３０の矩形Ｅ４１２が表す範囲と矩形Ｅ４２４が表す範囲の和を表す矩形Ｅ４５２が示されている。また、図３１には、矩形Ｅ４５２が表す範囲の符号を反転させた範囲を表す矩形Ｅ４６２も示されている。

[0575] 図３１に示すとおり、ｓ＝０の場合は、インデックスの範囲を示す矩形Ｅ

４５１と矩形Ｅ４６２は、図２３の矩形Ｅ２３１とＥ２４２と同じである。よって、第２実施形態と同じく、図３０の変形例でｓ＝０の場合、キャリー補正を考慮に入れると、－１から（2^{k-1}＋１）までのインデックスが使われる可能性がある。

[0576] つまり、第３実施形態には、乱数値ｓを非ゼロに限定することでメモリ消費量を第２実施形態よりも少なくする効果があるが、乱数生成部３０５が負の乱数値を生成する変形例では、乱数値ｓを非ゼロに限定しても、エントリ数は変わらない。逆の観点から説明すれば、単に乱数値ｓを非ゼロの値に限定するのではなく、第３実施形態のように乱数値ｓを正の非ゼロの値に限定することで、メモリ消費量の削減効果が生まれる。

[0577] 以上、いくつかの観点から様々な変形例について説明したが、第１～第３実施形態および以上の様々な変形例について概括すれば下記のとおりである。

処理部３０１は、個人鍵格納部３０４から楕円曲線暗号における個人鍵ｄを読み出し、処理対象のビット列Ｄを認識する。処理部３０１は、個人鍵ｄ自体を処理対象のビット列Ｄとして認識してもよい。あるいは、処理部３０１は、ＭＳＢの値が０になるように個人鍵ｄを加工し、加工して得られたビット列を処理対象のビット列Ｄとして認識してもよい。上記のダミー鍵ｅとゼロ埋めビット列ｒは、ＭＳＢの値が０になるように処理部３０１が個人鍵ｄを加工することで得られるビット列Ｄの具体例である。

[0578] また、乱数生成部３０５は符号つきまたは符号なしのｂビットの乱数値ｓを生成し、正整数ｍに関して、ビット列Ｄの長さｕとウィンドウサイズｋとの間にｕ＝ｍｋ＋ｂなる関係が成立する。処理部３０１は、正整数ｍに関して、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｋビット値のウィンドウ値ｗ［ｉ］と０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｂビット値の乱数値ｓ［ｉ］と補正値ｇを定める。具体的には、処理部３０１は、ビット列Ｄと乱数値ｓを用いて、各乱数値ｓ［ｉ］を＋ｓまたは－ｓに決定しながら、式（１２．７）が成立するという制約条件下でこれらの値を定める。

[0579] より具体的には、第１実施形態またはその変形例では、処理部３０１内の判断部３０２は、以下のいずれかの判定条件が満たされるか否かを判断する。

・乱数値ｓ［ｉ］が＋ｓであるという仮定の下でウィンドウ値ｗ［ｉ］として見積もられる値である第１の値が０以上２^{k-1}未満である、という第１の判定条件

・１つ上位のウィンドウ値ｗ［ｉ＋１］へのキャリー補正値と相殺するウィンドウ補正値により第１の値を補正した第２の値が０以上２^{k-1}以下である、という第２の判定条件

なお、上記の第１の値は、ｉ＝ｍ－１の場合は、ビット列Ｄの上位（ｋ＋ｂ）ビット（例えば、図１１のステップＳ３０１で設定される符号つき（ｋ＋ｂ）ビット値$ｄ_{H}$の初期値）から乱数値ｓを引いた値の上位ｋビットである。

[0580] また、ｉ＜ｍ－１の場合は、上記の第１の値は、符号つき（ｋ＋ｂ）ビット値から乱数値ｓを引いた値の、上位ｋビットである。なお、ここでの符号つき（ｋ＋ｂ）ビット値とは、ｉ＜ｊ≦ｍ－１なる各ｊについて決定部３０３が算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与をビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である。より正確には、ｊ＝ｉ＋１なるｊに関しては、ビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺されるウィンドウ値ｗ［ｊ］の寄与は、キャリー補正前のウィンドウ値ｗ［ｊ］の寄与である。また、この符号つき（ｋ＋ｂ）ビット値の具体例は、第１実施形態においては、図１１のステップＳ３１８で更新された符号つき（ｋ＋ｂ）ビット値$ｄ_{H}$である。

[0581] 上記の第１と第２の判定条件に関する判断の具体的手順は、例えば次のとおりである。すなわち、判断基準値（$ｄ_{H}$－ｓ）が範囲Ｒ３に属するとき、判断部３０２は、「第１の判定条件が満たされる」と判断してもよい。また、判断基準値（$ｄ_{H}$－ｓ）が範囲Ｒ１に属するとき、判断部３０２は、「第２の判定条件が満たされる」と判断してもよい。

[0582] そして、第１の判定条件が満たされる場合、処理部３０１内の決定部３０３は、図１１のステップＳ３１０、Ｓ３１１、Ｓ３１５、Ｓ３１６に例示したように、乱数値ｓ［ｉ］を＋ｓと決定し、ウィンドウ値ｗ［ｉ］を第１の値に決定する。また、第２の判定条件が満たされる場合、決定部３０３は、図１１のステップＳ３０５～Ｓ３０７、Ｓ３１５、Ｓ３１６に例示したように、乱数値ｓ［ｉ］を＋ｓと決定し、ウィンドウ値ｗ［ｉ］を第２の値に決定する。

[0583] 逆に、第１の判定条件も第２の判定条件も満たされない場合、決定部３０３は、乱数値ｓ［ｉ］をーｓと決定し、ーｓと決定した乱数値ｓ［ｉ］に応じてウィンドウ値ｗ［ｉ］を算出する。

[0584] その際、決定部３０３は、第１の値が正であれば、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負の値によるウィンドウ補正を行う。例えば、第１実施形態の例では、判断基準値（d_H－ｓ）が範囲Ｒ４に属していれば、決定部３０３は図１１のステップＳ３１３でウィンドウ補正値ｔ［ｉ］を負の値に設定することで、ウィンドウ補正を行う。

[0585] また、乱数値ｓ［ｉ］をーｓと決定した場合に、前記第１の値が負であれば、決定部３０３は、ウィンドウ補正を行わずに、ーｓに決定した乱数値ｓ［ｉ］に応じてウィンドウ値ｗ［ｉ］を算出する。例えば、第１実施形態の例では、判断基準値（d_H－ｓ）が範囲Ｒ２に属していれば、決定部３０３はウィンドウ補正を行わない。

[0586] 以上は、第１実施形態またはその変形例における処理部３０１の動作である。第２実施形態、第３実施形態、またはその変形例における処理部３０１の動作は、概括すれば以下のとおりである。

[0587] すなわち、判断部３０２は、符号つき（ｋ＋ｂ）ビット値d_Hが、０以上2^{k+b-1}－１以下の第１の範囲（つまり範囲Ｒ３）または－2^{k+b-1}以下の第２の範囲（つまり範囲Ｒ１）に含まれるか否かを判断する。

[0588] ここでの符号つき（ｋ＋ｂ）ビット値d_Hは、ｉ＝ｍ－１の場合は、ビット列Ｄの上位（ｋ＋ｂ）ビットであり、具体例は、図１９のステップＳ４０１

または図２４のステップＳ５０１で設定される値である。また、ｉ＜ｍ－１の場合は、符号つき（ｋ＋ｂ）ビット値$ｄ_H$は、ｉ＜ｊ≦ｍ－１なる各ｊについて算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与をビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である。具体例は、図１９のステップＳ４１８または図２４のステップＳ５１８で設定される値である。なお、より正確には、ｊ＝ｉ＋１なるｊに関しては、ビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺されるウィンドウ値ｗ［ｊ］の寄与は、キャリー補正前のウィンドウ値ｗ［ｊ］の寄与である。

[0589] そして、決定部３０３は、符号つき（ｋ＋ｂ）ビット値$ｄ_H$が第１の範囲に含まれる場合、乱数値ｓ［ｉ］を＋ｓと決定し、ウィンドウ値ｗ［ｉ］を、符号つき（ｋ＋ｂ）ビット値$ｄ_H$から乱数値ｓを減じた値の上位ｋビットに決定する。この動作の例は、図１９のステップＳ４１０、Ｓ４１１、Ｓ４１５、Ｓ４１６および図２４のステップＳ５１０、Ｓ５１１、Ｓ５１５、Ｓ５１６に示したとおりである。

[0590] また、符号つき（ｋ＋ｂ）ビット値$ｄ_H$が第２の範囲に含まれる場合、決定部３０３は、乱数値ｓ［ｉ］を＋ｓと決定し、ウィンドウ値ｗ［ｉ］を、符号つき（ｋ＋ｂ）ビット値$ｄ_H$から乱数値ｓを減じて$２^{ｋ+ｂ}$なるウィンドウ補正値を加えた値の上位ｋビットに決定する。この動作の例は、図１９のステップＳ４０５～Ｓ４０７、Ｓ４１５、Ｓ４１６および図２４のステップＳ５０５～Ｓ５０７、Ｓ５１５、Ｓ５１６に示したとおりである。

[0591] そして、符号つき（ｋ＋ｂ）ビット値$ｄ_H$が第１の範囲にも第２の範囲にも含まれない場合、決定部３０３は、乱数値ｓ［ｉ］を－ｓと決定し、－ｓと決定した乱数値ｓ［ｉ］に応じてウィンドウ値ｗ［ｉ］を算出する。

[0592] その際、決定部３０３は、具体的には、符号つき（ｋ＋ｂ）ビット値$ｄ_H$が正であれば、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負のウィンドウ補正値によるウィンドウ補正を行う。この動作の例は、図１９のステップＳ４１２～４１６および図２４のステップＳ５１２～Ｓ５１６に示したとおりである。

[0593] 逆に、乱数値ｓ［ｉ］を－ｓと決定した場合において、符号つき（ｋ＋ｂ）ビット値$ｄ_{Ｈ}$が負であれば、決定部３０３は、ウィンドウ補正を行わずに、乱数値ｓ［ｉ］に応じてウィンドウ値ｗ［ｉ］を算出する。この動作の例は、図１９のステップＳ４０８、Ｓ４０９、Ｓ４１５、Ｓ４１６および図２４のステップＳ５０８、Ｓ５０９、Ｓ５１５、Ｓ５１６に示したとおりである。

[0594] また、以上説明したいずれの実施形態においても、スカラ倍算部３０７が楕円曲線上の点（$２^{ｂ}ｈ$＋ｓ）Ａの座標を計算するインデックスｈの範囲は、次のとおりである。すなわち、インデックスｈは、処理部３０１が－ｓと決定する乱数値ｓ［ｉ］に対応するウィンドウ値ｗ［ｉ］の値域の最大値（例えば、第１実施形態では２、第２実施形態と第３実施形態では１）の符号を反転させた負数以上である。かつ、インデックスｈは、処理部３０１が＋ｓと決定する乱数値ｓ［ｉ］に対応するウィンドウ値ｗ［ｉ］の値域の最大値（例えば、第１実施形態と第２実施形態では$２^{ｋ－１}$＋１、第３実施形態では$２^{ｋ－１}$）以下である。

[0595] 最後に、以上説明した第１～第３実施形態およびそれらの変形例、ならびに２つの比較例におけるテーブルデータのエントリ数についてまとめると、図３２のとおりである。

すなわち、第１～第３実施形態は、いずれもＰＡ攻撃に対して安全（つまりＳＰＡ攻撃とＤＰＡ攻撃の双方に対して安全）である。また、第３の比較例として説明したランダム化ウィンドウ法も、ＰＡ攻撃に対して安全である。しかし、第４の比較例として説明した、メモリ節約効果のある符号つきウィンドウ法は、ＤＰＡ攻撃に対して脆弱である。

[0596] そして、第１実施形態では、ｓ≧０の場合のスカラ倍点情報のインデックスの範囲は、$－２≦ｈ≦２^{ｋ－１}＋１$であり、変形例として説明したｓ≦０の場合のインデックスの範囲は、$－１≦ｈ≦２^{ｋ－１}＋２$である。よって、スカラ倍点情報のエントリ数は、いずれにせよ（$２^{ｋ－１}＋４$）個である。例えば、ウィンドウサイズｋが２、３、４の場合それぞれのエントリ数は、６個、８個、

１２個である。

[0597] また、第２実施形態では、ｓ≧０の場合も、変形例として説明したｓ≦０の場合も、スカラ倍点情報のインデックスの範囲は、－１≦ｈ≦$２^{ｋ-１}$＋１である。よって、スカラ倍点情報のエントリ数は（$２^{ｋ-１}$＋３）個である。例えば、ウィンドウサイズｋが２、３、４の場合それぞれのエントリ数は、５個、７個、１１個である。

[0598] また、乱数値ｓを非ゼロの値に限定した第３実施形態では、ｓ＞０の場合とｓ＜０の場合でスカラ倍点情報のインデックスの範囲が異なり、したがって、エントリ数も異なる。具体的には、ｓ＞０の場合、インデックスの範囲は－１≦ｈ≦$２^{ｋ-１}$であり、エントリ数は（$２^{ｋ-１}$＋２）個である。例えば、ウィンドウサイズｋが２、３、４の場合それぞれのエントリ数は、４個、６個、１０個である。ｓ＜０の場合のインデックスの範囲とエントリ数は、第２実施形態と同じである。

[0599] 第３の比較例では、テーブルデータのインデックスの範囲は０≦ｈ≦$２^{ｋ}$－１であり、エントリ数は$２^{ｋ}$個である。例えば、ウィンドウサイズｋが２、３、４の場合それぞれのエントリ数は、４個、８個、１６個である。

[0600] よって、第１実施形態と第３の比較例を比べると、ｋが３以上のとき第１実施形態でのエントリ数は第３の比較例でのエントリ数以下であり、特にｋが４以上のとき第１実施形態でのエントリ数は第３の比較例でのエントリ数より少ない。

[0601] また、第２実施形態と第３の比較例を比べると、ｋが３以上のとき第２実施形態でのエントリ数は第３の比較例でのエントリ数より少ない。同様のことは、第３実施形態においてｓ＜０の場合にも成り立つ。

[0602] さらに、ｓ＞０の場合に関して第３実施形態と第３の比較例を比べると、ｋが２以上のとき第３実施形態でのエントリ数は第３の比較例でのエントリ数以下であり、特にｋが３以上のとき第３実施形態でのエントリ数は第３の比較例でのエントリ数より少ない。

[0603] すなわち、第１～第３実施形態でのエントリ数は、$２^{ｋ-１}$のオーダであるの

に対し、第３の比較例でのエントリ数は、2^kのオーダである。よって、第１～第３実施形態でのエントリ数は、ウィンドウサイズｋによっては第３の比較例でのエントリ数を超えることもありうるものの、多くの場合は、第３の比較例でのエントリ数より少ない。また、ウィンドウサイズは、ｋ≧３であることが多い。したがって、実用上はほとんどの場合において、第１～第３実施形態は、メモリ使用量の点で第３の比較例に対する優位性を持つ。

[0604] また、第４の比較例では、テーブルデータのインデックスの範囲は０≦ｈ≦2^{k-1}であり、エントリ数は（2^{k-1}＋１）個である。例えば、ウィンドウサイズｋが２、３、４の場合それぞれのエントリ数は、３個、５個、９個である。このように、ウィンドウサイズｋによらず、第４の比較例でのエントリ数は常に第１～第３実施形態でのエントリ数よりも１～３個少ないが、第４の比較例にはＤＰＡ攻撃に対する脆弱性がある。よって、第１～第３実施形態は、安全性において第４の比較例に対する優位性を持つ。

[0605] 以上、図３２に示したように、第１～第３実施形態およびその変形例によれば、ＳＰＡ攻撃とＤＰＡ攻撃の双方に対する安全性と、メモリ使用量の抑制が、ともに実現される。したがって、第１～第３実施形態およびその変形例の暗号処理装置は、様々な分野に好適であり、特に、スマートカードや組み込み機器などのメモリ容量が少ない装置に好適である。

請求の範囲

[請求項1] 楕円曲線暗号における個人鍵ｄを格納する個人鍵格納部と、

符号つきまたは符号なしのｂビットの乱数値ｓを生成する乱数生成部と、

前記個人鍵格納部から前記個人鍵ｄを読み出し、前記個人鍵ｄ、または最上位ビットの値が０になるように前記個人鍵ｄを加工して得られるビット列であるビット列Ｄの長さｕとウィンドウサイズｋとの間にｕ＝ｍｋ＋ｂなる関係が成立する正整数ｍに関して、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｋビット値のウィンドウ値ｗ［ｉ］と０≦ｉ≦（ｍ－１）なる各ｉに対応する符号つきｂビット値の乱数値ｓ［ｉ］と補正値ｇを、前記ビット列Ｄと前記乱数値ｓとを用いて、各乱数値ｓ［ｉ］を＋ｓまたはーｓに決定しながら、

[数16]

$$D = g + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i])$$

が成立するという制約条件下で定める処理部

を備えることを特徴とする暗号処理装置。

[請求項2] 前記乱数値ｓ［ｉ］が＋ｓであるという仮定の下で前記ウィンドウ値ｗ［ｉ］として見積もられる値である第１の値が０以上2^{k-1}未満である、という第１の判定条件、または、前記ウィンドウ値ｗ［ｉ］の１つ上位のウィンドウ値ｗ［ｉ＋１］へのキャリー補正値と相殺するウィンドウ補正値により前記第１の値を補正した第２の値が０以上2^{k-1}以下である、という第２の判定条件が満たされるか否かを判断する判断部と、

前記第１の判定条件が満たされる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を前記第１の値に決定し、前記

第２の判定条件が満たされる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を前記第２の値に決定し、前記第１の判定条件も前記第２の判定条件も満たされない場合、前記乱数値ｓ［ｉ］をーｓと決定し、ーｓと決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する決定部

を前記処理部が備えることを特徴とする請求項１に記載の暗号処理装置。

[請求項3] 前記第１の値は、

ｉ＝ｍ－１の場合は、前記ビット列Ｄの上位（ｋ＋ｂ）ビットから前記乱数値ｓを引いた値の上位ｋビットであり、

ｉ＜ｍ－１の場合は、ｉ＜ｊ≦ｍ－１なる各ｊについて前記決定部が算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与を前記ビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である符号つき（ｋ＋ｂ）ビット値から、前記乱数値ｓを引いた値の、上位ｋビットである

こと特徴とする請求項２に記載の暗号処理装置。

[請求項4] 前記決定部は、前記乱数値ｓ［ｉ］をーｓと決定した場合、

前記第１の値が正であれば、ーｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出するときに、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負の値によるウィンドウ補正を行い、

前記第１の値が負であれば、前記ウィンドウ補正を行わずに、ーｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

