

明 細 書

発明の名称：

暗号処理装置、および暗号処理方法、並びにプログラム

技術分野

[0001] 本開示は、暗号処理装置、および暗号処理方法、並びにプログラムに関する。さらに詳細には、共通鍵系暗号を実行する暗号処理装置、および暗号処理方法、並びにプログラムに関する。

背景技術

[0002] 情報化社会が発展すると共に、扱う情報を安全に守るための情報セキュリティ技術の重要性が増してきている。情報セキュリティ技術の構成要素の一つとして暗号技術があり、現在では様々な製品やシステムで暗号技術が利用されている。

[0003] 暗号処理アルゴリズムには様々なものがあるが、基本的な技術の一つとして、共通鍵ブロック暗号と呼ばれるものがある。共通鍵ブロック暗号では、暗号化用の鍵と復号用の鍵が共通のものとなっている。暗号化処理、復号処理共に、その共通鍵から複数の鍵を生成し、あるブロック単位、例えば64ビット、128ビット、256ビット等のブロックデータ単位でデータ変換処理を繰り返し実行する。

[0004] 代表的な共通鍵ブロック暗号のアルゴリズムとしては、過去の米国標準であるDES (Data Encryption Standard) や現在の米国標準であるAES (Advanced Encryption Standard) が知られている。他にも様々な共通鍵ブロック暗号が現在も提案され続けており、2007年にソニー株式会社が提案したCLEFIAも共通鍵ブロック暗号の一つである。

[0005] このような、共通鍵ブロック暗号のアルゴリズムは、主として、入力データの変換を繰り返し実行するラウンド関数実行部を有する暗号処理部と、ラウンド関数部の各ラウンドで適用するラウンド鍵を生成する鍵スケジュール

部とによって構成される。鍵スケジュール部は、秘密鍵であるマスター鍵（主鍵）に基づいて、まずビット数を増加させた拡大鍵を生成し、生成した拡大鍵に基づいて、暗号処理部の各ラウンド関数部で適用するラウンド鍵（副鍵）を生成する。

[0006] このようなアルゴリズムを実行する具体的な構造として、線形変換部および非線形変換部を有するラウンド関数を繰り返し実行する構造が知られている。例えば代表的な構造にFeistel構造や一般化Feistel構造がある。Feistel構造や一般化Feistel構造は、データ変換関数としてのF関数を含むラウンド関数の単純な繰り返しにより、平文を暗号文に変換する構造を持つ。F関数においては、線形変換処理および非線形変換処理が実行される。なお、Feistel構造を適用した暗号処理について記載した文献としては、例えば非特許文献1、非特許文献2がある。

[0007] ブロック暗号の安全性を評価する指標のひとつとしてdiffusion（拡散）特性と呼ばれるものがある。この特性は入力データの変化を出力データに波及させる（拡散させる）特性と考えることができ、安全なブロック暗号にはこのような入力データの変化の影響をできるだけ速く出力データに伝えることが求められる。

[0008] 拡散（diffusion）特性を向上させるためには、例えば、ラウンド関数の繰り返し回数を増加させるといったことが効果的であることは予測される。しかし、より少ないラウンド関数の繰り返し回数でいかに拡散（diffusion）特性を向上させるかについての技術を開示した従来技術はない。

先行技術文献

非特許文献

[0009] 非特許文献1：K. Nyberg, "Generalized Feistel networks", ASIACRYPT 96, Springer Verlag, 1996, pp. 91--104.

非特許文献2：Yuliang Zheng, Tsutomu Matsu

moto, Hideki Imai: On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses. CRYPTO 1989: 461-480

発明の概要

発明が解決しようとする課題

- [0010] 本開示は、例えば上述の状況に鑑みてなされたものであり、拡散（diffusion）特性を向上させた安全性の高い暗号処理装置、および暗号処理方法、並びにプログラム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

- [0011] 本開示の第1の側面は、
- データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、
- 前記暗号処理部は、
- 入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する構成であり、
- ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する暗号処理装置にある。
- [0012] さらに、本開示の暗号処理装置の一実施態様において、前記ラウンド関数は、ラウンド鍵を適用した演算と、非線形変換処理と、線形変換処理を含む F 関数、および F 関数の出力または入力に対する他ラインのデータとの排他的論理和演算を含む。
- [0013] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、

(1) F関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する、

(2) 排他的論理和側データ系列は必ず次のラウンド関数のF関数入力側データ系列に分配する、

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する、

上記(1)～(3)の分配条件を満たす処理によって前段のラウンド演算の演算結果を次段のラウンド演算の入力として設定する。

[0014] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、入力データの分割数 d を4以上とした一般化Feistel構造を有する。

[0015] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、ラウンド演算の出力データを有する d 個のライン各々の n/d ビットデータを $d/2$ 個に再分割して、 $d \times (n/d)$ 個の再分割データを生成し、分割数 d に応じた d 本のラインの異なるラインから選択した $d/2$ 個の再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する。

[0016] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、全出力ビットが以下に示す2つの条件を満足する拡散(diffusion)状態、すなわち、出力ビットを入力ビットの関係式として記述した場合に、

(条件1) 全入力ビットが関係式に含まれる

(条件2) 全入力ビットが少なくとも1度は前記ラウンド関数を通過している

上記2つの条件を満足する全拡散(full diffusion)状態を実現する構成である。

[0017] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は

、前記全拡散 (full diffusion) 状態を4ラウンドのラウンド演算によって実現する。

[0018] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部の前段のラウンド演算の出力データと次段のラウンド演算の再分割データの入出力関係を決定する接続構成は、ラウンド演算の出力データを有するラインの n/d ビットデータの再分割処理によって生成される $d \times (n/d)$ 個の再分割データの組み合わせデータである ($d/2$) 個の $2n/d$ ビットのデータセットを単位とした接続構成から選択される接続構成である。

[0019] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、暗号化処理と復号処理の双方に適用可能なインボリューション性を有する。

[0020] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部の各ラウンド演算の再分割データの再構成処理は、前段のラウンド関数入力側系列の再分割データを、予め決定した規則に従って次段の排他的論理和側系列に分配し、前段の排他的論理和側系列の再分割データを、予め決定した規則に従って次段のラウンド関数入力側系列に分配する構成である。

[0021] さらに、本開示の暗号処理装置の一実施態様において、前記暗号処理部は、入力データとしての平文を暗号文に変換する暗号化処理、または、入力データとしての暗号文を平文に変換する復号処理を実行する。

[0022] さらに、本開示の第2の側面は、

暗号処理装置において実行する暗号処理方法であり、

暗号処理部が、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理ステップを実行し、

前記暗号処理ステップは、

入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行し、

ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$

個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する暗号処理方法にある。

[0023] さらに、本開示の第3の側面は、

暗号処理装置において暗号処理を実行させるプログラムであり、

暗号処理部に、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理ステップを実行させ、

前記暗号処理ステップにおいて、

入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行させ、

ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行させるプログラムにある。

[0024] なお、本開示のプログラムは、例えば、様々なプログラム・コードを実行可能な情報処理装置やコンピュータ・システムに対して例えば記憶媒体によって提供されるプログラムである。このようなプログラムを情報処理装置やコンピュータ・システム上のプログラム実行部で実行することでプログラムに応じた処理が実現される。

[0025] 本開示のさらに他の目的、特徴や利点は、後述する本発明の実施例や添付する図面に基づくより詳細な説明によって明らかになるであろう。なお、本明細書においてシステムとは、複数の装置の論理的集合構成であり、各構成の装置が同一筐体内にあるものには限らない。

発明の効果

[0026] 本開示の一実施例によれば、拡散 (d i f f u s i o n) 特性を向上させた安全性の高い暗号処理が実現される。

具体的には、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、暗号処理部において、入力データである n ビットデータを分割数 d で分割した n/d ビットデータを各ラインに入力して、ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する。ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に分割し、該分割データを組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして処理を実行する。この構成によって、拡散 (diffusion) 特性を向上させた安全性の高い暗号処理が実現される。

図面の簡単な説明

- [0027] [図1] k ビットの鍵長に対応した n ビット共通鍵ブロック暗号アルゴリズムを説明する図である。
- [図2] 図1に示す k ビットの鍵長に対応した n ビット共通鍵ブロック暗号アルゴリズムに対応する復号アルゴリズムを説明する図である。
- [図3] 鍵スケジュール部とデータ暗号化部の関係について説明する図である。
- [図4] データ暗号化部の構成例について説明する図である。
- [図5] S P N構造のラウンド関数の例について説明する図である。
- [図6] F e i s t e l 構造のラウンド関数の一例について説明する図である。
- [図7] 拡張 F e i s t e l 構造の一例について説明する図である。
- [図8] 拡張 F e i s t e l 構造の一例について説明する図である。
- [図9] 非線形変換部の構成例について説明する図である。
- [図10] 線形変換処理部の構成例について説明する図である。
- [図11] F e i s t e l 構造を持つブロック暗号における拡散 (diffusion) 状態を説明する図である。
- [図12] F e i s t e l 構造を持つブロック暗号における拡散 (diffusion) 状態を説明する図である。

