(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2012-215813
(P2012-215813A)
(43)公開日　平成24年11月8日(2012.11.8)

(51)Int.Cl.		F I			テーマコード（参考）
G09C 1/00 (2006.01)		G09C	1/00	610A	5J104

審査請求 未請求　請求項の数 13　ＯＬ　（全 38 頁）

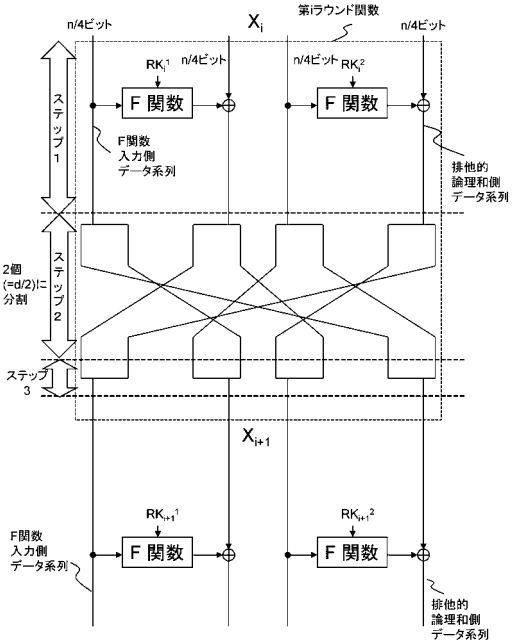
(21)出願番号　特願2011-207702 (P2011-207702)
(22)出願日　平成23年9月22日 (2011.9.22)
(31)優先権主張番号　特願2011-69182 (P2011-69182)
(32)優先日　平成23年3月28日 (2011.3.28)
(33)優先権主張国　日本国 (JP)

(71)出願人　000002185
ソニー株式会社
東京都港区港南１丁目７番１号
(74)代理人　100093241
弁理士　宮田　正昭
(74)代理人　100101801
弁理士　山田　英治
(74)代理人　100086531
弁理士　澤田　俊夫
(74)代理人　100095496
弁理士　佐々木　榮二
(74)代理人　110000763
特許業務法人大同特許事務所

最終頁に続く

(54)【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

(57)【要約】
【課題】拡散（ｄｉｆｆｕｓｉｏｎ）特性を向上させた安全性の高い暗号処理を実現する。
【解決手段】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。本構成により拡散特性を向上させた安全性の高い暗号処理が実現される。
【選択図】図２０



【特許請求の範囲】
【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Ｆｅｉｓｔｅｌ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×（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（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（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×（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。
【発明の詳細な説明】

【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，　ＳｐｒｉｎｇｅｒＶｅｒｌａｇ，　１９９６，　ｐｐ．９１－－１０４．
【非特許文献２】Ｙｕｌｉａｎｇ　Ｚｈｅｎｇ，　Ｔｓｕｔｏｍｕ　Ｍａｔｓｕｍｏｔｏ，　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×（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×（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（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。

【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】ＳＰＮ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】Ｆｅｉｓｔｅｌ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Ｆｅｉｓｔｅｌ構造の一例について説明する図である。

【図８】拡張Ｆｅｉｓｔｅｌ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】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１２】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】Ｆｅｉｓｔｅｌ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】Ｆｅｉｓｔｅｌ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－ｌｉｎｅ構造の一方式を示す図である。
【図３５】インボリ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－ｌｉｎｅ（ｄ＝４）の構成において、インボリ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－ｌｉｎｅ（ｄ＝６）の構成において、インボリ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（ｄｉｆｆｕｓｉｏｎ）特性の概要について
（２－１）拡散（ｄｉｆｆｕｓｉｏｎ）特性についての説明
（２－２）拡散（ｄｉｆｆｕｓｉｏｎ）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（ｄｉｆｆｕｓｉｏｎ）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（ｉｎｖｏｌｕｔｉｏｎ）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りうる。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Ｆｅｉｓｔｅｌ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Ｆｅｉｓｔｅｌ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ｍｓ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－ｂｏｘ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ｍｓビット入出力をもち、ＧＦ（2^s）の上で定義されるｍ×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（ｄｉｆｆｕｓｉｏｎ）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（ｄｉｆｆｕｓｉｏｎ）特性の概要について説明する。

【００４２】

（２－１）拡散（ｄｉｆｆｕｓｉｏｎ）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。

【００４３】

以下に、「ｄｉｆｆｕｓｉｏｎ状態」，「ｆｕｌｌ　ｄｉｆｆｕｓｉｏｎ状態」，「ｆｕｌｌ　ｄｉｆｆｕｓｉｏｎ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】

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

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

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

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

【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関数を通過していないため，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関数を通過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（ｄｉｆｆｕｓｉｏｎ）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でも全拡散（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．Ｓｕｚａｋｉ，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（ｄｉｆｆｕｓｉｏｎ）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（ｄｉｆｆｕｓｉｏｎ）状態となる。

【００７７】

以上の結果から、本方式では、全拡散（ｆｕｌｌ　ｄｉｆｆｕｓｉｏｎ）ラウンド数４を満たすことが可能になる。図２２は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。この構成例では従来構成の全拡散（ｆｕｌｌ　ｄｉｆｆｕｓｉｏｎ）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２２にこの構成におけるｄ＝４の場合のｆｕｌｌ　ｄｉｆｆｕｓｉｏｎを満たすパスの一例を示す。

【００７８】

また、図２３にｄ＝６とした場合の構成例、図２４にその場合の全拡散（ｆｕｌｌ　ｄｉｆｆｕｓｉｏｎ）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（ｄｉｆｆｕｓｉｏｎ）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×（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ト数、

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。
ラウンド数＝３＋［ｌｏｇ$_{p}$（ｄ／２）］
ただし、［ｘ］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、前述の各ラウント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］｜｜Ｙ$_{i}$［１×ｄ／２＋１］｜｜Ｙ$_{i}$［２×ｄ／２＋１］｜｜…｜｜Ｙ$_{i}$［（ｄ－１）×ｄ／２＋１］、
ＹＹ$_{i}$［２］＝Ｙ$_{i}$［２］｜｜Ｙ$_{i}$［１×ｄ／２＋２］｜｜Ｙ$_{i}$［２×ｄ／２＋２］｜｜…｜｜Ｙ$_{i}$［（ｄ－１）×ｄ／２＋２］、
…
ＹＹ$_{i}$［ｄ／２］＝Ｙ$_{i}$［ｄ／２］｜｜Ｙ$_{i}$［１×ｄ／２＋ｄ／２］｜｜Ｙ$_{i}$［２×ｄ／２＋ｄ／２］｜｜…｜｜Ｙ$_{i}$［（ｄ－１）×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、
ＺＺ$_{i+1}$［１］＝ＸＸ$_{i+1}$［１］＜＜＜１となるので、
ＺＺ$_{i+1}$［１］＝ＸＸ$_{i+1}$［１］＜＜＜１

$$=(X_{i+1}[1]\,||\,X_{i+1}[1\times d/2+1]\,||\,X_{i+1}[2\times d/2+1]\,||\,\cdots\,||\,X_{i+1}[(d-1)\times d/2+1])<<<1$$
$$=X_{i+1}[1\times d/2+1]\,||\,X_{i+1}[2\times d/2+1]\,||\cdots||\,X_{i+1}[(d-1)\times d/2+1]\,||\,X_{i+1}[1]$$

となる。

【０１０４】

以上のように定義されたＹＹ$_i$［ｊ］、およびＺＺ$_{i+1}$［ｊ］を用い、各ＹＹ$_i$［ｊ］より重複なくひとつずつＺＺ$_{i+1}$［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。

ＹＹ$_i$［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。

ＸＸ$_{i+1}$［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。

ＺＺ$_{i+1}$［ｔ］：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。

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ＹＹ$_i$［ｔ］：ＹＹ$_i$［１］、ＹＹ$_i$［２］、ＹＹ$_i$［３］、

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

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

ＹＹ$_i$［１］＝Ｙ$_i$［１］||Ｙ$_i$［４］||Ｙ$_i$［７］||Ｙ$_i$［１０］||Ｙ$_i$［１３］||Ｙ$_i$［１６］、

ＹＹ$_i$［２］＝Ｙ$_i$［２］||Ｙ$_i$［５］||Ｙ$_i$［８］||Ｙ$_i$［１１］||Ｙ$_i$［１４］||Ｙ$_i$［１７］、

ＹＹ$_i$［３］＝Ｙ$_i$［３］||Ｙ$_i$［６］||Ｙ$_i$［９］||Ｙ$_i$［１２］||Ｙ$_i$［１５］||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ＸＸ$_{i+1}$［ｔ］：ＸＸ$_{i+1}$［１］、ＸＸ$_{i+1}$［２］、ＸＸ$_{i+1}$［３］は、以下の設定となる。

ＸＸ$_{i+1}$［１］＝Ｘ$_{i+1}$［１］||Ｘ$_{i+1}$［４］||Ｘ$_{i+1}$［７］||Ｘ$_{i+1}$［１０］||Ｘ$_{i+1}$［１３］||Ｘ$_{i+1}$［１６］、