こと特徴とする請求項２または３に記載の暗号処理装置。

[請求項5] ｉ＝ｍ－１の場合は、前記ビット列Ｄの上位（ｋ＋ｂ）ビットの値であり、ｉ＜ｍ－１の場合は、ｉ＜ｊ≦ｍ－１なる各ｊについて算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与を前記ビット列Ｄ

の上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である、符号つき（ｋ＋ｂ）ビット値$ｄ_H$が、０以上$２^{ｋ+ｂ-１}$－１以下の第１の範囲または$－２^{ｋ+ｂ-１}$以下の第２の範囲に含まれるか否かを判断する判断部と、

前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$が前記第１の範囲に含まれる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を、前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$から前記乱数値ｓを減じた値の上位ｋビットに決定し、前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$が前記第２の範囲に含まれる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を、前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$から前記乱数値ｓを減じて$２^{ｋ+ｂ}$なるウィンドウ補正値を加えた値の上位ｋビットに決定し、前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$が前記第１の範囲にも前記第２の範囲にも含まれない場合、前記乱数値ｓ［ｉ］を－ｓと決定し、－ｓと決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する決定部

を前記処理部が備えることを特徴とする請求項１に記載の暗号処理装置。

[請求項6] 前記決定部は、前記乱数値ｓ［ｉ］を－ｓと決定した場合、

前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$が正であれば、－ｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出するときに、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負のウィンドウ補正値によるウィンドウ補正を行い、

前記符号つき（ｋ＋ｂ）ビット値$ｄ_H$が負であれば、前記ウィンドウ補正を行わずに、－ｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

ことを特徴とする請求項５に記載の暗号処理装置。

[請求項7] 前記乱数生成部は、前記乱数値ｓとして、非ゼロの正の値のみを生成することを特徴とする請求項５または６に記載の暗号処理装置。

[請求項8] 前記処理部が－ｓと決定する前記乱数値ｓ［ｉ］に対応する前記ウィンドウ値ｗ［ｉ］の値域の最大値の符号を反転させた負数以上であり、かつ前記処理部が＋ｓと決定する前記乱数値ｓ［ｉ］に対応する前記ウィンドウ値ｗ［ｉ］の値域の最大値以下である各インデックスｈに対して、楕円曲線上の点（２^{b}ｈ＋ｓ）Ａの座標を計算するスカラ倍算部と、

前記負数以上かつ前記最大値以下の各インデックスｈについて、前記スカラ倍算部が計算した前記点（２^{b}ｈ＋ｓ）Ａの前記座標を、前記インデックスｈと対応づけて記憶するスカラ倍点情報格納部

をさらに備えることを特徴とする請求項１から７のいずれか１項に記載の暗号処理装置。

[請求項9] 楕円曲線暗号における個人鍵ｄを読み出し、

前記個人鍵ｄを処理対象のビット列Ｄとして認識するか、または、最上位ビットの値が０になるように前記個人鍵ｄを加工し、加工して得られたビット列を前記ビット列Ｄとして認識し、

符号つきまたは符号なしのｂビットの乱数値ｓを生成し、

前記ビット列Ｄの長さｕとウィンドウサイズｋとの間にｕ＝ｍｋ＋ｂなる関係が成立する正整数ｍに関して、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号付きｋビット値のウィンドウ値ｗ［ｉ］と、０≦ｉ≦（ｍ－１）なる各ｉに対応する符号付きｂビット値の乱数値ｓ［ｉ］と、補正値ｇを、前記ビット列Ｄと前記乱数値ｓとを用いて、各乱数値ｓ［ｉ］を＋ｓまたは－ｓに決定しながら、

[数17]

$$D = g + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i])$$

が成立するという制約条件下で定める

ことを特徴とする暗号処理方法。

[請求項10] 前記乱数値ｓ［ｉ］が＋ｓであるという仮定の下で前記ウィンドウ値ｗ［ｉ］の値を見積もり、

見積もった値である第１の値が０以上２^{k-1}未満である、という第１の判定条件、または、前記ウィンドウ値ｗ［ｉ］の１つ上位のウィンドウ値ｗ［ｉ＋１］へのキャリー補正値と相殺するウィンドウ補正値により前記第１の値を補正した第２の値が０以上２^{k-1}以下である、という第２の判定条件が満たされるか否かを判断し、

前記第１の判定条件が満たされる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を前記第１の値に決定し、

前記第２の判定条件が満たされる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を前記第２の値に決定し、

前記第１の判定条件も前記第２の判定条件も満たされない場合、前記乱数値ｓ［ｉ］をーｓと決定し、ーｓと決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

ことを含むことを特徴とする請求項９に記載の暗号処理方法。

[請求項11] 前記第１の値は、

ｉ＝ｍ－１の場合は、前記ビット列Ｄの上位（ｋ＋ｂ）ビットから前記乱数値ｓを引いた値の上位ｋビットであり、

ｉ＜ｍ－１の場合は、ｉ＜ｊ≦ｍ－１なる各ｊについて算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与を前記ビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である符号つき（ｋ＋ｂ）ビット値から、前記乱数値ｓを引いた値の、上位ｋビットである

ことを特徴とする請求項１０に記載の暗号処理方法。

[請求項12] 前記乱数値ｓ［ｉ］をーｓと決定した場合、

前記第１の値が正であれば、ーｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出するときに、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負の値に

よるウィンドウ補正を行い、

前記第１の値が負であれば、前記ウィンドウ補正を行わずに、－ｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

ことを特徴とする請求項１０または１１に記載の暗号処理方法。

[請求項13] ｉ＝ｍ－１の場合は、前記ビット列Ｄの上位（ｋ＋ｂ）ビットの値であり、ｉ＜ｍ－１の場合は、ｉ＜ｊ≦ｍ－１なる各ｊについて算出したウィンドウ値ｗ［ｊ］と乱数値ｓ［ｊ］の寄与を前記ビット列Ｄの上位（ｋ（ｍ－ｉ）＋ｂ）ビットから減殺した値である、符号つき（ｋ＋ｂ）ビット値d_Hを計算し、

前記符号つき（ｋ＋ｂ）ビット値d_Hが０以上$2^{k+b-1}-1$以下の第１の範囲または-2^{k+b-1}以下の第２の範囲に含まれるか否かを判断し、

前記符号つき（ｋ＋ｂ）ビット値d_Hが前記第１の範囲に含まれる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を、前記符号つき（ｋ＋ｂ）ビット値d_Hから前記乱数値ｓを減じた値の上位ｋビットに決定し、

前記符号つき（ｋ＋ｂ）ビット値d_Hが前記第２の範囲に含まれる場合、前記乱数値ｓ［ｉ］を＋ｓと決定し、前記ウィンドウ値ｗ［ｉ］を、前記符号つき（ｋ＋ｂ）ビット値d_Hから前記乱数値ｓを減じて2^{k+b}なるウィンドウ補正値を加えた値の上位ｋビットに決定し、

前記符号つき（ｋ＋ｂ）ビット値d_Hが前記第１の範囲にも前記第２の範囲にも含まれない場合、前記乱数値ｓ［ｉ］を－ｓと決定し、－ｓと決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

ことを特徴とする請求項９に記載の暗号処理方法。

[請求項14] 前記乱数値ｓ［ｉ］を－ｓと決定した場合、

前記符号つき（ｋ＋ｂ）ビット値d_Hが正であれば、－ｓに決定

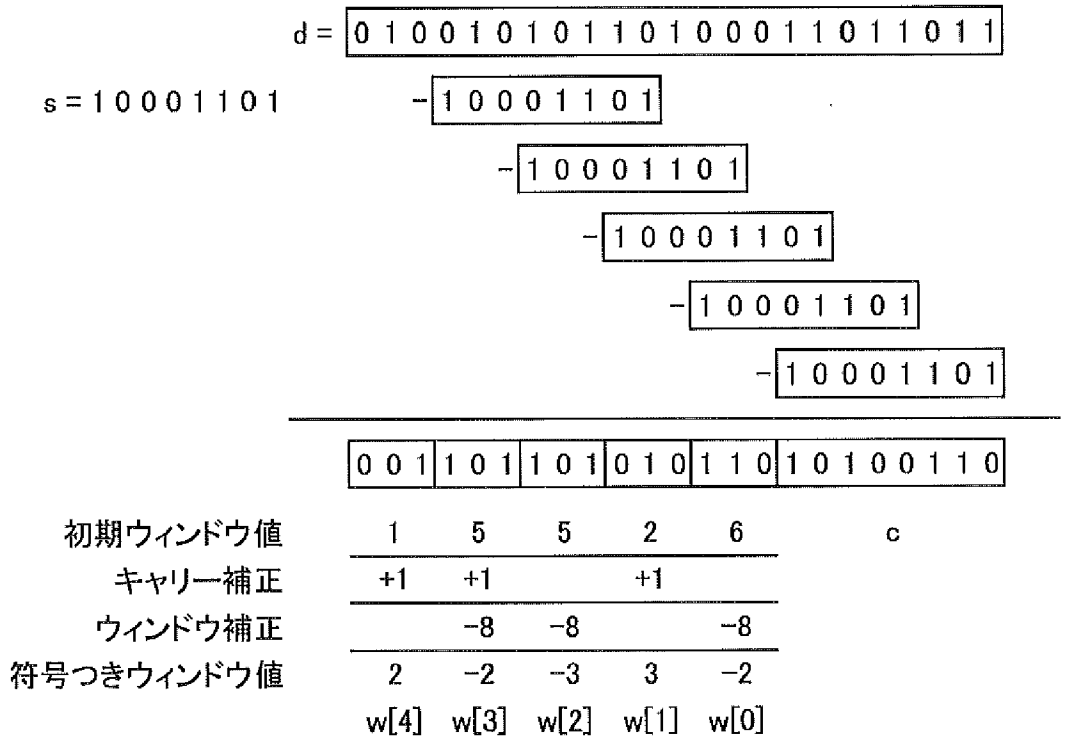
した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出するときに、１つ上位のウィンドウ値ｗ［ｉ＋１］への正のキャリー補正値と相殺する負のウィンドウ補正値による補正を行い、

前記符号つき（ｋ＋ｂ）ビット値d_Hが負であれば、前記ウィンドウ補正を行わずに、－ｓに決定した前記乱数値ｓ［ｉ］に応じて前記ウィンドウ値ｗ［ｉ］を算出する

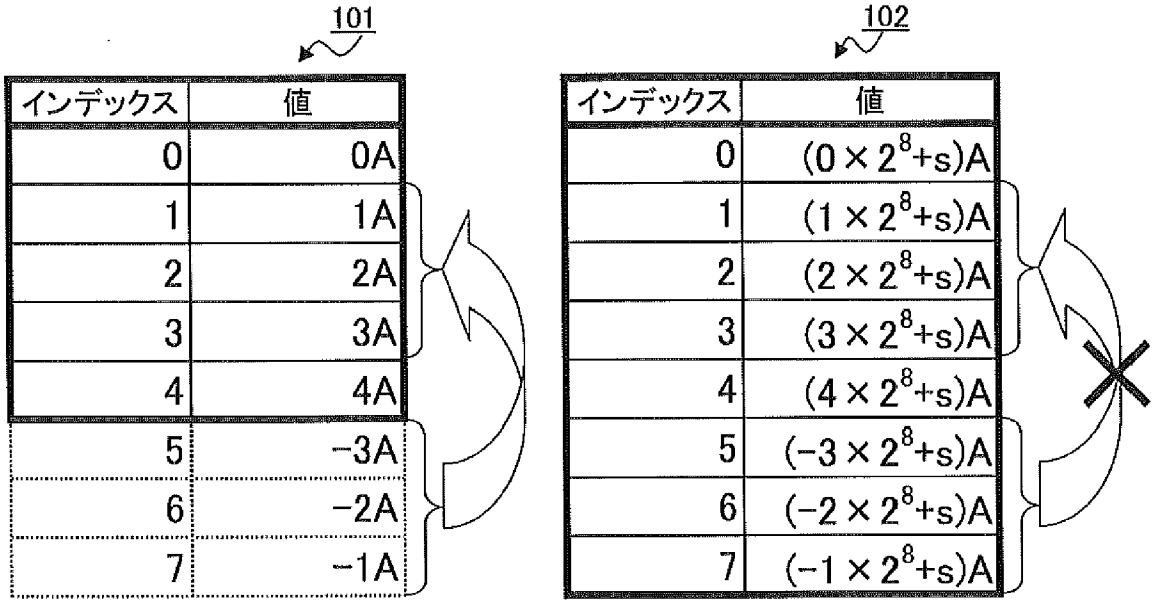
ことを特徴とする請求項１３に記載の暗号処理方法。

[請求項15] 前記乱数値ｓは非ゼロの正の値であることを特徴とする請求項１３または１４に記載の暗号処理方法。

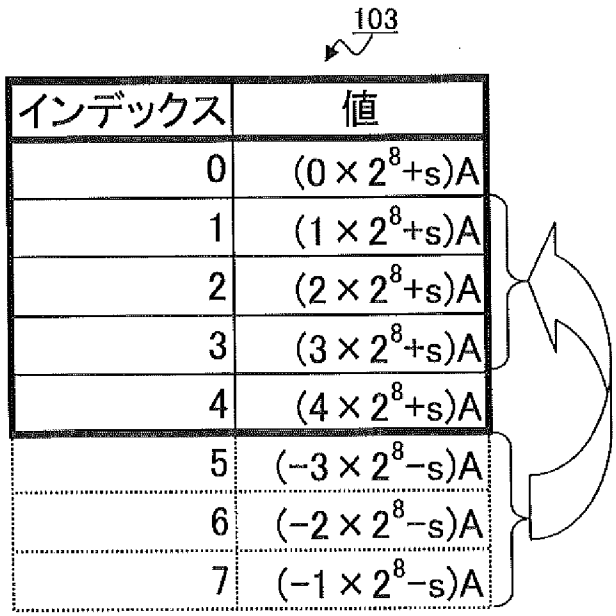
[図1]


d = 0 1 0 0 1 0 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 1
s = 1 0 0 0 1 1 0 1
- 1 0 0 0 1 1 0 1
- 1 0 0 0 1 1 0 1
- 1 0 0 0 1 1 0 1
- 1 0 0 0 1 1 0 1
- 1 0 0 0 1 1 0 1
0 0 1 1 0 1 1 0 1 0 1 0 1 1 0 1 0 0 1 1 0
初期ウィンドウ値 1 5 5 2 6 c
キャリー補正 +1 +1 +1
ウィンドウ補正 -8 -8 -8
符号つきウィンドウ値 2 -2 -3 3 -2
w[4] w[3] w[2] w[1] w[0]

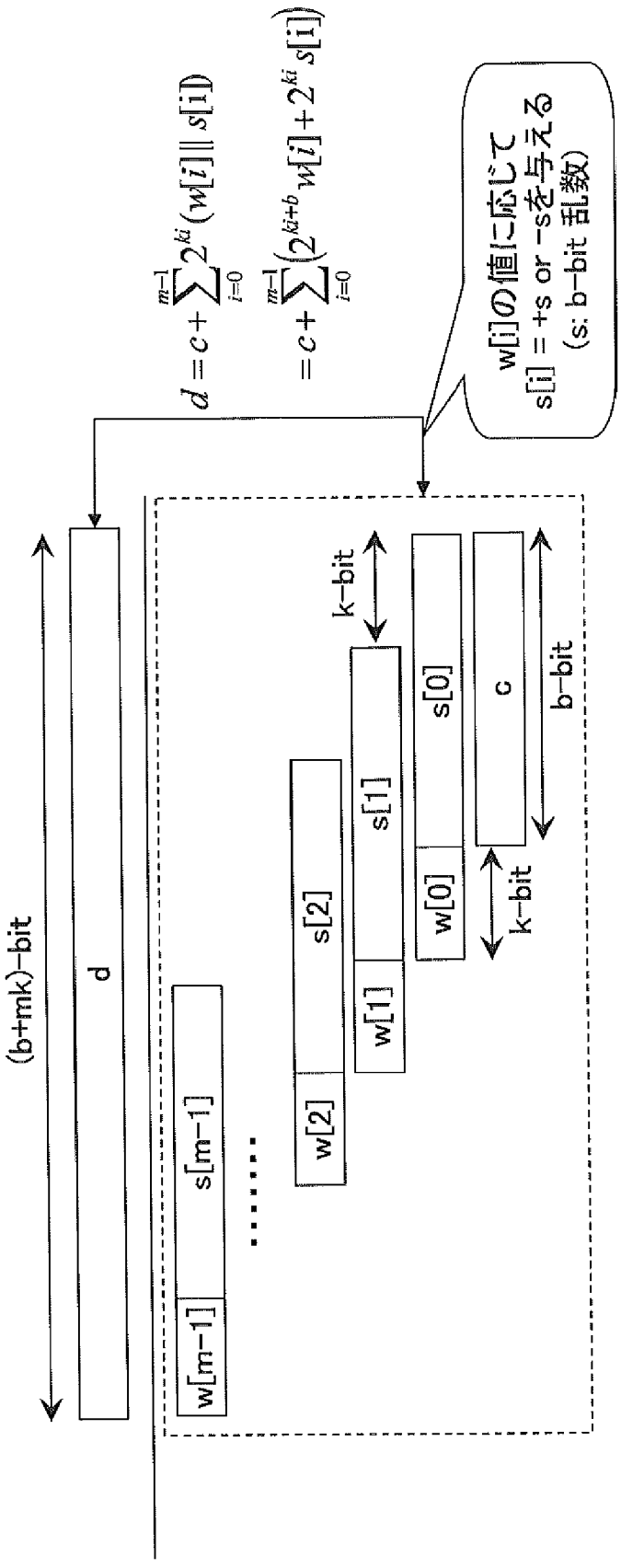
[図2]


101
102
インデックス
値
0
0A
1
1A
2
2A
3
3A
4
4A
5
−3A
6
−2A
7
−1A
インデックス
値
0
(0×2^8+s)A
1
(1×2^8+s)A
2
(2×2^8+s)A
3
(3×2^8+s)A
4
(4×2^8+s)A
5
(−3×2^8+s)A
6
(−2×2^8+s)A
7
(−1×2^8+s)A