[図13] 4-lineで1段に2つのF関数を使用する一般化Feistel構造を持つブロック暗号における拡散 (diffusion) 状態を説明する図である。

[図14] 4-lineで1段に2つのF関数を使用する一般化Feistel構造を持つブロック暗号における拡散 (diffusion) 状態を説明する図である。

[図15] 4-lineで1段に2つのF関数を使用する一般化Feistel構造を持つブロック暗号における拡散 (diffusion) 状態を説明する図である。

[図16] $d = 6$ とした場合の通常の一般化Feistel構造を説明する図である。

[図17] $d = 6$ とした場合の通常の一般化Feistel構造の全拡散 (full diffusion) 状態となるパスの一例を示す図である。

[図18] ラウンド間の配線を変更することで分割数 d が6以上の場合に、従来の構成より全拡散 (full diffusion) ラウンド数を減少させた構成例を説明する図である。

[図19] ラウンド間の配線を変更することで分割数 d が6以上の場合に、従来の構成より全拡散 (full diffusion) ラウンド数を減少させた構成において全拡散 (full diffusion) 状態となるパスの一例を示す図である。

[図20] 本開示の一実施例としての暗号処理構成について説明する図である。

[図21] 本開示の一実施例である $d = 4$ とした場合の例について説明する図である。

[図22] $d = 4$ の場合の本方式で全拡散 (full diffusion) 状態となるパスの一例を示す図である。

[図23] 本開示の一実施例である $d = 6$ とした場合の例について説明する図である。

[図24] $d = 6$ の場合の本方式で全拡散 (full diffusion) 状

態となるパスの一例を示す図である。

[図25] $d = 6$ の場合に各 n / d ビットデータを2分割した構成例を示す図である。

[図26] n ビット入力データの分割数 d , および各 n / d ビットデータの分割数 p における各全拡散 (full diffusion) ラウンド数の関係の一部を示す図である。

[図27] F 関数入力側データ系列と排他的論理和側データ系列とでそれぞれ異なる分割を行った構成例を示す図である。

[図28] 図21に記載の構成例で拡大鍵の挿入位置を変更した構成例を示す図である。

[図29] データ分配法のより効率的な手法について説明する図である。

[図30] データ分配法のより効率的な手法について説明する図である。

[図31] データ分配法のより効率的な手法について説明する図である。

[図32] $F e i s t e l$ 構造のインボリューション性について説明する図である。

[図33] $F e i s t e l$ 構造のインボリューション性について説明する図である。

[図34] 高い拡散 (diffusion) 特性を得られる $4 - l i n e$ 構造の一方式を示す図である。

[図35] インボリューション性を実現する処理について説明する図である。

[図36] $4 - l i n e$ ($d = 4$) の構成において、インボリューション性を有する構成の一例について説明する図である。

[図37] $6 - l i n e$ ($d = 6$) の構成において、インボリューション性を有する構成の一例について説明する図である。

[図38] インボリューション性を有する構成の一例について説明する図である。

[図39] インボリューション性を有する構成の一例について説明する図である。

[図40] 暗号処理装置としてのICモジュール700の構成例を示す図である。

発明を実施するための形態

[0028] 以下、図面を参照しながら本開示に係る暗号処理装置、および暗号処理方法、並びにプログラムの詳細について説明する。説明は、以下の項目に従って行う。

1. 共通鍵ブロック暗号の概要

2. 拡散 (d i f f u s i o n) 特性の概要について

(2-1) 拡散 (d i f f u s i o n) 特性についての説明

(2-2) 拡散 (d i f f u s i o n) 特性を考慮したこれまでの構成例について

3. 本開示に従った拡散 (d i f f u s i o n) 特性を向上させた構成例について

4. より効率的なデータ分配法を持つ構成例について

5. インボリューション (i n v o l u t i o n) 性を備えた構成例について

6. 暗号処理装置の構成例について

7. 本開示の構成のまとめ

[0029] [1. 共通鍵ブロック暗号の概要]

まず、共通鍵ブロック暗号の概要について説明する。

(1-1. 共通鍵ブロック暗号)

ここでは共通鍵ブロック暗号（以下ではブロック暗号）としては以下に定義するものを指すものとする。

ブロック暗号は入力として平文Pと鍵Kを取り、暗号文Cを出力する。平文と暗号文のビット長をブロックサイズと呼びここではnで著す。nは任意の整数値を取りうるが、通常、ブロック暗号アルゴリズムごとに、あらかじめひとつに決められている値である。ブロック長がnのブロック暗号のことをnビットブロック暗号と呼ぶこともある。

[0030] 鍵のビット長をkで表す。鍵は任意の整数値を取りうる。共通鍵ブロック暗号アルゴリズムは1つまたは複数の鍵サイズに対応することになる。例え

ば、あるブロック暗号アルゴリズム A はブロックサイズ $n = 128$ であり、 $k = 128$ または $k = 192$ または $k = 256$ の鍵サイズに対応するという構成もありうるものとする。

平文 P : n ビット

暗号文 C : n ビット

鍵 K : k ビット

[0031] 図 1 に k ビットの鍵長に対応した n ビット共通鍵ブロック暗号アルゴリズム E の図を示す。

暗号化アルゴリズム E に対応する復号アルゴリズム D は暗号化アルゴリズム E の逆関数 E^{-1} と定義でき、入力として暗号文 C と鍵 K を受け取り、平文 P を出力する。図 2 に図 1 に示した暗号アルゴリズム E に対応する復号アルゴリズム D の図を示す。

[0032] (1-2. 内部構成)

ブロック暗号は 2 つの部分に分けて考えることができる。ひとつは鍵 K を入力とし、

ある定められたステップによりビット長を拡大してできた拡大鍵 K' (ビット長 k') を出力する「鍵スケジュール部」と、もうひとつは平文 P と鍵スケジュール部から拡大された鍵 K' を受け取ってデータの変換を行い暗号文 C を出力する「データ暗号化部」である。

2 つの部分の関係は図 3 に示される。

[0033] (1-3. データ暗号化部)

以下の実施例において用いるデータ暗号化部はラウンド関数という処理単位に分割できるものとする。ラウンド関数は入力としての 2 つのデータを受け取り、内部で処理を施したのち、1 つのデータを出力する。入力データの一方は暗号化途中の n ビットデータであり、あるラウンドにおけるラウンド関数の出力が次のラウンドの入力として供給される構成となる。もう 1 つの入力データは鍵スケジュールから出力された拡大鍵の一部のデータが用いられ、この鍵データのことをラウンド鍵と呼ぶものとする。またラウンド関数

の総数は総ラウンド数と呼ばれ、暗号アルゴリズムごとにあらかじめ定められている値である。ここでは総ラウンド数を R で表す。

[0034] データ暗号化部の入力側から見て1ラウンド目の入力データを X_1 とし、 i 番目のラウンド関数に入力されるデータを X_i 、ラウンド鍵を RK_i とすると、データ暗号化部全体は図4のように示される。

[0035] (1-4. ラウンド関数)

ブロック暗号アルゴリズムによってラウンド関数はさまざまな形態をとる。ラウンド関数はその暗号アルゴリズムが採用する構造 (structure) によって分類できる。代表的な構造としてここではSPN構造、Feistel構造、拡張Feistel構造を例示する。

[0036] (ア) SPN構造ラウンド関数

n ビットの入力データすべてに対して、ラウンド鍵との排他的論理和演算、非線形変換、線形変換処理などが適用される構成。各演算の順番は特に決まっていない。図5にSPN構造のラウンド関数の例を示す。

[0037] (イ) Feistel構造

n ビットの入力データは $n/2$ ビットの2つのデータに分割される。うち片方のデータとラウンド鍵を入力として持つ関数 (F関数) が適用され、出力がもう片方のデータに排他的論理和される。そののちデータの左右を入れ替えたものを出力データとする。F関数の内部構成にもさまざまなタイプのものがあるが、基本的にはSPN構造同様にラウンド鍵データとの排他的論理和演算、非線形演算、線形変換の組み合わせで実現される。図6にFeistel構造のラウンド関数の一例を示す。

[0038] (ウ) 拡張Feistel構造

拡張Feistel構造はFeistel構造ではデータ分割数が2であったものを、3以上に分割する形に拡張したものである。分割数を d とすると、 d によってさまざまな拡張Feistel構造を定義することができる。F関数の入出力のサイズが相対的に小さくなるため、小型実装に向いているとされる。図7に $d=4$ でかつ、ひとつのラウンド内に2つのF関数が並

列に適用される場合の拡張 *F e i s t e l* 構造の一例を示す。また、図 8 に $d = 8$ でかつ、ひとつのラウンド内に 1 つの *F* 関数が適用される場合の拡張 *F e i s t e l* 構造の一例を示す。

[0039] (1-5. 非線形変換処理部)

非線形変換処理部は、入力されるデータのサイズが大きくなると実装上のコストが高くなる傾向がある。それを回避するために対象データを複数の単位に分割し、それぞれに対して非線形変換を施す構成がとられることが多い。例えば入力サイズを m s ビットとして、それらを s ビットずつの m 個のデータに分割して、それぞれに対して s ビット入出力を持つ非線形変換を行う構成である。それらの s ビット単位の非線形変換を *S - b o x* と呼ぶものとする。図 9 に例を示す。

[0040] (1-6. 線形変換処理部)

線形変換処理部はその性質上、行列として定義することが可能である。行列の要素は $GF(2^8)$ の体の要素や $GF(2)$ の要素など、一般的にはさまざまな表現ができる。図 10 に m s ビット入出力をもち、 $GF(2^s)$ の上で定義される $m \times m$ の行列により定義される線形変換処理部の例を示す。

[0041] [2. 拡散 (*d i f f u s i o n*) 特性の概要について]