ＸＸ$_{i+1}$［２］＝Ｘ$_{i+1}$［２］||Ｘ$_{i+1}$［５］||Ｘ$_{i+1}$［８］||Ｘ$_{i+1}$

$_{1}$［１１］｜｜X_{i+1}［１４］｜｜X_{i+1}［１７］、
XX_{i+1}［３］＝X_{i+1}［３］｜｜X_{i+1}［６］｜｜X_{i+1}［９］｜｜X_{i+1}［１２］｜｜X_{i+1}［１５］｜｜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ZZ_{i+1}［ｔ］：　ZZ_{i+1}［１］、ZZ_{i+1}［２］、ZZ_{i+1}［３］は、以下の設定となる。
ZZ_{i+1}［１］＝X_{i+1}［４］｜｜X_{i+1}［７］｜｜X_{i+1}［１０］｜｜X_{i+1}［１３］｜｜X_{i+1}［１６］｜｜X_{i+1}［１］、
ZZ_{i+1}［２］＝X_{i+1}［１１］｜｜X_{i+1}［１４］｜｜X_{i+1}［１７］｜｜X_{i+1}［２］｜｜X_{i+1}［５］｜｜X_{i+1}［８］、
ZZ_{i+1}［３］＝X_{i+1}［１８］｜｜X_{i+1}［３］｜｜X_{i+1}［６］｜｜X_{i+1}［９］｜｜X_{i+1}［１２］｜｜X_{i+1}［１５］。

【０１１０】

ここで、図３０に示すように、例えばYY_{i}［１］とZZ_{i+1}［１］を、YY_{i}［２］とZZ_{i+1}［２］をYY_{i}［３］とZZ_{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YY_{i}［１］とZZ_{i+1}［２］を、YY_{i}［２］とZZ_{i+1}［３］をYY_{i}［３］とZZ_{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×（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×ｄ／２種類のデータ系列をｄ／２×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ため、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造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（Ｇｅｎｅｒａｌｉｚｅｄ　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Ｆｅｉｓｔｅｌ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）、…、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）、…、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）$_{1}$、…、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）ｍｏｄ　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）ｍｏｄ　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）ｍｏｄ　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、ラウント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、暗号処理に必要となる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×（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×（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

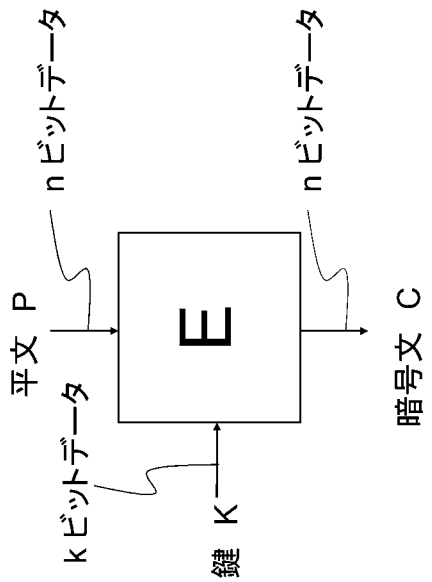
変換処理を含む演算をラウンド演算として繰り返し実行する。ラウンド演算の出力データを有するラインのｎ／ｄビットデータをｄ／２個に分割し、該分割データを組み合わせて、前段のラウンド演算の出力データと異なるｄ個のｎ／ｄビットデータを再構成して、次段のラウンド演算の入力データとして処理を実行する。この構成によって、拡散（ｄｉｆｆｕｓｉｏｎ）特性を向上させた安全性の高い暗号処理が実現される。

【符号の説明】

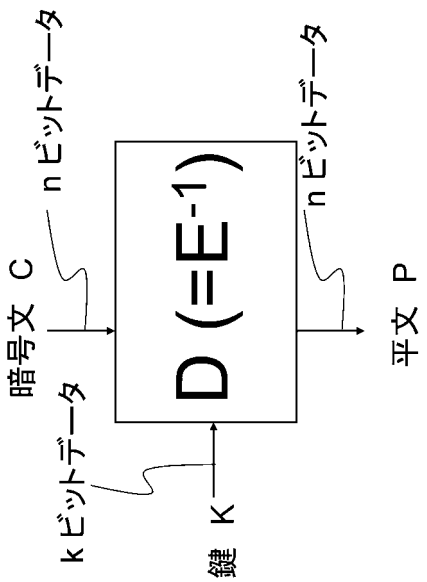
【０１５９】

７００　ＩＣモジュール
７０１　ＣＰＵ（Ｃｅｎｔｒａｌ　ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ）
７０２　メモリ
７０３　暗号処理部
７０４　乱数生成部
７０５　送受信部

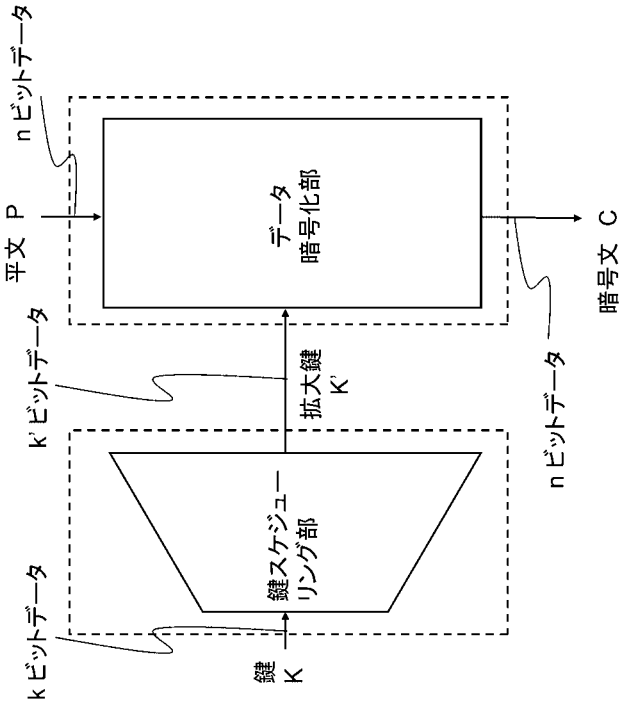
【図１】



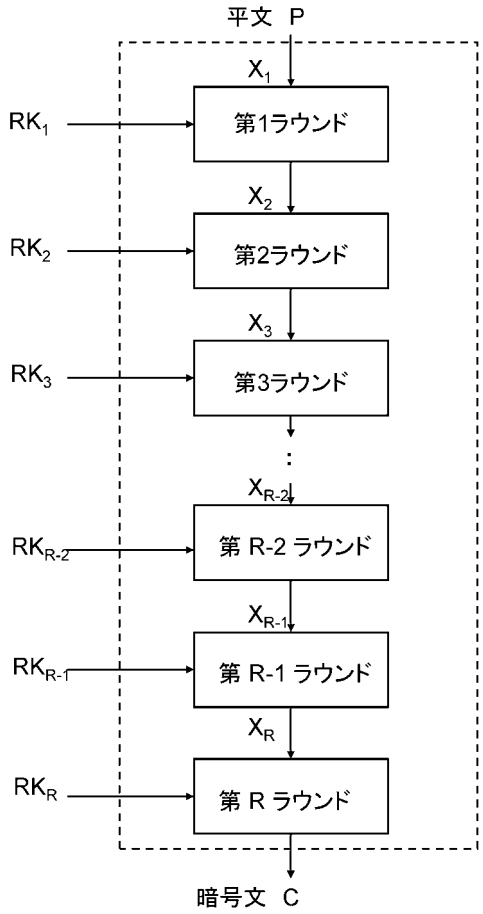
【図２】



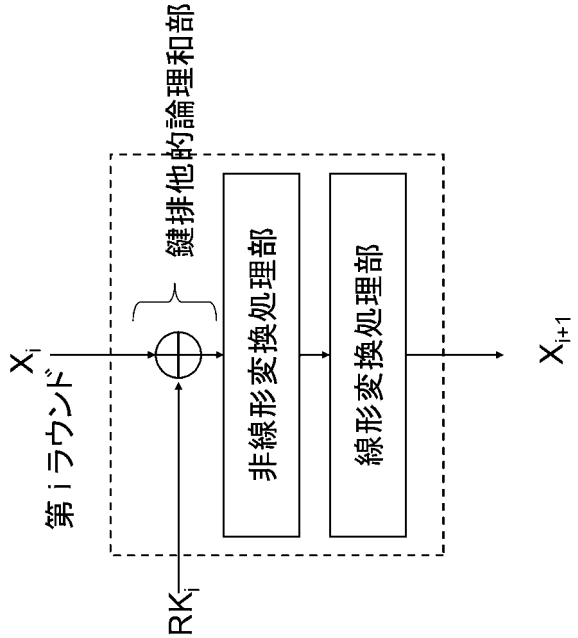
【図３】


nビットデータ
平文 P
データ
暗号化部
暗号文 C
k'ビットデータ
拡大鍵
K'
nビットデータ
鍵スケジュー
リング部
kビットデータ
鍵
K

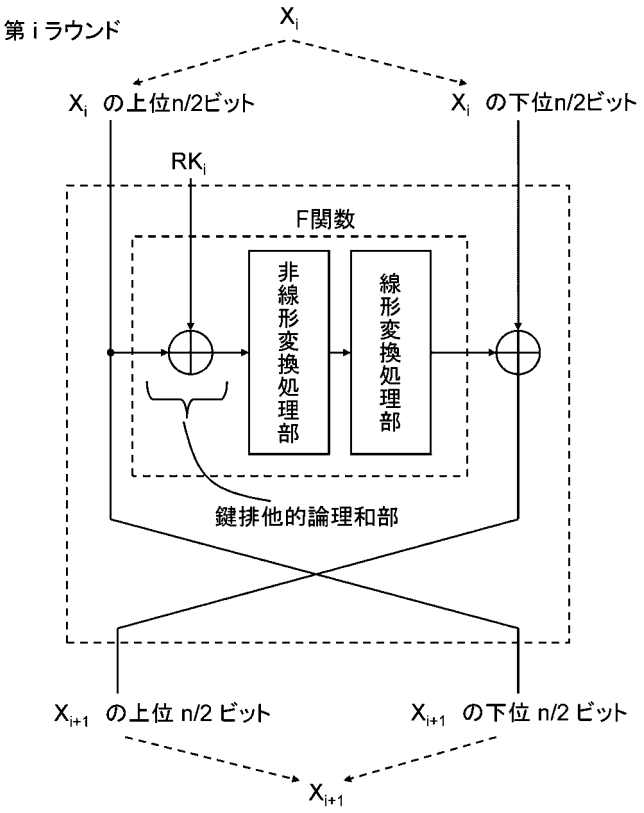
【図４】


平文 P
X1
RK1
第1ラウンド
X2
RK2
第2ラウンド
X3
RK3
第3ラウンド
XR-2
RKR-2
第 R-2 ラウンド
XR-1
RKR-1
第 R-1 ラウンド
XR
RKR
第 R ラウンド
暗号文 C