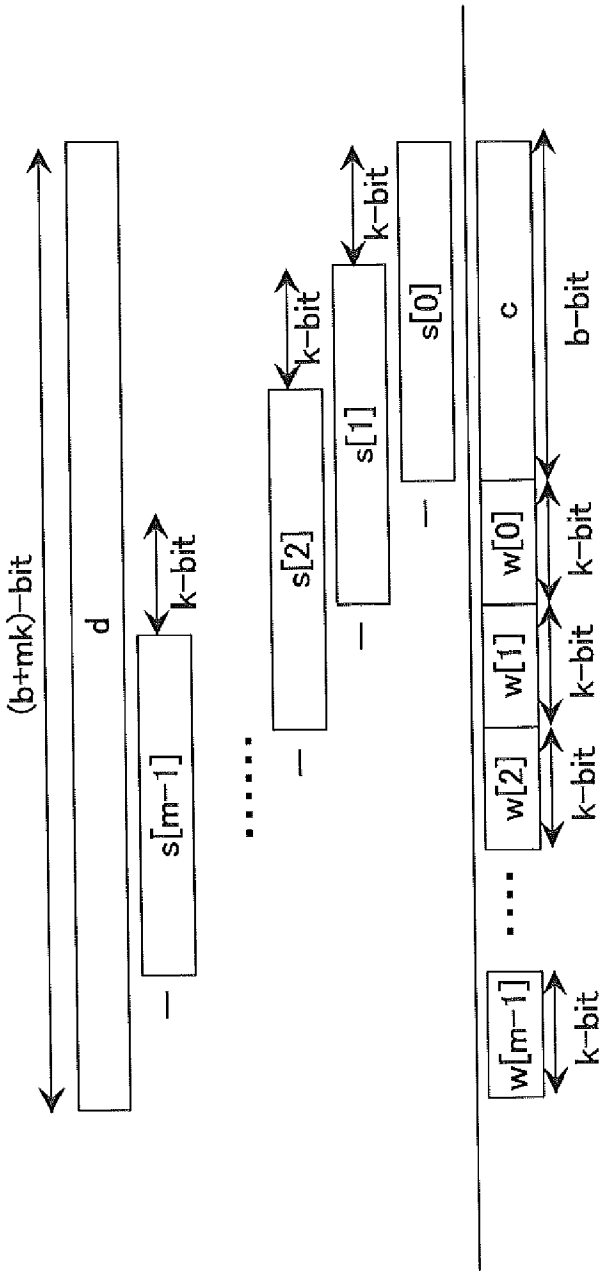
[図3]


103
インデックス
値
0
(0×2^8+s)A
1
(1×2^8+s)A
2
(2×2^8+s)A
3
(3×2^8+s)A
4
(4×2^8+s)A
5
(−3×2^8−s)A
6
(−2×2^8−s)A
7
(−1×2^8−s)A

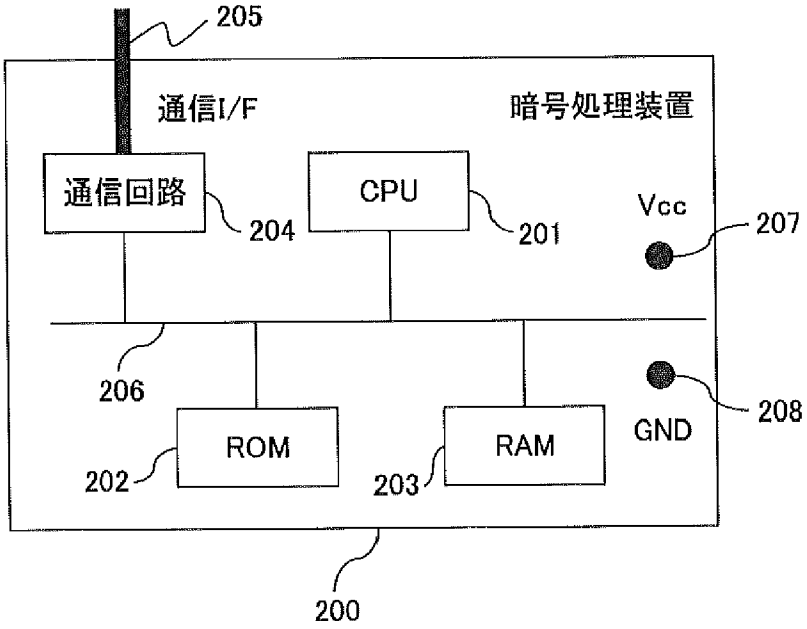
[図4]


$d = c + \sum_{i=0}^{m-1} 2^{ki}(w[i] \| s[i])$
$= c + \sum_{i=0}^{m-1} (2^{ki+b} w[i] + 2^{ki} s[i])$
w[i]の値に応じて
s[i] = +s or −sを与える
(s: b−bit 乱数)
(b+mk)−bit
d
w[m−1]
s[m−1]
w[2]
s[2]
w[1]
s[1]
w[0]
s[0]
c
k−bit
k−bit
b−bit

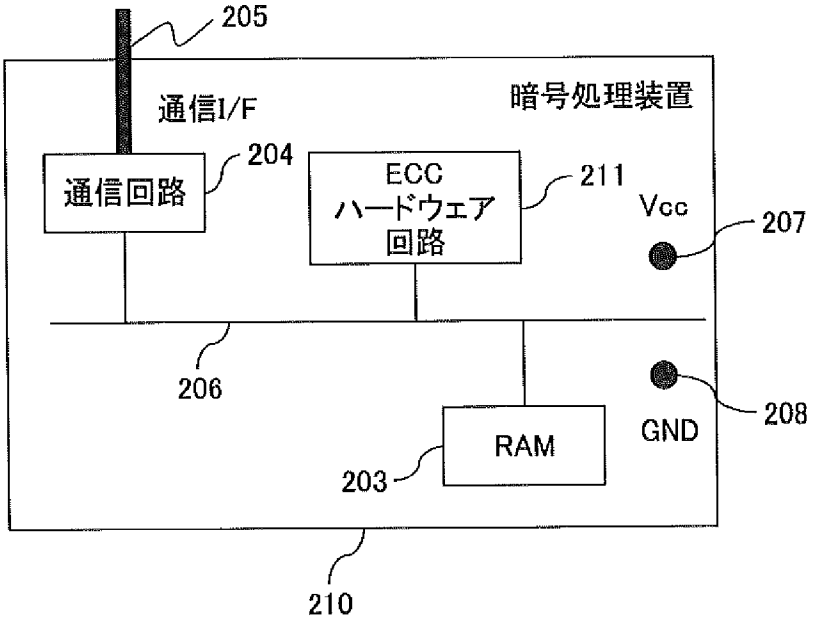
[図5]



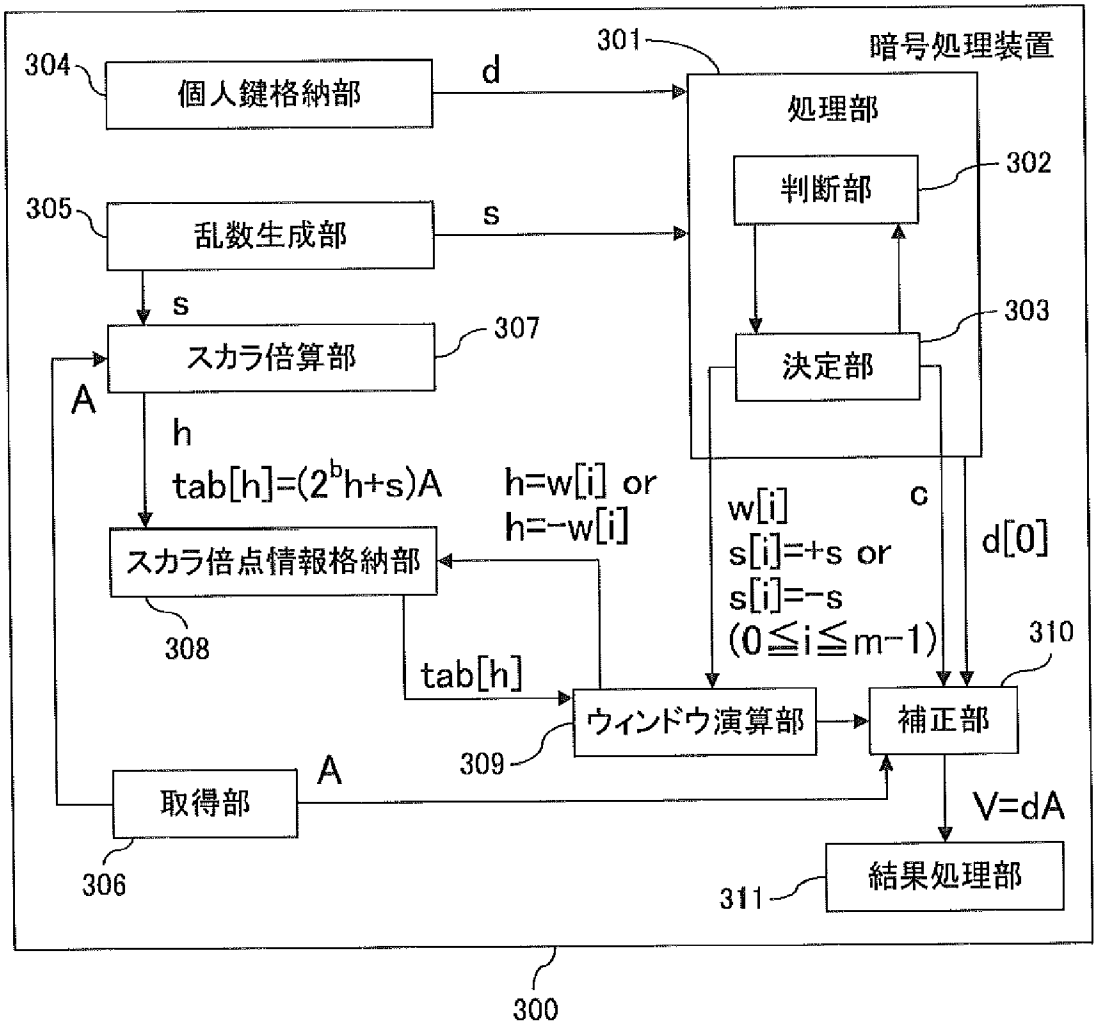
[図6]


205
通信I/F
暗号処理装置
通信回路
204
CPU
201
Vcc
207
206
ROM
202
RAM
203
GND
208
200

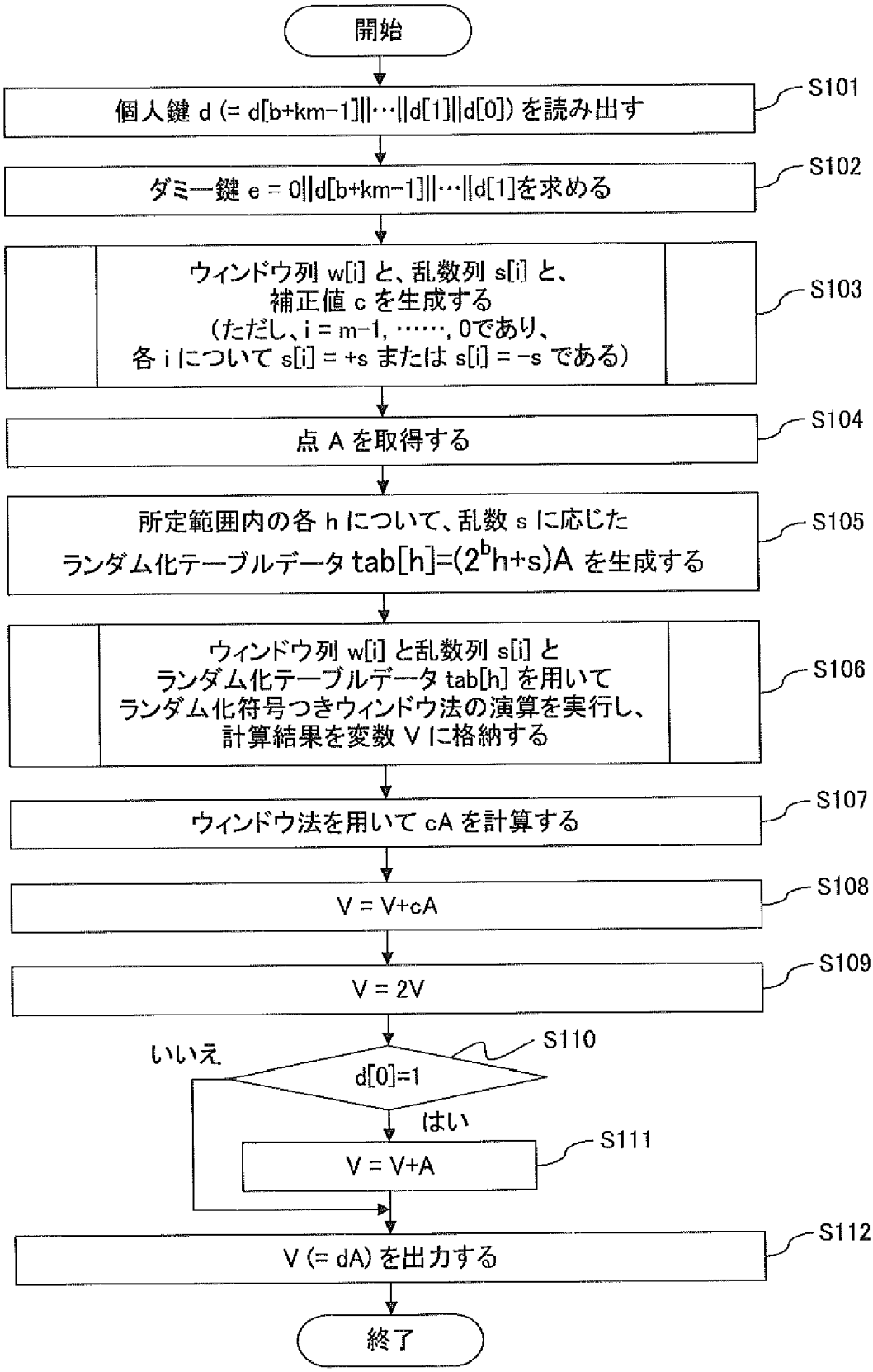
[図7]


205
通信I/F
暗号処理装置
通信回路
204
ECC
ハードウェア
回路
211
Vcc
207
206
RAM
203
GND
208
210

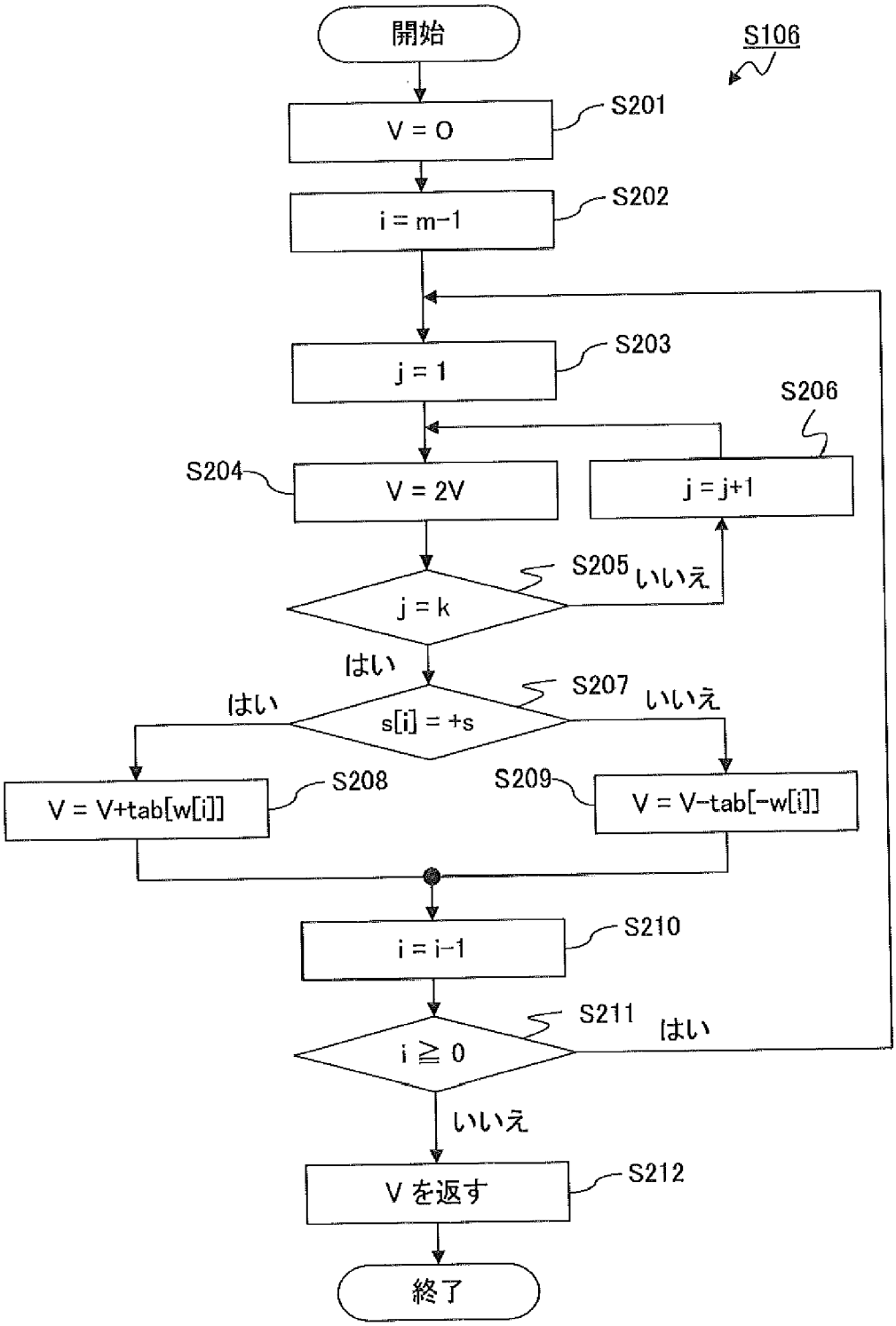
[図8]


暗号処理装置
301
304
個人鍵格納部
d
処理部
判断部
302
305
乱数生成部
s
s
スカラ倍算部
307
303
決定部
A
h
tab[h]=(2^b h+s)A
h=w[i] or
h=−w[i]
w[i]
s[i]=+s or
s[i]=−s
(0≦i≦m−1)
c
d[0]
スカラ倍点情報格納部
308
tab[h]
ウィンドウ演算部
309
補正部
310
A
取得部
306
V=dA
結果処理部
311
300