本開示の暗号処理の説明の前に拡散 (*d i f f u s i o n*) 特性の概要について説明する。

[0042] (2-1) 拡散 (*d i f f u s i o n*) 特性についての説明

先に簡単に説明したように、ブロック暗号の安全性を評価する指標のひとつとして *d i f f u s i o n* (拡散) 特性と呼ばれるものがある。この特性は入力データの変化を出力データに波及させる (拡散させる) 特性と考えることができ、安全なブロック暗号にはこのような入力データの変化の影響をできるだけ速く出力データに伝えることが求められる。

[0043] 以下に、「*d i f f u s i o n* 状態」, 「*full d i f f u s i o n* 状態」, 「*full d i f f u s i o n* ラウンド数」を定義する。

ある出力ビットを入力ビットの関係式として記述して以下の条件を満たす

場合、その出力ビットは「diffusion状態」になっていると定義する。

(条件1) 全入力ビットが関係式に含まれる。

(条件2) 全入力ビットが少なくとも1度はラウンド関数(F関数)を通過している。

[0044] さらに、

全出力ビットが拡散(diffusion)状態になっていることを「全拡散(full diffusion)状態」と定義する。

この全拡散(full diffusion)状態を満たすために最低限必要なラウンド数(繰り返し数)を「全拡散(full diffusion)ラウンド数」と定義する。

[0045] これらの定義について、具体的な例として図11、図12に記載のFeistel構造を持つブロック暗号を用いて詳しく説明を行う。図11はFeistel構造を持つブロック暗号のラウンド関数の構成例を表し、図12はそのラウンド関数を3ラウンド繰り返す構造を表している。

[0046] 図12より第*i*、*i*+1、*i*+2ラウンド出力の左側(第*i*+1、*i*+2、*i*+3入力の左側)の*n*/2ビットデータ X_{i+1}^1 、 X_{i+2}^1 、 X_{i+3}^1 、および第*i*、*i*+1、*i*+2ラウンド出力の右側(第*i*+1、*i*+2、*i*+3入力の右側)の*n*/2ビットデータ X_{i+1}^2 、 X_{i+2}^2 、 X_{i+3}^2 はそれぞれ第*i*ラウンド入力 X_i^1 、 X_i^2 とラウンド鍵 RK_i^1 、 RK_i^2 、 RK_i^3 を用いて以下のように表現できる。

$$[0047] \quad X_{i+1}^1 = F(RK_i^1, X_i^1) (+) X_i^2$$

$$X_{i+2}^1 = F(RK_{i+1}^1, X_{i+1}^1) (+) X_{i+1}^2 = F(RK_{i+1}^1, F(RK_i^1, X_i^1) (+) X_i^2) (+) X_i^1$$

$$X_{i+3}^1 = F(RK_{i+2}^1, X_{i+2}^1) (+) X_{i+2}^2 = F(RK_{i+2}^1, F(RK_{i+1}^1, X_{i+1}^1) (+) X_{i+1}^2) (+) X_{i+2}^2$$

$$= F(RK_{i+2}^1, F(RK_{i+1}^1, F(RK_i^1, X_i^1) (+) X_i^2) (+) X_i^1) (+) F(RK_i^1, X_i^1) (+) X_i^2$$

[0048] $X_{i+1}^2 = X_i^1$

$$X_{i+2}^2 = X_{i+1}^1 = F(RK_i^1, X_i^1) (+) X_i^2$$

$$X_{i+3}^2 = X_{i+2}^1 = F(RK_{i+1}^1, X_{i+1}^1) (+) X_{i+1}^2 = F(RK_{i+1}^1, F(RK_i^1, X_i^1) (+) X_i^2) (+) X_i^1$$

[0049] なお、上記式において、

$F(K, X)$ はデータ X をパラメータ K を用いて F 関数で変換したデータを表し、

(+) はビット毎の排他的論理和を表す。

[0050] X_{i+1}^1 は入力データ X_i^1 , X_i^2 を用いて表現されているが、 X_i^2 が F 関数を通していないため、拡散 (diffusion) 状態にはなっていない。 X_{i+2}^1 は入力データ X_i^1 , X_i^2 を用いて表現され、かつそれら入力データが F 関数の入力として与えられているので、拡散 (diffusion) 状態になっていると言える。同様に X_{i+3}^1 も拡散 (diffusion) 状態になっていると言える。

[0051] X_{i+1}^2 は X_i^1 のみで表現されているため、また、 X_{i+2}^2 は X_i^2 が F 関数を通していないため拡散 (diffusion) 状態にはなっていない。 X_{i+3}^2 は条件を満たすため拡散 (diffusion) 状態となっている。

[0052] 以上の結果から第 $i+3$ ラウンド出力 X_{i+3}^1 , X_{i+3}^2 はともに拡散 (diffusion) 状態になっているため、全拡散 (full diffusion) 状態であると言える。このように、図 11、図 12 に示される Feistel 構造を持つブロック暗号の全拡散 (full diffusion) ラウンド数は 3 ラウンドであることが分かる。

[0053] 全拡散 (full diffusion) 状態になっていない場合、特定の出力ビットは特定の入力ビットと非線形関数 (F 関数) の影響を受けていないということになるため、各種攻撃に対して脆弱になることが予想される。特に拡散 (diffusion) 特性は不能差分攻撃、飽和攻撃といった攻撃に対する安全性を直接的に評価する指標としても用いられる。全拡散 (full diffusion) ラウンド数は少ない方が拡散 (diffu

sion) 特性は高いと言える。

[0054] 図13、図14に別の例を挙げる。これらの図は4-lineで1段に2つのF関数を使用する一般化Feistel構造を示している。

図15に第iラウンド目のn-bit入力 X_i を $n/4$ -bitずつ4つに分割し、その4番目の $n/4$ -bitデータ(X_i^4 と表記)の一部のみに変化を与えた場合のその影響の波及を示す。

[0055] ただし、図中で太点線は入力の変化がF関数を通過しない状態で伝播しているデータを表し、太線は入力の変化が少なくとも1回F関数を通過した状態で伝播しているデータを表す。また、F関数の各出力ビットはF関数の全入力ビットの影響を受けると仮定する。このように、あらゆる入力ビットの変化を想定して、全出力ビットが影響を受けるまでのラウンド数を求めることも全拡散(full diffusion)ラウンド数は求めることができる。実際に $d=4$ として、1段に2つのF関数を使用する一般化Feistel構造の全拡散(full diffusion)ラウンド数は5ラウンドであることが知られている。

[0056] (2-2) 拡散(diffusion) 特性を考慮したこれまでの構成例について

次に、これまでに提案されている拡散(diffusion) 特性を考慮した処理構成についての概要を説明する。

[0057] 一般的に1段に $d/2$ 個のF関数を用いたd-line一般化Feistel構造は拡散(diffusion) 特性があまりよくなく、全拡散(full diffusion)ラウンド数は $d+1$ ラウンドになることが知られている。このことは、例えば、非特許文献3(T. Suzuki, K. Minematsu, "Improving the Generalized Feistel", FSE 2010, LNCS6147, pp. 19-39, 2010.)に記載がある。

[0058] 図16に $d=6$ とした場合の通常の一般化Feistel構造、図17に同構成の全拡散(full diffusion)状態となるパスの一例を

示す。

[0059] 非特許文献3ではこのような課題を解決するため、 n/d ビット毎 d 個に分割したデータそれぞれについてラウンド間の配線を変更することで分割数 d が6以上の場合に、従来の構成より全拡散 (full diffusion) ラウンド数を少なくできる構成法を提案している。

[0060] 図18に $d=6$ とした場合のこの構成法、また図19にその場合の全拡散 (full diffusion) 状態となるパスの一例を示す。

この構成で達成している全拡散 (full diffusion) ラウンド数は $d=6, 8, 10, 12, 14, 16$ の場合にそれぞれ5, 6, 7, 8, 8, 8ラウンドとなっており、通常の構成の $d+1$ ラウンドより良い拡散 (diffusion) 特性が得られていることが分かる。しかしながら、この構成では $d=4$ の場合には効果がなく、またこれ以上全拡散 (full diffusion) ラウンド数を削減することはできない。

[0061] また、非特許文献4 (洲崎, 角尾, 久保, 川幡, "一般化Feistel構造に拡散層を組み合わせた構造の提案", SCIS2008, 2008) では、 n/d ビット毎 d 個に分割したデータそれぞれについて線形演算を施すことで従来より拡散 (diffusion) 特性が高くなる構成を提案している。しかしながら、この構成の場合、線形演算のための実装手段が必要なたため、実装コストの増大を招いてしまう。

[0062] [3. 本開示に従った拡散 (diffusion) 特性を向上させた構成例について]

以上のような問題を鑑み、本開示では、一般化Feistel構造において、実装コストを増大させずにより高い拡散 (diffusion) 特性を得られるような構成法を提案する。

[0063] 本構成では、まず通常の構成と同様に d -lineの一般化Feistel構造において、 n ビット入力データを n/d ビット毎 d 個に分割しそれぞれ F 関数処理、排他的論理和処理を行う (図20: ステップ1)。

