学位申請論文

共通鍵ブロック暗号に対する攻撃手法の高速化とKASUMIの安全性評価への適用

平成２９年３月

杉尾　信行

目 次

図 目 次

表 目 次

第1章　序論

1.1　背景と目的

インターネットの普及により，パソコンや携帯電話を用いて電子情報のやり取りが容易になった．情報通信社会において，通信内容を第三者の脅威（通信内容の盗聴や改竄）から守る事は重要である．第三者の脅威を防ぐ技術として，暗号技術が活用されている（図 1.1）．

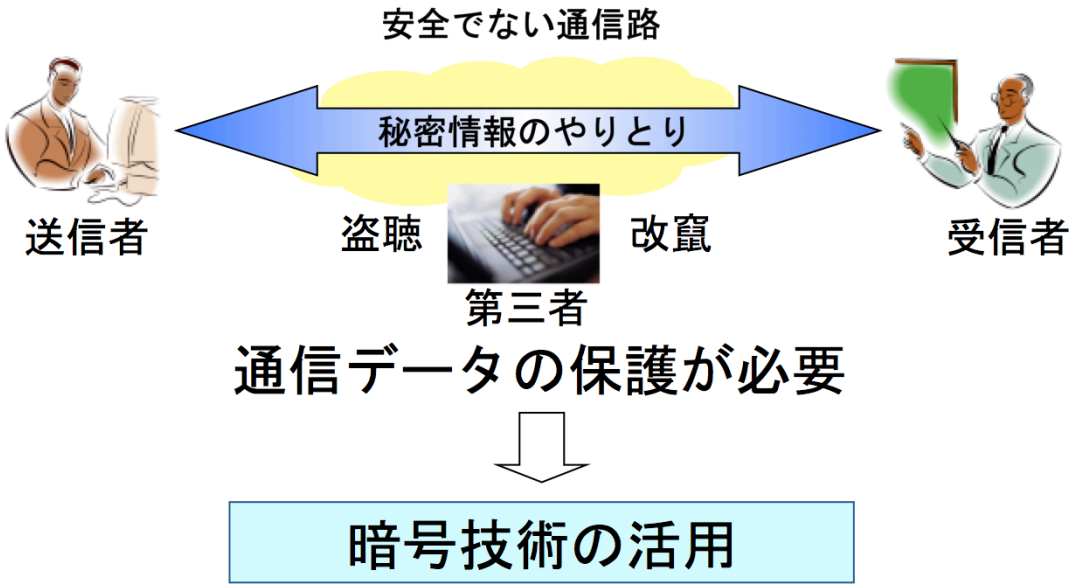

図 1.1: 暗号技術の活用

具体的には，第三者による盗聴に対しては通信内容の暗号化，改竄に対しては通信内容の完全性保証を実現する為に，暗号技術が活用されている．暗号技術は主に以下の 4 つに分類される．

- 共通鍵暗号（対称鍵暗号）: 送信者と受信者が同一の鍵（秘密鍵）を用いて暗号通信を行う方式
- 公開鍵暗号（非対称暗号）: 送信者と受信者が異なる鍵（秘密鍵，公開鍵）を用いて暗号通信を行う方式
- ハッシュ関数：長いビット列を比較的短いブロックに圧縮する関数

- 乱数生成：予測不可能な乱数を生成

通信内容の暗号化を行う代表的な共通鍵暗号アルゴリズムとして，AES（Advanced Encryption Standard）[36] が知られている．また，通信内容の完全性保証を実現する暗号アルゴリズムにメッセージ認証（HMAC, Keyed-Hashing for Message Authentication）[26] が知られている．第三者の脅威を防止する為に暗号技術を用いても，使用する暗号アルゴリズムに欠陥があれば，期待する安全性 (秘匿性や完全性) を保証する事が出来なくなる．従って，使用する暗号アルゴリズムが安全かどうかを正しく評価する事が重要である．日本では，暗号技術評価プロジェクト（CRYPTREC）が暗号技術に関する調査や評価を行い，電子政府における調達のために推奨すべき暗号リスト（CRYPTREC 暗号リスト）を公開している [10]．