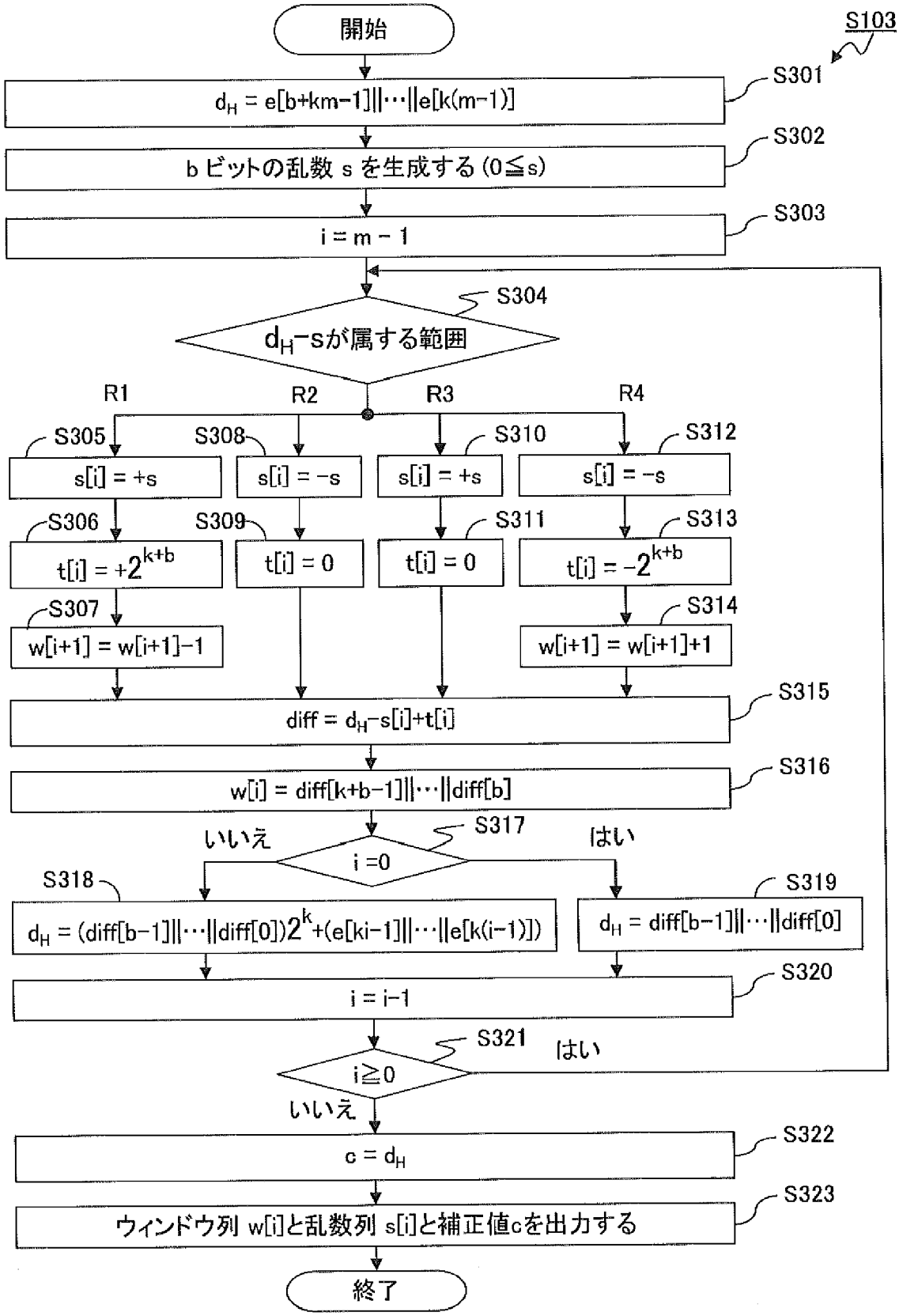
[図9]


開始
個人鍵 d (= d[b+km-1]||…||d[1]||d[0]) を読み出す
S101
ダミー鍵 e = 0||d[b+km-1]||…||d[1]を求める
S102
ウィンドウ列 w[i] と、乱数列 s[i] と、
補正値 c を生成する
(ただし、i = m-1, ……, 0であり、
各 i について s[i] = +s または s[i] = -s である)
S103
点 A を取得する
S104
所定範囲内の各 h について、乱数 s に応じた
ランダム化テーブルデータ tab[h]=(2^b h+s)A を生成する
S105
ウィンドウ列 w[i] と乱数列 s[i] と
ランダム化テーブルデータ tab[h] を用いて
ランダム化符号つきウィンドウ法の演算を実行し、
計算結果を変数 V に格納する
S106
ウィンドウ法を用いて cA を計算する
S107
V = V+cA
S108
V = 2V
S109
d[0]=1
S110
いいえ
はい
V = V+A
S111
V (= dA) を出力する
S112
終了

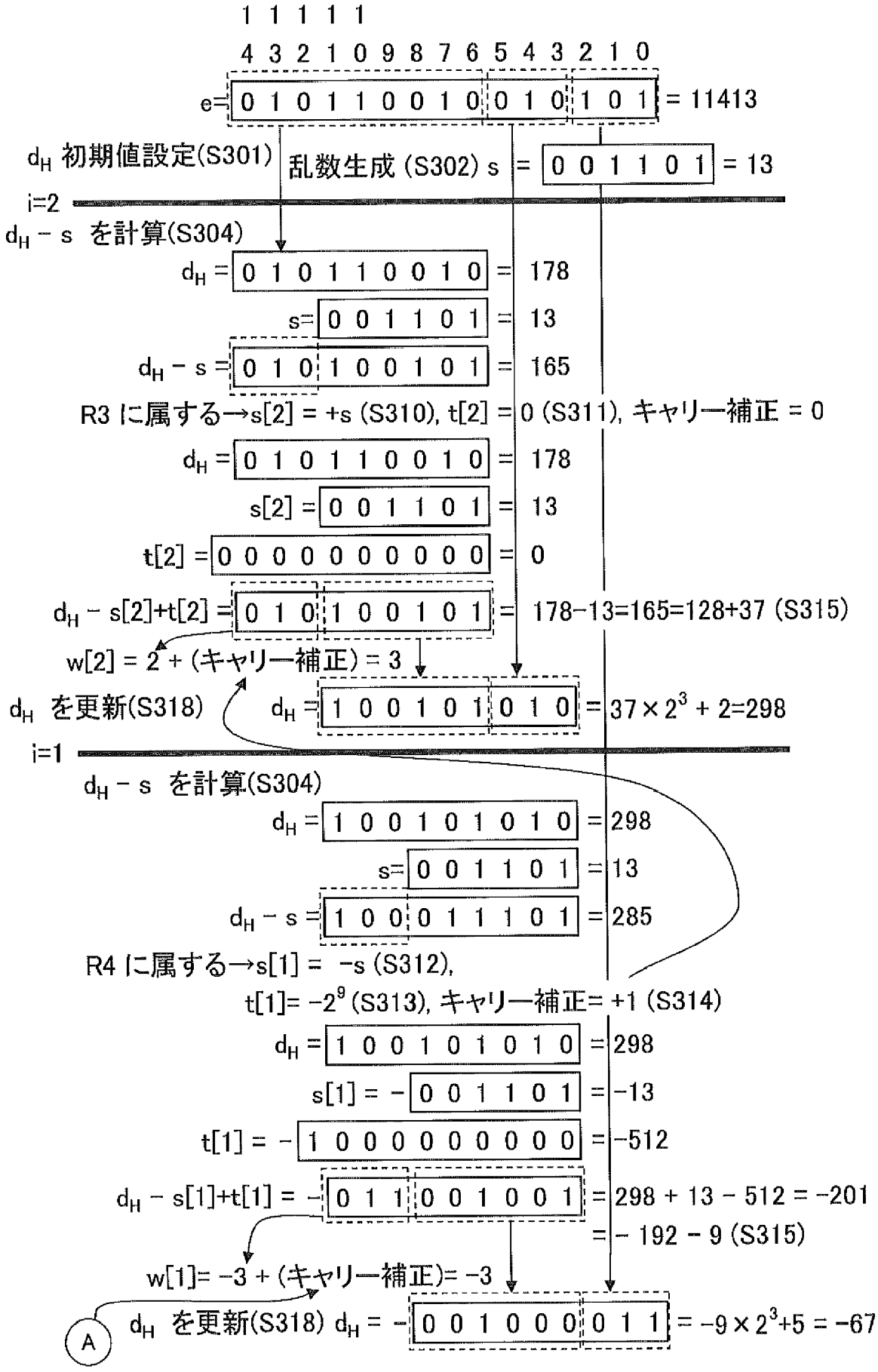
[図10]


開始
S106
V = 0
S201
i = m-1
S202
j = 1
S203
S206
S204
V = 2V
j = j+1
S205
いいえ
j = k
はい
S207
はい
いいえ
s[i] = +s
S208
S209
V = V+tab[w[i]]
V = V-tab[-w[i]]
i = i-1
S210
S211
はい
i ≧ 0
いいえ
V を返す
S212
終了

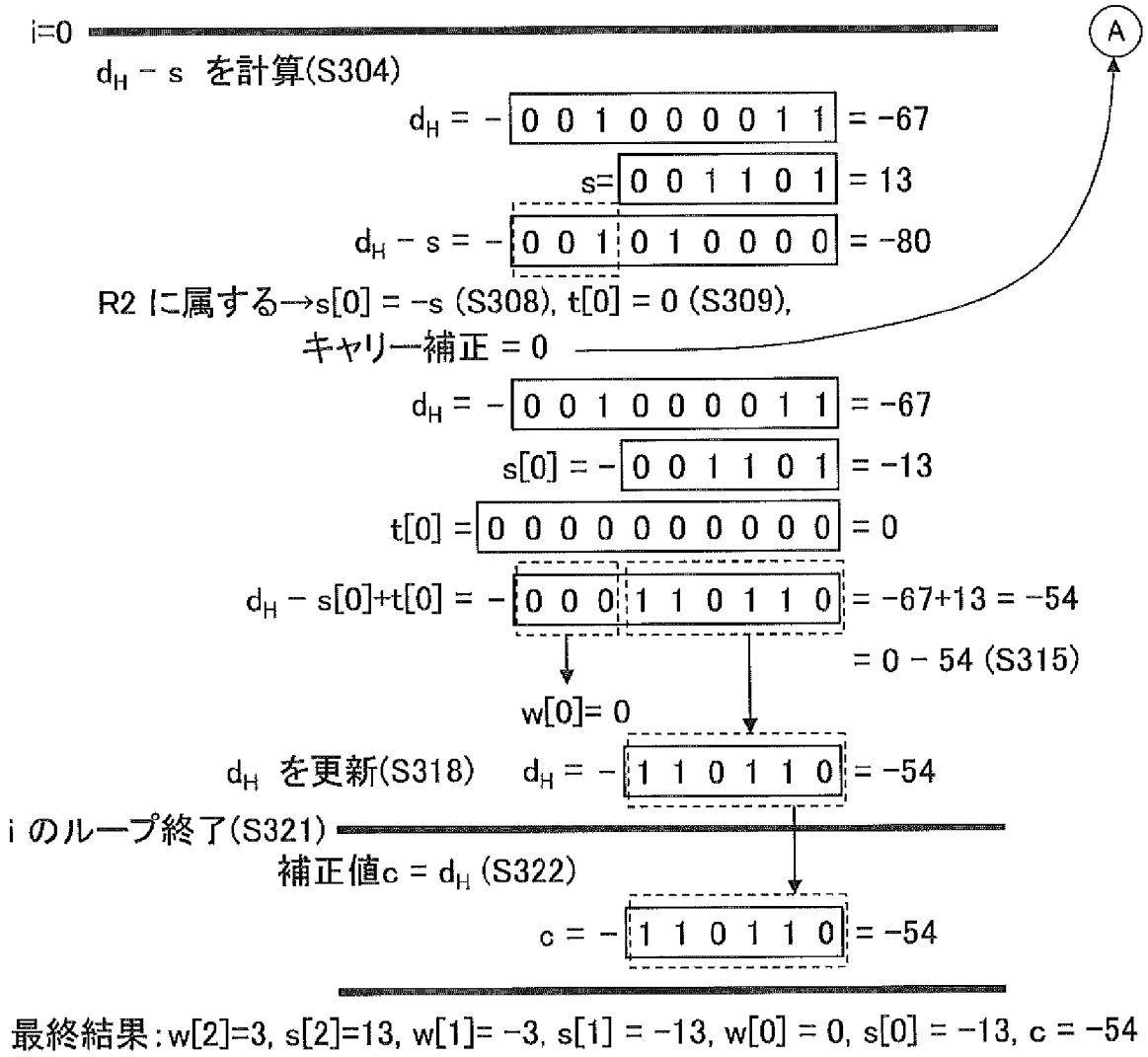
[図11]


開始
S103
S301
$d_H = e[b+km-1]||\cdots||e[k(m-1)]$
S302
b ビットの乱数 s を生成する (0≦s)
S303
i = m − 1
S304
d_H−sが属する範囲
R1
R2
R3
R4
S305
s[i] = +s
S308
s[i] = −s
S310
s[i] = +s
S312
s[i] = −s
S306
$t[i] = +2^{k+b}$
S309
t[i] = 0
S311
t[i] = 0
S313
$t[i] = -2^{k+b}$
S307
w[i+1] = w[i+1]−1
S314
w[i+1] = w[i+1]+1
S315
diff = d_H−s[i]+t[i]
S316
w[i] = diff[k+b−1]||…||diff[b]
S317
いいえ
i =0
はい
S318
$d_H = (diff[b-1]||\cdots||diff[0])2^k+(e[ki-1]||\cdots||e[k(i-1)])$
S319
d_H = diff[b−1]||…||diff[0]
S320
i = i−1
S321
i≧0
はい
いいえ
S322
c = d_H
S323
ウィンドウ列 w[i]と乱数列 s[i]と補正値cを出力する
終了

[図12A]

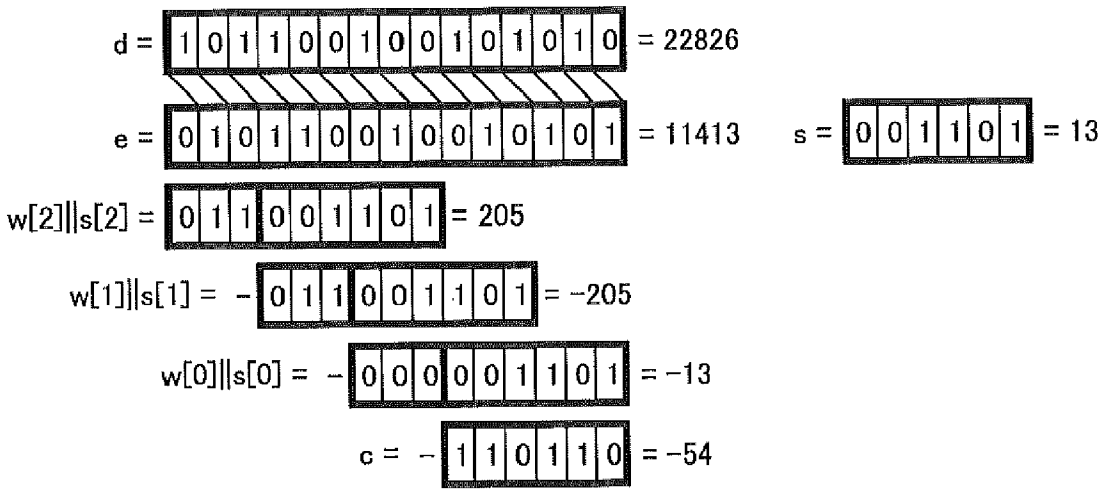

1 1 1 1 1
4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
e= 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 = 11413
dH 初期値設定(S301)
乱数生成 (S302) s = 0 0 1 1 0 1 = 13
i=2
dH − s を計算(S304)
dH = 0 1 0 1 1 0 0 1 0 = 178
s= 0 0 1 1 0 1 = 13
dH − s = 0 1 0 1 0 0 1 0 1 = 165
R3 に属する→s[2] = +s (S310), t[2] = 0 (S311), キャリー補正 = 0
dH = 0 1 0 1 1 0 0 1 0 = 178
s[2] = 0 0 1 1 0 1 = 13
t[2] = 0 0 0 0 0 0 0 0 0 0 = 0
dH − s[2]+t[2] = 0 1 0 1 0 0 1 0 1 = 178−13=165=128+37 (S315)
w[2] = 2 + (キャリー補正) = 3
dH を更新(S318) dH = 1 0 0 1 0 1 0 1 0 = 37×2^3 + 2=298
i=1
dH − s を計算(S304)
dH = 1 0 0 1 0 1 0 1 0 = 298
s= 0 0 1 1 0 1 = 13
dH − s = 1 0 0 0 1 1 1 0 1 = 285
R4 に属する→s[1] = −s (S312),
t[1]= −2^9 (S313), キャリー補正= +1 (S314)
dH = 1 0 0 1 0 1 0 1 0 = 298
s[1] = − 0 0 1 1 0 1 = −13
t[1] = − 1 0 0 0 0 0 0 0 0 0 = −512
dH − s[1]+t[1] = − 0 1 1 0 0 1 0 0 1 = 298 + 13 − 512 = −201
= − 192 − 9 (S315)
w[1]= −3 + (キャリー補正)= −3
A
dH を更新(S318) dH = − 0 0 1 0 0 0 0 1 1 = −9×2^3+5 = −67

[図12B]



最終結果：w[2]=3, s[2]=13, w[1]= −3, s[1] = −13, w[0] = 0, s[0] = −13, c = −54

[図13]



104

インデックス	値
-2	(-2×64+s)A = -115A
-1	(-1×64+s)A = -51A
0	(0×64+s)A = 13A
1	(1×64+s)A = 77A
2	(2×64+s)A = 141A
3	(3×64+s)A = 205A
4	(4×64+s)A = 269A
5	(5×64+s)A = 333A

[図14]

107

インデックス	値
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$
5	$(5\times 2^{b}+s)A$
6	$(6\times 2^{b}+s)A$
7	$(7\times 2^{b}+s)A$

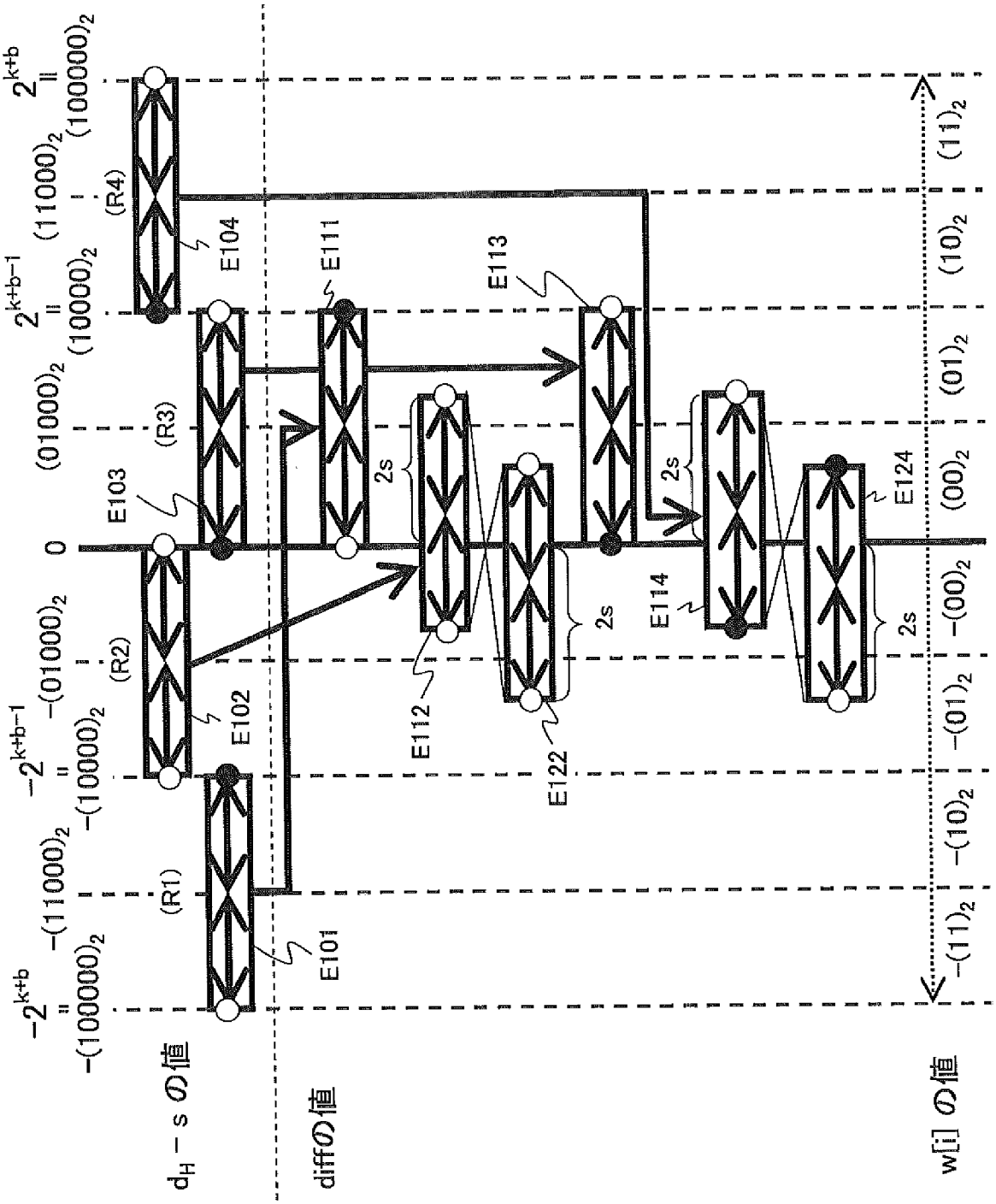
105

インデックス	値
−2	$(-2\times 2^{b}+s)A$
−1	$(-1\times 2^{b}+s)A$
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$
5	$(5\times 2^{b}+s)A$