[0064] このとき、

F関数に入力されるデータ系列をF関数入力側データ系列、
排他的論理和されるデータ系列を排他的論理和側データ系列、
と呼ぶ。

その後、各系列（各ライン）において転送される n/d ビットデータ各々
について、さらに $d/2$ 個に再分割する（この際の分割は等分割でなくても
良い）。

各系列（各ライン）各々で $d/2$ 個に再分割されたデータを次のルールに
したがって分配する（図20：ステップ2）。

[0065] (1) F関数入力側データ系列は必ず次のラウンド関数の排他的論理和側
データ系列に分配する

 (2) 排他的論理和側データ系列は必ず次のラウンド関数のF関数入力側
データ系列に分配する

 (3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関
数のデータ系列にそれぞれ重複なく分配する

[0066] このように分配した後、各 $d/2$ 個に分割されたデータをそれぞれ1つの
データに結合する（図20：ステップ3）。

これを必要回数繰り返す。

[0067] 分割数 d が、 $d=4$ の場合の構成例を図20に示す。

 このように構成した場合、データの分割数 d によらず全拡散（full
diffusion）ラウンド数4を実現することが可能になる。

[0068] 本方式が分割数 d に関わらず全拡散（full diffusion）を
実現するラウンド数=4を満たす理由を以下に示す。

 (1) 第 i ラウンドのあらゆる入力データの変化は第 $i+1$ ラウンド目の
少なくとも1つのF関数に影響を与える。

 (2) 第 $i+1$ ラウンド目の排他的論理和側データ系列のうち少なくとも
1つはdiffusion状態となっている。

 (3) 第 $i+1$ ラウンド目の排他的論理和側データ系列で拡散（diffusion）
状態となっているデータはさらに $d/2$ 個に分割され、第 $i+$

2 ラウンド目の $d/2$ 個の全 F 関数に影響を与える。そのため第 $i+2$ ラウンド目の全ての排他的論理和側データ系列は拡散 (diffusion) 状態になる。

(4) 第 $i+2$ ラウンド目の排他的論理和側データ系列は第 $i+3$ ラウンド目の F 関数入力側データ系列となるため、第 $i+3$ ラウンド目の F 関数入力側データ系列は全て拡散 (diffusion) 状態となる。また、これら拡散 (diffusion) 状態であるデータは第 $i+3$ ラウンド目の全ての F 関数に入力されるため、その出力と排他的論理和される排他的論理和側データ系列も全て拡散 (diffusion) 状態となる。

[0069] 以上の理由から、必ず第 $i+3$ ラウンド後、つまり 4 ラウンドで全拡散 (full diffusion) 状態になることが分かる。図 21, 図 22 を用いて具体例を示す。

[0070] 図 21 は $d=4$ とした場合の本方式の一例である。 $d=4$ なので、4 つに分割された各 $n/4$ ビットデータをさらに 2 ($=d/2$) 分割している。

第 i ラウンド入力 X_i^1 に対応する第 i ラウンド出力の 2 分割データをそれぞれ Y_i^{1L} , Y_i^{1R} とし、同様に X_i^2 , X_i^3 , X_i^4 についても対応する各分割データを Y_i^{2L} , Y_i^{2R} , Y_i^{3L} , Y_i^{3R} , Y_i^{4L} , Y_i^{4R} とする。これら分割データのサイズは等分割した際は $n/8$ ビットとなる。

[0071] しかし、各ラインのデータの再分割処理は、必ずしも等分割であることは必要でない。例えば入力ビット数 = 256 ビットし、分割数 $d=4$ である場合、

各ラインのビット数は、 $n/d = 256/4 = 64$ ビットとなり、この再分割データは等分割すれば 32 ビットとなり、2 つの 32 ビットデータが生成される。

しかし、等分割することは必須ではなく、64 ビットのデータから生成する再分割データを 20 ビットと 44 ビット等、任意の組み合わせに分割してもよい。

ただし、次のラウンド演算部に入力する際には、異なるラインにおいて分

割された20ビットと44ビットとの組み合わせによって分割数である $d = 4$ 個の64ビット単位のデータを再構成して各分割ラインに入力する。

[0072] すなわち分割数 d の構成において、 d 本の各ラインにおける再分割処理においては、等分割することは必須ではないが、 d 本の各ラインにおける再分割の態様（分割比率）は一致させる必要がある。

[0073] このとき、第 i ラウンド入力データのどこかに変化を与えた場合、その影響は第 $i + 1$ ラウンド目の2つのF関数のうち少なくとも1つに必ず入力される。例えば、 X_i^4 のLSB1ビットのみに変化を与えた場合、その影響は Y_i^{4R} のみに伝播し、次にその影響が X_{i+1}^1 に伝播し、この X_{i+1}^1 が第 $i + 1$ ラウンド目の左側F関数に入力されることになる（図22）。入力ビットの他の位置に変化を与えた場合にも同様に考えることができ、少なくとも1つのF関数に影響が伝播することが保証される。

[0074] 第 $i + 1$ ラウンド目のF関数の少なくとも1つのF関数には入力の変化が影響しているので、その出力と排他的論理和されるデータは拡散（diffusion）状態となる。つまり、 $(Y_{i+1}^{2L}, Y_{i+1}^{2R})$ 、 $(Y_{i+1}^{4L}, Y_{i+1}^{4R})$ のどちらかの組は拡散（diffusion）状態となることが保証される。図22では $(Y_{i+1}^{2L}, Y_{i+1}^{2R})$ がdiffusion状態となっている例を示している。

[0075] 本方式のルール2、3より、第 $i + 1$ ラウンド目の出力である Y_{i+1}^{2L} 、 Y_{i+1}^{2R} は第 $i + 2$ ラウンド目の2（ $= d / 2$ ）個の全F関数に入力される。同様に Y_{i+1}^{4L} 、 Y_{i+1}^{4R} についても第 $i + 2$ ラウンド目の全F関数に入力される。つまり、第 $i + 1$ ラウンド目において $(Y_{i+1}^{2L}, Y_{i+1}^{2R})$ 、 $(Y_{i+1}^{4L}, Y_{i+1}^{4R})$ のどちらの組が拡散（diffusion）状態となっていたとしても、第 $i + 2$ ラウンド目の全F関数には拡散（diffusion）状態のデータが入力されることになる。よって、それらF関数の出力と排他的論理和されるデータ Y_{i+2}^{2L} 、 Y_{i+2}^{2R} 、 Y_{i+2}^{4L} 、 Y_{i+2}^{4R} は全て拡散（diffusion）状態になり、これらはルール2、3より X_{i+3}^1 、 X_{i+3}^3 に供給されるため、 X_{i+3}^1 、 X_{i+3}^3 も拡散（diffusion）状態

となることが分かる。

[0076] これら拡散 (d i f f u s i o n) 状態にある X_{i+3}^1 , X_{i+3}^3 は第 $i+3$ ラウンド目のそれぞれの F 関数に入力されるため、その出力と排他的論理和された結果である Y_{i+2}^{2L} , Y_{i+2}^{2R} , Y_{i+2}^{4L} , Y_{i+2}^{4R} 、つまり、 X_{i+4}^1 , X_{i+4}^3 も拡散 (d i f f u s i o n) 状態となる。

[0077] 以上の結果から、本方式では、全拡散 (f u l l d i f f u s i o n) ラウンド数 4 を満たすことが可能になる。図 2 2 は $d=4$ の場合の本方式で全拡散 (f u l l d i f f u s i o n) 状態となるパスの一例を示す。この構成例では従来構成の全拡散 (f u l l d i f f u s i o n) ラウンド数である 5 ラウンドより少ない 4 ラウンドを実現できていることが分かる。図 2 2 にこの構成における $d=4$ の場合の f u l l d i f f u s i o n を満たすパスの一例を示す。

[0078] また、図 2 3 に $d=6$ とした場合の構成例、図 2 4 にその場合の全拡散 (f u l l d i f f u s i o n) 状態となるパスの一例を示す (図中のラウンド間置換の前後に記述されている数字は置換後にどの位置にデータが配置されているかを示すインデックスを表す)。

[0079] 以上、説明した本開示に係る構成では、従来構成と比較して非常に高い拡散 (d i f f u s i o n) 特性が得られていることが分かる。また、本開示の構成では、線形演算を含まないため、実装コストを増大させることもない。

[0080] このように、本実施例は、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、この暗号処理部において以下のようなデータの再分割と再構成を伴うラウンド演算を繰り返し実行する。

[0081] すなわち、暗号処理部は、入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する。

このラウンド演算の繰り返し処理に際して、ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する。

[0082] 具体的には、以下の条件を満足する再分割と再構成処理を行う。

(1) F 関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する

(2) 排他的論理和側データ系列は必ず次のラウンド関数の F 関数入力側データ系列に分配する

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する

[0083] このような条件を満たす処理を行う。

[0084] 例えば、暗号処理部は、ラウンド演算の出力データを有する d 個のライン各々の n/d ビットデータを $d/2$ 個に再分割して、 $d \times (n/d)$ 個の再分割データを生成し、分割数 d に応じた d 本のラインの異なるラインから選択した $d/2$ 個の再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する。

[0085] このような処理を行うことで、前述した全拡散 (full diffusion) 状態を満たすために最低限必要なラウンド数 (繰り返し数)、すなわち、「全拡散 (full diffusion) ラウンド数」を 4 に設定することが可能となる。

[0086] なお、前述したように、ある出力ビットを入力ビットの関係式として記述して以下の条件を満たす場合、その出力ビットは「diffusion 状態」になっていると定義する。

(条件 1) 全入力ビットが関係式に含まれる。

(条件 2) 全入力ビットが少なくとも 1 度はラウンド関数 (F 関数) を

通過している。

さらに、全出力ビットが拡散 (diffusion) 状態になっている場合が「全拡散 (full diffusion) 状態」である。

[0087] 本開示の実施例として説明した構成例では、入力 n ビットを d 分割したデータを転送する各ライン、すなわち $(d/2)$ 本の F 関数入力側データ系列ラインと、 $(d/2)$ 本の排他的論理和側系列ライン各々で転送される各 n/d ビットデータをそれぞれ $d/2$ 個に分割した例を説明した。

