(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3591857号
(P3591857)
(45)発行日　平成16年11月24日(2004.11.24)　(24)登録日　平成16年9月3日(2004.9.3)

(51)Int.Cl.[7]　F I
G09C 1/00　G09C 1/00 650B
G06F 7/58　G06F 7/58 A

請求項の数 5　(全 24 頁)

(21)出願番号　特願平5-331241
(22)出願日　平成5年12月27日(1993.12.27)
(65)公開番号　特開平7-193565
(43)公開日　平成7年7月28日(1995.7.28)
審査請求日　平成12年12月27日(2000.12.27)

前置審査

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(72)発明者　山本　貴久
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

最終頁に続く

(54)【発明の名称】擬似乱数生成方法及び装置、通信方法及び装置

(57)【特許請求の範囲】

【請求項１】

所定の整数Ｎと、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1}=R^{-1}\cdot X_i^2 \bmod N$（i=0,1,2,…,s）をモンゴメリ法に基づいて$X_{i+1}=(X_i^2+(X_i^2\cdot(-N^{-1} \bmod R) \bmod R)\cdot N)/R$によって計算するようにして、所定の初期値$X_0$から当該漸化式により数列$X_1,X_2,X_3,\cdots X_{s+1}$を順次求める計算手段と、

前記漸化式$X_{i+1}=R^{-1}\cdot X_i^2 \bmod N$により求められた数列の各$X_i$から所定のビット部分を抽出し、その抽出された抽出データを疑似乱数データとする抽出手段とを備えることを特徴とする疑似乱数生成装置。

【請求項２】

前記Ｎは奇数であって、前記整数Ｒとして、ｔを整数とする2^tを選択し、

前記計算手段は、前記モンゴメリ法に基づく演算における前記整数Ｒによる除算項を、その除算項の被除数をｔビットシフトすることで計算することを特徴とする請求項１に記載の疑似乱数生成装置。

【請求項３】

任意のυ，νに対して、$\upsilon\cdot\nu\cdot R^{-1} \bmod N$をモンゴメリ法に基づいて$(\upsilon\cdot\nu+(\upsilon\cdot\nu\cdot(-N^{-1} \bmod R) \bmod R)\cdot N)/R$によって演算する処理をＭｏｎｔ（υ，ν）と表すとき、所定の整数Ｎと、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1}=R^{-(e-1)}\cdot X_i^e \bmod N$（i=0,1,2,…,s）をモンゴメリ法に基づいてＭｏｎｔ（X_{i+1}，X_i）及びＭｏｎｔ（X_{i+1}，X_{i+1}）を用いて計算するようにして、所定

の初期値X_0から当該漸化式により数列$X_1, X_2, X_3, \cdots X_{s+1}$を順次求める計算手段と、
前記漸化式$X_{i+1} = R^{-(e-1)} \cdot X_i^e \mod N$により求められた数列の各$X_i$から所定のビット部分を抽出し、その抽出された抽出データを疑似乱数データとする抽出手段と、
を備えることを特徴とする疑似乱数生成装置。
【請求項４】
前記Ｎは奇数であって、前記整数Ｒとして、ｔを整数とする2^tを選択し、
前記計算手段は、前記モンゴメリ法に基づく演算における前記整数Ｒによる除算項を、その除算項の被除数をｔビットシフトすることで計算することを特徴とする請求項３に記載の疑似乱数生成装置。
【請求項５】
請求項１または３に記載の疑似乱数生成装置と、
前記疑似乱数生成装置により生成された疑似乱数データを用いて通信データを暗号化して送信する送信手段と、
暗号化されたデータを受信して、当該暗号化されたデータを前記疑似乱数生成装置により生成された疑似乱数データを用いて復号する受信手段とを有することを特徴とする通信装置。
【発明の詳細な説明】
【０００１】
【産業上の利用分野】
本発明は、暗号通信、特に、データの秘匿、発信者・着信者の認証、暗号鍵の共有、零知識証明プロトコル等で必要な乱数生成を用いた通信方法とその装置に関する。また、例えば、モンテカルロシミュレーション等で必要な乱数生成方法とその装置に関する。
【０００２】
【従来の技術】
暗号通信で用いる乱数列は、ある時点までの生成した乱数列のみからその時点以降に生成する乱数が容易に予測できないことが必要である。文献「Ｐｒｉｍａｌｉｔｙ ａｎｄ Ｃｒｙｐｔｏｇｒａｐｈｙ」（Ｅｖａｎｇｅｌｏｓ Ｋｒａｎａｋｉｓ著、ＪＯＨＮ ＷＩＬＥＹ＆ＳＯＮＳ発行、ｐｐ．１０８－１３７）には、上の条件を満たす疑似乱数系列が掲載されている。
【０００３】
つまり、疑似乱数列を$b_1, b_2, \cdots$とするとビットb_iは，X_0を任意に与える初期値、ｐ，ｑを$p \equiv q \equiv 3 \pmod 4$である素数として、

$$X_{i+1} = X_i^2 \mod N \quad (i = 0, 1, 2, \cdots) \qquad (式１)$$

$$b_i = lsb(X_i) \quad (i = 1, 2, \cdots) \qquad (式２)$$

によって与えれる（ただし、Ｎ＝ｐ・ｑ，ｌｓｂは最下位ビットを表わす）。
【０００４】
また、文献「暗号と情報セキュリティ」（辻重男、笠原正雄編著、昭晃堂発行、ｐｐ．８６）には、別の疑似乱数列生成方法が掲載されている。
【０００５】
つまり、疑似乱数列を$b_1, b_2, \cdots$とするとビットb_iは、x_0を任意に与える初期値、ｐ，ｑを素数、ｅをＬ（Ｌはｐ－１とｑ－１の最小公倍数）と互いに素な数として、

$$x_{i+1} = x_i^e \mod N \quad (i = 0, 1, 2, \cdots) \qquad (式３)$$

$$b_i = lsb(x_i) \quad (i = 1, 2, \cdots) \qquad (式４)$$

によって与えられる（ただし、Ｎ＝ｐ・ｑ，ｌｓｂは最下位ビットを表わす）。
【０００６】
これらの方法から生成された疑似乱数列$b_1, b_2, \cdots, b_i$のみからb_{i+1}を求めることは、Ｎを因数分解するのと同程度の手間が必要であることが知られている。つまり、ある時点までに生成された疑似乱数列のみからその時点以降に生成されるべき乱数を求めるための計算量は、Ｎを因数分解するのに必要な計算量と同等であることが知ら

れている。ただし、Ｎを因数分解することを計算量的に困難にするためにはｐ，ｑを数百ビット程度にする必要がある。このように、ある時点までに生成された乱数列のみからその時点以降に生成されるべき乱数を予測することが計算量的に困難となるような方法により生成された乱数は、暗号学的に安全な疑似乱数と呼ばれている。

【０００７】

式（１）、式（３）のような演算は、次式で示される乗算剰余と呼ばれる演算に含まれる。

Ｑ＝υ・ν　ｍｏｄ　Ｎ　（Ｑ，υ，νは整数）　（式５）

このような乗算剰余を効率的に演算する方法としてモンゴメリ法（Ｐ．Ｌ．Ｍｏｎｔｇｏｍｅｒｙ“Ｍｏｄｕｌａｒ　ｍｕｌｔｉｐｌｉｃａｔｉｏｎ　ｗｉｔｈｏｕｔ　ｔｒｉａｌ　ｄｉｖｉｓｉｏｎ”Ｍａｔｈ．ｏｆ　Ｃｏｍｐｕｔａｔｉｏｎ，ｖｏｌ．４４，１９８５，ｐｐ．５１９－５２１　参照）が知られている。このモンゴメリ法を用いれば、法Ｎによる除算を行うことなく演算できるので、通常の乗算剰余より効率的に処理できる。

【０００８】

モンゴメリ法を用いた場合の乗算剰余をＭｏｎｔ（υ，ν）とすると、Ｍｏｎｔ（υ，ν）は互いに素な数であるＲを用いて、

$$Mont(\upsilon, \nu) \equiv \upsilon \cdot \nu \cdot R^{-1} \pmod{N} \quad (式6)$$

により与えられる。

【０００９】

モンゴメリ法では、上式の計算結果Ｍｏｎｔ（υ，ν）を得るために

$$Mont(\upsilon, \nu) = (\upsilon \cdot \nu + M \cdot N) / R \quad (式7)$$

ただし、

$$M = \upsilon \cdot \nu \cdot N' \bmod R \quad (式8)$$

$$N' = -N^{-1} \bmod R \quad (式9)$$

という演算を行っている。

【００１０】

Ｎが奇数の場合、$R = 2^t$　（ｔは任意の整数）とすればＲとＮは互いに素な整数になる。この場合、Ｒによる除算及び剰余演算は実質的な演算を必要とせず、Ｍｏｎｔ（υ，ν）を乗算と加算のみで高速に計算できる。

【００１１】

モンゴメリ法を用いた場合の自乗剰余演算の計算手順は、式（１）と同じパラメータとＮと互いに素な数であるＲを用いて、

$$y_0 = R \cdot X_0 \bmod N \quad (式10)$$

$$y_{i+1} = R^{-1} \cdot y_i^2 \bmod N \quad (i = 0, 1, 2, \cdots) \quad (式11)$$

によって与えられる。

【００１２】

この場合、式（１）と式（１１）で生成される系列を比較すると、

$$y_i = R \cdot X_i \bmod N \quad (i = 0, 1, 2, \cdots) \quad (式12)$$

となり、式（１１）で生成される系列ｙｉ　（ｉ＝０，１，２，…）は、式（１）で生成される系列X_i　（ｉ＝０，１，２…）にＲを掛けたものになっている。よって従来では、暗号学的に安全な疑似乱数系列として、X_i　の最下位ビットの系列であるb_i　を生成するために、演算で得られたy_i　に対し、

$$X_i = R^{-1} \cdot y_i \bmod N \quad (i = 0, 1, 2, \cdots) \quad (式13)$$

という演算を行う必要があった。

【００１３】

一方、式（３）は、式（５）に示したべき乗剰余演算を繰り返すことによって実行することができる。具体的に乗算剰余の繰り返しによって、

べき乗剰余$x_{i+1} = x_i^e \bmod N$（ｉ＝０，１，２，…，ｓ）を順次計算する手順は以下の［アルゴリズム１］に示すようになる。また、そのフローチャートを図１３