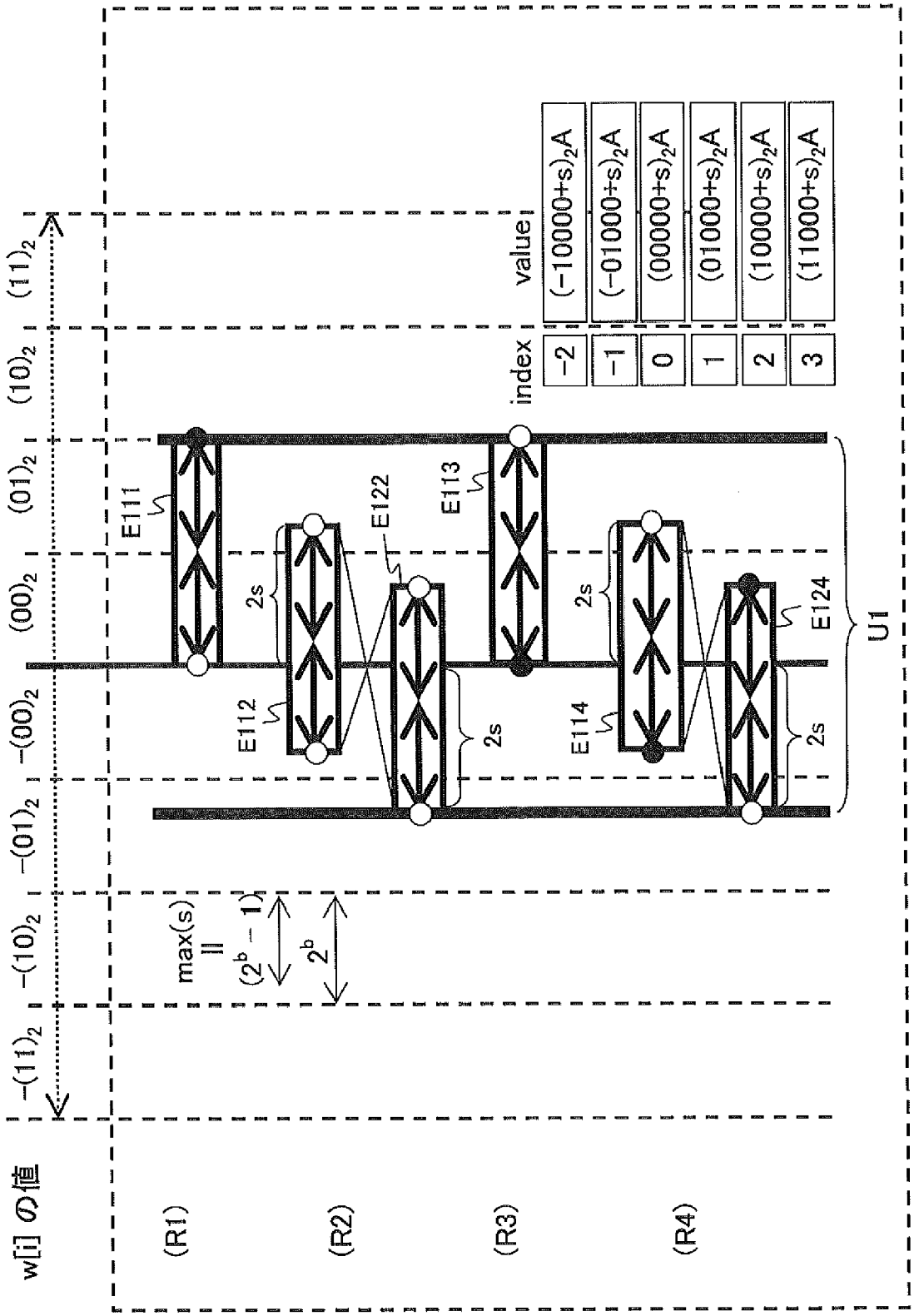
106

インデックス	値
0	0A
1	1A
2	2A
3	3A
4	4A

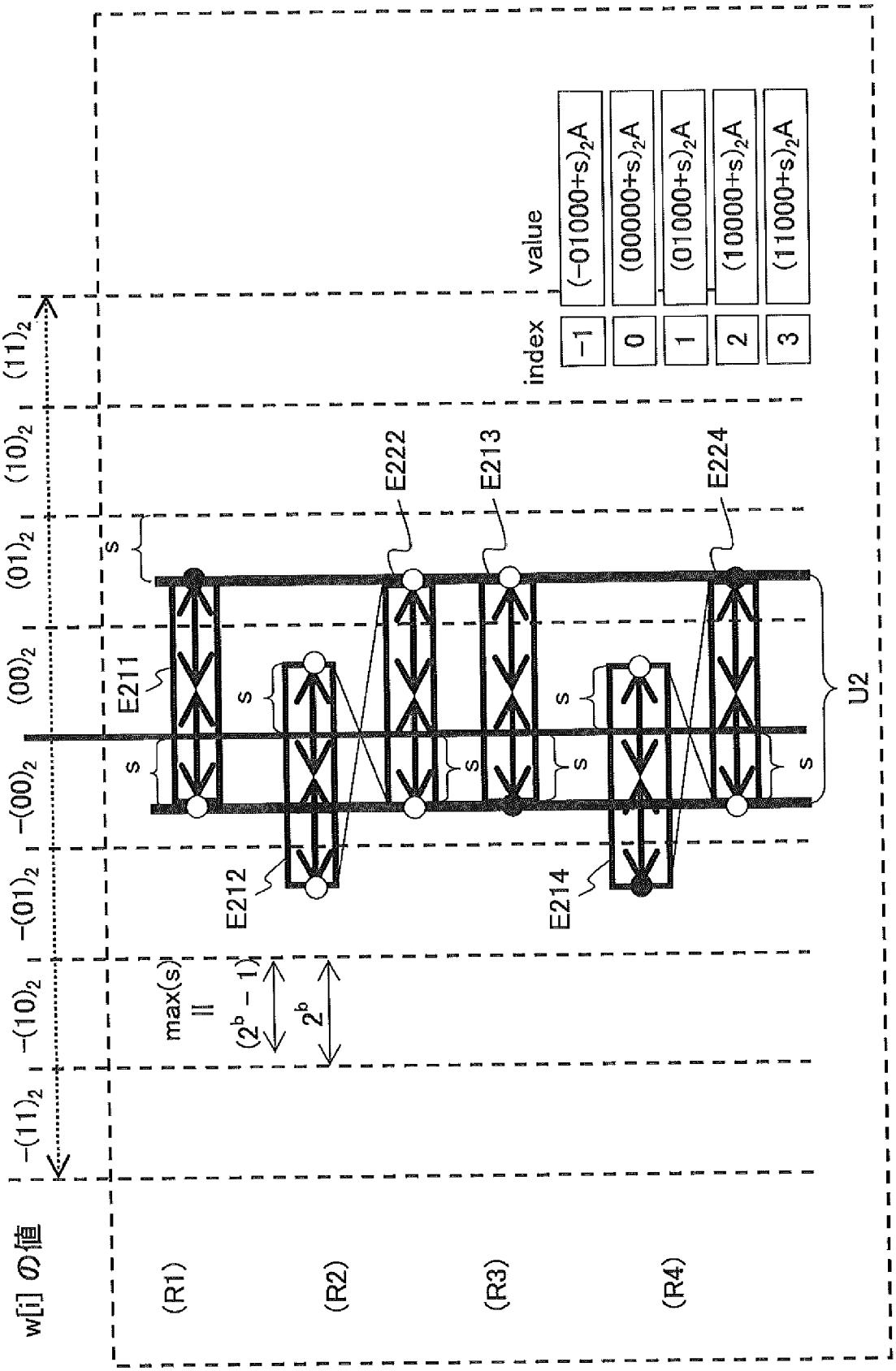
[図15]


2^{k+b} = $(100000)_2$
2^{k+b-1} = $(10000)_2$
$(11000)_2$
$(01000)_2$
0
$-(01000)_2$
-2^{k+b-1} = $-(10000)_2$
$-(11000)_2$
-2^{k+b} = $-(100000)_2$
(R4)
(R3)
(R2)
(R1)
E101
E102
E103
E104
E111
E112
E113
E114
E122
E124
2s
$d_H - s$ の値
diffの値
w[i] の値
$(11)_2$
$(10)_2$
$(01)_2$
$(00)_2$
$-(00)_2$
$-(01)_2$
$-(10)_2$
$-(11)_2$

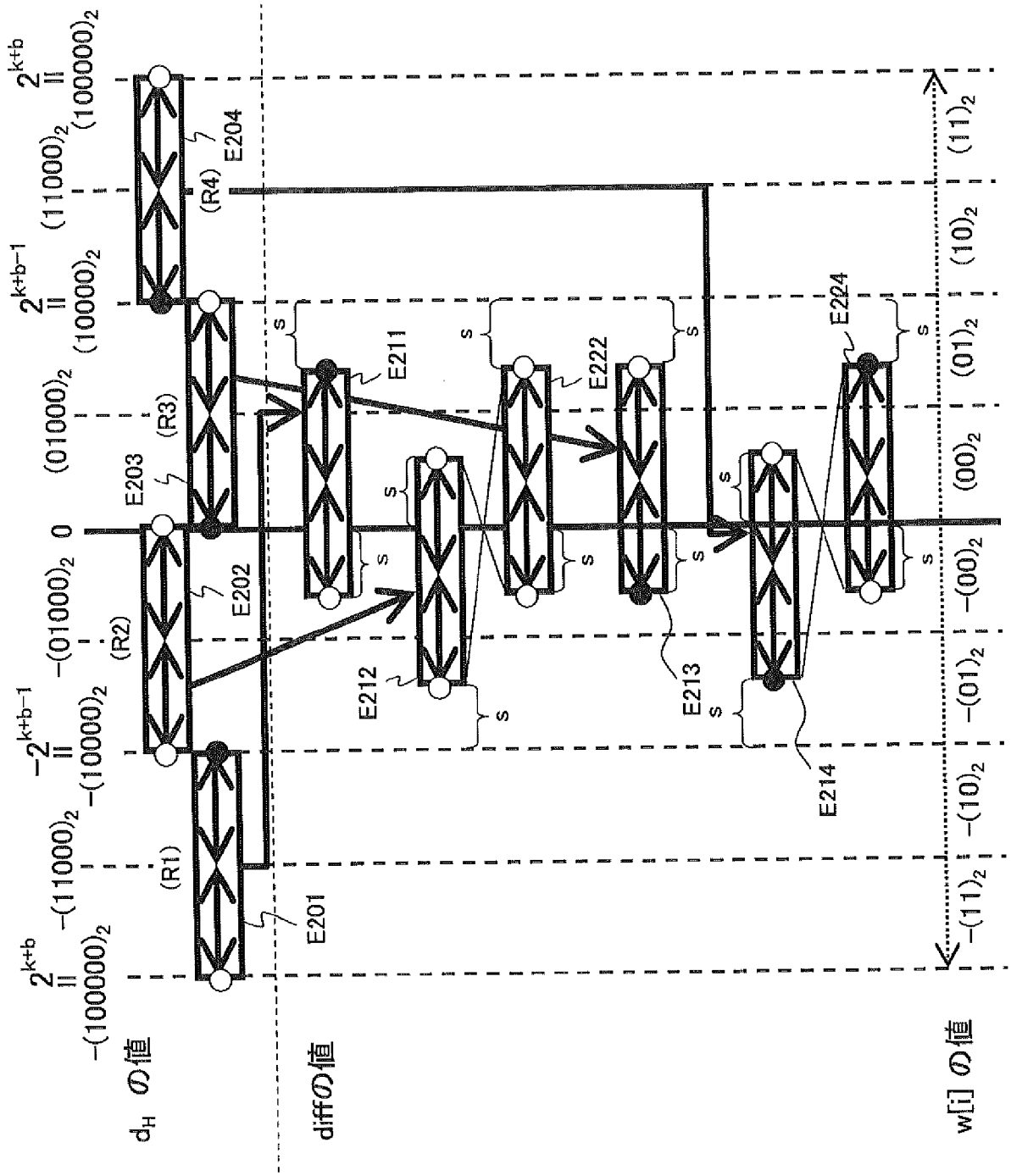
[図16]


w[i] の値
(11)2
(10)2
(01)2
(00)2
−(00)2
−(01)2
−(10)2
−(11)2
max(s)
(2b − 1)
2b
(R1)
(R2)
(R3)
(R4)
index
value
−2
−1
0
1
2
3
(−10000+s)2A
(−01000+s)2A
(00000+s)2A
(01000+s)2A
(10000+s)2A
(11000+s)2A
E111
E112
E113
E114
E122
E124
2s
U1

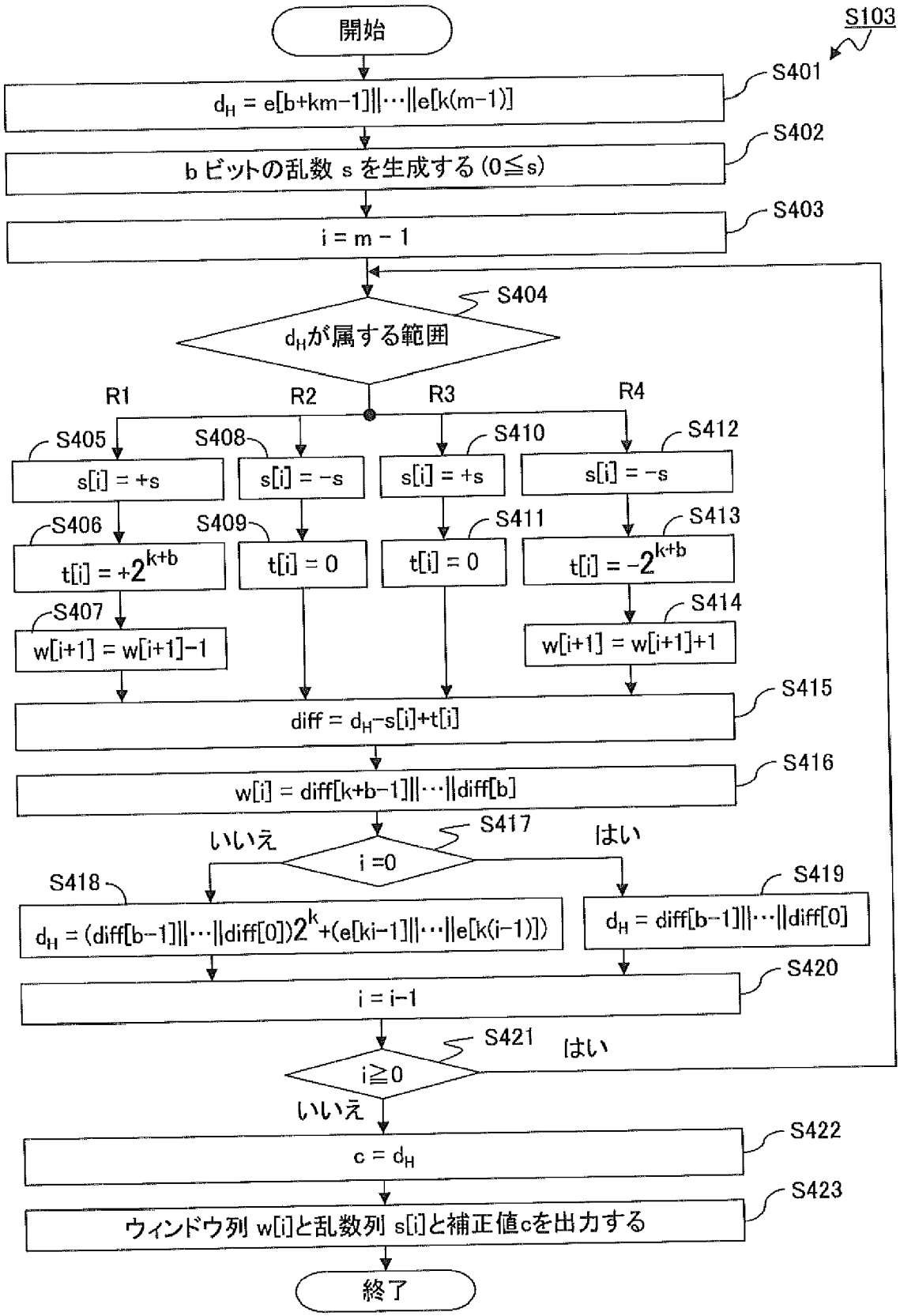
[図17]


w[i] の値
(R1)
(R2)
(R3)
(R4)
$(11)_2$
$(10)_2$
$(01)_2$
$(00)_2$
$-(00)_2$
$-(01)_2$
$-(10)_2$
$-(11)_2$
max(s)
=
$(2^b - 1)$
2^b
s
E211
E212
E213
E214
E222
E224
U2
index
value
-1
0
1
2
3
$(-01000+s)_2A$
$(00000+s)_2A$
$(01000+s)_2A$
$(10000+s)_2A$
$(11000+s)_2A$