1990 年代から徐々に普及し始めた移動体通信サービスは，現在，第二世代（２ G，GSM 方式），及び第三世代（３ G，W-CDMA 方式）システムが世界中で利用されており，国内においても主要ネットワークオペレータによる第三世代移動体通信サービスが提供されている．総務省が公開した「電気通信サービスの契約数及びシェアに関する四半期データの公表」に拠れば，2016 年 9 月時点で携帯電話の契約数は 1 億 5,955 万であり，その内３ G の契約数は 6,547 万である [59]．電気通信サービスの契約数及びシェアに関する四半期データの公表別紙に記載の「3G・LTE・PHS・BWA の各契約数の推移」を抜粋したものを図 1.2 に示す．３ G 携帯電話[1] と無線基地局間の無線通信において，第三者による通信データ (利用者の音声データや制御信号等) の盗聴・改竄を防止する為，共通鍵暗号 KASUMI を用いた秘匿・完全性保証が実現されている [1]．

(注意) 下記は電気通信サービスの契約数及びシェアに関する四半期データの公表 [59] 別紙，頁 1 の「3G・LTE・PHS・BWA の各契約数の推移」を抜粋したものである．

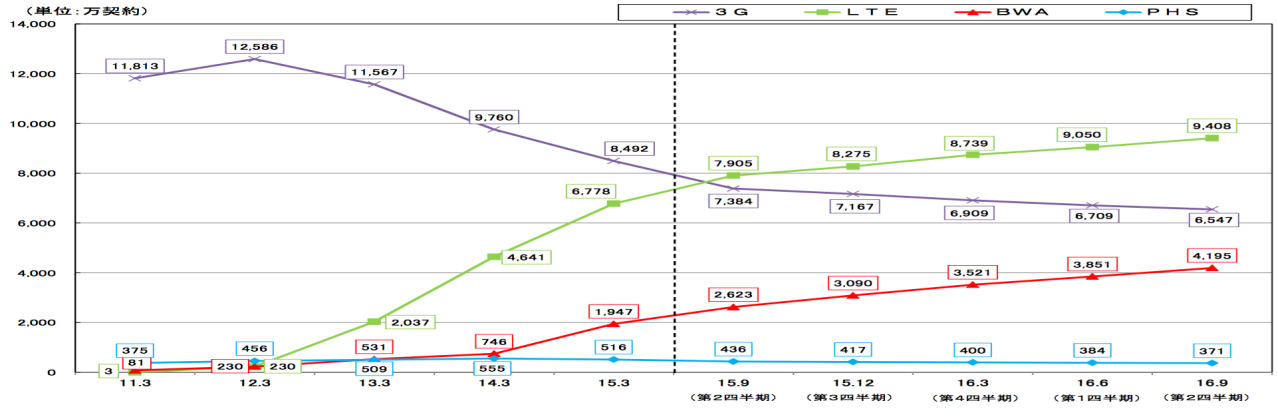

図 1.2: 3G・LTE・PHS・BWA の各契約数の推移

共通鍵暗号の安全性は暗号攻撃手法を用いて，どの程度耐性を有しているかで評価を行う．共

[1] ＬＴＥの携帯電話は下位互換性を有する為、３ G を用いた無線通信が可能である．

通鍵暗号に対する代表的な攻撃手法に差分攻撃 [5]，線形攻撃 [33] が知られている．近年の暗号はこれらの攻撃手法に配慮した設計がなされており，差分攻撃と線形攻撃に対して数学的に安全であること（差分攻撃と線形攻撃に対する証明可能安全性）を示した共通鍵暗号 MISTY[34] が提案された．共通鍵暗号 KASUMI はこの共通鍵暗号 MISTY を改良して開発されている為，差分攻撃と線形攻撃に対する耐性を有している．そこで本論文では，差分攻撃と線形攻撃以外の攻撃手法として強力，且つ汎用的に利用可能な高階差分攻撃 [28] と積分攻撃 [25] を取り扱い，攻撃手法の高速化を行う．適用例として KASUMI を挙げ，安全性評価を行う．また，高階差分攻撃と積分攻撃の関係性を明らかにする事を目的とする．