に示す。ただし、ただし、ｅをｋビットからなる整数でｅ＝［e_k　e_{k-1}　…e_2　e_1　］で表わすとする。

[アルゴリズム1]

```
INPUT x0, e, N, s                          (**1)
FOR i=0 TO s                               (**2)
  x(i+1)=1
  FOR j=k TO 1
    IF ej=1 THEN x(i+1)=x(i+1)・xi mod N
    IF j>1  THEN x(i+1)=x(i+1)・x(i+1) mod N
  NEXT
  OUTPUT  x(i+1) (=xi^e mod N)
NEXT                                       (**9)
```

【００１４】

ライン（＊＊１）のＩＮＰＵＴ文では、x_0、ｅ、Ｎ、ｓの各値を入力する。ここで、ｓは剰余演算の繰り返し数である。ライン（＊＊２）のＦＯＲ文は、ライン（＊＊９）までの処理を、変数ｉに関して「０」から「ｓ」まで繰り返すコマンドであり、べき乗剰余x_{i+1}（ｉ＝０，１，２，…，ｓ）を順次求める処理を繰り返させる。

【００１５】

次に、モンゴメリ法の計算手順を用いた乗算剰余の繰り返しによるべき乗剰余$x_{i+1}=x_i^e \bmod N$を計算する手順は以下のようになる。ただし、ＲはＮと互いに素である整数とし、ｅは先程と同じくｋビットからなる整数でｅ＝［e_k　e_{k-1}　…e_2　e_1　］で表わせるとする。このアルゴリズムを実行すれば、式（２）で得られる系列x_i（ｉ＝０，１，２，…，ｓ）を得ることができる。

[アルゴリズム2]

```
INPUT x0, e, N, s, RR=R^2 mod N
FOR i=0 TO s
      yi=Mont (xi, RR)                     (*1)
      y(i+1)=Mont (1, RR)                  (*2)
      FOR j=k TO 1
      IF ej=1 THEN y(i+1)=Mont (y(i+1), yi)
      IF j>1  THEN y(i+1)=Mont (y(i+1), y(i+1))
NEXT
x(i+1)=Mont (y(i+1), 1)                    (*3)
OUTPUT  x(i+1) (=xi^e mod N)
NEXT
```

［アルゴリズム２］に従って式（２）をモンゴメリ法により計算する場合に、ｊに対するＦＯＲ－ＮＥＸＴ部分の出力として得られる系列y_{i+1}（ｉ＝０，１，２，…，ｓ）は、式（１）と同じパラメータとＮと互いに素な数であるＲを用いて、

$y_0 = R \cdot x_0 \bmod N$　　（式１４）

$y_{i+1} = R^{-(e-1)} \cdot y_i^e \mod N$ （i＝0，1，2，…） （式１５）
と表わされる。
【００１６】
この場合、式（３）で生成される系列x_{i+1}（i＝0，1，2，…）と式（１５）で生成される系列y_{i+1}（i＝0，1，2，…）とを比較すると、
$y_i = R \cdot x_i \mod N$ （i＝0，1，2，…） （式１６）
となっている。つまり、［アルゴリズム２］に従って式（３）をモンゴメリ法により計算する場合に、ｊに対するＦＯＲ－ＮＥＸＴ部分の出力として得られる系列y_{i+1}（i＝0，1，2，…ｓ）は、式（３）で得られる系列x_{i+1}（i＝0，1，2，…ｓ）に対して式（１６）のような関係にある。
【００１７】
よって、入力x_iに対してモンゴメリ法を用いないべき乗剰余演算の［アルゴリズム１］で得られる演算結果x_{i+1}（$x_i^e \mod N$）を、モンゴメリ法を用いたべき剰余演算の［アルゴリズム２］により得るためには、［アルゴリズム２］の式（＊１）により、x_iをy_i＝Ｍｏｎｔ（x_i，R_R）（＝$R \cdot x_i \mod N$）に補正し、式（＊３）により出力x_{i+1}としてｊに対するＦＯＲ－ＮＥＸＴ部分の出力として得られるy_{i+1}からx_{i+1}＝Ｍｏｎｔ（y_{i+1}，１）（＝$R^{-1} \cdot y_{i+1} \mod N$）と補正する必要があった。
【００１８】
【発明が解決しようとする課題】
しかしながら、上述した安全な疑似乱数生成方法を用いた場合には、ｐ，ｑを数百ビット程度にする必要があるため演算量が大きく、特に式（１）、式（３）部分の計算量が大きいため、高速に疑似乱数を生成できず、その疑似乱数に基づいて通信データの生成／再生を高速に行えないという問題があった。
【００１９】
本発明は上記従来例に鑑みてなされたもので、安全な疑似乱数をより高速にかつ容易に生成可能とすることを目的とする。また、そのようにして生成された擬似乱数を用いて通信データの暗号化／復号を高速に実行可能とすることを目的とする。
【００２０】
【課題を解決するための手段】
上記の目的を達成するための本発明による擬似乱数発生装置は以下の構成を備える。即ち、
所定の整数Ｎと、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1} = R^{-1} \cdot X_i^2 \mod N$ (i=0,1,2,…,s) をモンゴメリ法に基づいて$X_{i+1} = (X_i^2 + (X_i^2 \cdot (-N^{-1} \mod R) \mod R) \cdot N) / R$によって計算するようにして、所定の初期値$X_0$から当該漸化式により数列$X_1, X_2, X_3, \cdots X_{s+1}$を順次求める計算手段と、
前記漸化式$X_{i+1} = R^{-1} \cdot X_i^2 \mod N$により求められた数列の各$X_i$から所定のビット部分を抽出し、その抽出された抽出データを疑似乱数データとする抽出手段とを備える。
【００２１】
また、本発明の他の形態による擬似乱数発生装置は以下の構成を備える。即ち、
任意のυ，νに対して、$\upsilon \cdot \nu \cdot R^{-1} \mod N$をモンゴメリ法に基づいて$(\upsilon \cdot \nu + (\upsilon \cdot \nu \cdot (-N^{-1} \mod R) \mod R) \cdot N) / R$によって演算する処理をＭｏｎｔ（υ，ν）と表すとき、所定の整数Ｎと、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1} = R^{-(e-1)} \cdot X_i^e \mod N$ (i=0,1,2,…,s) をモンゴメリ法に基づいてＭｏｎｔ（X_{i+1}，X_i）及びＭｏｎｔ（X_{i+1}，X_{i+1}）を用いて計算するようにして、所定の初期値X_0から当該漸化式により数列$X_1, X_2, X_3, \cdots X_{s+1}$を順次求める計算手段と、
前記漸化式$X_{i+1} = R^{-(e-1)} \cdot X_i^e \mod N$により求められた数列の各$X_i$から所定のビット部分を抽出し、その抽出された抽出データを疑似乱数データとする抽出手段とを備える。

【００２２】
また、本発明の他の形態による通信装置は、
上記疑似乱数生成装置により生成された疑似乱数データを用いて通信データを暗号化して送信する送信手段と、
暗号化されたデータを受信して、当該暗号化されたデータを前記疑似乱数生成装置により生成された疑似乱数データを用いて復号する受信手段とを有する。
【００２４】
【作用】
上記構成の擬似乱数生成装置によれば、所定の整数Ｎと、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1}=R^{-1}\cdot X_i^2 \bmod N$（i=0,1,2,…,s）をモンゴメリ法に基づいて$X_{i+1}=(X_i^2+(X_i^2\cdot(-N^{-1}\bmod R)\bmod R)\cdot N)/R$によって計算するようにして、所定の初期値$X_0$から当該漸化式により数列$X_1,X_2,X_3,\cdots X_{s+1}$を順次求める。そして、求めた数列の各$X_i$から所定のビット部分を抽出することにより疑似乱数データが得られる。
【００２５】
また、本発明の他の態様による擬似乱数発生装置によれば、任意のυ，νに対して、$\upsilon\cdot\nu\cdot R^{-1} \bmod N$をモンゴメリ法に基づいて$(\upsilon\cdot\nu+(\upsilon\cdot\nu\cdot(-N^{-1}\bmod R)\bmod R)\cdot N)/R$によって演算する処理をＭｏｎｔ（υ，ν）と表すとき、所定の整数Ｎと、所定の初期値X_0と、前記Ｎと互いに素な整数Ｒとを用いて、漸化式$X_{i+1}=R^{-(e-1)}\cdot X_i^e \bmod N$（i=0,1,2,…,s）をモンゴメリ法に基づいてＭｏｎｔ（X_{i+1}，X_i）及びＭｏｎｔ（X_{i+1}，X_{i+1}）を用いて計算するようにして、所定の初期値から当該漸化式により数列$X_1,X_2,X_3,\cdots X_{s+1}$を順次求める。そして、この数列の各$X_i$から所定のビット部分を抽出することにより擬似乱数データが得られる。
【００２６】
更に、本発明の通信装置は、上述の擬似乱数生成装置によって生成された疑似乱数データを用いて通信データを暗号化して送信する。また、そのように暗号化されたデータを受信して、当該暗号化されたデータを上述の擬似乱数生成装置によって生成された疑似乱数データを用いて復号する。
【００２８】
【実施例】
以上説明した課題を解決する本発明の第１の実施例について、以下、適宜図を参照しながら説明する。
【００２９】
本実施例による疑似乱数生成方法は、暗号学的に暗算な疑似乱数系列として式（１２）のy_i　から直接、
α_i　＝ｌｓｂ（y_i　）（ｉ＝０，１，２，…）　　（式１７）
によって得られたαｉ　を用いることにより、式（１３）の演算を省くことを可能にし、出力乱数の安全性を低下させることなく疑似乱数の生成速度を向上させたものである。
【００３０】
式（１）で用いた疑似乱数生成の安全性は、X_{i+1}　からb_i　を求めることが非常に困難であること、つまりb_i　がX_{i+1}のハードコアビットであることを利用している。式（１１），（１７）を用いた疑似乱数生成の場合もＸｉ＋１　にある定数Ｒを掛けたy_{i+1}　からα_i　を求めることは非常に困難である。つまり、α_iはy_{i+1}　のハードコアビットであるので、式（１１），式（１７）を用いた疑似乱数生成の安全性も、式（１）を用いた疑似乱数生成と同程度である。
【００３１】
以上のように本実施例によれば、式（１３）の処理を行うことなく、式（１）と同程度の安全性を有する疑似乱数列をより高速に、或はより小さな回路規模で生成することを可能にし、本実施例の方法を用いることで、高速に通信データの生成／再生を行うことができる。