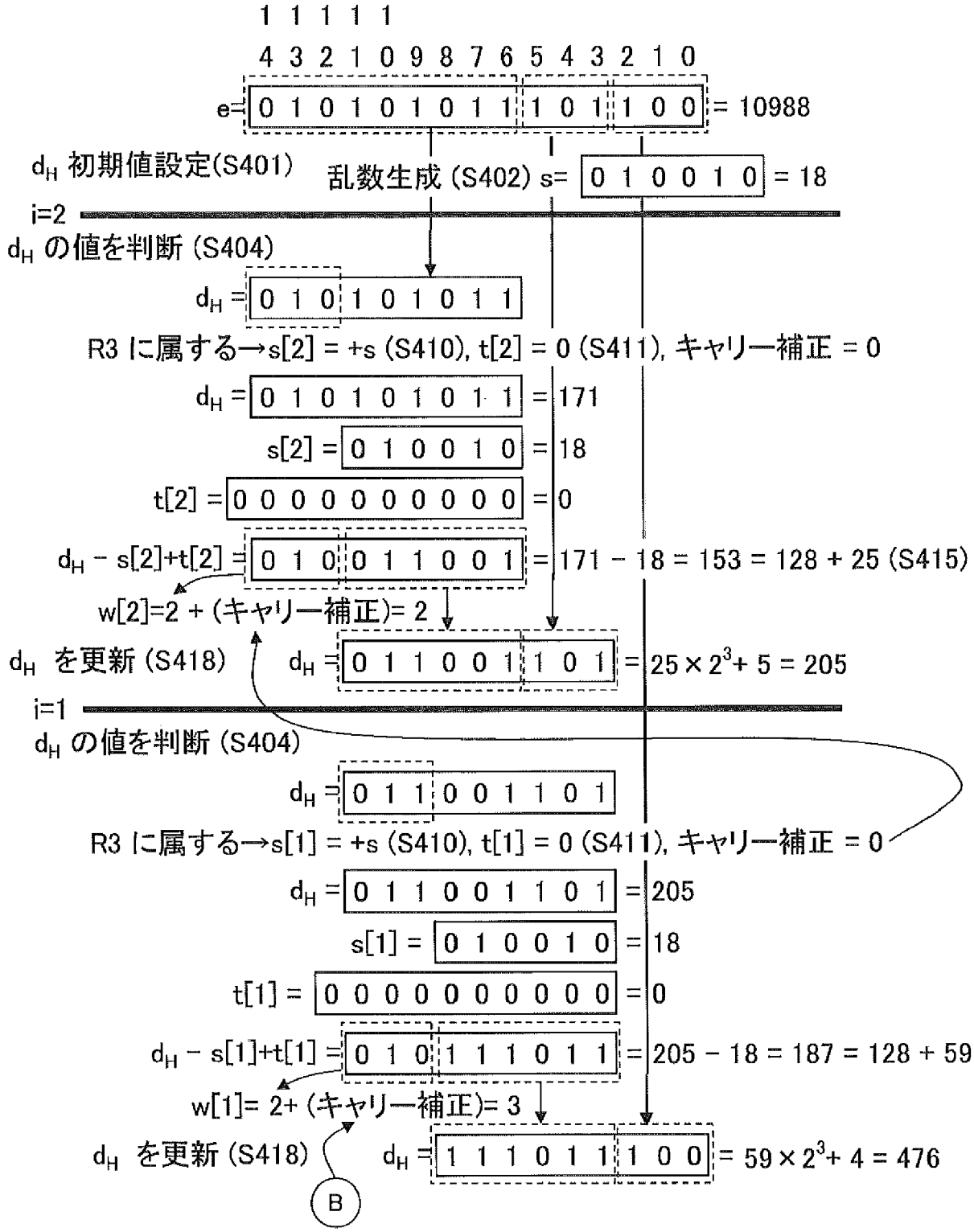
[図18]


d_H の値
diffの値
w[i] の値
2^{k+b} = $(100000)_2$
2^{k+b-1} = $(10000)_2$
-2^{k+b-1} = $-(10000)_2$
-2^{k+b} = $-(100000)_2$
$(11000)_2$
$(01000)_2$
0
$-(01000)_2$
$-(11000)_2$
$(11)_2$
$(10)_2$
$(01)_2$
$(00)_2$
$-(00)_2$
$-(01)_2$
$-(10)_2$
$-(11)_2$
(R1)
(R2)
(R3)
(R4)
E201
E202
E203
E204
E211
E212
E213
E214
E222
E224
S

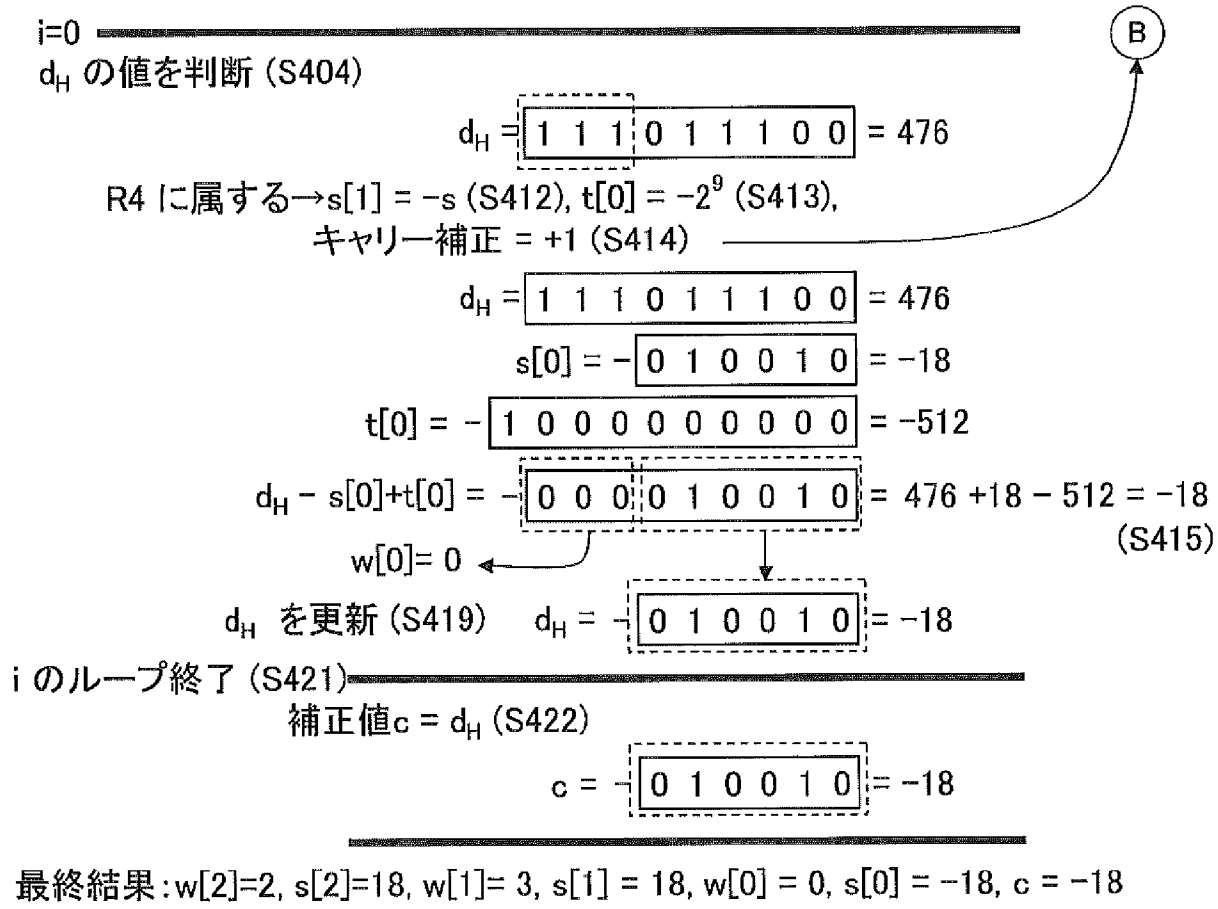
[図19]


S103
開始
S401
$d_H = e[b+km-1]||\cdots||e[k(m-1)]$
S402
b ビットの乱数 s を生成する (0≦s)
S403
i = m - 1
S404
d_Hが属する範囲
R1
R2
R3
R4
S405
s[i] = +s
S408
s[i] = -s
S410
s[i] = +s
S412
s[i] = -s
S406
$t[i] = +2^{k+b}$
S409
t[i] = 0
S411
t[i] = 0
S413
$t[i] = -2^{k+b}$
S407
w[i+1] = w[i+1]-1
S414
w[i+1] = w[i+1]+1
S415
$diff = d_H - s[i] + t[i]$
S416
w[i] = diff[k+b-1]||…||diff[b]
S417
i =0
いいえ
はい
S418
$d_H = (diff[b-1]||\cdots||diff[0])2^k + (e[ki-1]||\cdots||e[k(i-1)])$
S419
d_H = diff[b-1]||…||diff[0]
S420
i = i-1
S421
i≧0
はい
いいえ
S422
$c = d_H$
S423
ウィンドウ列 w[i]と乱数列 s[i]と補正値cを出力する
終了

[図20A]

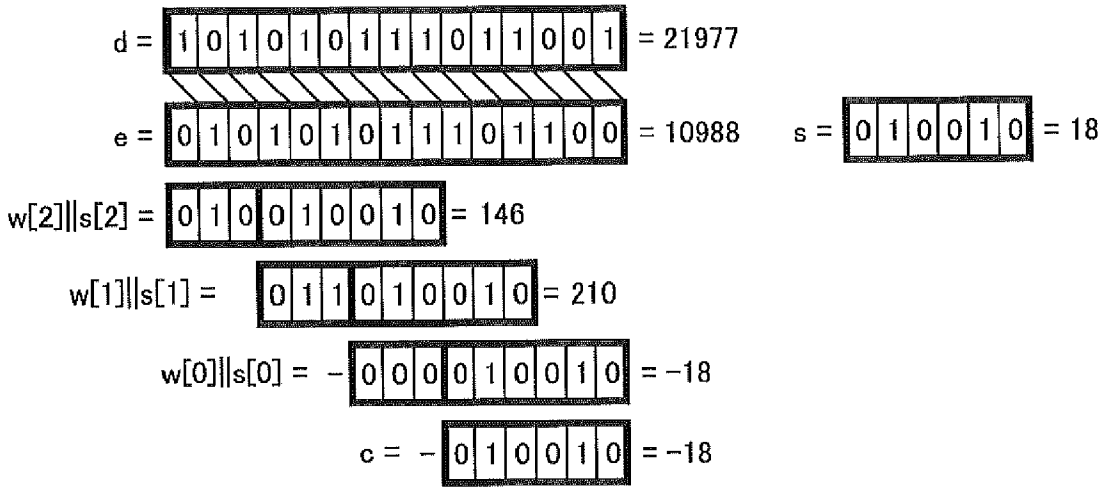

1 1 1 1 1
4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
e= 0 1 0 1 0 1 0 1 1 1 0 1 1 0 0 = 10988
dH 初期値設定(S401)
乱数生成 (S402) s= 0 1 0 0 1 0 = 18
i=2
dH の値を判断 (S404)
dH = 0 1 0 1 0 1 0 1 1
R3 に属する→s[2] = +s (S410), t[2] = 0 (S411), キャリー補正 = 0
dH = 0 1 0 1 0 1 0 1 1 = 171
s[2] = 0 1 0 0 1 0 = 18
t[2] = 0 0 0 0 0 0 0 0 0 0 = 0
dH − s[2]+t[2] = 0 1 0 0 1 1 0 0 1 = 171 − 18 = 153 = 128 + 25 (S415)
w[2]=2 + (キャリー補正)= 2
dH を更新 (S418)
dH = 0 1 1 0 0 1 1 0 1 = 25 × 2^3 + 5 = 205
i=1
dH の値を判断 (S404)
dH = 0 1 1 0 0 1 1 0 1
R3 に属する→s[1] = +s (S410), t[1] = 0 (S411), キャリー補正 = 0
dH = 0 1 1 0 0 1 1 0 1 = 205
s[1] = 0 1 0 0 1 0 = 18
t[1] = 0 0 0 0 0 0 0 0 0 0 = 0
dH − s[1]+t[1] = 0 1 0 1 1 1 0 1 1 = 205 − 18 = 187 = 128 + 59
w[1]= 2+ (キャリー補正)= 3
dH を更新 (S418)
dH = 1 1 1 0 1 1 1 0 0 = 59 × 2^3 + 4 = 476
B

[図20B]



最終結果:w[2]=2, s[2]=18, w[1]= 3, s[1] = 18, w[0] = 0, s[0] = −18, c = −18

[図21]



108

インデックス	値
−1	(−1 × 64+s)A = −46A
0	(0 × 64+s)A = 18A
1	(1 × 64+s)A = 82A
2	(2 × 64+s)A = 146A
3	(3 × 64+s)A = 210A
4	(4 × 64+s)A = 274A
5	(5 × 64+s)A = 338A

[図22]

107

インデックス	値
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$
5	$(5\times 2^{b}+s)A$
6	$(6\times 2^{b}+s)A$
7	$(7\times 2^{b}+s)A$

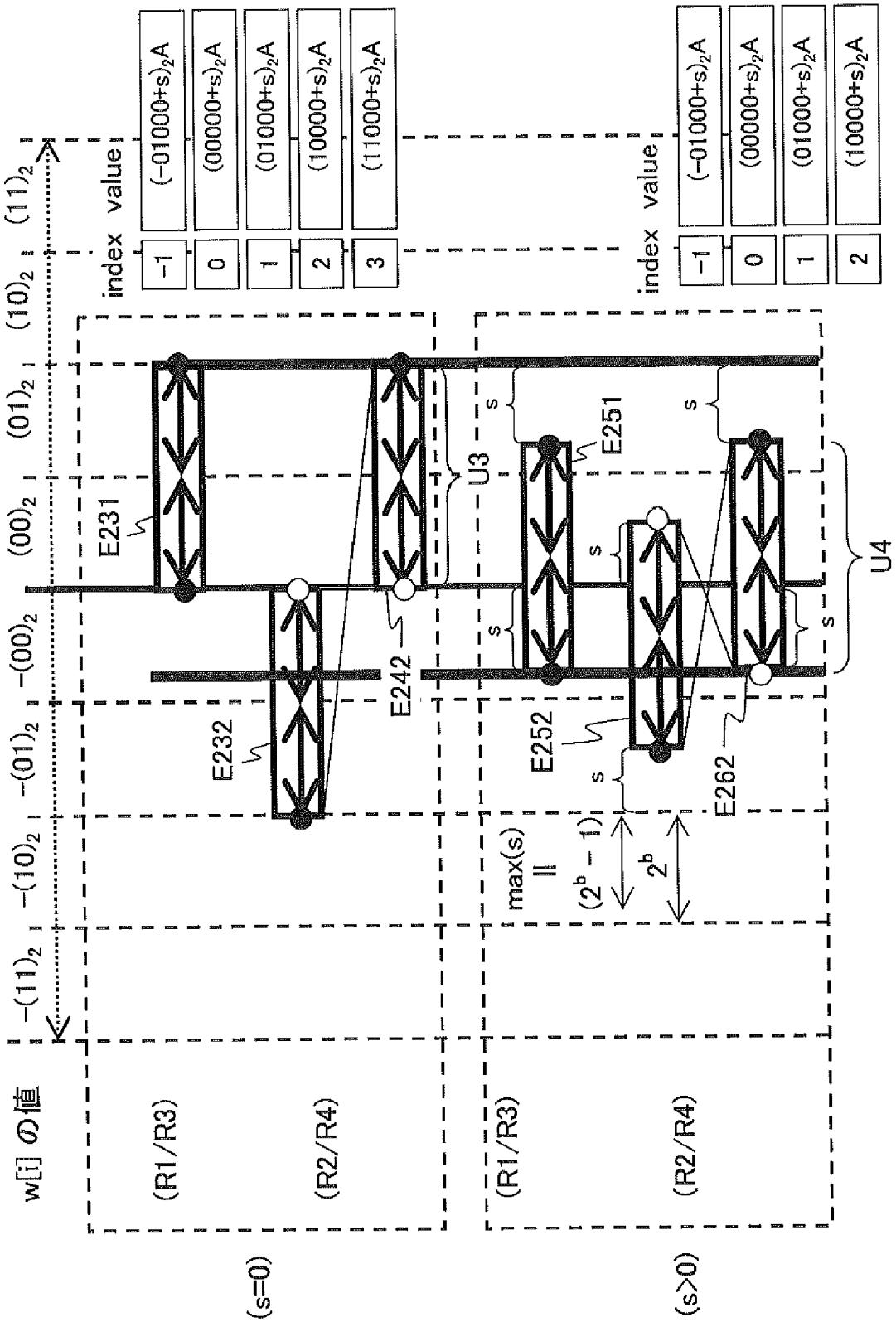
109

インデックス	値
−1	$(-1\times 2^{b}+s)A$
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$
5	$(5\times 2^{b}+s)A$