1.2 概要

共通鍵暗号 KASUMI に対する先行研究として，高階差分攻撃 [43, 48]，不能差分攻撃 [27]，及び，Multidimensional Zero-Correlation 攻撃 [55] 等が知られている．本論文では，高階差分攻撃を行う際の特性探索法と鍵回復の両面で高速化が可能な技術の提案と，その適用結果を示す．具体的には，田中らが示した 32 階差分の 4 段に対し，効果的な選択平文の探索を行う事で，16 階差分の 4 段特性が存在する事を示す．この手法を用いる事により，1 組の 4 段特性を構築する為に必要な選択平文数を最大 $\frac{1}{2^{16}}$ に削減可能である．また，鍵回復を行う際には，全数探索法よりも効率的な線形化手法 [42] を採用し，更に線形化手法の高速化を行う事で，5 段の KASUMI の鍵回復に必要な計算量を $2^{85.5}$ 倍高速化する事が可能である事を示す．線形化手法の高速化技術は，(1) 事前に方程式を解析し，鍵（未知項）の係数導出を高速化する技法と，(2) 高次項の鍵（未知項）の係数を 0 に制御し，導出する鍵（未知項）の数を削減する技法の 2 つから成る．この 2 つの高速化技術は，KASUMI 以外の共通鍵暗号にも適用可能な汎用性を有する技術である．

共通鍵暗号 MISTY に対する先行研究として，藤堂は積分特性を探索する新たな手法として Division 属性を新たに提案し，世界で初めてフルラウンドの MISTY1 が全数探索法よりも効率的に解読可能である事をを示した [50]．本論文では，この手法を用いて KASUMI の安全性評価を行い，7 段の KASUMI が 2^{63} の選択平文数と $2^{63.3}$ で攻撃可能である事を示す．

また，本論文では，高階差分攻撃と積分攻撃の関係性について明らかにする．m bit 入力の d 次関数に対し，Division 属性から導出される最良の積分攻撃は，高階差分攻撃の $(d+1)$ 階差分と等しく，2^{d+1} の選択平文数を必要とする．また，2^{d+1} の選択平文数の選び方は $2^{m-d-1}\prod_{i=0}^{d}\frac{(2^{m-i}-1)}{(2^{d+1-i}-1)}$ 通りであり，必要な選択平文数を最小とする積分攻撃は高階差分攻撃と一致する事を示す．

また，本論文で示す KASUMI の鍵回復攻撃に必要な計算量に対し，現在，及び今後の計算機能力の向上を見込んだ実行可能性に基づく安全性評価を行う．具体的には，今後の計算機能力の向上について，スーパーコンピューターのベンチマーク結果の 1 位から 500 位を 1993 年から半

年毎に集計している Web サイト TOP500.Org[51] と CRYPTREC が公表している報告書 [11] を参考にした．第 3 章で示す 5 段の KASUMI を攻撃する為に必要な計算量は $T = 2^{31.5}$ 回の暗号化計算量 [ENC.] であり，この計算量は汎用 PC の計算機で実行可能である．また，第 4 章で示す 7 段の KASUMI を攻撃する為に必要な計算量は $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] であり，この計算量は現在の世界 Top500 位のスーパーコンピューターが有する計算能力で実行可能な計算量である．従って，実際のシステムにおいて，7 段以下の KASUMI を用いる事は，秘密鍵に対する鍵回復攻撃の影響を受ける可能性がある為，危険である．尚，暗号解読技術は日々進化している為，今後も新たな攻撃手法に対する耐性を評価していく事が重要である．

1.3 本論文の構成

本論文の構成を以下に示す．第 2 章では，暗号の安全性評価に関わる予備知識について説明すると共に，本論文の研究対象である共通鍵暗号 KASUMI と，KASUMI の開発元暗号 MISTY について説明する．第 3 章では，高階差分攻撃の高速化技法の提案とその評価結果について纏める．第 4 章では，2015 年に新たに提案された積分特性を効率的に発見する手法（Division 属性）を用いた積分特性探索と，発見した新たな特性を用いた積分攻撃耐性評価を実施する．第 5 章では，高階差分攻撃と積分攻撃の関係性を明らかにする．第 6 章では，本論文の評価結果を加味し，共通鍵ブロック暗号 KASUMI の安全性評価を行う．最後に，第 7 章で，本論文を纏める．