【００３２】
次に、式（１１）、式（１７）を用いた疑似乱数生成の安全性も式（１）を用いた疑似乱数生成と同程度であることの証明を行う。その前に用いる記号等を簡単に定義する。詳しくは、文献「現代暗号理論」（池野、小山著、昭和６１年発行、電子情報通信学会、１４－１５、９５－９６）参照されたい。
・平方剰余：
$X_2 \equiv c \pmod{p}$ に解があるときｃはｐの平方剰余、解がないときは平方非剰余という。
【００３３】
・ルジャンドル記号（ｘ／ｐ）：
ｐを素数とし、$X \neq 0 \pmod{p}$ のとき、
（ｘ／ｐ）＝　１　：ｘがｐの平方剰余の時
　　－１　：ｘがｐの平方非剰余の時
・Z^*_N：
Ｎと互いに素で０からＮ－１までの範囲の整数。
【００３４】
・ヤコビ記号（ｘ／Ｎ）：
$x \in Z^*_N$ と、Ｎ＝ｐ・ｑ（ｐ，ｑは素数）に対し、ヤコビ記号（ｘ／Ｎ）は、ルジャンドル記号（ｘ／ｐ）を用いて、
（ｘ／Ｎ）＝（ｘ／ｐ）（ｘ／ｑ）
で表される。
・$Z^*_N(+1) = \{x \in Z^*_N \mid (x/N) = 1\}$
・$Z^*_N(-1) = \{x \in Z^*_N \mid (x/N) = -1\}$
・$Q_1 = \{x \in Z^*_N \mid (x/p) = (x/q) = 1\}$
・$Q_2 = \{x \in Z^*_N \mid (x/p) = (x/q) = -1\}$
・$Q_3 = \{x \in Z^*_N \mid (x/p) = -(x/q) = 1\}$
・$Q_4 = \{x \in Z^*_N \mid (x/p) = -(x/q) = -1\}$
式（１１）、式（１７）を用いた疑似乱数生成の安全性が式（１）を用いた疑似乱数生成の安全性と同じであることを示すのは次の命題を証明することに等しい。
［命題］
「式（１１）、（１７）より生成された疑似乱数列 α_{i+1} ，α_{i+2} ，…から α_i を正しく推測できるなら、任意のｃ（$c \in Z^*_N(+1)$）に対して、その平方剰余を判定できる。」
［証明］
$R \in Q_1$ の時、任意のｃ（$c \in Z^*_N(+1)$）に対して、
$b = R \cdot c \mod N$
$y_{i+1} = R^{-1} \cdot b^2 \mod N$
とし、y_{i+1} を初期値として、式（１１）、式（１７）により α_{i+1} ，α_{i+2} ，…を生成する。
【００３５】
このとき、
$b \in Z^*_N(+1)$
$y_{i+1} \in Q_1$
となる。このとき、
$y_{i+1} = R^{-1} \cdot z^2 \mod N$
$y_{i+1} \cdot R = z^2 \mod N$
の解を Z_j （ｊ＝１，２，３，４）（ただし、$Z_j \in Q_j$ ）とすると、

$Z_1 \equiv -Z_2 \pmod N$
$Z_3 \equiv -Z_4 \pmod N$
より、仮定を用いて、
$\alpha_i = lsb(z_1)$
を予想することができる。よって、
$\alpha_i = lsb(b)$ なら $b \in Q_1$ そのとき $c \in Q_1$
$\alpha_i \neq lsb(b)$ なら $b \in Q_2$ そのとき $c \in Q_2$
同様に $R \in Q_2$ ，$R \in Q_3$ ，$R \in Q_4$ の時も、cの平方剰余性を判定できる。
【００３６】
証明終わり。
図１は本実施例による疑似乱数生成装置１の構成を示す図である。疑似乱数生成装置１は、自乗剰余演算装置２と、論理演算装置３から構成される。
【００３７】
自乗剰余演算装置２は、初期値 y_0 と剰余演算の法であるNと互いに素である任意の定数Rから、次式に示す演算を連鎖的に行い、y_1 ，y_2 ，…を生成する。
$y_{i+1} = R^{-1} \cdot y_i^2 \mod N \quad (i = 0, 1, 2, \cdots)$ （式１８）
$N = p \cdot q$ （式１９）
ただし、p、qは、$p \equiv q \equiv 3 \pmod 4$ である素数。
【００３８】
Rは、Nと互いに素な任意の数。
生成された y_1 ，y_2 ，…は、y_{i+1} 出力線５にシーケンシャルに出力される。
【００３９】
上式で表わされる演算方式は、前述のようにモンゴメリ法と呼ばれるものである。モンゴメリ法では、実際に計算結果 y_{i+1} を得るために、次式の演算を行う。
$y_{i+1} = (y_i^2 + M \cdot N) / R \quad (i = 0, 1, 2, \cdots)$ （式２０）
ただし、
$M = y_i^2 \cdot N' \bmod R$ （式２１）
$N' = -N^{-1} \bmod R$ （式２２）
ここで、Nは奇数であるので、$R = 2^t$ （tは任意の整数）とすればRとNは互いに素な整数になる。この場合、Rによる除算及び剰余演算は実質的な演算を必要とせず、y_{i+1} を乗算と加算演算で高速に計算できる。
【００４０】
自乗剰余演算装置２の構成としては、例えばハードウエアで構成する場合には、加算機と、乗算器と、Rによる除算及び剰余演算のためのビットシフトを行うシフタ等の基本演算器を備えることによって、容易にモンゴメリの方法による演算を実行できる。さらに、文献「モンゴメリー法を用いたべき乗アルゴリズムとシストリックアレイ」（岩村、松本、今井；ｖｏｌ．９２，Ｎｏ．１３４，ｐｐ．４９－５４，１９９２ ）に示される公知のモンゴメリ演算回路を用いることもできる。
【００４１】
自乗剰余演算装置２の y_{i+1} 出力線５に、シーケンシャルに出力される y_1 ，y_2 ，…の各値は論理演算装置３に入力される。論理演算装置３では、入力された y_1，y_2，…のそれぞれ下位 $log_2 n$（但し、nはNの２進表示による桁数）ビットの範囲にある任意の（単数／複数）ビットを切り出して、それを疑似乱数として出力線６に疑似乱数を出力する。例えば、下位 $log_2 n$ ビット全てを疑似乱数としてもよいし、最下位ビットだけを疑似乱数として出力してもよい。
【００４２】
論理演算装置３を、例えばハードウエアにより構成する場合には、入力された入力された y_i （ｉ＝１，２，…）をパラレルでラッチし、その下位 $log_2 n$ ビットを純にシリアルに出力するようなパラレル入力シリアル出力のシフトレジスタを用いることができる。

【００４３】
図２は、第１の実施例である疑似乱数生成方法をソフトウエアで実現するための疑似乱数生成プログラムを搭載したデータ処理装置１５の構成を示す図である。ここで、ＣＰＵ１０は、疑似乱数生成プログラムを搭載したデータ処理装置１５全体の制御を行う。キーボード１１は、疑似乱数生成プログラムの起動コマンドや疑似乱数生成プログラムのための各種パラメータ値などの入力を行う。ＲＯＭ１３には、本実施例の疑似乱数を生成する疑似乱数生成プログラム等が予め格納されており、ＣＰＵ１０によって読み出されながら実行される。ＲＡＭ１４は、疑似乱数生成プログラムを実行させるなどの作業領域であり、また疑似乱数生成結果が格納される。
【００４４】
第３図は、疑似乱数生成プログラムの処理過程を説明するフローチャートである。以下このフローチャートに基づき疑似乱数生成プログラムの処理過程を説明する。疑似乱数生成プログラムは、式（７）～式（９）の演算の実行を基本としている。
【００４５】
ステップＳ１では、３（ｍｏｄ　４）である素数を任意に選び、ｐとｑに設定する。そして、ｐ・ｑの乗算を行い、結果をＮに設定する。そして、Ｎと互いに素な任意の数をＲに設定する。さらに、$R=2^{t}$ を満たす「ｔ」値を計算する。またさらに、「$-N^{-1}$　ｍｏｄ　Ｒ」を計算して、結果をＮ’に設定する。
【００４６】
ステップＳ２では、乱数生成のための任意の初期値をy_0にセットする。
【００４７】
ステップＳ３では、$y_i^{2}\cdot N' \bmod 2^{t}$ の演算を行い、結果をＮ’に設定する。
【００４８】
ステップＳ４では、乱数生成の繰り返し数を示すｉを「０」に初期化する。
【００４９】
ステップＳ５では、「$y_i^{2}\cdot N' \bmod 2^{t}$」を計算して、結果をＭに設定する。尚、ここで、「$y_i^{2}\cdot N'$」を$2^{t}$で割る処理が基本になるので、除算は「$y_i^{2}\cdot N'$」を「ｔ」ビットシフト処理すればよいので高速に処理できる。
【００５０】
ステップＳ６では、「$(y_i^{2}+M\cdot N)/2^{t}$」の演算を行い、結果をy_{i+1}に設定する。ここでの除算も「ｔ」ビットシフトが基本となるので高速に処理できる。
【００５１】
ステップＳ７では、y_iから、所定数の所定位置のビット列を切り出して、ｒａｎｄ（ｉ）に設定する。尚、ｒａｎｄ（ｉ）は１次元の配列である。
【００５２】
ステップＳ８では、ｉ値を所定のＥｎｄ値と比較して、ｉ値がＥｎｄ値より小さければ、ステップＳ９へ進む。尚、Ｅｎｄ値は、予め生成したい乱数列の量に対応して設定されている。これは、例えば、キーボード１１からでもよいし、本疑似乱数生成プログラムを呼び出す別のプログラムで設定してもよい。
【００５３】
ステップＳ９では、次の乱数を生成するためにｉをカウントアップする。そして、ステップＳ５からの処理に戻り、次の乱数の生成を継続する。
【００５４】
以上の処理を行うことで、配列領域ｒａｎｄに一連の疑似乱数列が生成される。
【００５５】
尚、モンゴメリ法による疑似乱数生成方法を示してきたが、一般に自乗演算結果X^{2}に任意の定数Ｃを掛けた数$C\cdot X^{2}$に対し剰余演算を行い、その結果得られる$y=C\cdot X^{2}$ ｍｏｄＮの所定ビットから疑似乱数を生成することもできる。図４は、第２の実施例による疑似乱数生成装置２０の構成を示す図である。疑似乱数生成装置２０は、自乗剰余演算装置２１と論理演算装置２２から構成される。