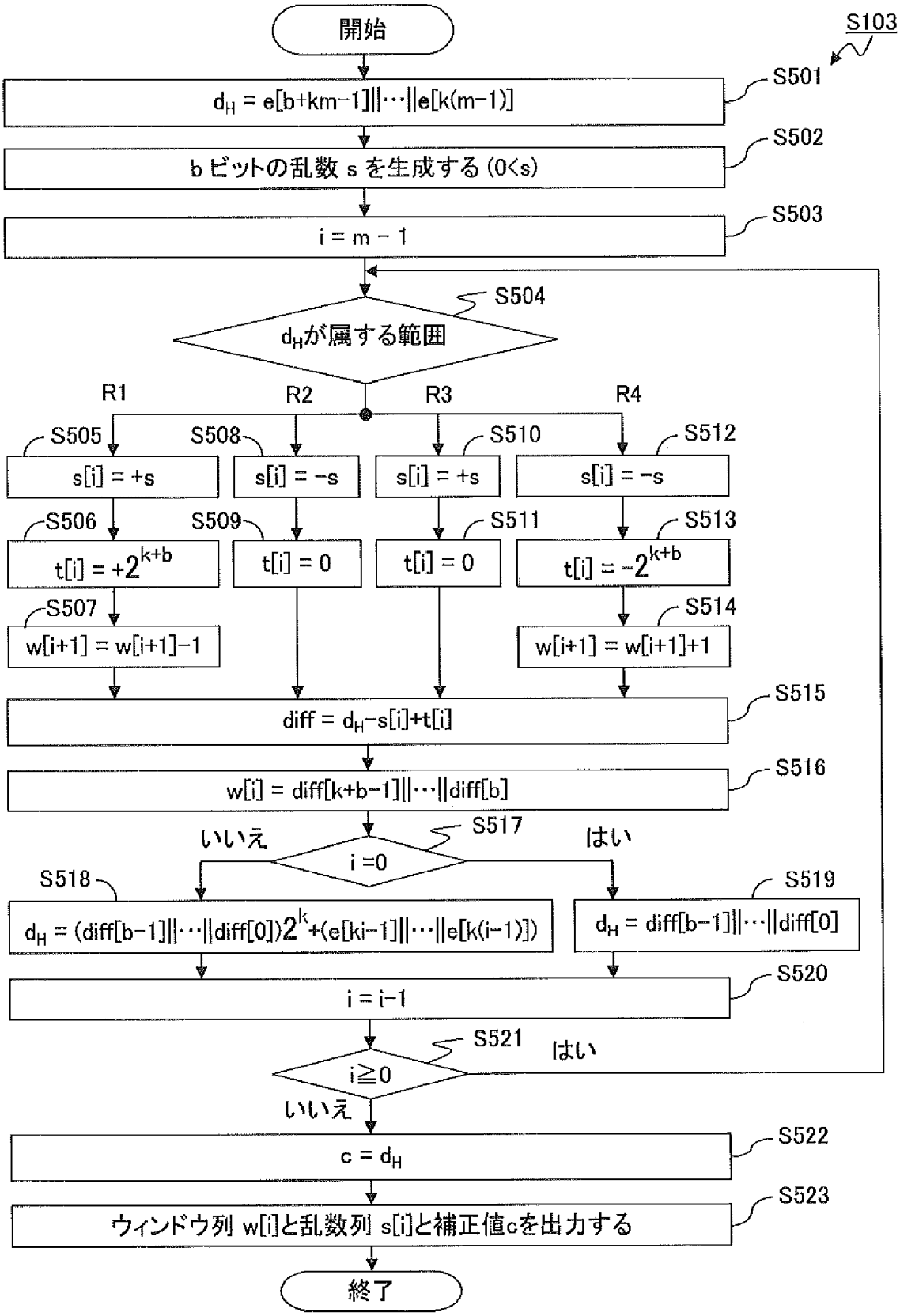
106

インデックス	値
0	0A
1	1A
2	2A
3	3A
4	4A

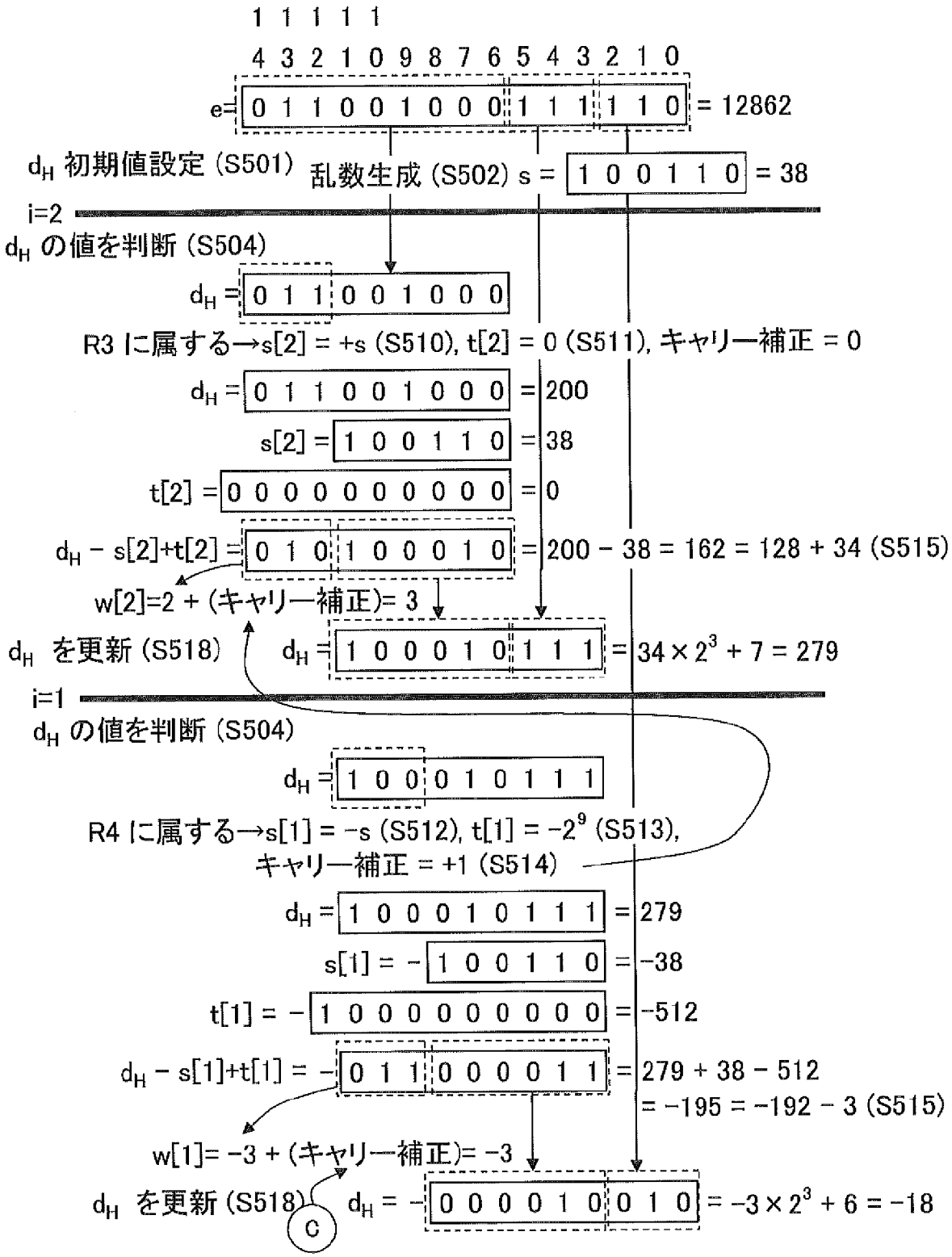
[図23]


w[i] の値
(R1/R3)
(R2/R4)
(s=0)
(s>0)
index
value
(−01000+s)2A
(00000+s)2A
(01000+s)2A
(10000+s)2A
(11000+s)2A
E231
E232
E242
E251
E252
E262
U3
U4
s
max(s)
(2^b − 1)
2^b

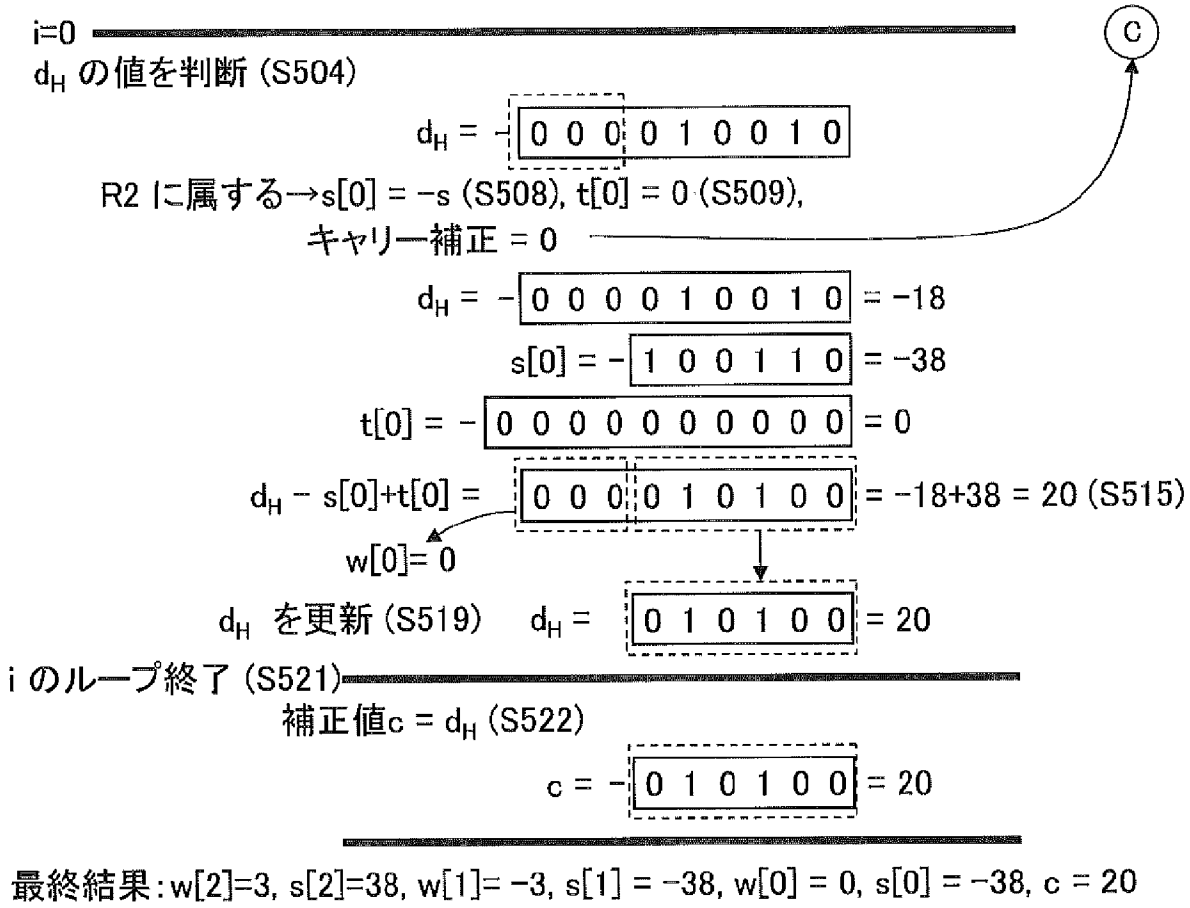
[図24]


開始
S103
S501
$d_H = e[b+km-1]||\cdots||e[k(m-1)]$
S502
b ビットの乱数 s を生成する (0<s)
S503
i = m - 1
S504
d_Hが属する範囲
R1
R2
R3
R4
S505
s[i] = +s
S508
s[i] = -s
S510
s[i] = +s
S512
s[i] = -s
S506
$t[i] = +2^{k+b}$
S509
t[i] = 0
S511
t[i] = 0
S513
$t[i] = -2^{k+b}$
S507
w[i+1] = w[i+1]-1
S514
w[i+1] = w[i+1]+1
S515
$diff = d_H - s[i]+t[i]$
S516
w[i] = diff[k+b-1]||…||diff[b]
S517
いいえ
i =0
はい
S518
$d_H = (diff[b-1]||\cdots||diff[0])2^k+(e[ki-1]||\cdots||e[k(i-1)])$
S519
$d_H = diff[b-1]||\cdots||diff[0]$
S520
i = i-1
S521
はい
i≧0
いいえ
S522
$c = d_H$
S523
ウィンドウ列 w[i]と乱数列 s[i]と補正値cを出力する
終了

[図25A]

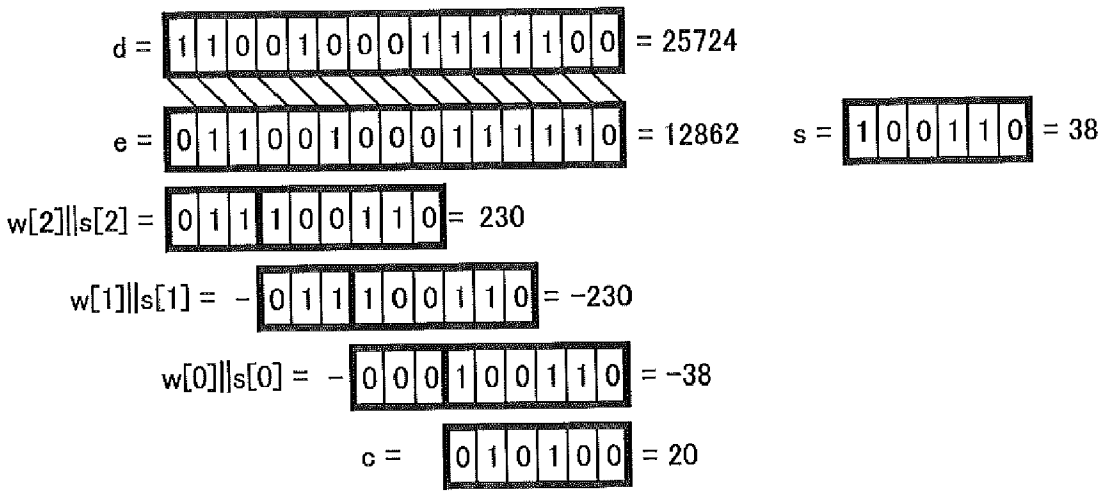

1 1 1 1 1
4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
e= 0 1 1 0 0 1 0 0 0 1 1 1 1 1 0 = 12862
d_H 初期値設定 (S501)
乱数生成 (S502) s = 1 0 0 1 1 0 = 38
i=2
d_H の値を判断 (S504)
d_H = 0 1 1 0 0 1 0 0 0
R3 に属する→s[2] = +s (S510), t[2] = 0 (S511), キャリー補正 = 0
d_H = 0 1 1 0 0 1 0 0 0 = 200
s[2] = 1 0 0 1 1 0 = 38
t[2] = 0 0 0 0 0 0 0 0 0 0 = 0
d_H − s[2]+t[2] = 0 1 0 1 0 0 0 1 0 = 200 − 38 = 162 = 128 + 34 (S515)
w[2]=2 + (キャリー補正)= 3
d_H を更新 (S518)
d_H = 1 0 0 0 1 0 1 1 1 = $34 \times 2^3 + 7 = 279$
i=1
d_H の値を判断 (S504)
d_H = 1 0 0 0 1 0 1 1 1
R4 に属する→s[1] = −s (S512), t[1] = -2^9 (S513),
キャリー補正 = +1 (S514)
d_H = 1 0 0 0 1 0 1 1 1 = 279
s[1] = − 1 0 0 1 1 0 = −38
t[1] = − 1 0 0 0 0 0 0 0 0 0 = −512
d_H − s[1]+t[1] = − 0 1 1 0 0 0 0 1 1 = 279 + 38 − 512
= −195 = −192 − 3 (S515)
w[1]= −3 + (キャリー補正)= −3
d_H を更新 (S518)
C
d_H = − 0 0 0 0 1 0 0 1 0 = $-3 \times 2^3 + 6 = -18$

[図25B]



最終結果:w[2]=3, s[2]=38, w[1]= −3, s[1] = −38, w[0] = 0, s[0] = −38, c = 20

[図26]



110

インデックス	値
-1	(-1 × 64+s)A = -26A
0	(0 × 64+s)A = 38A
1	(1 × 64+s)A = 102A
2	(2 × 64+s)A = 166A
3	(3 × 64+s)A = 230A
4	(4 × 64+s)A = 294A

[図27]

107

インデックス	値
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$
5	$(5\times 2^{b}+s)A$
6	$(6\times 2^{b}+s)A$
7	$(7\times 2^{b}+s)A$

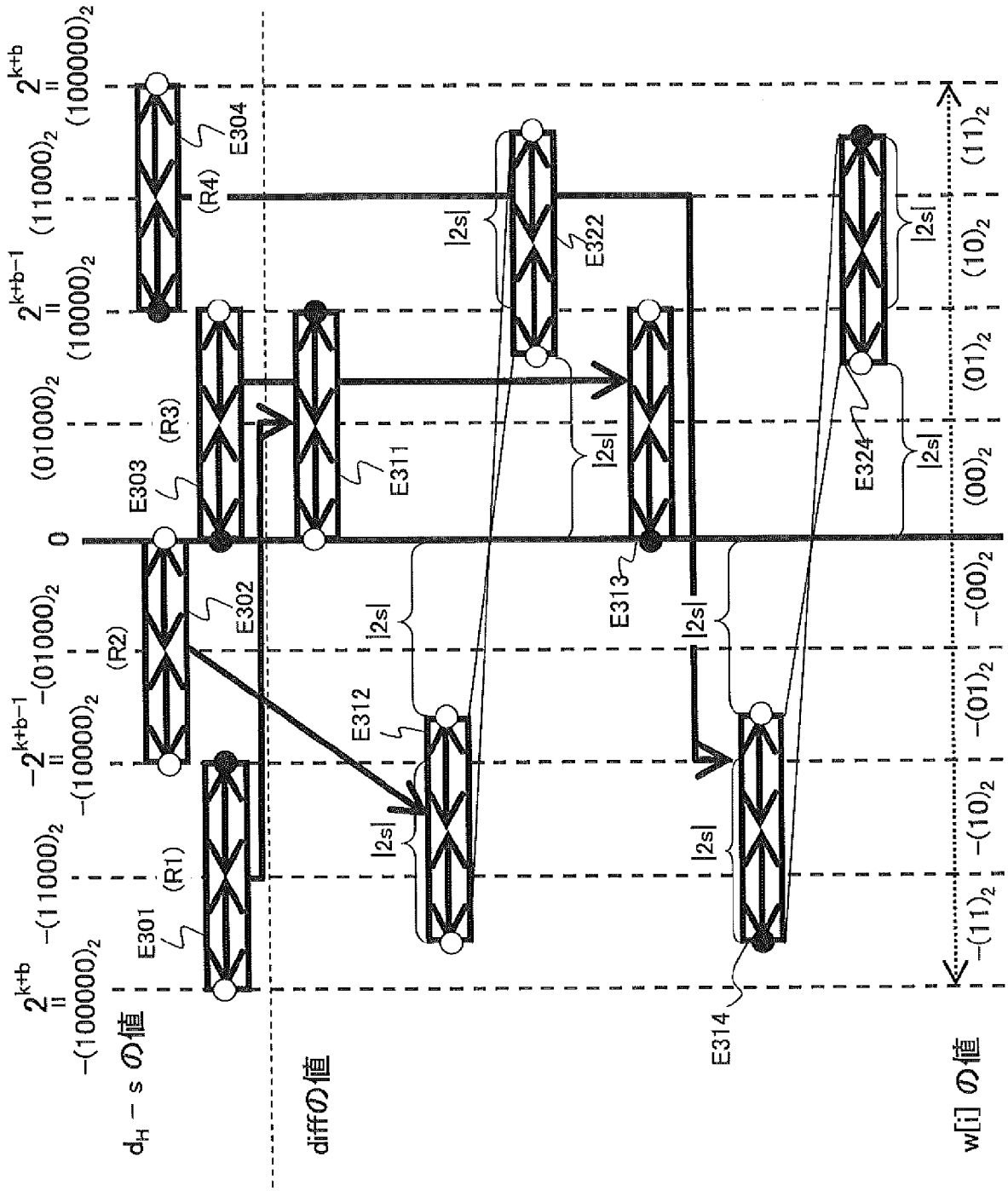
111

インデックス	値
-1	$(-1\times 2^{b}+s)A$
0	$(0\times 2^{b}+s)A$
1	$(1\times 2^{b}+s)A$
2	$(2\times 2^{b}+s)A$
3	$(3\times 2^{b}+s)A$
4	$(4\times 2^{b}+s)A$