第2章　暗号の安全性評価に関わる予備知識

本章では，暗号の安全性評価に関わる予備知識について説明する．また，本論文の研究対象である共通鍵暗号 KASUMI と MISTY について説明する．本論文で使用する代表的な表記法を表 2.1 に記す．また，第 2 章で用いる記号の一覧を表 2.2 に示す．

表 2.1: 本論文で使用する代表的な表記法

記号	説明
$\mathbb{Z}$	整数の全体
$+$	$\mathbb{Z}$ 上で行う算術加算
$GF(2)$	ガロア体（有限体）．2 つの数字 $\{0,1\}$ と mod 2 上の計算法を示す
$GF(2)^n$	2 つの数字 $\{0,1\}$ を n 個並べた値
$\oplus$	$GF(2)$ 上の加算で，排他的論理和（XOR）を示す
$\boldsymbol{a}$	小文字のボールド表記は任意のベクトルを示す
$\boldsymbol{A}$	大文字のボールド表記は任意の行列を示す

表 2.2: 第 2 章で用いる記号一覧

記号	説明
E	m bit 入力の暗号
P	平文 64 bit．上位 (左側)32 bit を P_L，下位 (右側)32 bit を P_R とする
C	暗号文 64 bit．上位 (左側)32 bit を C_L，下位 (右側)32 bit を C_R とする
K	秘密鍵 128 bit
L_i	i 段目 $(i = 1, 2, \ldots)$ 出力の上位 (左側)32 bit
R_i	i 段目 $(i = 1, 2, \ldots)$ 出力の下位 (右側)32 bit
$\|$	結合
$\lll i$	i bit $(i = 1, 2, \ldots)$ 左巡回シフト
$\cap$	bit 単位の AND 演算
$\cup$	bit 単位の OR 演算
$\mathcal{H}(X_R(i))$	R 段目出力 X_R の特徴量，$(i = 1, 2, \cdots)$

2.1 共通鍵ブロック暗号

共通鍵暗号は，事前に通信相手と秘密の情報 (秘密鍵) を共有し，暗号通信を行う方式である．共通鍵暗号は，更にブロック暗号とストリーム暗号に大別される．以下では，本論文で対象とするブロック暗号について説明を行う．

ブロック暗号は，メッセージを一定長のブロックに分割し，対応する暗号文ブロックに変換する方式である．米国の標準暗号である AES[36] が有名である．ブロック暗号の基本構造を図 2.1 に示す．