【００５６】
自乗剰余演算装置２１は、初期値y_0と任意の定数Ｃから、次式に基づく演算を連鎖的に行い、ｙ１，ｙ２，…を生成する。
$y_{i+1} = C \cdot y_i^2 \mod N$　（ｉ＝０，１，２，…）
$N = p \cdot q$
ただし、ｐ，ｑはｐ≡ｑ≡３（ｍｏｄ４）である素数。
図４の自乗剰余演算装置２１に示された演算、即ち、
$y_i^2 \mod N$
$C \cdot y_i^2 \mod N$
は、両者とも乗算剰余演算であるため、自乗剰余装置２１を構成としては、乗算剰余演算を行えるものであればよい。特にハードウエアにより構成する場合には、文献「並列処理によるＲＳＡ暗号装置の構成法」（岩村、松本、今井；電子情報通信学会論文Ａ、ｖｏｌ．Ｊ７５－Ａ，Ｎｏ．８，ｐｐ．１３０１－１３１１，１９９２）に示される公知の乗算剰余演算回路を用いることもできる。自乗剰余演算装置からの出力は、論理演算装置に入力される。論理演算装置では入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$（ただし、ｎはＮの２進表示による桁数）ビットの範囲にある任意の（複数／単数）ビットから疑似乱数を生成し出力する。

【００５７】
論理演算装置２２は入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$ビットの範囲にある任意のビットを疑似乱数として出力することができる。例えば、下位$\log_2 n$ビット全てを疑似乱数とすることも、最下位ビットだけを疑似乱数として出力することもできる。

【００５８】
以上、詳細に説明した様に本発明の実施例によれば、暗号学的に安全な疑似乱数をモンゴメリ法により生成する場合、式（１１）により得られるｙｉ　の所定のビットを疑似乱数として用いることにより、安全性を低下させることなく、従来必要であった式（１３）の演算を不要にできる。そのことにより、従来と同程度の安全性を有する疑似乱数列を高速に、或はより小さな回路規模で生成することが可能である。

【００５９】
以上説明した様に、本実施例の方法とその装置で生成された疑似乱数を用いることで、高速に通信データの生成／再生を行うことができる。

【００６０】
［第２の実施例］
第２の実施例による疑似乱数生成器は、［アルゴリズム２］の式（＊３）で示された演算を省くことにより、式（３）と同程度の安全性を有する疑似乱数列をより高速に、或はより小さな回路規模で生成することを可能とすることを目的としている。

【００６１】
まず、［アルゴリズム２］の式（＊３）で示された演算を省いた時の［アルゴリズム３］を以下に示す。

[アルゴリズム３]

INPUT x_0, e, N, $R_R = R^2 \bmod N$
$y_0 = Mont(x_0, R_R)$
FOR i=0 TO s
　　$y_{i+1} = Mont(1, R_R)$
FOR j=k TO 1
　　IF $e_j = 1$ THEN $y_{i+1} = Mont(y_{i+1}, y_i)$
　　IF j>1 THEN $y_{i+1} = Mont(y_{i+1}, y_{i+1})$
NEXT
OUTPUT y_{i+1} $(= R^{-(e-1)} \cdot y_i^e \bmod N)$
NEXT

この［アルゴリズム３］を実行することによりえられる系列y_i（ｉ＝０，１，２，…，ｓ）は、式（１５）により示される。式（＊３）を省略した場合の第５の実施例による疑似乱数生成器は、暗号学的に疑似乱数系列としての式（１５）のy_iから

$$\alpha_i = lsb(y_i) \quad (i=0, 1, 2, \cdots) \qquad (式２３)$$

によって得られたα_iを用いる。

【００６２】

ここで、式（１５）と式（２３）を用いた疑似乱数生成の安全性について考察する。式（３）で用いた疑似乱数生成の安全性は、x_{i+1}からb_iを求めることが困難であること、つまりb_iがx_{i+1}のハードコアビットであることを利用している。式（１５）で得られる系列y_i（ｉ＝０，１，２，…）は、式（３）で得られる系列x_i（ｉ＝０，１，２，…）に定数Ｒを掛けてＮで剰余をとった値である（式（１６）参照）。よって、x_{i+1}から$b_i = lsb(x_i)$を求めることが非常に困難であれば、x_{i+1}に定数Ｒを掛けてＮで剰余をとったy_{i+1}（$= R \cdot x_{i+1} \bmod N$）から$\alpha_i = lsb(y_i)$（$lsb(R \cdot x_i \bmod N)$）を求めることも非常に困難である。つまり、$a_i$は$y_{i+1}$のハードコアビットであるので、式（１５）と式（２３）を用いた疑似乱数生成の安全性も式（３）を用いた疑似乱数生成と同程度である。

【００６３】

次に、図５を用いて、［アルゴリズム３］を実行するための処理フローを説明する。

【００６４】

ステップＳ１０では、x_0、ｅ、Ｎ、ｓの各値を入力する。ここで、ｓは剰余演算の繰り返し数である。

【００６５】

ステップＳ１１では、Ｍｏｎｔ（x_0，R_R）の演算を行い、その結果をy_0に設定する。ここで、Ｍｏｎｔ（x_0，R_R）は、x_0，R_Rを変数とし、式（７）、式（８）、式（９）の演算を行う関数である。但し、式（９）は、予め計算しておくことができ、Ｎ’は定数として扱えるため、実際には、式（７）、式（８）を演算してＭｏｎｔ（x_0，R_R）を求める。式（７）、式（８）の演算において、法Ｒは予め2^tで表現されているため、必要な除算は、ビットシフトで実行できる。

【００６６】

ステップＳ１２では、剰余演算の繰り返し数のカウンタｉを、まず「０」に初期化する。そして、このステップにエントリするごとに、カウンタｉをカウントアップし、次のステップへ進む。カウントアップはｓまで行う。

【００６７】

ステップＳ１３では、Ｍｏｎｔ（１，R_R）の演算を行い、その結果をy_{i+1}に設定す

る。
【００６８】
ステップＳ１４では、ｅの各ビットをポイントするビットポインタｊを、まずｋに設定する。ここで、ｋはｅのビット長であり予め設定されているとする。そして、このステップにエントリするごとに、ビットポインタｊをカウントダウンし、次のステップへ進む。カウントダウンは１まで行う。
【００６９】
ステップＳ１５では、ビットポインタｊで指定されるｅのビットが「１」であるかどうかチェックする。そして、「１」であればステップＳ１６へ進み、
Ｍｏｎｔ（y_{i+1}，y_i）の演算を行い、その結果をy_{i+1}に設定する。「０」であればステップＳ１７へ進む。
【００７０】
ステップＳ１８では、ビットポインタｊが「１」以上かどうかチェックし、「１」以上でなければ、ステップＳ２８へ進む。「１」以上であればステップＳ１８へ進み、「Ｍｏｎｔ（y_{i+1}，y_{i+1}）」の演算をおこない、その結果をy_{i+1}に設定する。
【００７１】
ステップＳ１９では、ビットポインタｊが「ｋ～１」区間の値であるかチェックし、その区間値であれば、ステップＳ１４からの剰余演算処理へ戻る。その区間値以外であればステップＳ１００へ進む。
【００７２】
ステップＳ１００では、y_{i+1}　（ｉ＝０，１，２，．．．）を記憶装置等に格納する。
【００７３】
ステップＳ１０１では、カウンタｉが「０～ｓ」区間の値であるかチェックし、その区間値であれば、ステップＳ１２からの処理に戻り、次の剰余演算処理を行う。その区間値以外であれば、剰余演算処理を終了する。
【００７４】
図６は、第２の実施例の疑似乱数生成装置９３である。べき乗余剰演算装置９３は演算はアルゴリズム３に従って初期値x_0　，剰余演算の法であるＮ，Ｎと互いにそである任意の定数Ｒ、べきｅから、以下の演算を連鎖的に行ない、y_1，y_2，…を生成する。

$$y_0 = R \cdot x_0 \mod N \quad \text{（式２４）}$$

$$y_{i+1} = R^{-(e-1)} \cdot y_i^{\,e} \mod N \quad (i=0,1,2,\cdots) \quad \text{（式２５）}$$

ここで、Ｎ＝ｐ・ｑ
ただし、ｐ，ｑは素数。
【００７５】
ｅ（≧２）は任意の定数。
べき乗余剰演算装置９０では、アルゴリズム３を実行している。べき乗余剰演算装置９０の入力は初期値x_0，べきｅ，演算の法Ｎ，定数Ｒ，$R_R = R^2 \mod N$，ｉに対する繰り返し演算回数ｓである。べき乗余剰演算装置９０からはy_{i+1}　（ｉ＝０，１，…，ｓ）が順次出力される。べき乗余剰演算装置９０は、入力バッファ９４、判定部９６、モンゴメリ演算器９５、メモリ９８、出力部９７からなる。べき乗余剰演算装置９０の動作手順を次に述べる。
１．入力バッファ９４にはx_0　，ｅ，Ｎ，Ｒ，R_R　，ｓが入力される。入力バッファ９４に入力されたｅ，ｓは判定部９６に入力される。判定部９６では、ｅを［e_k　e_{k-1}　…e_2　e_1　］のｋビットに分解する。更に判定部９６が備えるｉ及びｊに対する２つのカウンタに対してｉ＝０，ｊ＝ｋと設定する。入力バッファ９４に入力されたＲ，Ｎはモンゴメリ演算器９５に設定され、モンゴメリ演算の初期値x_0　，R_R　はメモリ９８に保持される。
【００７６】