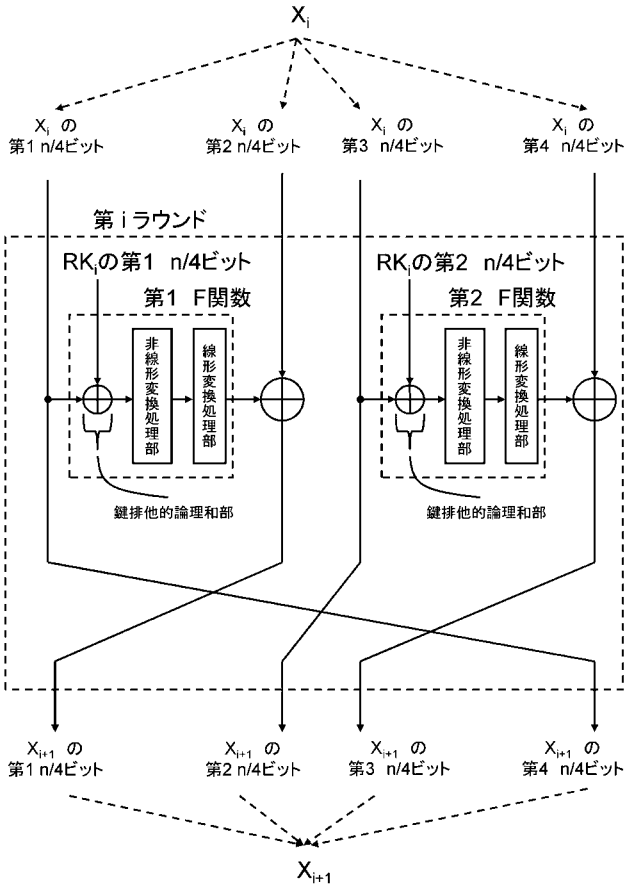
【図５】


鍵排他的論理和部
Xi
第 i ラウンド
非線形変換処理部
線形変換処理部
Xi+1
RKi

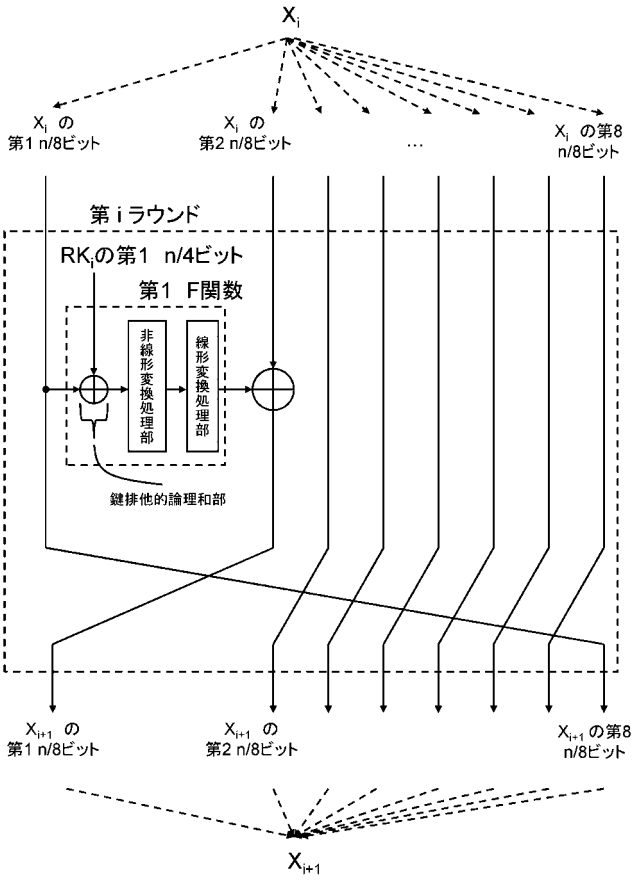
【図６】


第 i ラウンド
Xi
Xi の上位n/2ビット
Xi の下位n/2ビット
RKi
F関数
非線形変換処理部
線形変換処理部
鍵排他的論理和部
Xi+1 の上位 n/2 ビット
Xi+1 の下位 n/2 ビット
Xi+1

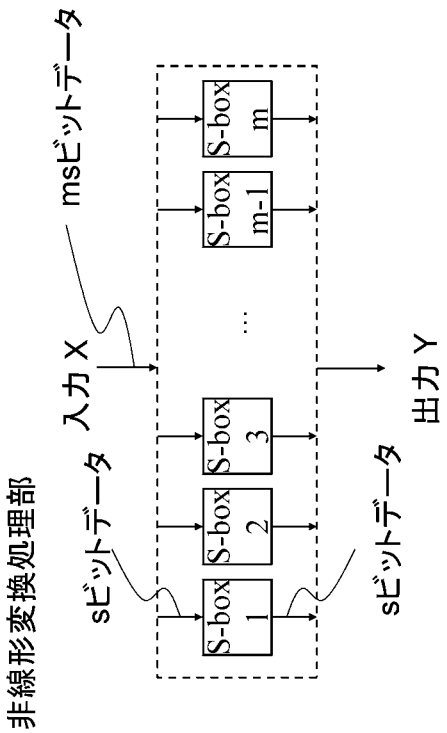
【図７】


Xi
Xi の 第1 n/4ビット
Xi の 第2 n/4ビット
Xi の 第3 n/4ビット
Xi の 第4 n/4ビット
第 i ラウンド
RKiの第1 n/4ビット
第1 F関数
非線形変換処理部
線形変換処理部
鍵排他的論理和部
RKiの第2 n/4ビット
第2 F関数
非線形変換処理部
線形変換処理部
鍵排他的論理和部
Xi+1 の 第1 n/4ビット
Xi+1 の 第2 n/4ビット
Xi+1 の 第3 n/4ビット
Xi+1 の 第4 n/4ビット
Xi+1

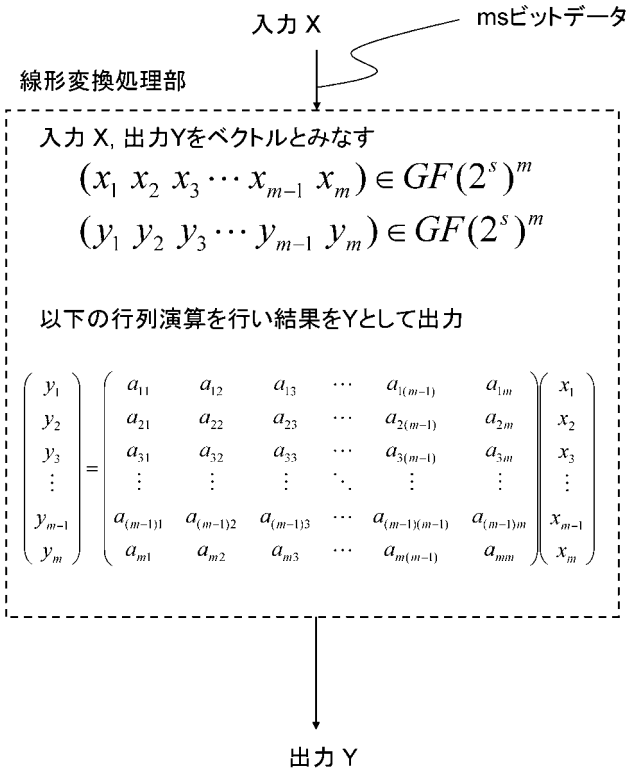
【図８】


Xi
Xi の 第1 n/8ビット
Xi の 第2 n/8ビット
…
Xi の第8 n/8ビット
第 i ラウンド
RKiの第1 n/4ビット
第1 F関数
非線形変換処理部
線形変換処理部
鍵排他的論理和部
Xi+1 の 第1 n/8ビット
Xi+1 の 第2 n/8ビット
Xi+1 の第8 n/8ビット
Xi+1

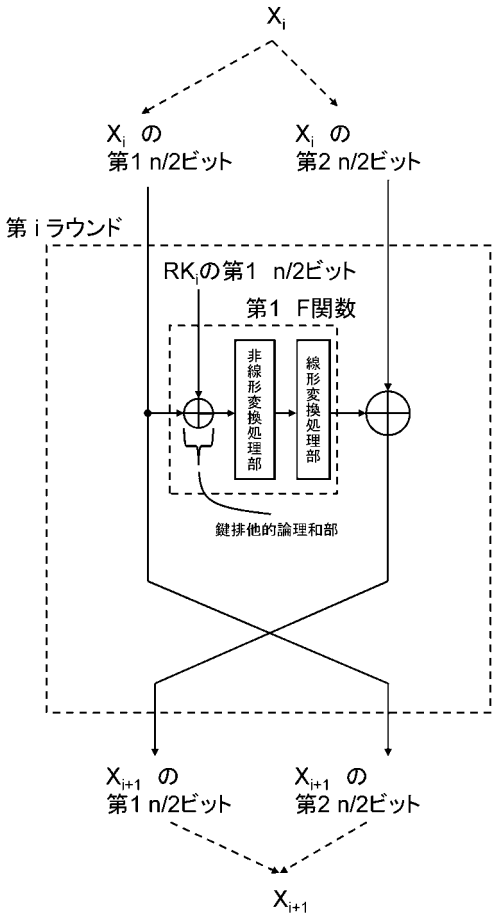
【図９】


msビットデータ
入力 X
sビットデータ
非線形変換処理部
S-box m
S-box m-1
S-box 3
S-box 2
S-box 1
出力 Y
sビットデータ