これは、全拡散 (full diffusion) を実現するための最小ラウンド数 = 4 を実現するための設定である。

図 25 に分割数 $d : d = 6$ の場合に各ラインで転送される n/d ビットデータ ($n/6$ ビットデータ) を 2 分割、すなわち $d/12$ ビットに分割してラウンド間の転送を行う構成例を示す。

[0088] しかし、ラウンド数を 4 に限定しなければ、全拡散 (full diffusion) を実現するための分割数は、各ライン各々について $(d/2)$ 個に限定されるものではない。

入力 n ビットの分割数 : d と、

各ライン各々の分割数 : p と、

全拡散 (full diffusion) を実現するためのラウンド数、これらの対応関係を示す図を図 26 に示す。

適切にデータ系列を分配すれば、全拡散 (full diffusion) ラウンド数は、以下の式で算出できる。

$$\text{ラウンド数} = 3 + \lceil \log_p (d/2) \rceil$$

ただし、 $\lceil x \rceil$ は x 以上の最小の整数とする。

[0089] さらに拡張例として F 関数入力側データ系列と排他的論理和側データ系列とでそれぞれ異なる分割を行った構成を図 27 に示す。

図 27 に示す構成は、

分割数 $d : d = 6$ の場合に、

$(d/2) = 3$ 本の F 関数入力側データ系列ラインについては、2 分割、

($d/2$) = 3本の排他的論理和側系列ラインについては、3分割した例である。

このような構成としても、適切にデータ系列を分配すれば、全拡散 (full diffusion) ラウンド数は4とすることが可能である。

[0090] また、本方式は拡大鍵 (ラウンド鍵) の挿入位置によらず効果が得られる。図21に記載の構成例で拡大鍵 (ラウンド鍵) の挿入位置を変更した構成例を図28に示す。

図21においては、拡大鍵 (ラウンド鍵) を各F関数に挿入して適用している。

これに対して、図28に示す構成では、拡大鍵 (ラウンド鍵) をF関数出力と排他的論理和側データ系列との排他的論理和演算な部に挿入している。

このような構成においても、前述の各ラウンド間のデータの再分割処理による転送構成によって、全拡散 (full diffusion) ラウンド数の削減が実現される。

[0091] [4. より効率的なデータ分配法を持つ構成例について]

上記の項目 [3. 本開示に従った拡散 (diffusion) 特性を向上させた構成例について] では、一般化Feistel構造において、実装コストを増大させずにより高い拡散 (diffusion) 特性を得られるような構成法について説明した。

[0092] すなわち、入力データである n ビットデータを分割数 d で分割した n/d ビットデータを各ラインに入力して、ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する構成において、ラウンド演算の出力データを有するライン各々の n/d ビットデータをさらに $d/2$ 個に再分割し、この再分割データを組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして処理を実行する構成である。

[0093] 以下では、上述した実施例において説明したデータ分配法のより効率的な手法について説明する。

まず、図29を参照して本実施例の基本的な構成例について説明する。

本構成では、まず前述した上記項目[3]において説明した実施例と同様、*d-line*の一般化 *Feistel* 構造において、*n*ビット入力データを分割数 *d* に従って、 n/d ビット毎の *d* 個に分割し、それぞれ、ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する。

[0094] *i* 段のラウンド演算後、*d* 個の各ライン（系列）各々の n/d ビット系列をさらにそれぞれ $d/2$ 個に再分割する。

i 段目のラウンド演算後の、各ライン（系列）各々の $d/2$ 個に分割された出力データのデータ中、*j* 番目のデータを $Y_i[j]$ と表す（ただし、*j* は1以上 $d^2/2$ 以下の整数）。

[0095] 同様に *i* + 1 段目のラウンド演算部における分割数：*d* の各ライン単位の n/d ビットをそれぞれ $d/2$ 個に再分割したデータ中、*j* 番目のデータを $X_{i+1}[j]$ と表す（ただし、*j* は1以上 $d^2/2$ 以下の整数）。

[0096] また、 $YY_i[t]$ を $Y_i[j]$ のうち、 $j = (d/2)s + t$ を満たすものを順に連結したデータとする。

ただし *s* は0以上 (*d* - 1) 以下、*t* は1以上 $d/2$ 以下の整数である。

[0097] 具体的には、例えば、

$$YY_i[1] = Y_i[1] \parallel Y_i[1 \times d/2 + 1] \parallel Y_i[2 \times d/2 + 1] \parallel \dots \parallel Y_i[(d-1) \times d/2 + 1],$$

$$YY_i[2] = Y_i[2] \parallel Y_i[1 \times d/2 + 2] \parallel Y_i[2 \times d/2 + 2] \parallel \dots \parallel Y_i[(d-1) \times d/2 + 2],$$

...

$$YY_i[d/2] = Y_i[d/2] \parallel Y_i[1 \times d/2 + d/2] \parallel Y_i[2 \times d/2 + d/2] \parallel \dots \parallel Y_i[(d-1) \times d/2 + d/2]$$

となる。

[0098] 同様に $XX_i[t]$ を $X_i[j]$ のうち、 $j = (d/2)s + t$ を満たすものを順に連結したデータとする。

ただし、

$X_i[j]$ は、 i 段目の入力データを n/d ビット毎 d 個に分割したものをそれぞれ $d/2$ 個に分割したデータの j 番目のデータ、

s は 0 以上 $(d-1)$ 以下、 t は 1 以上 $d/2$ 以下の整数である。

[0099] 前記の項目 [3] において説明した手法では、分配のパターンが多くなってしまい評価に時間がかかる。すなわち、前記の項目 [3] において説明した手法では、以下の (1) ~ (3) の条件を満足する分配パターンであればよい。

(1) F 関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する

(2) 排他的論理和側データ系列は必ず次のラウンド関数の F 関数入力側データ系列に分配する

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する

[0100] この条件 (1) ~ (3) を満足する分配パターンは、多数存在する。特に入力データの分割数 d が大きいような場合は分配のパターンが非常に多くなるため容易に決定できない。また、分配法によっては実装時のコストが大きくなってしまうものもある。

[0101] 以下では、選択許容可能な分配パターンを予め限定し、さらに実装コストも低減できる分配法を提案する。本提案方式は前記の項目 [3] で説明した分配方式をさらに改良した方式となる。

[0102] まず、 $i+1$ 段目の入力データを n/d ビット毎 d 個に分割したものをそれぞれ $d/2$ 個に再分割したデータの j 番目のデータ $X_{i+1}[j]$ のうち、 $j = (d/2)s + t$ を満たすものを順に連結したデータ $XX_{i+1}[t]$ のデータ系列を $2t-1$ データ分だけ左に巡回シフトさせたデータ系列を、

$ZZ_{i+1}[t]$

とする。

[0103] 具体的には、例えば、

$$\begin{aligned}
Z Z_{i+1} [1] &= X X_{i+1} [1] \lll 1 \text{ となるので、} \\
Z Z_{i+1} [1] &= X X_{i+1} [1] \lll 1 \\
&= (X_{i+1} [1] \parallel X_{i+1} [1 \times d / 2 + 1] \parallel X_{i+1} \\
&\quad [2 \times d / 2 + 1] \parallel \cdots \parallel X_{i+1} [(d-1) \times d / 2 + 1]) \lll \\
&\quad < 1 \\
&= X_{i+1} [1 \times d / 2 + 1] \parallel X_{i+1} [2 \times d / 2 + 1] \parallel \\
&\quad \cdots \parallel X_{i+1} [(d-1) \times d / 2 + 1] \parallel X_{i+1} [1] \\
&\text{となる。}
\end{aligned}$$

[0104] 以上のように定義された $Y Y_i [j]$ 、および $Z Z_{i+1} [j]$ を用い、各 $Y Y_i [j]$ より重複なくひとつずつ $Z Z_{i+1} [j]$ を選択し、接続することで第 i ラウンド目のデータから第 $i+1$ ラウンド目のデータへのデータ置換を決めることができる。

入力データの分割数 d : $d = 6$ とした場合を図 30 に例示する。

図 30 での各中間変数は以下のように定義される。

[0105] なお、中間変数とは、以下の 3 種類のデータである。

$Y Y_i [t]$: i 段目の $d/2$ 個に分割された出力データのうち j 番目のデータ $Y_i [j]$ (ただし、 j は 1 以上 $d^2/2$ 以下の整数) のうち、 $j = (d/2) s + t$ (ただし、 s は 0 以上 $(d-1)$ 以下、 t は 1 以上 $d/2$ 以下の整数) を満たすものを順に連結したデータ。

$X X_{i+1} [t]$: $i+1$ 段目の入力データを n/d ビット毎 d 個に分割したものをそれぞれ $d/2$ 個に分割したデータの j 番目のデータ $X_{i+1} [j]$ のうち、 $j = (d/2) s + t$ (ただし、 s は 0 以上 $(d-1)$ 以下、 t は 1 以上 $d/2$ 以下の整数) を満たすものを順に連結したデータ。