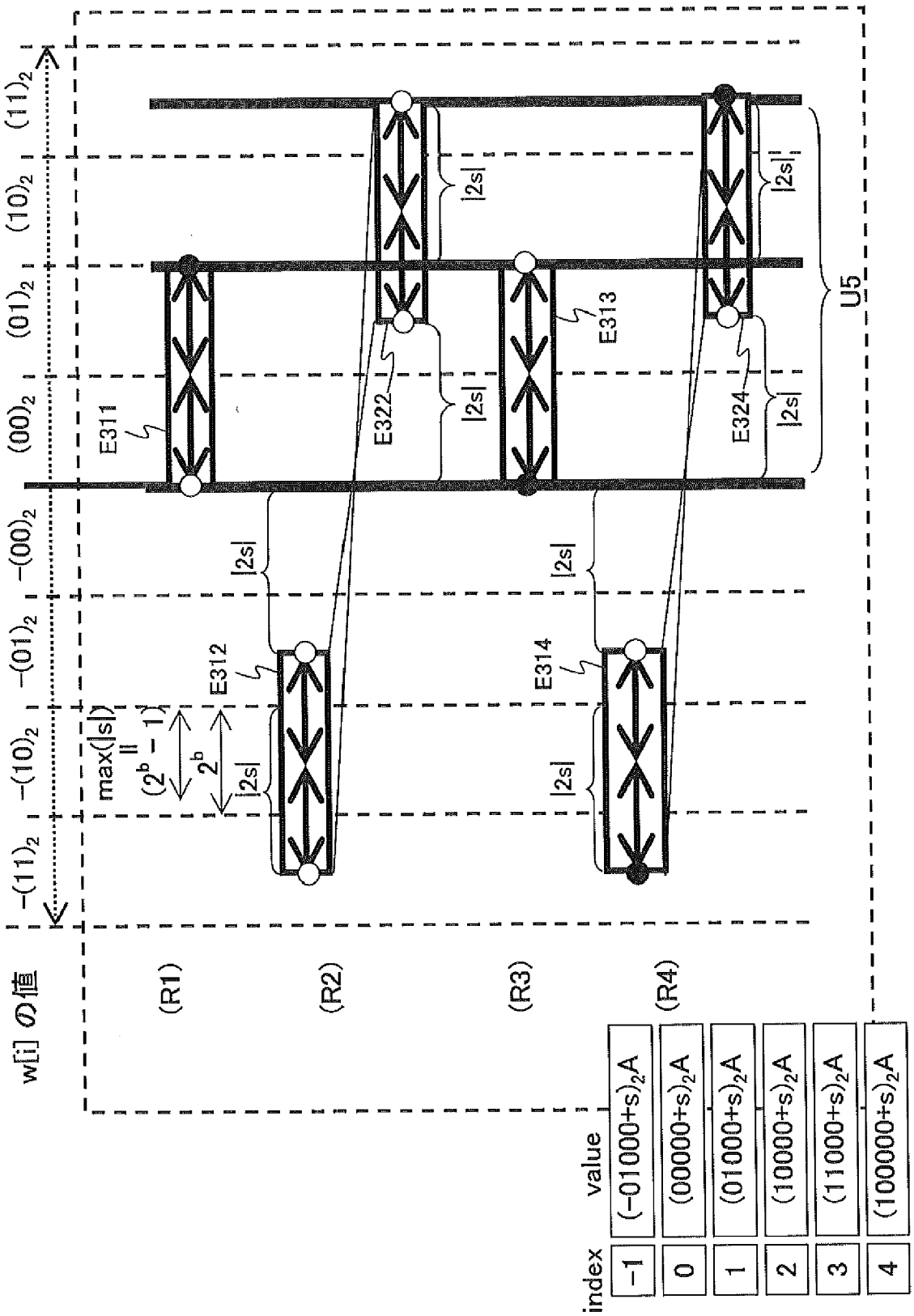
106

インデックス	値
0	0A
1	1A
2	2A
3	3A
4	4A

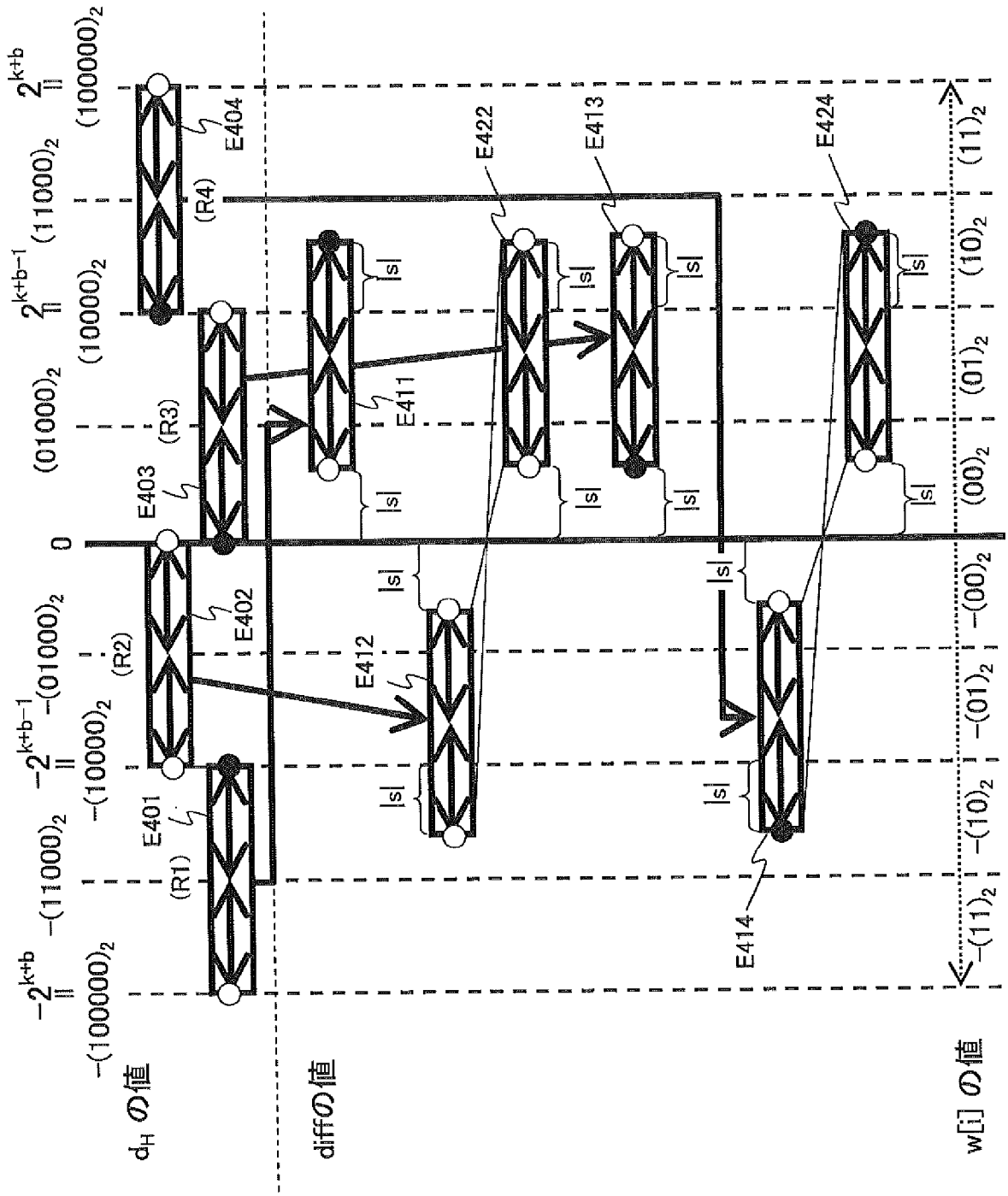
[図28]


2^{k+b} $(100000)_2$
2^{k+b-1} $(11000)_2$
$(10000)_2$
$(01000)_2$
0
$-(01000)_2$
-2^{k+b-1} $-(10000)_2$
$-(11000)_2$
-2^{k+b} $-(100000)_2$
$d_H - s$ の値
diffの値
w[i] の値
E301
E302
E303
E304
E311
E312
E313
E314
E322
E324
(R1)
(R2)
(R3)
(R4)
|2s|
$(11)_2$
$(10)_2$
$(01)_2$
$(00)_2$
$-(00)_2$
$-(01)_2$
$-(10)_2$
$-(11)_2$

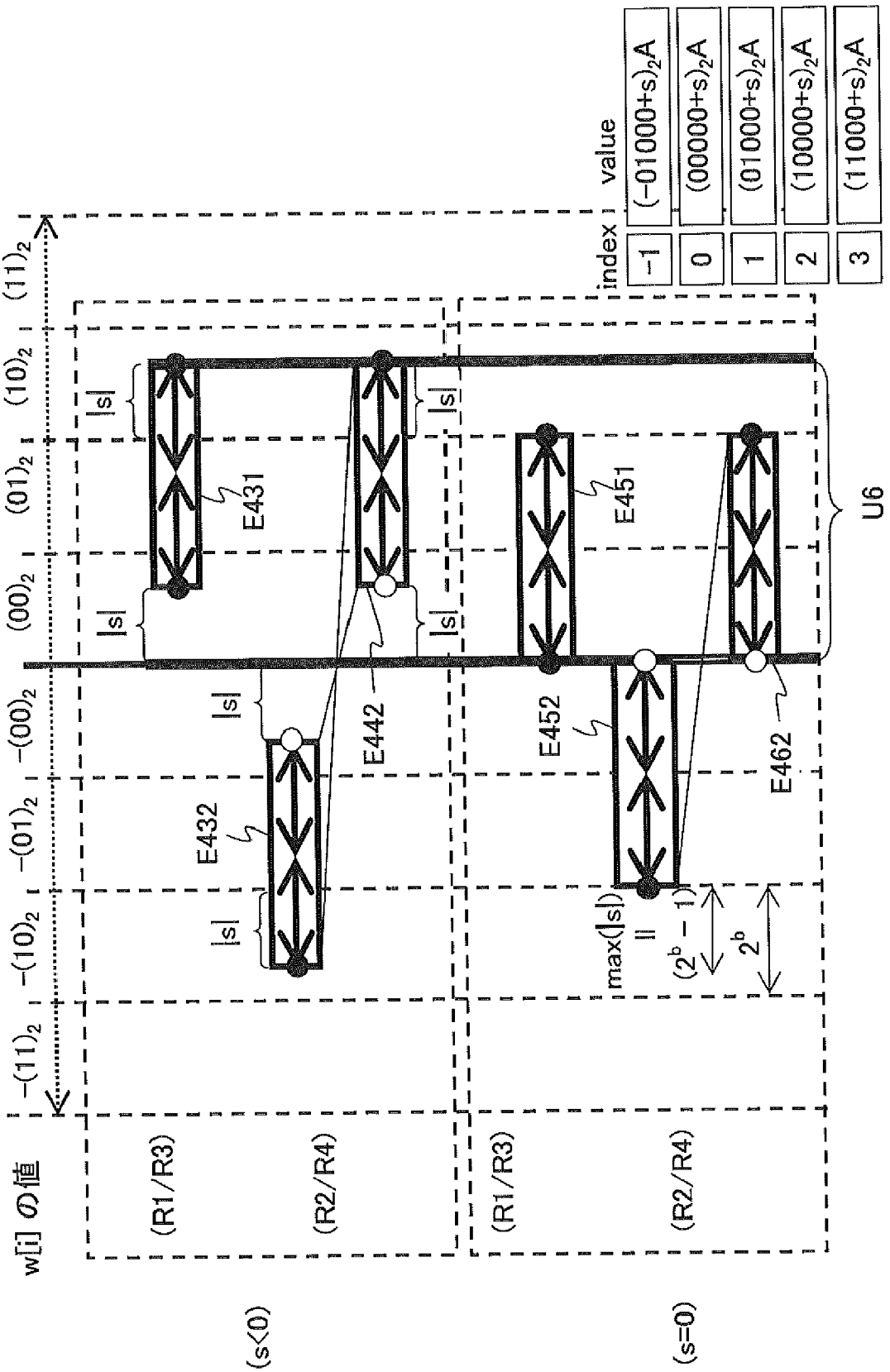
[図29]


w[i] の値
(11)2
(10)2
(01)2
(00)2
−(00)2
−(01)2
−(10)2
−(11)2
max(|s|)
(2b − 1)
2b
|2s|
E311
E312
E313
E314
E322
E324
U5
(R1)
(R2)
(R3)
(R4)
index
value
−1
(−01000+s)2A
0
(00000+s)2A
1
(01000+s)2A
2
(10000+s)2A
3
(11000+s)2A
4
(100000+s)2A

[図30]


d_H の値
diffの値
w[i] の値
-2^{k+b}
$-(100000)_2$
-2^{k+b-1}
$-(11000)_2$
$-(10000)_2$
$-(01000)_2$
0
$(01000)_2$
$(10000)_2$
$(11000)_2$
2^{k+b-1}
2^{k+b}
$(100000)_2$
$-(11)_2$
$-(10)_2$
$-(01)_2$
$-(00)_2$
$(00)_2$
$(01)_2$
$(10)_2$
$(11)_2$
E401
E402
E403
E404
E411
E412
E413
E414
E422
E424
(R1)
(R2)
(R3)
(R4)
|s|

[図31]


index
value
-1
(-01000+s)2A
0
(00000+s)2A
1
(01000+s)2A
2
(10000+s)2A
3
(11000+s)2A
w[i] の値
(11)2
(10)2
(01)2
(00)2
-(00)2
-(01)2
-(10)2
-(11)2
|s|
E431
E432
E442
E451
E452
E462
U6
max(|s|)
=
(2b − 1)
2b
(R1/R3)
(R2/R4)
(R1/R3)
(R2/R4)
(s<0)
(s=0)

[図32]

		PA攻撃に対して	テーブルのエントリ数				インデックスの範囲
			k=2	k=3	k=4	任意のk	
第1実施形態	$s \geqq 0$	安全	6	8	12	$2^{k-1}+4$	$-2 \leqq h \leqq 2^{k-1}+1$
	$s \leqq 0$						$-1 \leqq h \leqq 2^{k-1}+2$
第2実施形態	$s \geqq 0$	安全	5	7	11	$2^{k-1}+3$	$-1 \leqq h \leqq 2^{k-1}+1$
	$s \leqq 0$						
第3実施形態	$s > 0$	安全	4	6	10	$2^{k-1}+2$	$-1 \leqq h \leqq 2^{k-1}$
	$s < 0$		5	7	11	$2^{k-1}+3$	$-1 \leqq h \leqq 2^{k-1}+1$
ランダム化ウィンドウ法（第3の比較例）		安全	4	8	16	2^k	$0 \leqq h \leqq 2^k-1$
符号つきウィンドウ法（第4の比較例）		脆弱	3	5	9	$2^{k-1}+1$	$0 \leqq h \leqq 2^{k-1}$

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/002363

A. CLASSIFICATION OF SUBJECT MATTER
G09C1/00(2006.01)i

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)
G09C1/00

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

Jitsuyo Shinan Koho	1922-1996	Jitsuyo Shinan Toroku Koho	1996-2010
Kokai Jitsuyo Shinan Koho	1971-2010	Toroku Jitsuyo Shinan Koho	1994-2010

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.
A	Kouichi Itoh et.al., DPA Countermeasures by Improving the Window Method, [online], CHES2002, 2002, p.303-317, [retrieved on 2010.05.18], Retrieved from the Internet:<URL:http://eref.uqu.edu.sa/files/Others/Elliptic%20Curves/SCA/DPA%20Countermeasures%20by%20Improving%20the%20Window%20Method.pdf>	1-15
A	Fan Zhang et.al., An Efficient Window-Based Countermeasure to Power Analysis of ECC Algorithms, [online], Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008, p.120-126, [retrieved on 2010.05.18], Retrieved from the Internet:<URL:http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04492465>	1-15

☒ 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 but 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
18 May, 2010 (18.05.10)	01 June, 2010 (01.06.10)
Name and mailing address of the ISA/ Japanese Patent Office Facsimile No.	Authorized officer Telephone No.

INTERNATIONAL SEARCH REPORT

International application No.
PCT/JP2010/002363

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 2007/005563 A1 (MICROSOFT CORP.), 11 January 2007 (11.01.2007), paragraphs [0018] to [0074] & US 2007/0064931 A1	1-15

国際調査報告

国際出願番号　ＰＣＴ／ＪＰ２０１０／００２３６３

Ａ．　発明の属する分野の分類（国際特許分類（ＩＰＣ））

Int.Cl.　G09C1/00(2006.01)i

Ｂ．　調査を行った分野

調査を行った最小限資料（国際特許分類（ＩＰＣ））

Int.Cl.　G09C1/00

最小限資料以外の資料で調査を行った分野に含まれるもの

日本国実用新案公報	１９２２－１９９６年
日本国公開実用新案公報	１９７１－２０１０年
日本国実用新案登録公報	１９９６－２０１０年
日本国登録実用新案公報	１９９４－２０１０年

国際調査で使用した電子データベース（データベースの名称、調査に使用した用語）

Ｃ．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	Kouichi Itoh et.al., DPA Countermeasures by Improving the Window Method, [online], CHES2002, 2002, p.303-317, [retrieved on 2010.05.18], Retrieved from the Internet:<URL: http://eref.uqu.edu.sa/files/Others/Elliptic%20Curves/SCA/DPA%20Countermeasures%20by%20Improving%20the%20Window%20Method.pdf>	1-15

☑　Ｃ欄の続きにも文献が列挙されている。　　☐　パテントファミリーに関する別紙を参照。

＊　引用文献のカテゴリー
「Ａ」特に関連のある文献ではなく、一般的技術水準を示すもの
「Ｅ」国際出願日前の出願または特許であるが、国際出願日以後に公表されたもの
「Ｌ」優先権主張に疑義を提起する文献又は他の文献の発行日若しくは他の特別な理由を確立するために引用する文献（理由を付す）
「Ｏ」口頭による開示、使用、展示等に言及する文献
「Ｐ」国際出願日前で、かつ優先権の主張の基礎となる出願の日の後に公表された文献
「Ｔ」国際出願日又は優先日後に公表された文献であって出願と矛盾するものではなく、発明の原理又は理論の理解のために引用するもの
「Ｘ」特に関連のある文献であって、当該文献のみで発明の新規性又は進歩性がないと考えられるもの
「Ｙ」特に関連のある文献であって、当該文献と他の１以上の文献との、当業者にとって自明である組合せによって進歩性がないと考えられるもの
「＆」同一パテントファミリー文献

国際調査を完了した日 １８．０５．２０１０	国際調査報告の発送日 ０１．０６．２０１０
国際調査機関の名称及びあて先 日本国特許庁（ＩＳＡ／ＪＰ） 郵便番号１００－８９１５ 東京都千代田区霞が関三丁目４番３号	特許庁審査官（権限のある職員）　５Ｓ　３８５７ 新田　亮 電話番号　０３－３５８１－１１０１　内線　３５４６

Ｃ（続き）．　関連すると認められる文献

引用文献のカテゴリー＊	引用文献名　及び一部の箇所が関連するときは、その関連する箇所の表示	関連する請求項の番号
A	Fan Zhang et.al., An Efficient Window-Based Countermeasure to Power Analysis of ECC Algorithms, [online], Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008, p.120-126, [retrieved on 2010.05.18], Retrieved from the Internet:<URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=04492465>	1-15
A	WO 2007/005563 A1 (MICROSOFT CORPORATION) 2007.01.11, 段落【0018】～【0074】 & US 2007/0064931 A1	1-15