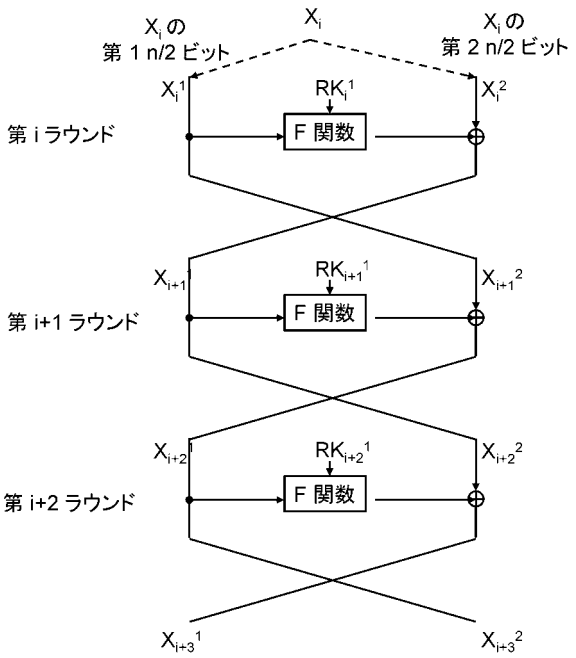
【図１０】


入力 X
msビットデータ
線形変換処理部
入力 X, 出力Yをベクトルとみなす
$(x_1\ x_2\ x_3 \cdots x_{m-1}\ x_m) \in GF(2^s)^m$
$(y_1\ y_2\ y_3 \cdots y_{m-1}\ y_m) \in GF(2^s)^m$
以下の行列演算を行い結果をYとして出力
$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(m-1)} & a_{1m} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2(m-1)} & a_{2m} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3(m-1)} & a_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{(m-1)1} & a_{(m-1)2} & a_{(m-1)3} & \cdots & a_{(m-1)(m-1)} & a_{(m-1)m} \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{m(m-1)} & a_{mm} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{m-1} \\ x_m \end{pmatrix}$
出力 Y