２．メモリ９８にあるx_0，R_Rによりモンゴメリ演算器９５においてy_0が算出され、y_1＝Ｒと共にモンゴメリ演算の初期値としてメモリ９８に保持される。

【００７７】

３．ｉ＝０，ｊ＝ｋに対し、判定回路はe_j＝１及びｊ＞１を判定し、判定結果に応じてメモリ９８に対するアドレス信号を出力する。メモリ９８はy_iとy_{i+1}を保持するが、y_{i+1}はモンゴメリ演算器９５の出力結果により随時更新される。メモリ９８は、判定部９６からのアドレス信号に応じて、モンゴメリ演算の出力ｙｉ＋１を記憶し、判定部９６からのアドレス信号に応じて読み出し、モンゴメリ演算器９５へy_{i+1}あるいはy_iを出力する。モンゴメリ演算器９５はメモリ９８からの出力に応じてモンゴメリ演算を行なう。判定部９６のｊに対するカウンタを１カウントダウンさせ、ｊ＝０となるまでこの手順を繰り返す。

【００７８】

４．ｊ＝０となったら、判定部９６は出力部９７に対しイネーブル信号を出し、出力部９７はｊ＝０の時のy_{i+1}をラッチする。出力部９７はラッチしたy_{i+1}をべき乗余剰演算結果として出力し、同時にy_{i+1}をモンゴメリ演算器９５の次の入力としてメモリ９８で保持する。判定部９６のｉに対するカウンタを１カウントアップさせ、ｉ＝ｓとなるまで手順３．、手順４．を繰り返す。

【００７９】

５．終了。

べき乗余剰演算装置９０からの出力は、論理演算装置９１に入力される。論理演算装置９１では、入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$（但し、ｎはＮの２進表示による桁数）ビットの範囲にある任意の（複数／単位）ビットから疑似乱数を生成し、出力する。

【００８０】

論理演算装置９１は、入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$ビットの範囲にある任意のビットを疑似乱数として出力することができる。例えば、下位$\log_2 n$ビット全てを疑似乱数とすることも、最下位ビットだけを疑似乱数として出力することもできる。

【００８１】

論理演算装置として例えば、ハードウエアにより構成する場合には、入力されたy_i（ｉ＝１，２，…）をパラレルでラッチし、その下位$\log_2 n$ビットを順次シリアルに出力するパラレル入力シリアル出力のシフトレジスタを用いることができる。

【００８２】

以上のように第２の実施例によれば、［アルゴリズム２］において従来必要であった式（＊３）の処理を省いた［アルゴリズム３］の計算手順により、式（３）と同程度の安全性を有する疑似乱数列をより高速に、或はより小さな回路規模で生成できる。

【００８３】

以上説明した様に、本実施例の方法とその装置で生成された疑似乱数を用いることで、高速に通信データの生成／再生を行うことができる。

【００８４】

［第３の実施例］

第３の実施例による疑似乱数生成器は、［アルゴリズム２］の式（＊１）、式（＊３）で示された演算を両方とも省くことにより、式（３）と同程度の安全性を有する疑似乱数列をより高速に、或はより小さな回路規模で生成することを可能とすることを目的としている。

【００８５】

第３の実施例の［アルゴリズム４］を以下に示す。

[アルゴリズム４]

```
INPUT  y0 (=x0), e, N, R
FOR  i=0  TO  s
      yi+1=R
      FOR  j=k  TO  1
      IF  ej=1  THEN  yi+1=Mont (yi+1, yi)
      IF  j>1  THEN  yi+1=Mont (yi+1, yi+1)
NEXT
OUTPUT   yi+1  (=R^-(e-1)・yi^e mod N)
NEXT
```

このアルゴリズムを実行することによりえられる系列y_i　（ｉ＝０，１，２，…，ｓ）は、Ｎと互いに素であるＲを用いて、

$y_0 = x_0 \mod N$　（式２６）

$y_{i+1} = R^{-(e-1)} \cdot y_i^{e} \mod N$　（ｉ＝０，１，２，…）　（式２７）

と表わされる。この場合、式（３）で生成される系列x_i　（ｉ＝０，１，２，…）と式（２７）で生成される系列y_i　（ｉ＝０，１，２，…）とを比較すると、

$y_i = R^{**}(1-e^{i}) \cdot x_i \mod N$（ｉ＝０，１，２，…）　（式２８）

となっている。尚、ここで、「Ｒ＊＊（１－e^{i}）」は、Ｒの（１－e^{i}）乗を意味するものとする。

【００８６】

式（＊１）、式（＊３）を省略した場合の本実施例による疑似乱数生成器は、暗号学的に安全な疑似乱数系列として式（２７）のy_i　から、

$a_i' = lsb(y_i)$　（ｉ＝０，１，２，…）　（式２９）

によって得られたa_i'を用いる。

【００８７】

この場合も安全性に関して前述の式（＊３）を省略した場合と同じことが言える。式（２７）で得られる系列y_i　（ｉ＝０，１，２，…）は、式（３）で得られる系列x_i　（ｘ＝０，１，２，…）に定数$R^{**}(1-e^{i+1})$を掛けてＮで剰余をとった値である（式（２８）参照）。

【００８８】

よって、x_{i+1}　から$b_i = lsb(x_i)$を求めることが非常に困難であれば、x_{i+1}　に定数$R^{**}(1-e^{i+1})$を掛けてＮで剰余をとった

y_{i+1}（＝（$R^{**}(1-e^{i+1})$・ｘｉ＋１）　ｍｏｄ　Ｎ）から

$a_i' = lsb(y_i)$（＝ｌｓｂ（（$R^{**}(1-e^{i+1})$・ｘｉ）　ｍｏｄ　Ｎ））を求めることも非常に困難である。つまり、a_i'はy_{i+1}のハードコアビットであるので、式（２７）と式（２９）を用いた疑似乱数生成の安全性も式（３）を用いた疑似乱数生成と同程度である。

【００８９】

次に、図７を用いて、［アルゴリズム３］を実行するための処理フローを説明する。

【００９０】

ステップＳ２００では、y_0、ｅ、Ｎ、ｓの各値を入力する。ここで、ｓは剰余演算の繰り返し数である。

【００９１】

ステップＳ２０１では、剰余演算の繰り返し数のカウンタｉを、まず「０」に初期化する。そして、このステップにエントリするごとに、カウンタｉをカウントアップし、次のステップへ進む。カウントアップはｓまで行う。

【００９２】

ステップＳ２０２では、Ｒをy_{i+1}に設定する。

【００９３】

ステップＳ２０３では、ｅの各ビットをポイントするビットポインタｊを、まずｋに設定する。ここで、ｋはｅのビット長であり予め設定されているとする。そして、このステップにエントリするごとに、ビットポインタｊをカウントダウンし、次のステップへ進む。カウントダウンは１まで行う。

ステップＳ２０４では、ビットポインタｊで指定されるｅのビットが「１」であるかどうかチェックする。そして、「１」であればステップＳ２０５へ進み、Ｍｏｎｔ（y_{i+1}，y_i）の演算を行い、その結果をy_{i+1}に設定する。「０」であればステップＳ２０６へ進む。

【００９４】

ステップＳ２０６では、ビットポインタｊが「１」以上かどうかチェックし、「１」以上でなければ、ステップＳ２０７進む。「１」以上であればステップＳ２０８へ進み、「Ｍｏｎｔ（y_{i+1}，y_{i+1}）」の演算をおこない、その結果をy_{i+1}に設定する。

【００９５】

ステップＳ２０８では、ビットポインタｊが「ｋ～１」区間の値であるかチェックし、その区間値であれば、ステップＳ２０３からの剰余演算処理へ戻る。その区間値以外であればステップＳ２０９へ進む。

【００９６】

ステップＳ２０９では、y_{i+1}　（ｉ＝０，１，２，．．．）を記憶装置等に格納する。

【００９７】

ステップＳ２１０では、カウンタｉが「０～ｓ」区間の値であるかチェックし、その区間値であれば、ステップＳ１２からの処理に戻り、次の剰余演算処理を行う。その区間値以外であれば、剰余演算処理を終了する。

【００９８】

図８は、第３の実施例の疑似乱数生成装置８３の構成を示す図である。疑似乱数生成装置８３は、べき乗剰余演算装置８０と論理演算装置８１を備える。べき乗剰余演算装置８０は演算を［アルゴリズム４］に従って、初期値y_0　，剰余演算の法であるＮ，Ｎと互いに素である任意の定数Ｒ、べきｅ等を入力バッファ８４に入力する。この入力は、例えばキーボード１１を用いて入力する。モンゴメリ演算器８５では、入力バッファ８４から、Ｎ，Ｒ、y_0の各値を入力し、次式に相当する演算を行いy_{i+1}　（ｉ＝０，１，２，．．．）の数列、即ち、ｙ１　，ｙ２　，…を順次求める。

$$y_{i+1} = R^{-(e-1)} \cdot y_i^{e} \mod N \quad (i=0,1,2,\cdots) \qquad (式30)$$

ただし、Ｎ＝ｐ・ｑ

ｐ，ｑは素数

ｅ：　ｅ≧２を満たす任意の定数

尚、ｐ，ｑは予め設定されているとする。また、Ｎ（＝ｐ・ｑ）も予め計算されているとする。モンゴメリ演算器８５での実際の演算方法は、上式を直接計算するのではなく、モンゴメリ法を適用した以下に示す等価式に基づいて演算を行う。

【００９９】

［アルゴリズム４］で用いられているモンゴメリ法による演算Ｍｏｎｔ（ｕ，ｖ）は前述のように、式（７）～式（９）の演算を行なっている。

【０１００】

上述のようにＮは奇数であるので、$R=2^t$　（ｔは任意の整数）とすればＲとＮは互い

に素な整数になる。この場合、Ｒによる乗算及び剰余演算は実質的な演算を必要とせず、Ｍｏｎｔ（ｕ，ｖ）を乗算と加算のみで高速に計算できる。よって、このモンゴメリ法による演算の繰り返しにより実現できるべき乗剰余演算も高速に計算できる。