$Z Z_{i+1} [t]$: $X X_{i+1} [t]$ のデータ系列を $2t-1$ データ分だけ左に巡回シフトさせたデータ系列。

これらの中間変数である。

[0106] 図 30 に示すように、入力データの分割数 d : $d = 6$ とした場合、

t は 1 以上 $d/2$ 以下の整数であり、 $t = 1, 2, 3$ の各値が設定され、

中間変数として、

i 段目出力データ対応の中間変数 $Y Y_i [t] : Y Y_i [1]、Y Y_i [2]、Y Y_i [3]、$

$i + 1$ 段目のシフト前入力データ対応の中間変数 $X X_{i+1} [t] : X X_{i+1} [1]、X X_{i+1} [2]、X X_{i+1} [3]$

$i + 1$ 段目のシフト後入力データ対応の中間変数 $Z Z_{i+1} [t] : Z Z_{i+1} [1]、Z Z_{i+1} [2]、Z Z_{i+1} [3]$

これらが以下のように算出される。

[0107] i 段目出力データ対応の中間変数 $Y Y_i [t] : Y Y_i [1]、Y Y_i [2]、Y Y_i [3]$ は、以下の設定となる。

$Y Y_i [1] = Y_i [1] \parallel Y_i [4] \parallel Y_i [7] \parallel Y_i [10] \parallel Y_i [13] \parallel Y_i [16]、$

$Y Y_i [2] = Y_i [2] \parallel Y_i [5] \parallel Y_i [8] \parallel Y_i [11] \parallel Y_i [14] \parallel Y_i [17]、$

$Y Y_i [3] = Y_i [3] \parallel Y_i [6] \parallel Y_i [9] \parallel Y_i [12] \parallel Y_i [15] \parallel Y_i [18]、$

[0108] $i + 1$ 段目のシフト前入力データ対応の中間変数 $X X_{i+1} [t] : X X_{i+1} [1]、X X_{i+1} [2]、X X_{i+1} [3]$ は、以下の設定となる。

$X X_{i+1} [1] = X_{i+1} [1] \parallel X_{i+1} [4] \parallel X_{i+1} [7] \parallel X_{i+1} [10] \parallel X_{i+1} [13] \parallel X_{i+1} [16]、$

$X X_{i+1} [2] = X_{i+1} [2] \parallel X_{i+1} [5] \parallel X_{i+1} [8] \parallel X_{i+1} [11] \parallel X_{i+1} [14] \parallel X_{i+1} [17]、$

$X X_{i+1} [3] = X_{i+1} [3] \parallel X_{i+1} [6] \parallel X_{i+1} [9] \parallel X_{i+1} [12] \parallel X_{i+1} [15] \parallel X_{i+1} [18]、$

[0109] $i + 1$ 段目のシフト後入力データ対応の中間変数 $Z Z_{i+1} [t] : Z Z_{i+1} [1]、Z Z_{i+1} [2]、Z Z_{i+1} [3]$ は、以下の設定となる。

$Z Z_{i+1} [1] = X_{i+1} [4] \parallel X_{i+1} [7] \parallel X_{i+1} [10] \parallel X_{i+1} [13] \parallel X_{i+1} [16] \parallel X_{i+1} [1]、$

$$\begin{aligned}
 Z Z_{i+1} [2] &= X_{i+1} [1\ 1] \parallel X_{i+1} [1\ 4] \parallel X_{i+1} [1\ 7] \parallel \\
 &\parallel X_{i+1} [2] \parallel X_{i+1} [5] \parallel X_{i+1} [8], \\
 Z Z_{i+1} [3] &= X_{i+1} [1\ 8] \parallel X_{i+1} [3] \parallel X_{i+1} [6] \parallel X_{i+1} [9] \\
 &\parallel X_{i+1} [1\ 2] \parallel X_{i+1} [1\ 5]。
 \end{aligned}$$

[0110] ここで、図30に示すように、例えば $Y Y_i [1]$ と $Z Z_{i+1} [1]$ を、 $Y Y_i [2]$ と $Z Z_{i+1} [2]$ を $Y Y_i [3]$ と $Z Z_{i+1} [3]$ を接続する。この場合、図31上部の(a)に示されるようなデータ分配法となる。

[0111] また、例えば $Y Y_i [1]$ と $Z Z_{i+1} [2]$ を、 $Y Y_i [2]$ と $Z Z_{i+1} [3]$ を $Y Y_i [3]$ と $Z Z_{i+1} [1]$ を接続した場合は図31下部の(b)に示されるようなデータ分配法となる。

[0112] このように、本実施例の暗号処理装置の暗号処理部の設定、すなわち、前段のラウンド演算の出力データと次段のラウンド演算の再分割データの入出力関係を決定する接続構成は、ラウンド演算の出力データを有するラインの n/d ビットデータの再分割処理によって生成される $d \times (n/d)$ 個の再分割データの組み合わせデータである $(d/2)$ 個の $2n/d$ ビットのデータセットを単位とした接続構成から選択される。

[0113] 以上のように決められたデータ分配法は、前述した項目[3]において説明した条件、すなわち、

(1) F関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する

(2) 排他的論理和側データ系列は必ず次のラウンド関数のF関数入力側データ系列に分配する

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する

この条件(1)～(3)を満足する。

したがって、拡散(diffusion)特性の改善が可能となる。

[0114] また、前述の項目[3]の方法では、 $d/2 \times d/2$ 種類のデータ系列を $d/2 \times d/2$ 種類のデータ系列から1つずつ選んで分配する必要があった

が（実際にはこの処理をさらにもう1回繰り返す必要がある）、本方式では $d/2$ 種類のデータ系列を $d/2$ 種類のデータ系列から1つずつ選んで分配すればよい。また、選択しなければならないパターンを大幅に削減できる。さらに、上記方法で選択された分配法は、データの分配について規則性があるため、実装コストを削減することが可能となる。

[0115] [5. インボリューション (involution) 性を備えた構成例について]

通常の Feistel 構造における暗号化関数は図32のように表現することができる。また、通常の Feistel 構造における復号関数は図33のように表現することができる。

[0116] これらの図32、図33の構成から理解されるように、F 関数に挿入される拡大鍵（ラウンド鍵）の順序を適切に入れ替えるだけで暗号化関数と復号関数は全く同じ関数を使用することが可能である。このように、拡大鍵（ラウンド鍵）の挿入を除き、暗号化関数と復号関数を同一構成で実現可能とした性質をインボリューション性と呼ぶ。

[0117] インボリューション性を持つ暗号処理構成においては、拡大鍵（ラウンド鍵）の順序を適切に入れ替えれば暗号化関数、復号関数を共有可能であり、復号関数を別途用意する必要はない。したがって、一般的にインボリューション性を持つ暗号は、インボリューション性を持たない暗号に比べて少ない実装コストで実装することが可能であると言える。

また、暗号化関数と復号関数で同じ関数を用いられることは、検証コストが半減できる（検証は暗号化関数、復号関数のどちらかで検証できればよい）、コードサイズが半減できる、などのメリットもある。

[0118] このように、図32、図33の分割数 $d=2$ の通常の Feistel 構造では、F 関数に挿入される拡大鍵（ラウンド鍵）の順序を適切に入れ替えるだけで暗号化関数と復号関数は全く同じ関数を使用することが可能であり、インボリューション性を持つ構成を容易に構成できる。

[0119] しかし、分割数 d が2以外の構成を含む一般化 Feistel 構造 (G

eneralized Feistel Networks)でのインボリ
ューション性は容易に構成可能とは言えない。

[0120] 1段に $d/2$ 個のF関数を用いた d -line一般化Feistel構造
のインボリューション性について考える（ただし、 $d > 2$ ）。

先に説明した図14に示すような4-line一般化Feistel構造
では、処理ラウンド数が奇数であれば、インボリューション性を持つことが
知られている。例えば3ラウンドで構成された4-line一般化Feis
tel構造はインボリューション性を持つ（5ラウンド構成も同様）。しか
しながら処理ラウンド数が偶数の場合は、インボリューション性を持たない
。

[0121] 一般的に d -line一般化Feistel構造においては、構成ラウン
ド数を、分割数： d で割ったときの剰余が、

1、もしくは、

$$(d/2) + 1$$

である場合のみインボリューション性を持つ。

[0122] 例えば、図14に示す4-line一般化Feistel構造では、構成
ラウンド数を、分割数 $d = 4$ で割ったときの剰余が、

1、もしくは、

$$(d/2) + 1 = (4/2) + 1 = 3$$

である場合のみインボリューション性を持つ。

具体的には、

構成ラウンド数 = 1, 3, 5, 7, 9, ...、

つまり構成ラウンド数が奇数である場合のみインボリューション性を持つ
。

[0123] また、例えば、分割数 $d = 6$ とした6-line一般化Feistel構
造では、構成ラウンド数を、分割数 $d = 6$ で割ったときの剰余が、

1、もしくは、

$$(d/2) + 1 = (6/2) + 1 = 4$$

である場合のみインボリューション性を持つ。

具体的には、

構成ラウンド数 = 1, 4, 7, 10, 13 . . . 、

構成ラウンド数がこれらの数である場合のみインボリューション性を持つ。
。

[0124] このように、一般的に d -line 一般化 Feistel 構造においては構成ラウンド数に関わらずインボリューション性を持つということはない。

[0125] 図 34 は、前述の項目 [3. 本開示に従った拡散 (diffusion) 特性を向上させた構成例について] において説明したラウンド演算間のデータ再分割、再合成処理によって高い拡散 (diffusion) 特性を得られる 4 -line 構造の一方式を示す図である。