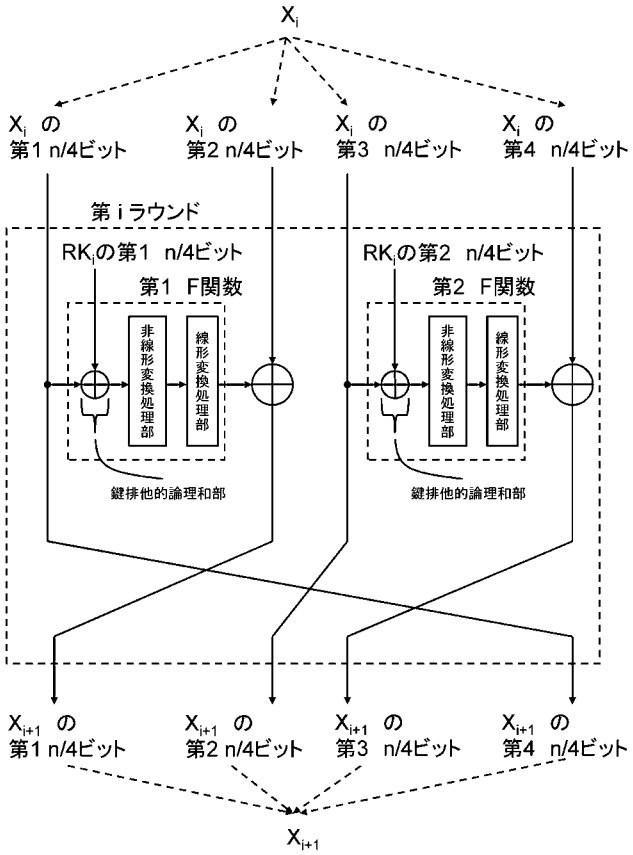
【図１１】



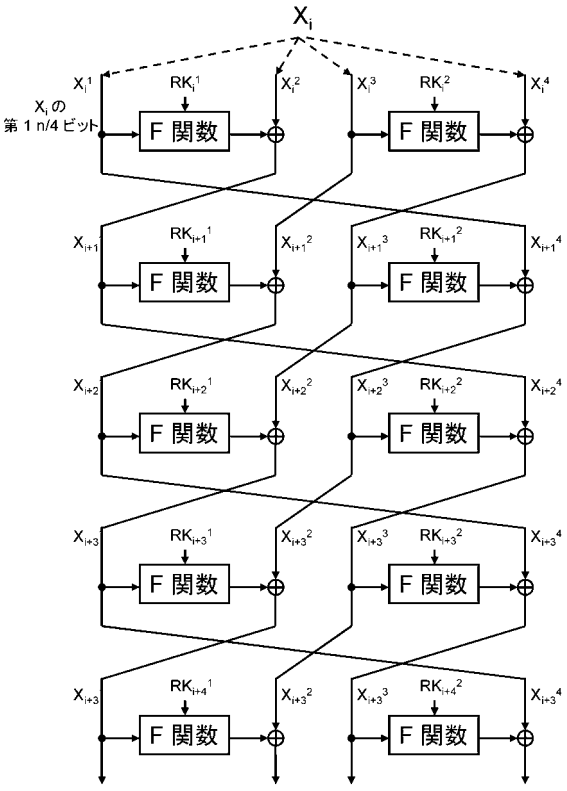
【図１２】



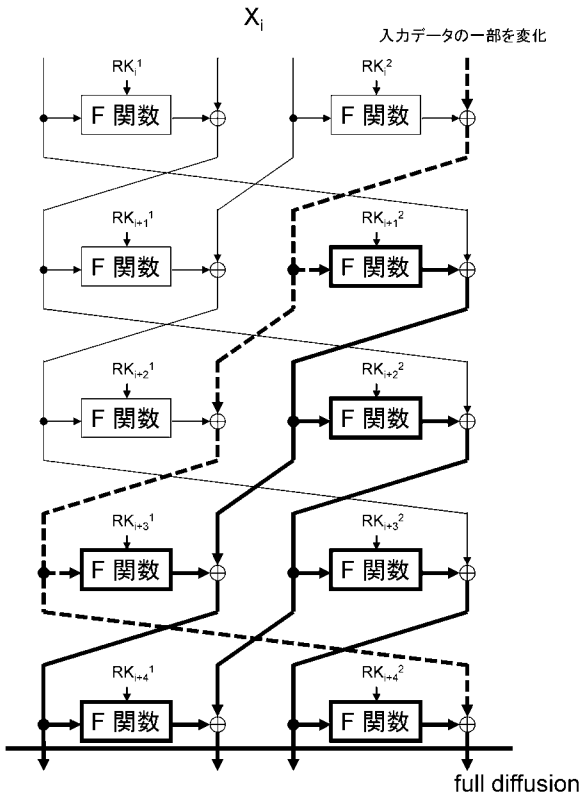
【図１３】



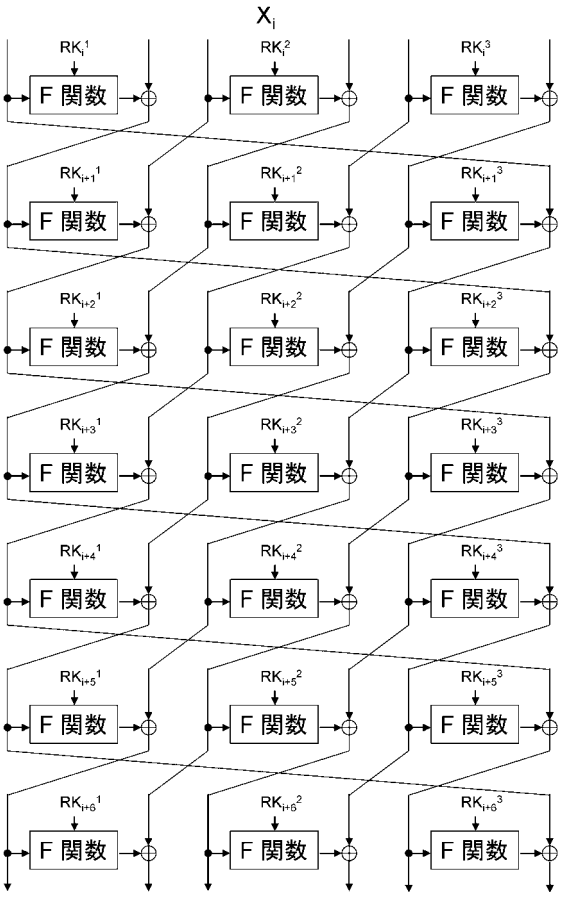
【図１４】



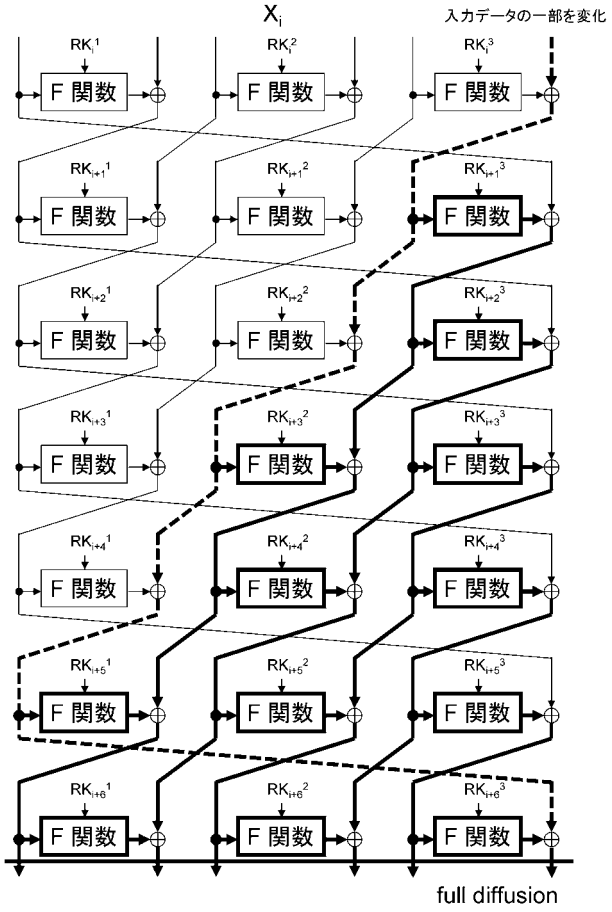
【図１５】



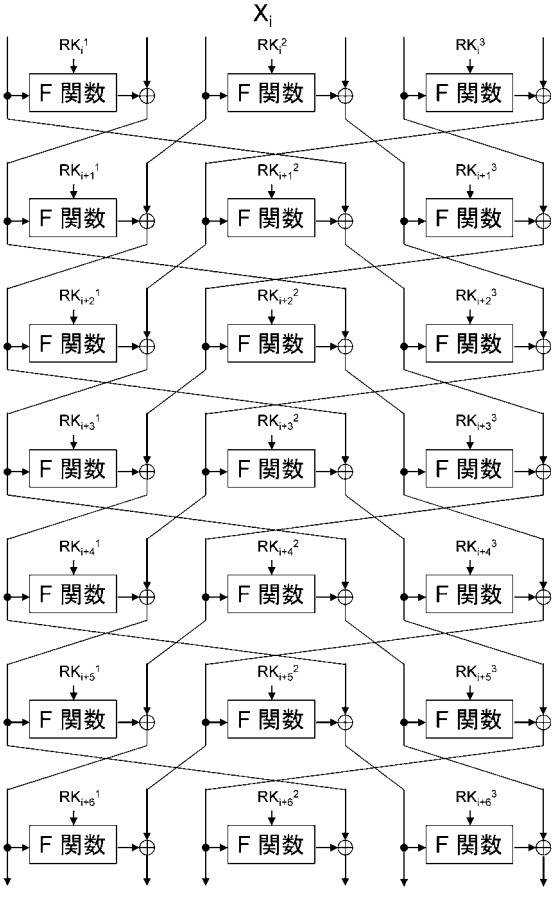
【図１６】



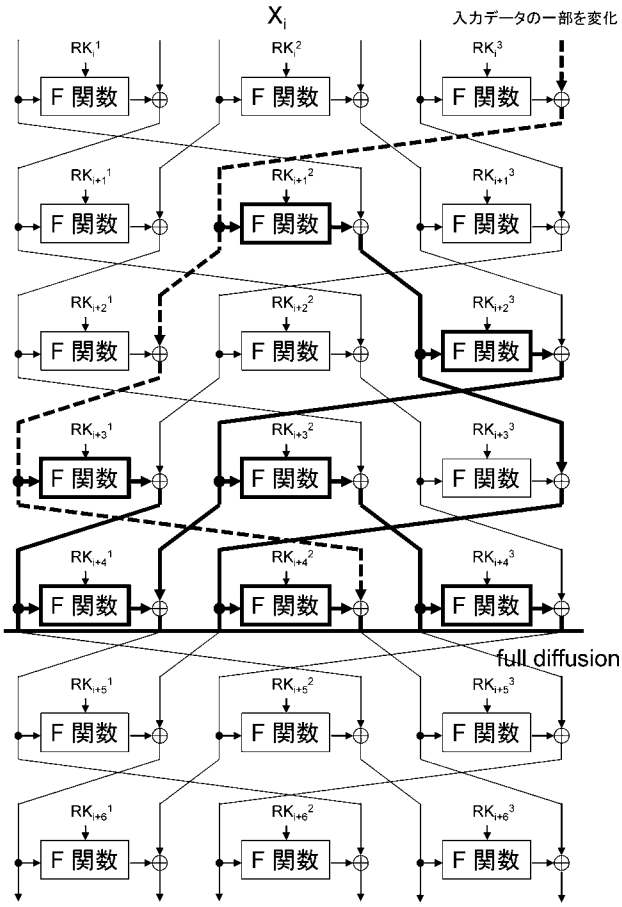
【図１７】



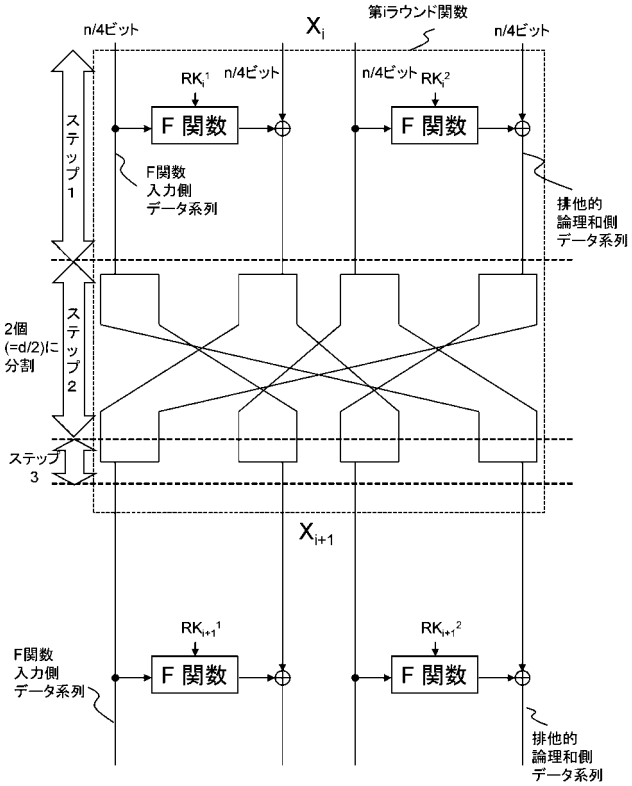
【図１８】



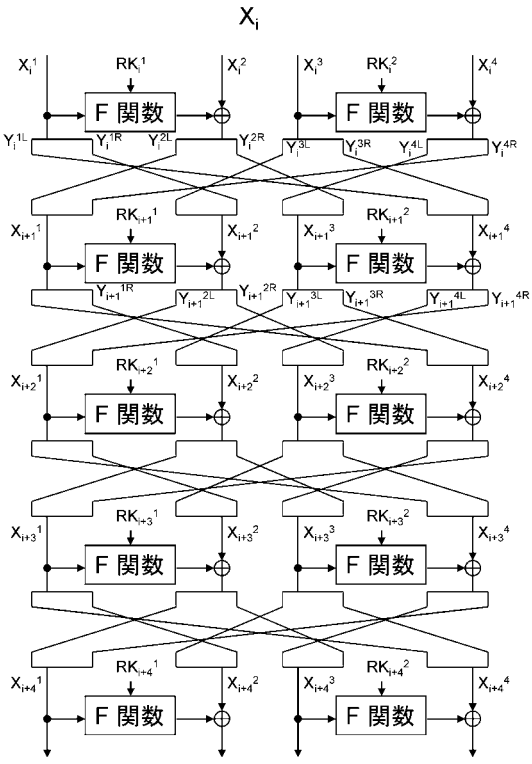
【図１９】



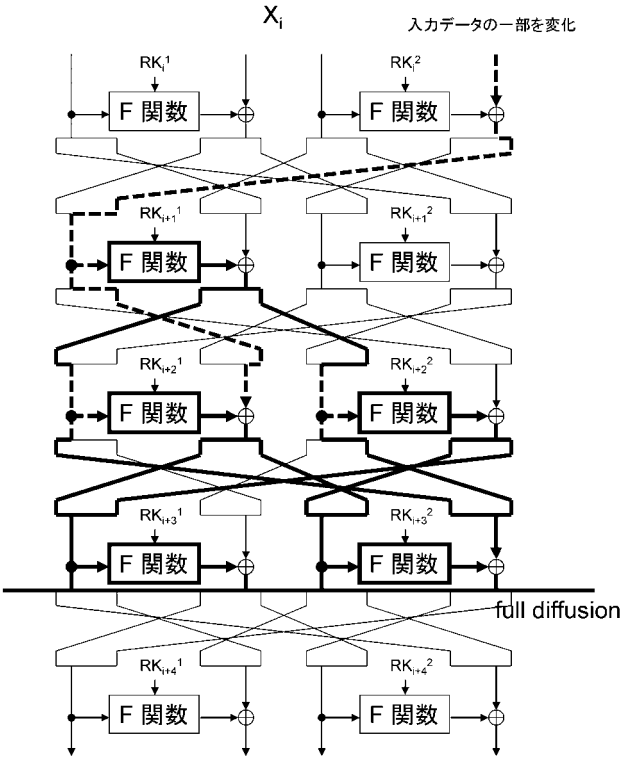
【図２０】



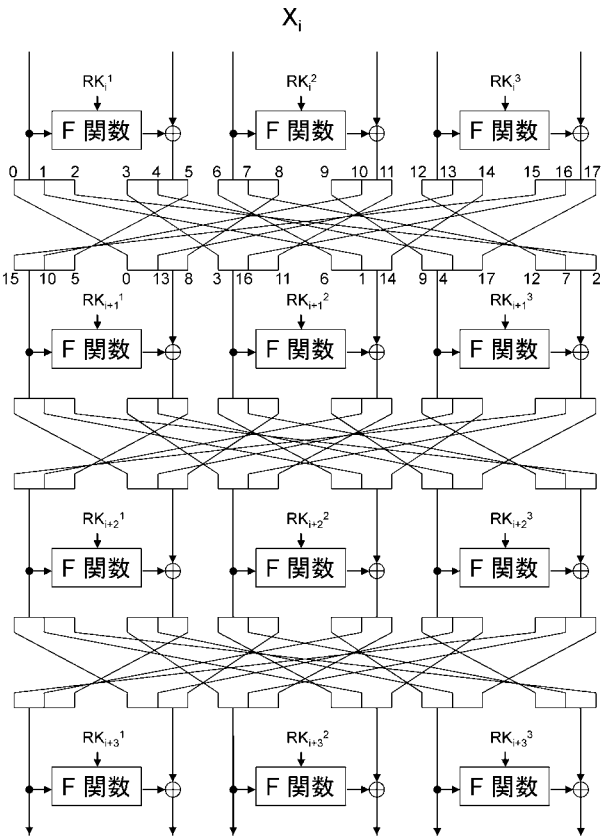
【図２１】



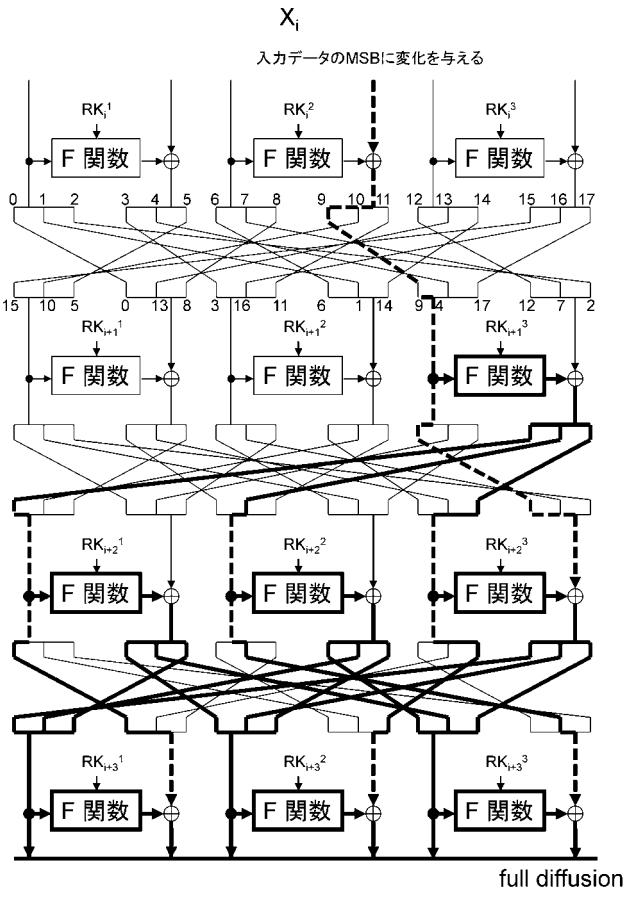