【０１０１】

べき乗剰余演算では、［アルゴリズム４］を実行している。べき乗剰余演算装置８０の入力バッファ８４に対する入力は初期値y_0（＝x_0），べきｅ，演算の法Ｎ，定数Ｒ，ｉに対する繰り返し演算回数ｓである。べき乗剰余演算装置８０の出力部８７からはy_{i+1}（ｉ＝０，１，…，ｓ）が順次出力される。べき乗剰余演算装置８０は、入力バッファ８４、判定部８６、モンゴメリ演算器８５、メモリ８８、出力部８７を備える。べき乗剰余演算装置８０の動作の手順を述べる。

１．入力バッファにはy_0＝（x_0），ｅ，Ｎ，Ｒ，ｓが入力される。入力手段に入力されたｅ，ｓは判定部８６に入力される。判定部８６では、ｅを［ｅｋ　ｅｋ－１　…　ｅ２　ｅ１］のｋビットに分解する。更に判定部８６が備えるｉ及びｊに対する２つのカウンタに対してｉ＝０，ｊ＝ｋと設定する。入力バッファ８４に入力されたＲ，Ｎはモンゴメリ演算器８５に設定され、モンゴメリ演算の初期値y_0，y_1＝Ｒはメモリ８８に保持される。

【０１０２】

２．ｉ＝０，ｊ＝ｋに対し、判定回路はe_j＝１及びｊ＞１を判定し、判定結果に応じてメモリ８８に対するアドレス信号を出力する。メモリ８８はy_iとy_{i+1}を保持するが、ｙｉ＋１はモンゴメリ演算器８５の出力結果により随時更新される。メモリ８８は判定部８６からのアドレス信号に応じてモンゴメリ演算器８５の出力ｙｉ＋１を記憶し、判定部８６からのアドレス信号に応じて読み出しモンゴメリ演算器８５へｙｉ＋１あるいはｙｉを出力する。モンゴメリ演算器８５はメモリ８８からの出力に応じてモンゴメリ演算を行なう。判定部８６のｊに対するカウンタを１カウンタダウンさせ、ｊ＝０となるまでこの手順を繰り返す。

【０１０３】

３．ｊ＝０となったら判定部８６は出力部８７に対し、イネーブル信号を出し、出力部８７はｊ＝０の時のy_{i+1}をラッチする。出力部８７はラッチしたy_{i+1}をべき乗剰余演算結果として出力し、同時にｙｉ＋１をモンゴメリ演算器８５の次の入力としてメモリ８８で保持する。判定部８６のｉに対するカウンタを１カウントアップさせ、ｉ＝ｓとなるまで手順２．と手順３．を繰り返す。

【０１０４】

４．終了。

上記入力バッファ８４としてはy_0（＝x_0），ｅ，Ｎ，Ｒの各入力値をラッチし保持するレジスタ、判定部８６としてはe_j＝１及びｊ＞１を判定する比較器とｉ及びｊに対するカウンタとアドレス指定信号やイネーブル信号を出力する論理回路で構成できる。メモリ８８は書き込み／読み出しがランダムに行なえるＲＡＭ、出力部８７としては判定部８６からのイネーブル信号に応じてy_{i+1}の出力値をラッチし保持するレジスタ等で構成できる。

【０１０５】

上記モンゴメリ法の演算を行なう手段として、例えばハードウエアにより構成する場合には、加算器と、乗算器と、Ｒによる除算剰余演算のためのビットシフトをシフタ等の基本要素から構成できる。更に、文献、「モンゴメリ法を用いたべき乗アルゴリズムとシストリックアレイ」（岩村，松本，今井；信学技報，ｖｏｌ．９２，Ｎｏ．１３４，ｐｐ．４９－５４，１９９２）に示される公知のモンゴメリ演算回路を用いることもできる。

【０１０６】

べき乗余剰演算装置８０からの出力は、論理演算装置８１に入力される。論理演算装置８１では、入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$（但し、ｎはＮの２進表示による桁数）ビットの範囲にある任意の（複数／単位）ビットから疑似乱数を生成し、出力する。

【０１０７】
論理演算装置８１は、入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$ビットの範囲にある任意のビットを疑似乱数として出力することができる。例えば、下位$\log_2 n$ビット全てを疑似乱数とすることも、最下位ビットだけを疑似乱数として出力することもできる。
【０１０８】
論理演算装置として例えば、ハードウエアにより構成する場合には、入力されたy_i（$i=1, 2, \cdots$）をパラレルでラッチし、その下位$\log_2 n$ビットを順次シリアルに出力するパラレル入力シリアル出力のシフトレジスタを用いることができる。
【０１０９】
以上説明したように、第３の実施例によれば、［アルゴリズム２］において従来必要であった式（＊１），式（＊３）の処理を省いた［アルゴリズム４］の計算手順により、式（３）と同程度の安全性を有する疑似乱数列をより高速に、あるいはより小さな回路規模で生成することを可能にしたものである。
【０１１０】
この場合、モンゴメリ法を用いたことによる乗算剰余演算の高速化に加えて、［アルゴリズム２］に必要であった入力に対する変換と出力を得るための変換が不要になるので、演算全体としての高速化が期待できる。
【０１１１】
以上説明した様に、本実施例の方法とその装置で生成された疑似乱数を用いることで、高速に通信データの生成／再生を行うことができる。
【０１１２】
［第４の実施例］
以上説明して実施例では、モンゴメリ法による疑似乱数生成方法を示したが、一般にべき乗演算結果x^eに任意の定数Ｃを掛けた数$C \cdot x^e$に対し剰余演算を行ない、その結果得られる$y = C \cdot x^e \mod N$の所定ビットから疑似乱数を生成することもできる。
【０１１３】
図９は、第４の実施例による疑似乱数生成装置の構成図７３である。べき乗余剰演算装置７０は初期値x_0，剰余演算の法であるN，べきｅから、次式の演算を連鎖的に行ない、x_1，x_2，…を生成する。

$$x_{i+1} = x_i^e \mod N \quad (i = 0, 1, 2, \cdots) \qquad \text{（式３１）}$$

ここで、$N = p \cdot q$
ただし、ｐ，ｑを素数
ｅ（≧２）は任意の定数。
乗算剰余演算器７０は入力値x_{i+1}（$i=0, 1, 2, \cdots$）、剰余演算の法であるNから、

$$y_{i+1} = C \cdot x_{i+1} \mod N \quad (i = 0, 1, 2, \cdots) \qquad \text{（式３２）}$$

という演算を行ない、ｙ１，ｙ２，…を生成する。
【０１１４】
べき乗剰余演算装置７０では、［アルゴリズム１］を実行している。べき乗剰余演算装置７０の入力は、初期値x_0，べきｅ，演算の法N，ｉに対する繰り返し演算回数ｓである。べき乗剰余演算装置７０からはx_{i+1}（$i=0, 1, \cdots, s$）が順次出力される。べき乗剰余演算装置７０は、入力バッファ７４、判定部７６、乗算剰余演算器７５、メモリ７８、出力部７７からなる。
【０１１５】
次に、べき乗剰余演算装置７０の動作の手順を述べる。
１．入力バッファ７４には、x_0，ｅ，Ｎ，ｓが入力される。入力バッファ７４に入力されたｅ，ｓは判定部７６に入力される。判定部７６ではｅを［e_k　e_{k-1}　…e_2　e_1　］のｋビットに分解する。更に判定部７６が備えるｉ及びｊに対する２つのカウ

ンタに対してｉ＝０，ｊ＝ｋと設定する。入力バッファ７４に入力されたＲ，Ｎは乗算剰余演算器７５に設定され、乗算剰余演算の初期値x_0，x_1＝１はメモリ７８に保持される。

【０１１６】

２．ｉ＝０，ｊ＝ｋに対し、判定部７６は、e_j＝１及びｊ＞１を判定し、判定結果に応じてメモリ７８に対するアドレス信号を出力する。メモリ７８はx_iとx_{i+1}を保持するが、x_{i+1}は乗算剰余演算器７５の出力結果により随時更新される。メモリ７８は判定部７６からのアドレス信号に応じて乗算剰余演算の出力x_{i+1}を記憶し、判定部７６からのアドレス信号に応じて読み出し、乗算剰余演算器７５へ、x_{i+1}あるいはx_iを出力する。乗算剰余演算器７５は、メモリ７８からの出力に応じて乗算剰余演算を行なう。判定部７６のｊに対するカウンタを１カウントダウンさせ、ｊ＝０となるまでこの手順を繰り返す。

【０１１７】

３．ｊ＝０となったら判定部７６は出力部７７に対しイネーブル信号を出し、出力部７７はｊ＝０の時のx_{i+1}をラッチする。出力部７７はラッチしたy_{i+1}をべき乗剰余演算結果として出力し、同時にx_{i+1}を乗算剰余演算器７５の次の入力としてメモリ７８で保持する。判定部７６のｉに対するカウンタを１カウントアップさせ、ｉ＝ｓとなるまで手順２．、手順３．を繰り返す。

【０１１８】

４．終了。

入力バッファ７４としては、ｘ０，ｅ，Ｎ，ｓの各入力値をラッチし保持するレジスタ、判定部７６としてはe_j＝１及びｊ＞１を判定する比較器とｉ及びｊに対するカウンタとアドレス指定信号やイネーブル信号を出力する論理回路で構成できる。メモリ７８は書き込み／読み出しがランダムに行なえるＲＡＭ、出力部７７としては判定部７６からのイネーブル信号に応じてx_{i+1}の出力値をラッチし保持するレジスタ等で容易に構成できる。

【０１１９】

前述のように、べき乗剰余演算は乗算剰余演算の繰り返しで実現できる。べき乗剰余演算装置７０及び乗算剰余器での乗算剰余演算を、例えばハードウエアにより構成する場合には、文献「並列処理によるＲＳＡ暗号装置の構成ｂ法」（岩村，松本，今井；電子情報通信学会論文誌Ａ，ｖｏｌ．Ｊ７５－Ａ，Ｎｏ．８，ｐｐ．１３０１－１３１１，１９９２）に示される公知の乗算剰余演算方法を用いることもできる。

【０１２０】

乗算剰余演算器７２からの出力は、論理演算装置７１に入力される。論理演算装置７１では入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$（但し、ｎはＮの２進表示による桁数）ビットの範囲にある任意の（複数／単数）ビットから疑似乱数を生成し、出力する。