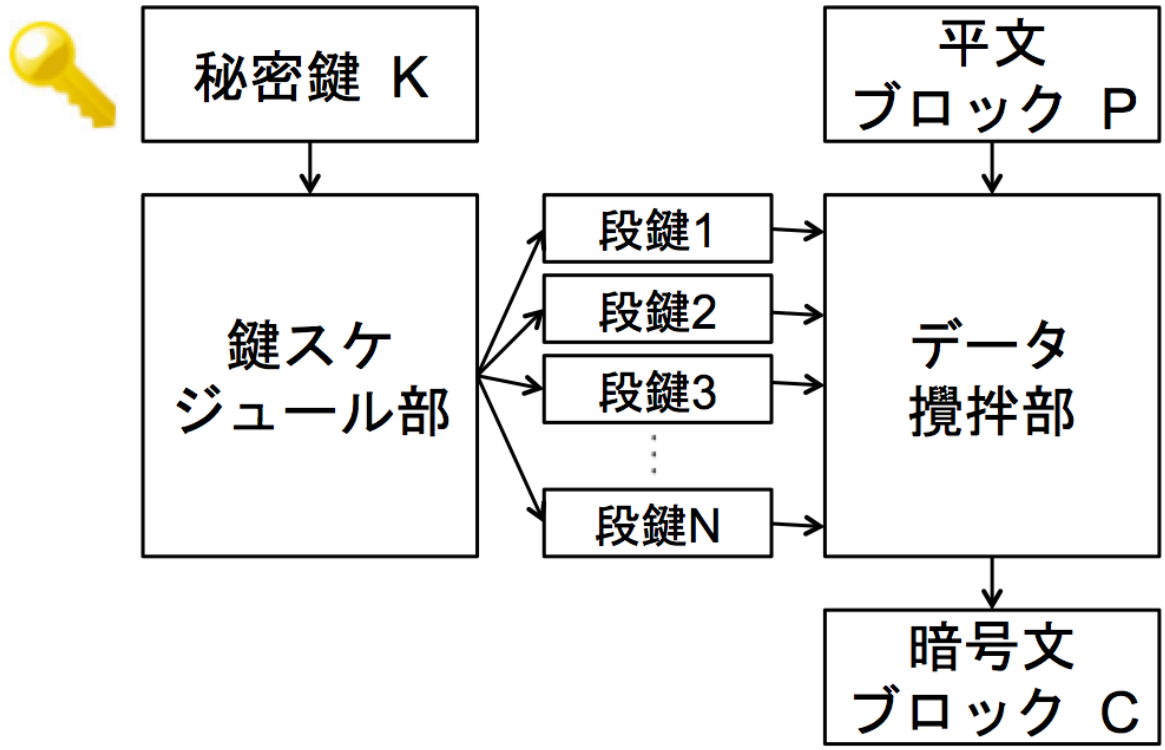

図 2.1: ブロック暗号の基本構造

ブロック暗号のブロックサイズは 64, 128 bit 等の固定長で，アルゴリズムによって異なる．秘密鍵は 128, 192, 256 bit 等であり，アルゴリズムによって異なる一つ，又は複数の秘密鍵長をサポートしている．通常，ブロック暗号は秘密鍵 K から鍵スケジュール部で段鍵 (副鍵や拡大鍵とも呼ばれる) を生成し，段鍵を用いて平文 P を攪拌する．ブロック暗号の代表的なデータ攪拌部の構造として，Feistel 構造，SPN（Substitution Permutation Network）構造が存在する．以下では，それぞれの構造とその特徴について説明する．

Feistel 構造を図 2.2 に示す．Feistel 構造の特徴として，以下の特徴が存在する．

- 暗号化・復号化が同じロジック，段鍵の順序を入れ替えるだけで暗号化，復号化が可能．

- F 関数 (段関数) の制約が少ない．復号時に逆関数を計算する必要が無い為，F 関数は全単射である必要がない．
- 1 ラウンドでブロックの半分しか変換されない．