【図２２】



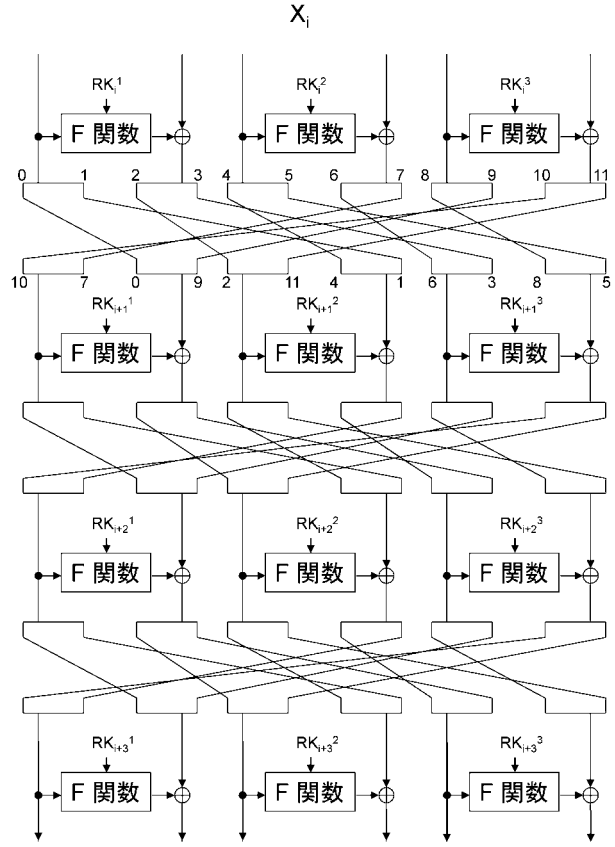
【図２３】



【図２４】



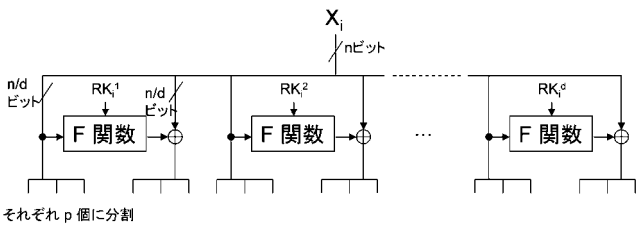
【図２５】



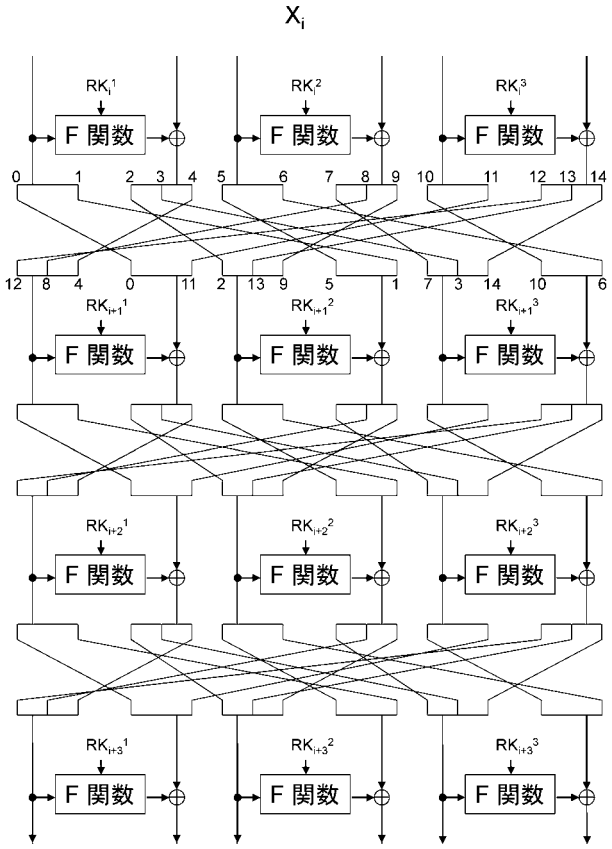
【図２６】

n ビットデータ分割数 d, d/n ビットデータ分割数 p と full diffusion ラウンド数の関係

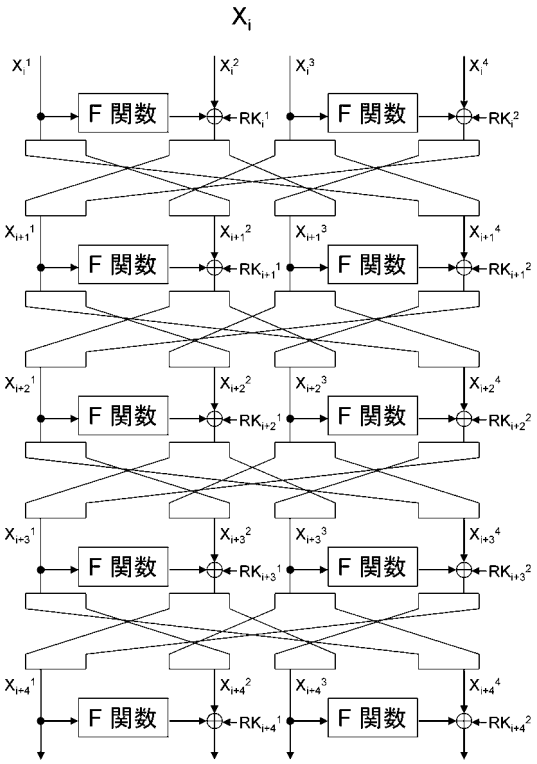
d \ p	2	3	4	5	6	7	8	9	10	11	12
4	4	-	-	-	-	-	-	-	-	-	-
6	5	4	-	-	-	-	-	-	-	-	-
8	5	5	4	-	-	-	-	-	-	-	-
10	6	5	5	4	-	-	-	-	-	-	-
12	6	5	5	5	4	-	-	-	-	-	-
14	6	5	5	5	5	4	-	-	-	-	-
16	6	5	5	5	5	5	4	-	-	-	-
18	7	5	5	5	5	5	5	4	-	-	-
20	7	6	5	5	5	5	5	5	4	-	-
22	7	6	5	5	5	5	5	5	5	4	-
24	7	6	5	5	5	5	5	5	5	5	4