【０１２１】

論理演算装置７１は、入力されたy_1，y_2，…のそれぞれ下位$\log_2 n$ビットの範囲にある任意のビットを疑似乱数として出力することができる。例えば、下位$\log_2 n$ビット全てを疑似乱数とすることも、最下位ビットだけを疑似乱数として出力することもできる。

【０１２２】

以上説明した様に、本実施例の方法とその装置で生成された疑似乱数を用いることで、高速に通信データの生成／再生を行うことができる。

【０１２３】

［第５の実施例］

これまで述べてきたように、上記の疑似乱数生成方法によって生成された疑似乱数は解析に対して強いので、この疑似乱数を暗号化方式に用いることにより安全な暗号通信が実現できる。以下、通信文と乱数との間でビット毎に排他的論理和をとる暗号化方式（ストリ

ーム暗号）による暗号通信ネットワークにおいて、以上説明してきた実施例の乱数生成器を用いた暗号通信での応用例を以下説明する。

【０１２４】

図１０は、ネットワークの加入者間で固有かつ秘密の暗号鍵を共有している共通鍵暗号通信ネットワーク３０を示す。Ａ，Ｂ，Ｃ，…，Ｎはそのネットワークの各加入者である。通信ネットワーク３４は、ネットワークの加入者Ａ，Ｂ，Ｃ，…，Ｎ３１間の通信を可能とするネットワークである。各加入者Ａ，Ｂ，Ｃ，…，Ｎの下の各サークル内の各符号K_{AB}，K_{AC}，…は、それぞれの加入者が共有している暗号鍵を意味する。例えば、各符号K_{AB}，K_{AC}，…は、それぞれ加入者Ａ－Ｂ間、加入者Ａ－Ｃ間で共有している暗号鍵，…を示している。

【０１２５】

図１１は、本実施例の乱数生成器を用いた暗号装置及び複合装置を含む通信装置の構成をブロック図により示したものである。

【０１２６】

図１２は、図１０、図１１で示された暗号通信システムにおけるＡ、Ｂ間の秘匿通信の様子を示している。

【０１２７】

図１２において、送信者Ａが使用する送信器４５から受信者Ｂの利用する受信器４６への暗号通信は以下の手順で行う。

【０１２８】

１．通信の送信者Ａは、送信者Ｂと共有している秘密の鍵K_{AB}の全て又は一部を乱数生成器４０の初期値として、乱数生成器４０に設定し、乱数系列k_i（４１）を生成させる。

【０１２９】

２．送信器Ａで生成した乱数系列k_i（４１）と、予め作られた通信文m_i（４２）をビット毎に排他的論理和「m_i（＋）k_i」を排他的論理和器４３で計算し、その結果の暗号文c_iを受信器４６に送信する。

【０１３０】

３．通信の受信者Ｂは、送信元Ａと共有している秘密の鍵K_{AB}の全て又は一部を受信器Ｂの乱数生成器４７の初期値として乱数生成器４７に設定し、乱数系列k_i　を生成させる。

【０１３１】

４．受信器Ｂでは、生成した乱数系列k_iと受信した暗号文c_iをビット毎に排他的論理和「c_i（＋）k_i」をとり、その出力が通信文m_i（４８）として復元される。

【０１３２】

この手順に従えば、正規の受信者Ｂだけがその秘密の鍵K_{AB}を知っているので受け取った暗号文を本来の通信文に複合でき、それ以外の加入者（Ｃ～Ｎ）はその暗号文をする際に用いられた秘密の鍵を知らないのでその内容を知ることができない。このことにより秘匿通信が実現される。

【０１３３】

また、図１０のようにあらかじめ暗号鍵が配布されているのではなく、暗号通信を行うに先立って送・受信者間で暗号鍵を共有する必要がある携帯のネットワークにおいても、公知の鍵共有を行えば同じ手順で暗号通信を実現することができる。

【０１３４】

尚、第５の実施例に示した暗号通信ネットワークでは通信文の送信者と受信者の間で固有かつ秘密の鍵を共有しているので、暗号文を受け取り、意味をなす通信文に複号できるということは、通信文がその鍵のもう一人の所有者から送信されたということを受診者に保証している。そのため、第３の実施例に示した秘匿通信システムでは、通信の発信者及び着信者の認証も行うことができる。

【０１３５】

以上説明した様に、本実施例の方法とその装置で生成された疑似乱数を用いることで、高速に通信データの生成／再生を行う暗号通信ネットワークを実現できる。

【０１３６】

［第６の実施例］

第５の実施例のようにあらかじめ暗号鍵が配布されているのではなく、暗号通信を行うに先立って送・受信者間で暗号鍵を共有する必要がある形態のネットワークにおいて、盗聴の可能性のある通信路を介した場合でも安全に暗号鍵を共有できる方式としてＤｉｆｆｉｅ－Ｈｅｌｌｍａｎの方式（Ｗ．Ｄｏｆｆｉｅ ａｎｄ Ｍ．Ｅ．Ｈｅｌｌｍａｎ “Ｄｉｒｅｃｔｉｏｎ ｉｎ ｃｒｙｔｏｇｒａｐｈｙ ”，ＩＥＥＥ，ＩＴ，ｖｏｌ．ＩＴ－２２，Ｎｏ．６，１９７６ ）が良く知られている。その際に用いる乱数として本実施例により生成した乱数を用いることができる

。

【０１３７】

その場合に用いる乱数は、送信者と着信者で同じものを持つ必要はないため、乱数発生器に設定する初期値は任意の値を用いればよい。

【０１３８】

尚、本発明は、複数の機器から構成されるシステムに適用しても１つの機器から成る装置に適用しても良い。また、本発明は、システム或は装置にプログラムを供給することによって達成される場合にも適用できることはいうまでもない。

【０１３９】

以上、詳細に説明した実施例によって、暗号学的に安全な疑似乱数をモンゴメリ法により生成する場合、式（１５あるいは式（２５により得られるｙｉ の所定ビットを疑似乱数として用いることにより、安全性を低下させることなく、従来必要であった式（＊３）あるいは式（＊１），式（＊３）の演算を不要にしたものである。そのことにより、従来と同程度の安全性を有する疑似乱数をより高速に、あるいはより小さな回路規模で生成することが可能である。そのため、その生成された疑似乱数を用いることで、高速に通信データの生成／再生を行うことができる。

【０１４０】

【発明の効果】

以上説明したように本発明によれば、安全な疑似乱数をより高速にかつ容易に生成することができる。またそのようにして生成された疑似乱数を用いて通信データの暗号化／復号を高速に行うことができる。