Feistel 構造の代表例として DES[37] が有名である．

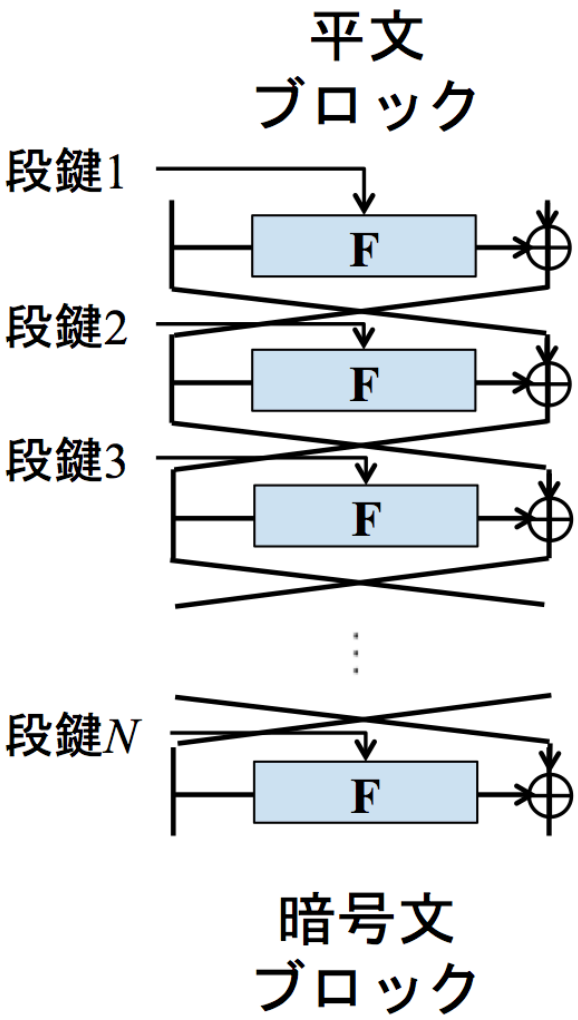

図 2.2: Feistel 構造

SPN 構造は，小さな非線形置換（Substitution）とそれを処理ブロック単位に拡大する転置（Permutation）からなる構造である．SPN 構造を図 2.3 に示す．SPN 構造の特徴として，以下の特徴が存在する．

- 暗号化，復号化が違うロジックで構成される．
- 段鍵の順序を変えるだけでは復号ができない．
- S 層 (Substitution 層) や P 層 (Permutation 層) は全単射である必要がある．

- 復号関数が暗号化関数の逆関数となる為の必須条件である．
- データ幅がブロック長と等しく，高速実装に向いている．

SPN 構造の代表例として AES[36] が有名である．

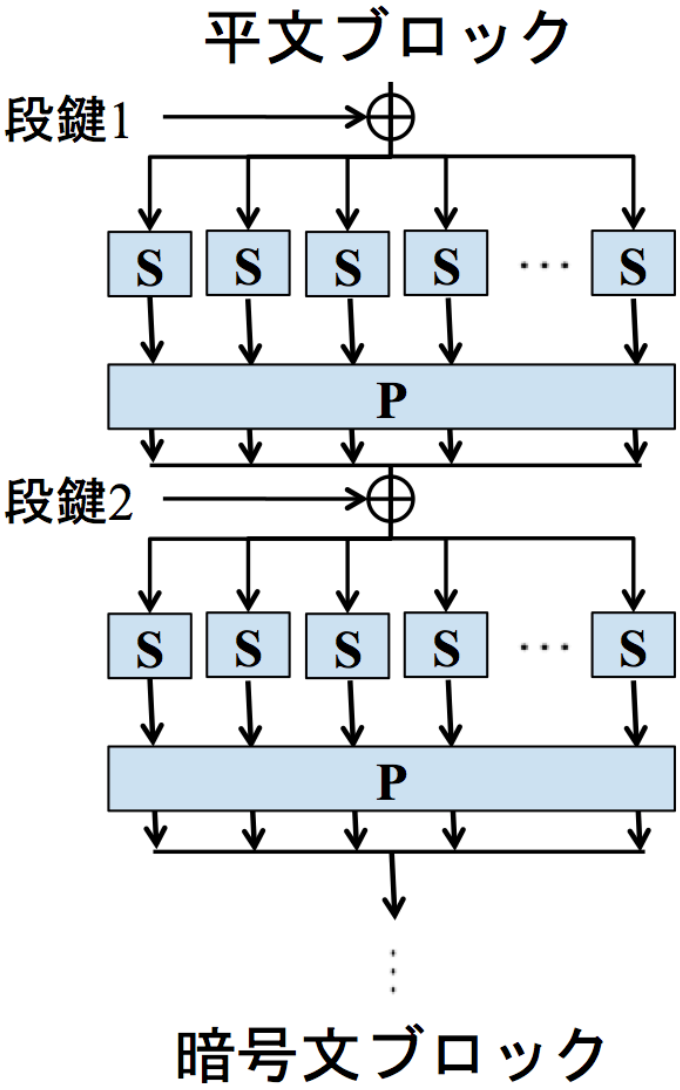

図 2.3: SPN 構造

2.2 KASUMI

KASUMI[1] はブロック単位で暗号化，復号化を行う共通鍵ブロック暗号アルゴリズムであり，MISTY1 を元に開発された．入出力長は 64 bit，秘密鍵長は 128 bit であり，8 段の Feistel 構造を有している．KASUMI は W-CDMA（Wideband Code Division Multiple Access）方式の国際標準暗号として，2000 年に 3rd Generation Partnership Project（3GPP）によって制定された．現