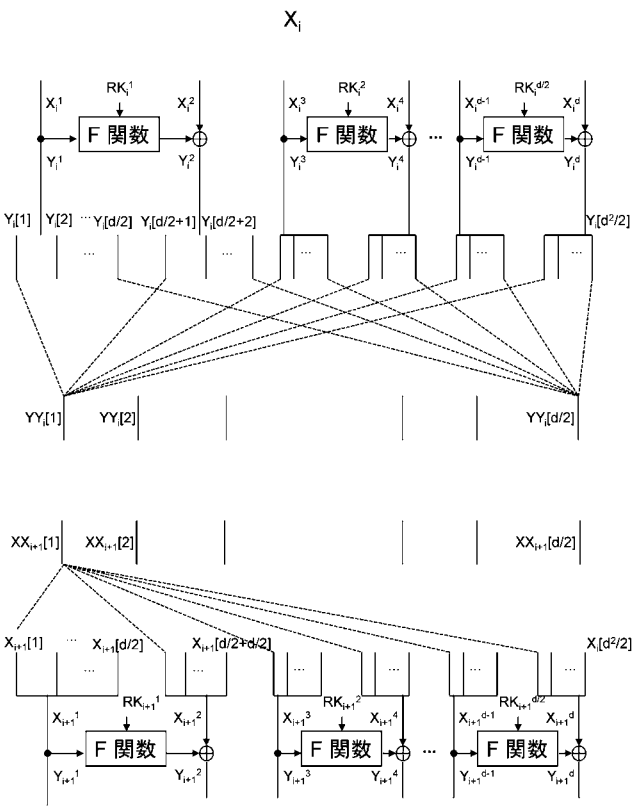
【図２７】



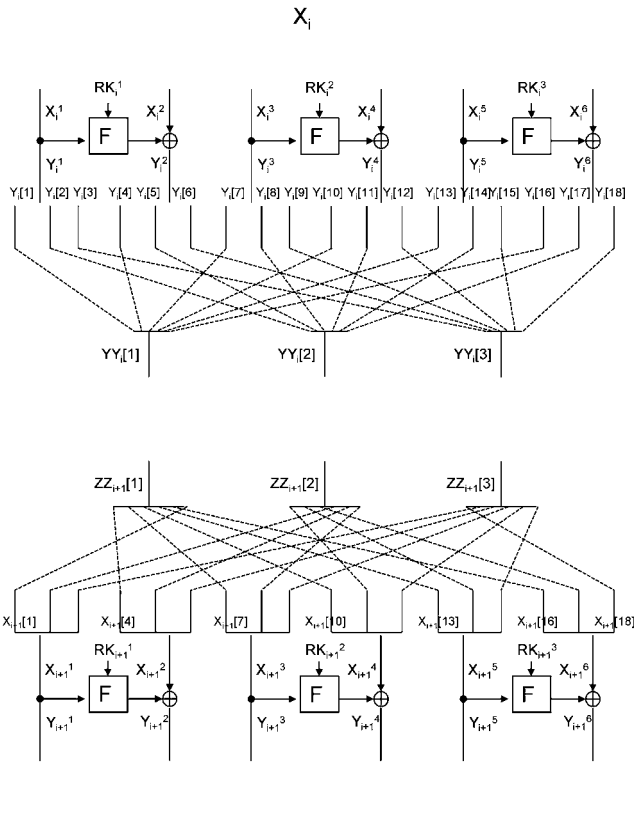
【図２８】



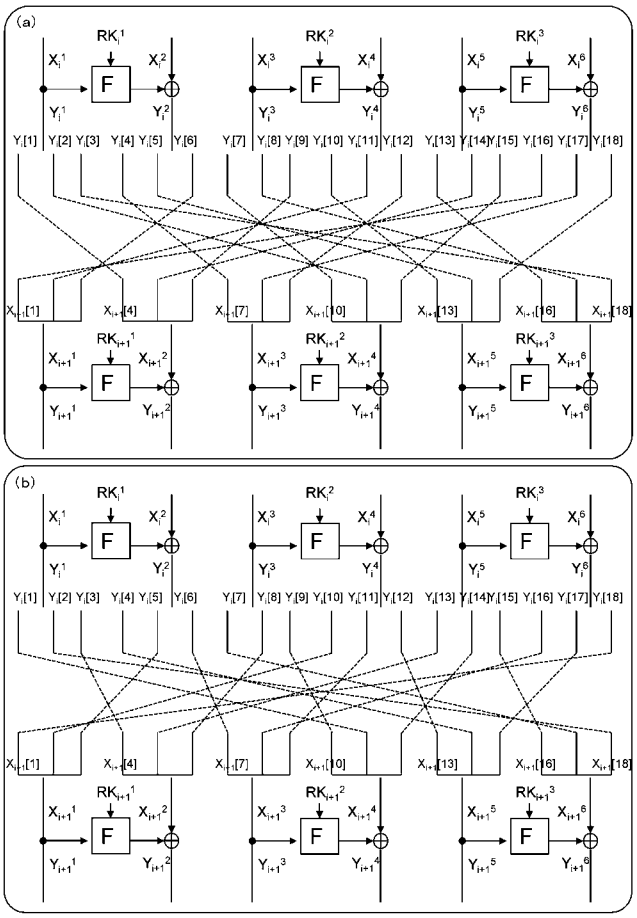
【図２９】



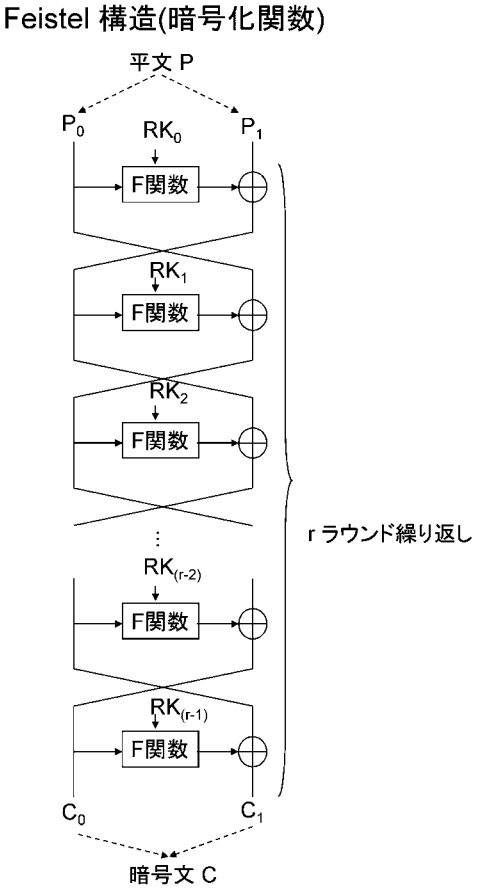
【図３０】



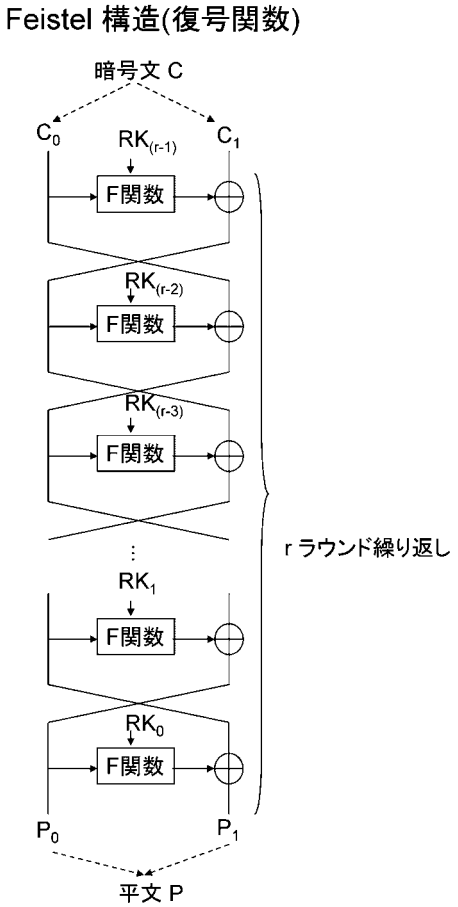
【図３１】



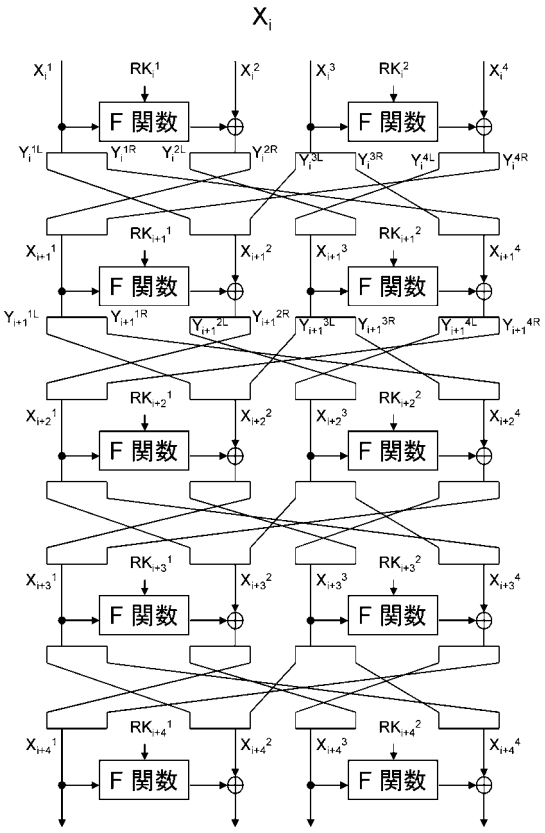
【図３２】



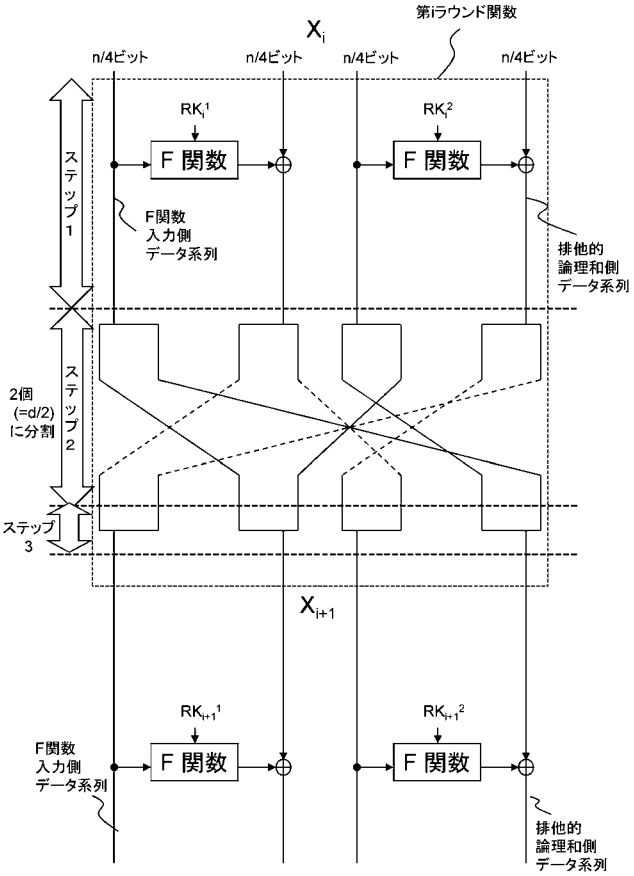
【図３３】



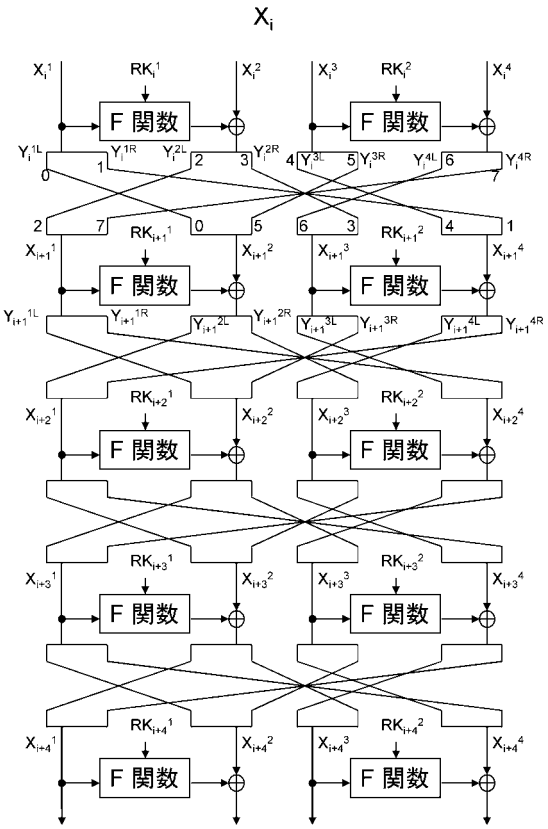
【図３４】



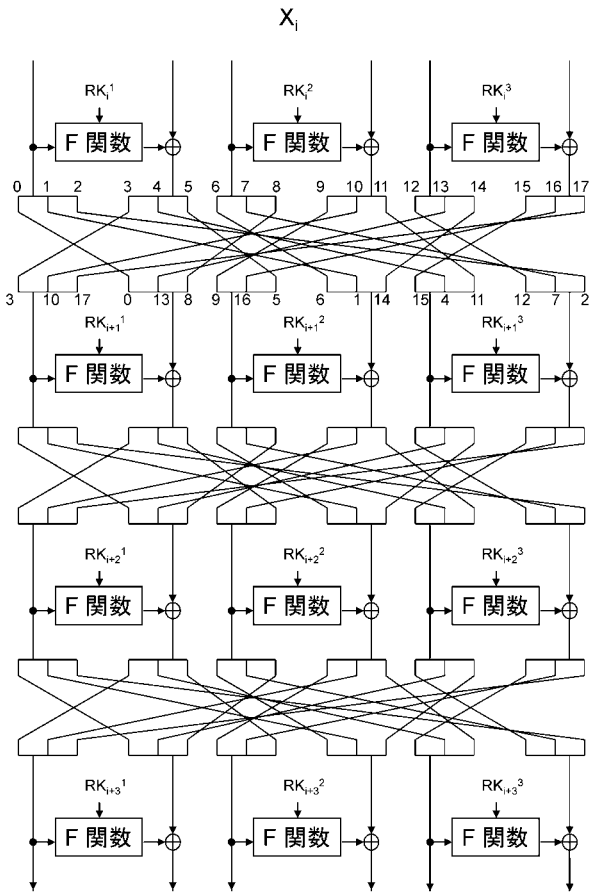
【図３５】



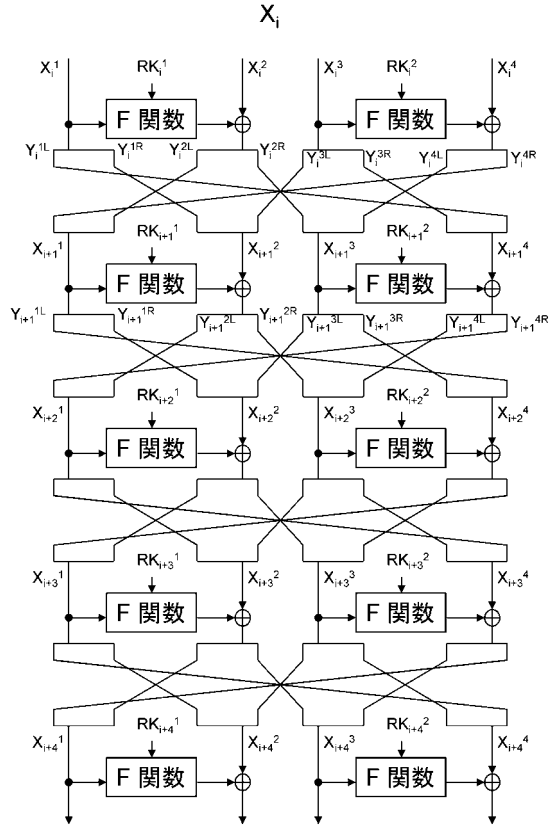
【図３６】



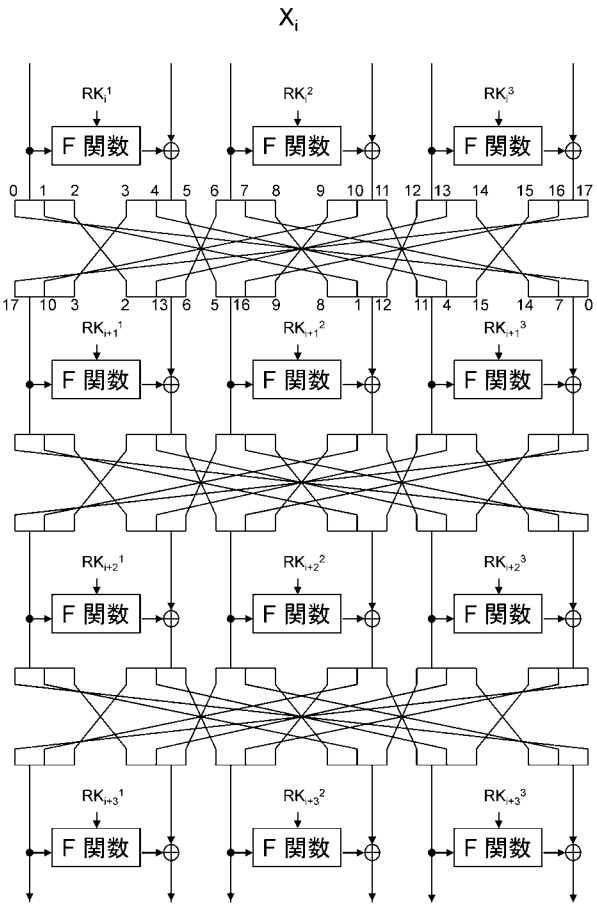
【図３７】



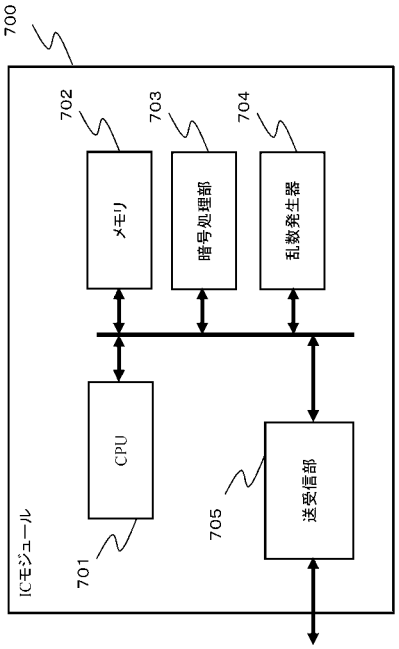
【図３８】



【図３９】


X_i
RK_i^1
RK_i^2
RK_i^3
F 関数
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
17 10 3 2 13 6 5 16 9 8 1 12 11 4 15 14 7 0
RK_{i+1}^1
RK_{i+1}^2
RK_{i+1}^3
RK_{i+2}^1
RK_{i+2}^2
RK_{i+2}^3
RK_{i+3}^1
RK_{i+3}^2
RK_{i+3}^3

【図４０】


700
702
703
704
メモリ
暗号処理部
乱数発生器
CPU
705
送受信部
701
ICモジュール

フロントページの続き

(72)発明者　渋谷　香士
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　秋下　徹
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　五十部　孝典
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　白井　太三
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　樋渡　玄良
東京都港区港南１丁目７番１号　ソニー株式会社内
(72)発明者　三津田　敦司
東京都港区港南１丁目７番１号　ソニー株式会社内

Ｆターム(参考) 5J104 JA09　NA02　NA09　NA10　NA39