【０１４１】

【図面の簡単な説明】

【図１】第１の実施例による疑似乱数生成装置の構成を示す図である。

【図２】本発明の疑似乱数生成処理を実行するデータ処理装置の図である。

【図３】第１の実施例による疑似乱数生成プログラムのフローチャートである。

【図４】モンゴメリ法以外の方法による疑似乱数発生方法の一例を示す図である。

【図５】第２の実施例による疑似乱数生成プログラムのフローチャートである。

【図６】第２の実施例による疑似乱数生成装置の構成を示す図である。

【図７】第３の実施例による疑似乱数生成のフローチャートである。

【図８】第３の実施例による疑似乱数生成装置の構成を示す図である。

【図９】第４の実施例による疑似乱数生成装置の構成を示す図である。

【図１０】第５の実施例による共通鍵暗号通信ネットワークを示す図である。

【図１１】第５の実施例による通信装置の構成図である。

【図１２】第５の実施例による暗号通信システムでの秘匿通信の様子を示す図である。

【符号の説明】

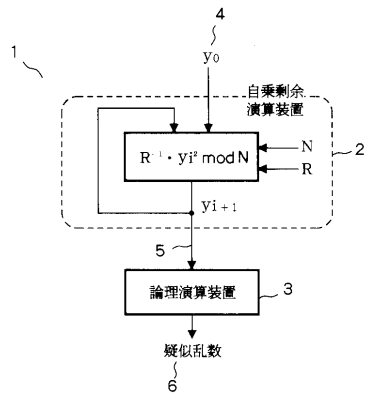
１０　ＣＰＵ

１１　キーボード

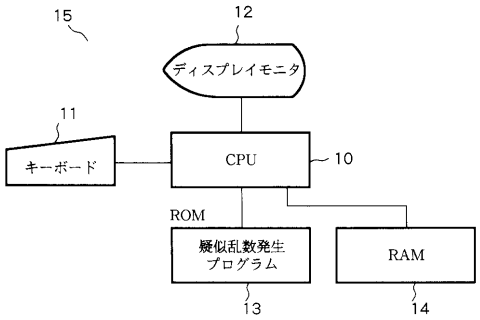
１２　デイスプレイモニタ

１３　疑似乱数発生プログラム
１４　ＲＡＭ

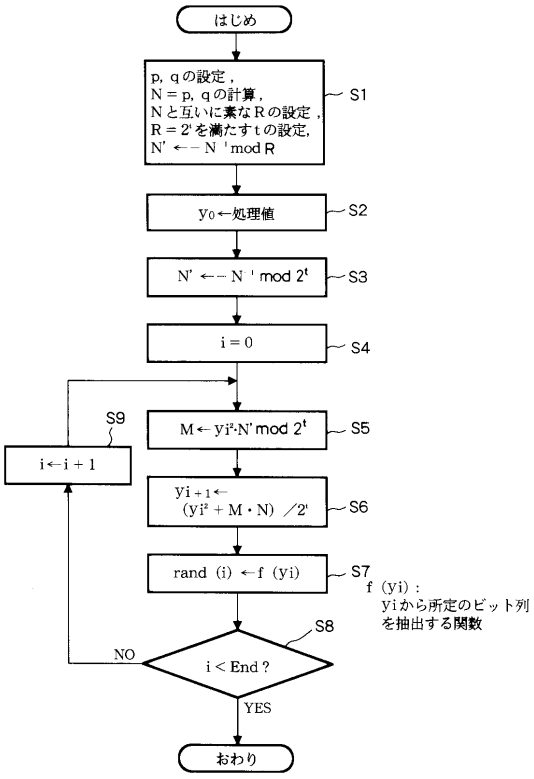
【図１】



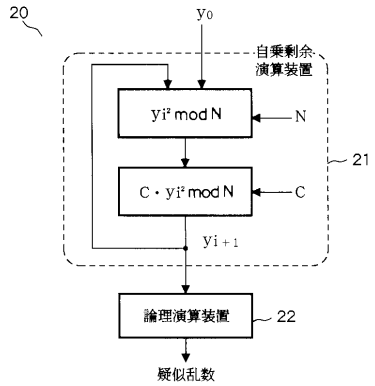
【図２】



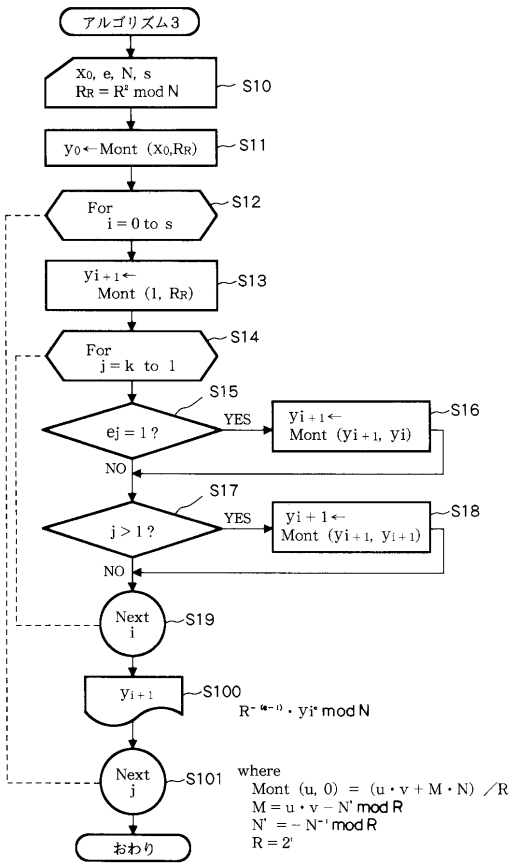
【図３】



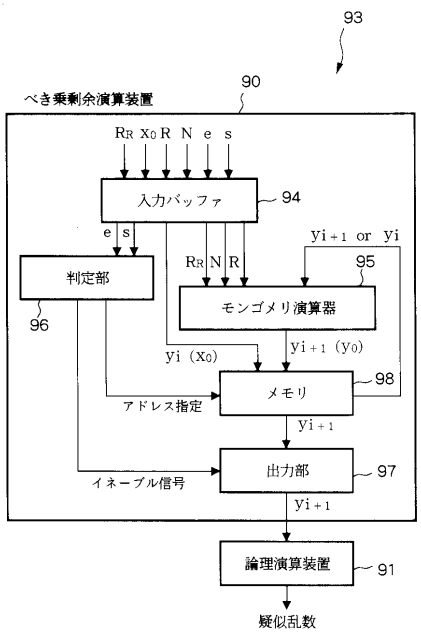
【図４】


20
y0
自乗剰余演算装置
yi² mod N
N
C・yi² mod N
C
21
yi+1
論理演算装置
22
疑似乱数

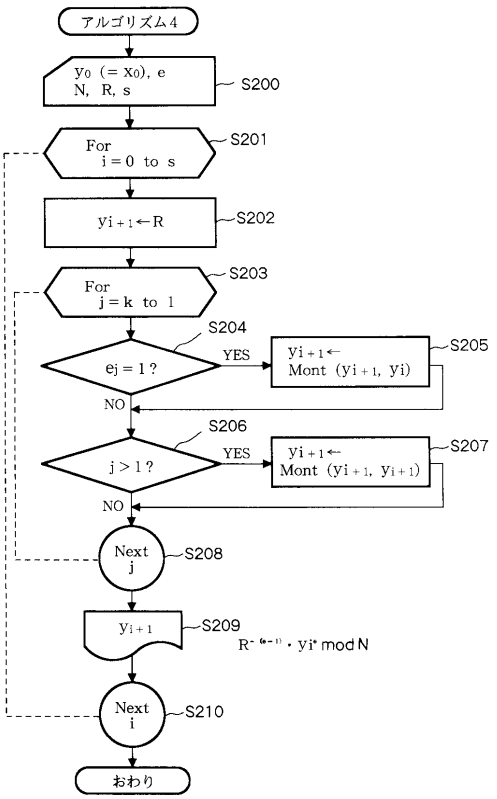
【図５】


アルゴリズム3
x0, e, N, s
RR = R² mod N
S10
y0←Mont (x0,RR)
S11
For i = 0 to s
S12
yi+1← Mont (1, RR)
S13
For j = k to 1
S14
ej = 1 ?
S15
YES
yi+1← Mont (yi+1, yi)
S16
NO
j > 1 ?
S17
YES
yi+1← Mont (yi+1, yi+1)
S18
NO
Next i
S19
yi+1
S100
R^-(e-1) · yi^e mod N
Next j
S101
where
Mont (u, 0) = (u・v + M・N) ／R
M = u・v − N' mod R
N' = − N^-1 mod R
R = 2^i
おわり

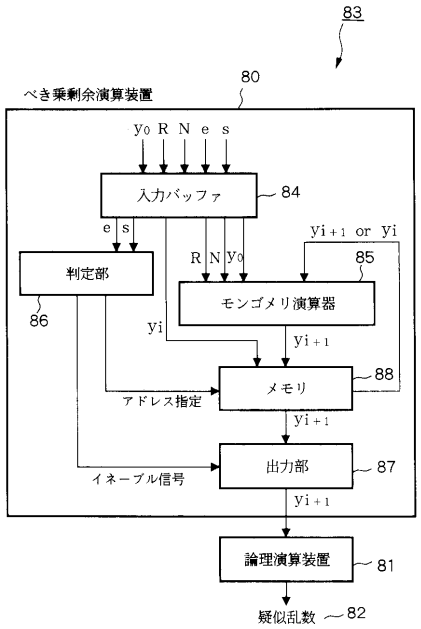
【図６】


93
90
べき乗剰余演算装置
RR x0 R N e s
入力バッファ
94
e s
RR N R
yi+1 or yi
95
判定部
96
モンゴメリ演算器
yi (x0)
yi+1 (y0)
メモリ
98
アドレス指定
yi+1
出力部
97
イネーブル信号
yi+1
論理演算装置
91
疑似乱数

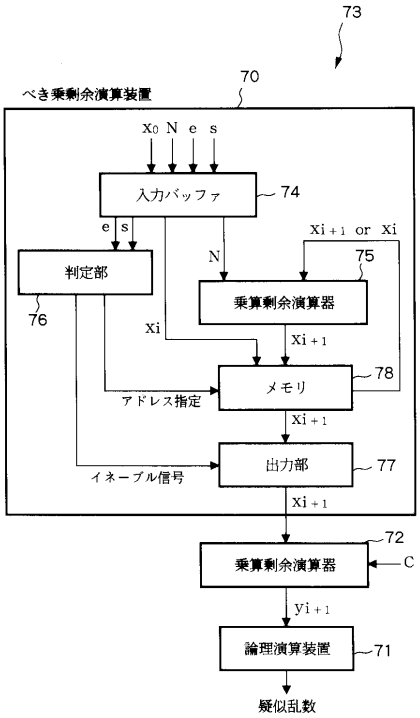
【図７】


アルゴリズム4
y0 (= x0), e
N, R, s
S200
For i = 0 to s
S201
yi+1←R
S202
For j = k to 1
S203
ej = 1 ?
S204
YES
yi+1← Mont (yi+1, yi)
S205
NO
j > 1 ?
S206
YES
yi+1← Mont (yi+1, yi+1)
S207
NO
Next j
S208
yi+1
S209
R^-(e-1) · yi^e mod N
Next i
S210
おわり

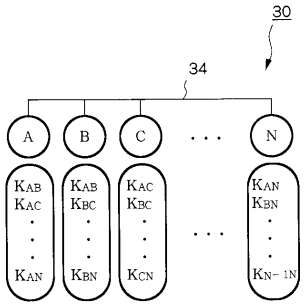
【図８】



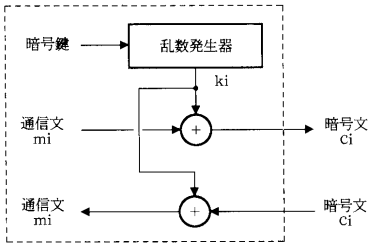
【図９】



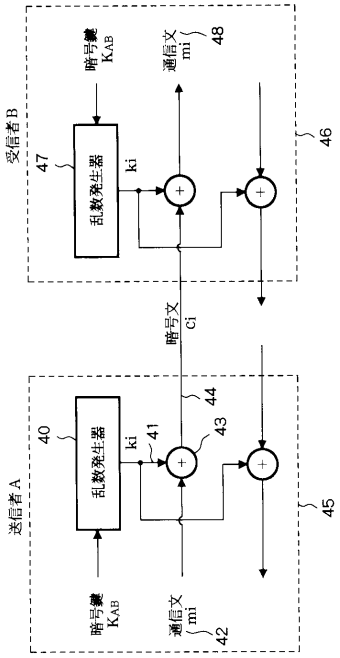
【図１０】



【図１１】



【図１２】



フロントページの続き

(72)発明者　岩村　恵市
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　青木　重徳

(56)参考文献　特開平５－３２４２７７（ＪＰ，Ａ）
山本貴久，岩村恵市，松本勉，今井秀樹，“２乗型疑似乱数生成器とブロック暗号を用いた実用的暗号方式”，電子情報通信学会技術研究報告（ＩＳＥＣ９３－２９），日本，社団法人電子情報通信学会，１９９３年　8月30日，Ｖｏｌ．９３，Ｎｏ．２０８，ｐ．６５－７５
岩村恵市，松本勉，今井秀樹，“べき乗剰余に適したモンゴメリ法による剰余乗算法とそれを実現するシストリックアレー”，電子情報通信学会論文誌，日本，社団法人電子情報通信学会，１９９３年　8月２５日，Ｖｏｌ．Ｊ７６－Ａ，Ｎｏ．８，ｐｐ．１２１４－１２２３

(58)調査した分野(Int.Cl.^{7}，ＤＢ名)
G09C　1/00　650
G06F　7/58