この方式では、構成ラウンド数が $1 + 3n$ (ただし n は 0 以上の整数) の場合のみインボリューション性を満たす。

[0126] 構成ラウンド数は安全性、実装性能に大きな関連がある。構成ラウンド数に関わらずインボリューション性を持つような構造であれば、より柔軟にラウンド数を設定でき、安全性、実装性能を柔軟に変更できる上、インボリューション性の特性により、小型な実装も可能となる。

[0127] 実装コストを増大させずにより高い拡散 (diffusion) 特性を得られ、さらに構成ラウンド数に関わらずインボリューション性を満たすようなラウンド間置換の構成法について説明する。

以下において説明する方式は、前記の項目 [3. 本開示に従った拡散 (diffusion) 特性を向上させた構成例について] において説明した高い拡散 (diffusion) 特性を得られる構成に、さらにインボリューション性を持たせる方式となる。

[0128] 本方式の一例について、図 35 を参照して説明する。本方式は、図 35 に示すように、以下のステップ 1 ~ 3 の処理を繰り返し実行することになる。

(ステップ 1)

まず通常の構成と同様に分割数 d の d -line の一般化 Feistel

構造において、 n ビット入力データを分割数 d に応じて、 n/d ビット毎、 d 個に分割し、各 n/d ビットを各分割ラインに入力して、それぞれ F 関数処理、排他的論理和処理を行う。

このとき、 F 関数に入力されるデータ系列を F 関数入力側データ系列、排他的論理和されるデータ系列を排他的論理和側データ系列と呼ぶ。

[0129] F 関数入力側データ系列のうち最も左側にある F 関数入力側データ系列を $L(0)$ とし、例えば左側から順に $L(1)$ 、 $\dots$ 、 $L((d/2) - 1)$ と表す。

同様に排他的論理和側データ系列を例えば左側から順に $R(0)$ 、 $\dots$ 、 $R((d/2) - 1)$ と表す。

[0130] (ステップ2)

その後、各系列(ライン)の転送データである n/d ビットデータをさらに $d/2$ 個に再分割する。この再分割は、等分割でなくてもよい。

この再分割されたデータを、 F 関数入力側データ系列、排他的論理和側データ系列各々について、それぞれ $L(i)_j$ 、 $R(i)_j$ と表現する。

i は、各系列(ライン)の識別子(番号)、

j は、1つの系列(ライン)の再分割データ各々識別子(番号)、

である。

例えば、最も左側の F 関数入力データ系列を $d/2$ 個に再分割した最も左側のデータを $L(0)_0$ とし、順に $L(0)_1$ 、 $\dots$ 、 $L(0)_{d/2-1}$ と表現する。

[0131] そしてそれぞれの各系列(ライン)において $d/2$ 個に再分割されたデータを次のルールにしたがって分配する。

ルール(2-1)

最も左側の F 関数入力データ系列、すなわち $i=0$ の $L(0)$ のデータを分配する。

$L(0)_0$ を次のラウンド関数の $R(0)_0$ に分配、

$L(0)_1$ を次のラウンド関数の $R(1)_1$ に分配、

同様に $i = (d/2) - 1$ まで $L(0)_i$ を $R(i)_i$ に分配する。

すなわち、

$$L(0)_0 = R(0)_0,$$

$$L(0)_1 = R(1)_1,$$

$$L(0)_2 = R(2)_2,$$

...

となる。

[0132] ルール (2-2)

次に $L(1)$ のデータを分配する。

$L(1)_0$ を次のラウンド関数の $R(1)_0$ に分配、

$L(1)_1$ を次のラウンド関数の $R(2)_1$ に分配、

同様に $i = (d/2) - 1$ まで $L(1)_i$ を $R((i+1) \bmod d/2)_i$ に分配する。

[0133] ルール (2-3)

以下、 $L((d/2) - 1)$ のデータまで、上記 2 と同様の処理を繰り返す。つまり、 $L(i)_j$ を次のラウンドの $R((i+j) \bmod d/2)_j$ に分配する (ただし、 i, j はそれぞれ 0 以上 $(d/2) - 1$ 以下)。

[0134] ルール (2-4)

排他的論理和側データ系列についても同様の処理を繰り返す。つまり、 $R(i)_j$ を次のラウンド関数の $L((d/2) + i - j) \bmod d/2)_j$ に分配する。ただし、 i, j はそれぞれ 0 以上 $(d/2) - 1$ 以下である。

[0135] (ステップ 3)

このように分配した後、各 $d/2$ 個に分割されたデータをそれぞれ 1 つのデータに結合する。

以上の処理を、ラウンド演算の実行回数に応じて必要回数繰り返す。

[0136] 図 36 に 4-line ($d=4$) の場合に上記処理を実行する構成の一構成例、

図 37 に 6-line ($d=6$) の場合に上記処理を実行する構成の一構成例、

成例を示す。

これら、図36、図37に示す構成では、各系列（ライン）において $d/2$ 個に再分割されたデータを上記ステップ2に示すルール（2-1）～（2-4）に従って月のラウンドに対して再分配する構成を有する。このデータの再分割と再分配構成によって、これらの方式は構成ラウンド数に関わらずインボリューション性を持つ。

[0137] また、図38、図39は上記構成例とは異なる構成法であるが、構成ラウンド数に関わらずインボリューション性を持つ方式である。なお、先の項目[3]において説明した図21に示されている方式は、構成ラウンド数が奇数であればインボリューション性を持つ構成の一例である。

[0138] このように、暗号処理部の各ラウンド演算の再分割データの再構成処理は、前段のラウンド関数入力側系列の再分割データを、上述した予め決定した規則に従って次段の排他的論理和側系列に分配し、前段の排他的論理和側系列の再分割データを、上述した予め決定した規則に従って次段のラウンド関数入力側系列に分配する構成としている。この構成によって、同一の構成で暗号化処理と復号処理双方に適用可能なインボリューション性を持つ構成を実現することができる。

[0139] [6. 暗号処理装置の構成例について]

最後に、上述した実施例に従った暗号処理を実行する暗号処理装置の実相例について説明する。

上述した実施例に従った暗号処理を実行する暗号処理装置は、暗号処理を実行する様々な情報処理装置に搭載可能である。具体的には、PC、TV、レコーダ、プレーヤ、通信機器、さらに、RFID、スマートカード、センサネットワーク機器、デンチ／バッテリー認証モジュール、健康、医療機器、自立型ネットワーク機器等、例えばデータ処理や通信処理に伴う暗号処理を実行する様々な危機において利用可能である。

[0140] 本開示の暗号処理を実行する装置の一例としてのICモジュール700の構成例を図40に示す。上述の処理は、例えばPC、ICカード、リーダラ

イタ、その他、様々な情報処理装置において実行可能であり、図40に示すICモジュール700は、これら様々な機器に構成することが可能である。

[0141] 図40に示すCPU (Central processing Unit) 701は、暗号処理の開始や、終了、データの送受信の制御、各構成部間のデータ転送制御、その他の各種プログラムを実行するプロセッサである。メモリ702は、CPU701が実行するプログラム、あるいは演算パラメータなどの固定データを格納するROM (Read-Only-Memory)、CPU701の処理において実行されるプログラム、およびプログラム処理において適宜変化するパラメータの格納エリア、ワーク領域として使用されるRAM (Random Access Memory) 等からなる。また、メモリ702は暗号処理に必要な鍵データや、暗号処理において適用する変換テーブル (置換表) や変換行列に適用するデータ等の格納領域として使用可能である。なおデータ格納領域は、耐タンパ構造を持つメモリとして構成されることが好ましい。

[0142] 暗号処理部703は、上記において説明した暗号処理構成、すなわち、例えば一般化Feistel構造や、Feistel構造を適用した共通鍵ブロック暗号処理アルゴリズムに従った暗号処理、復号処理を実行する。

[0143] なお、ここでは、暗号処理手段を個別モジュールとした例を示したが、このような独立した暗号処理モジュールを設けず、例えば暗号処理プログラムをROMに格納し、CPU701がROM格納プログラムを読み出して実行するように構成してもよい。

[0144] 乱数発生器704は、暗号処理に必要な鍵の生成などにおいて必要となる乱数の発生処理を実行する。

[0145] 送受信部705は、外部とのデータ通信を実行するデータ通信処理部であり、例えばリーダライタ等、ICモジュールとのデータ通信を実行し、ICモジュール内で生成した暗号文の出力、あるいは外部のリーダライタ等の機器からのデータ入力などを実行する。

[0146] なお、上述した実施例において説明した暗号処理装置は、入力データとし

ての平文を暗号化する暗号化処理に適用可能であるのみならず、入力データとしての暗号文を平文に復元する復号処理にも適用可能である。

暗号化処理、復号処理、双方の処理において、上述した実施例において説明した構成を適用することが可能である。

[0147] [7. 本開示の構成のまとめ]

以上、特定の実施例を参照しながら、本開示の実施例について詳解してきた。しかしながら、本開示の要旨を逸脱しない範囲で当業者が実施例の修正や代用を成し得ることは自明である。すなわち、例示という形態で本発明を開示してきたのであり、限定的に解釈されるべきではない。本開示の要旨を判断するためには、特許請求の範囲の欄を参酌すべきである。

[0148] なお、本明細書において開示した技術は、以下のような構成をとることができる。

(1) データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、

前記暗号処理部は、

入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する構成であり、

ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する暗号処理装置。

[0149] (2) 前記ラウンド関数は、ラウンド鍵を適用した演算と、非線形変換処理と、線形変換処理を含む F 関数、および F 関数の出力または入力に対する他ラインのデータとの排他的論理和演算を含む前記 (1) に記載の暗号処理装置。

[0150] (3) 前記暗号処理部は、

(1) F関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する、

(2) 排他的論理和側データ系列は必ず次のラウンド関数のF関数入力側データ系列に分配する、

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する、

上記(1)～(3)の分配条件を満たす処理によって前段のラウンド演算の演算結果を次段のラウンド演算の入力として設定する前記(1)または(2)に記載の暗号処理装置。

[0151] (4) 前記暗号処理部は、入力データの分割数 d を4以上とした一般化 Feistel 構造を有する前記(1)～(3)いずれかに記載の暗号処理装置。

(5) 前記暗号処理部は、ラウンド演算の出力データを有する d 個のライン各々の n/d ビットデータを $d/2$ 個に再分割して、 $d \times (n/d)$ 個の再分割データを生成し、分割数 d に応じた d 本のラインの異なるラインから選択した $d/2$ 個の再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する前記(1)～(4)いずれかに記載の暗号処理装置。

[0152] (6) 前記暗号処理部は、全出力ビットが以下に示す2つの条件を満足する拡散 (diffusion) 状態、すなわち、出力ビットを入力ビットの関係式として記述した場合に、

(条件1) 全入力ビットが関係式に含まれる

(条件2) 全入力ビットが少なくとも1度は前記ラウンド関数を通過している

上記2つの条件を満足する全拡散 (full diffusion) 状態を実現する構成である前記(1)～(5)いずれかに記載の暗号処理装置。

(7) 前記暗号処理部は、前記全拡散 (full diffusion)

状態を4ラウンドのラウンド演算によって実現する前記(1)～(6)いずれかに記載の暗号処理装置。

[0153] (8) 前記暗号処理部の前段のラウンド演算の出力データと次段のラウンド演算の再分割データの入出力関係を決定する接続構成は、ラウンド演算の出力データを有するラインの n/d ビットデータの再分割処理によって生成される $d \times (n/d)$ 個の再分割データの組み合わせデータである($d/2$)個の $2n/d$ ビットのデータセットを単位とした接続構成から選択される接続構成である前記(1)～(7)いずれかに記載の暗号処理装置。

[0154] (9) 前記暗号処理部は、暗号化処理と復号処理の双方に適用可能なインボリューション性を有する前記(1)～(8)いずれかに記載の暗号処理装置。

(10) 前記暗号処理部の各ラウンド演算の再分割データの再構成処理は、前段のラウンド関数入力側系列の再分割データを、予め決定した規則に従って次段の排他的論理和側系列に分配し、前段の排他的論理和側系列の再分割データを、予め決定した規則に従って次段のラウンド関数入力側系列に分配する構成である前記(1)～(9)いずれかに記載の暗号処理装置。

(11) 前記暗号処理部は、入力データとしての平文を暗号文に変換する暗号化処理、または、入力データとしての暗号文を平文に変換する復号処理を実行する前記(1)～(10)いずれかに記載の暗号処理装置。

[0155] さらに、上記した装置およびシステムにおいて実行する処理の方法や、処理を実行させるプログラムも本開示の構成に含まれる。

[0156] また、明細書中において説明した一連の処理はハードウェア、またはソフトウェア、あるいは両者の複合構成によって実行することが可能である。ソフトウェアによる処理を実行する場合は、処理シーケンスを記録したプログラムを、専用のハードウェアに組み込まれたコンピュータ内のメモリにインストールして実行させるか、あるいは、各種処理が実行可能な汎用コンピュータにプログラムをインストールして実行させることが可能である。例えば、プログラムは記録媒体に予め記録しておくことができる。記録媒体からコ

ンピュータにインストールする他、LAN (Local Area Network)、インターネットといったネットワークを介してプログラムを受信し、内蔵するハードディスク等の記録媒体にインストールすることができる。

- [0157] なお、明細書に記載された各種の処理は、記載に従って時系列に実行されるのみならず、処理を実行する装置の処理能力あるいは必要に応じて並列的にあるいは個別に実行されてもよい。また、本明細書においてシステムとは、複数の装置の論理的集合構成であり、各構成の装置が同一筐体内にあるものには限らない。

産業上の利用可能性

- [0158] 上述したように、本開示の一実施例の構成によれば、拡散 (diffusion) 特性を向上させた安全性の高い暗号処理が実現される。

具体的には、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、暗号処理部において、入力データである n ビットデータを分割数 d で分割した n/d ビットデータを各ラインに入力して、ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する。ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に分割し、該分割データを組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして処理を実行する。この構成によって、拡散 (diffusion) 特性を向上させた安全性の高い暗号処理が実現される。

符号の説明

- [0159] 700 ICモジュール
701 CPU (Central processing Unit)
702 メモリ
703 暗号処理部

7 0 4 乱数生成部

7 0 5 送受信部

請求の範囲

[請求項1] データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理部を有し、

前記暗号処理部は、

入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行する構成であり、

ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する暗号処理装置。

[請求項2] 前記ラウンド関数は、

ラウンド鍵を適用した演算と、非線形変換処理と、線形変換処理を含む F 関数、および F 関数の出力または入力に対する他ラインのデータとの排他的論理和演算を含む請求項 1 に記載の暗号処理装置。

[請求項3] 前記暗号処理部は、

(1) F 関数入力側データ系列は必ず次のラウンド関数の排他的論理和側データ系列に分配する、

(2) 排他的論理和側データ系列は必ず次のラウンド関数の F 関数入力側データ系列に分配する、

(3) 各 $d/2$ 個に分割されたデータ系列は $d/2$ 箇所の次のラウンド関数のデータ系列にそれぞれ重複なく分配する、

上記 (1) ~ (3) の分配条件を満たす処理によって前段のラウンド演算の演算結果を次段のラウンド演算の入力として設定する請求項 2 に記載の暗号処理装置。

- [請求項4] 前記暗号処理部は、
入力データの分割数 d を 4 以上とした一般化 $F e i s t e l$ 構造を有する請求項 1 に記載の暗号処理装置。
- [請求項5] 前記暗号処理部は、
ラウンド演算の出力データを有する d 個のライン各々の n/d ビットデータを $d/2$ 個に再分割して、 $d \times (n/d)$ 個の再分割データを生成し、分割数 d に応じた d 本のラインの異なるラインから選択した $d/2$ 個の再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行する請求項 1 に記載の暗号処理装置。
- [請求項6] 前記暗号処理部は、
全出力ビットが以下に示す 2 つの条件を満足する拡散 ($d i f f u s i o n$) 状態、すなわち、出力ビットを入力ビットの関係式として記述した場合に、
(条件 1) 全入力ビットが関係式に含まれる
(条件 2) 全入力ビットが少なくとも 1 度は前記ラウンド関数を通過している
上記 2 つの条件を満足する全拡散 ($f u l l d i f f u s i o n$) 状態を実現する構成である請求項 1 に記載の暗号処理装置。
- [請求項7] 前記暗号処理部は、
前記全拡散 ($f u l l d i f f u s i o n$) 状態を 4 ラウンドのラウンド演算によって実現する請求項 6 に記載の暗号処理装置。
- [請求項8] 前記暗号処理部の前段のラウンド演算の出力データと次段のラウンド演算の再分割データの入出力関係を決定する接続構成は、
ラウンド演算の出力データを有するラインの n/d ビットデータの再分割処理によって生成される $d \times (n/d)$ 個の再分割データの組み合わせデータである ($d/2$) 個の $2n/d$ ビットのデータセット

を単位とした接続構成から選択される接続構成である請求項 1 に記載の暗号処理装置。

[請求項9] 前記暗号処理部は、
暗号化処理と復号処理の双方に適用可能なインボリューション性を有する請求項 1 に記載の暗号処理装置。

[請求項10] 前記暗号処理部の各ラウンド演算の再分割データの再構成処理は、
前段のラウンド関数入力側系列の再分割データを、予め決定した規則に従って次段の排他的論理和側系列に分配し、
前段の排他的論理和側系列の再分割データを、予め決定した規則に従って次段のラウンド関数入力側系列に分配する構成である請求項 9 に記載の暗号処理装置。

[請求項11] 前記暗号処理部は、
入力データとしての平文を暗号文に変換する暗号化処理、または、
入力データとしての暗号文を平文に変換する復号処理を実行する請求項 1 に記載の暗号処理装置。

[請求項12] 暗号処理装置において実行する暗号処理方法であり、
暗号処理部が、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理ステップを実行し、
前記暗号処理ステップは、
入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行し、
ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行す

る暗号処理方法。

[請求項13]

暗号処理装置において暗号処理を実行させるプログラムであり、

暗号処理部に、データ処理対象となるデータの構成ビットを複数のラインに分割して入力し、各ラインのデータに対してラウンド関数を適用したデータ変換処理を繰り返して実行する暗号処理ステップを実行させ、

前記暗号処理ステップにおいて、

入力データである n ビットデータを分割数 d で分割した n/d ビットデータを前記各ラインに入力して、前記ラウンド関数を適用したデータ変換処理を含む演算をラウンド演算として繰り返し実行させ、

ラウンド演算の出力データを有するラインの n/d ビットデータを $d/2$ 個に再分割し、該再分割データを再度組み合わせて、前段のラウンド演算の出力データと異なる d 個の n/d ビットデータを再構成して、次段のラウンド演算の入力データとして設定する処理を実行させるプログラム。