在，第二世代（GSM 方式）と第三世代（W-CDMA 方式）の移動体通信システムで，無線区間の秘匿・完全性保証に利用されている．

2.2.1 データ攪拌部

KASUMI の内部構造を図 2.4 に示す．KASUMI の内部構造は 32 bit の非線形関数である FO 関数と，鍵依存の線形関数である FL 関数を段関数とした 8 段の Feistel 構造である．FO 関数は 16 bit 入出力の FI 関数を 3 段繰り返す変形 Feistel 構造となっている．FI 関数は入力 16 bit を 9 bit と 7 bit に不等分割し，2 種類の非線形関数（S7, S9）を 4 段繰り返す構造となっている．S7 と S9 のブール代数次数はそれぞれ 3 次と 2 次である．非線形関数 S7 と S9 のブール展開式を表 2.3，表 2.4 に示す．FL 関数は，鍵との AND 演算，OR 演算，1bit 左巡回シフトから構成される．

表 2.3: S7-box のブール展開式 (KASUMI)

$y_0 = x_1x_3 \oplus x_4 \oplus x_0x_1x_4 \oplus x_5 \oplus x_2x_5 \oplus x_3x_4x_5 \oplus x_6 \oplus x_0x_6 \oplus x_1x_6 \oplus x_3x_6 \oplus x_2x_4x_6 \oplus x_1x_5x_6 \oplus x_4x_5x_6$
$y_1 = x_0x_1 \oplus x_0x_4 \oplus x_2x_4 \oplus x_5 \oplus x_1x_2x_5 \oplus x_0x_3x_5 \oplus x_6 \oplus x_0x_2x_6 \oplus x_3x_6 \oplus x_4x_5x_6 \oplus 1$
$y_2 = x_0 \oplus x_0x_3 \oplus x_2x_3 \oplus x_1x_2x_4 \oplus x_0x_3x_4 \oplus x_1x_5 \oplus x_0x_2x_5 \oplus x_0x_6 \oplus x_0x_1x_6 \oplus x_2x_6 \oplus x_4x_6 \oplus 1$
$y_3 = x_1 \oplus x_0x_1x_2 \oplus x_1x_4 \oplus x_3x_4 \oplus x_0x_5 \oplus x_0x_1x_5 \oplus x_2x_3x_5 \oplus x_1x_4x_5 \oplus x_2x_6 \oplus x_1x_3x_6$
$y_4 = x_0x_2 \oplus x_3 \oplus x_1x_3 \oplus x_1x_4 \oplus x_0x_1x_4 \oplus x_2x_3x_4 \oplus x_0x_5 \oplus x_1x_3x_5 \oplus x_0x_4x_5 \oplus x_1x_6 \oplus x_3x_6 \oplus x_0x_3x_6 \oplus x_5x_6 \oplus 1$
$y_5 = x_2 \oplus x_0x_2 \oplus x_0x_3 \oplus x_1x_2x_3 \oplus x_0x_2x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_4x_5 \oplus x_1x_6 \oplus x_1x_2x_6 \oplus x_0x_3x_6 \oplus x_3x_4x_6 \oplus x_2x_5x_6 \oplus 1$
$y_6 = x_1x_2 \oplus x_0x_1x_3 \oplus x_0x_4 \oplus x_1x_5 \oplus x_3x_5 \oplus x_6 \oplus x_0x_1x_6 \oplus x_2x_3x_6 \oplus x_1x_4x_6 \oplus x_0x_5x_6$

表 2.4: S9-box のブール展開式 (KASUMI)

$y_0 = x_0x_2 \oplus x_3 \oplus x_2x_5 \oplus x_5x_6 \oplus x_0x_7 \oplus x_1x_7 \oplus x_2x_7 \oplus x_4x_8 \oplus x_5x_8 \oplus x_7x_8 \oplus 1$
$y_1 = x_1 \oplus x_0x_1 \oplus x_2x_3 \oplus x_0x_4 \oplus x_1x_4 \oplus x_0x_5 \oplus x_3x_5 \oplus x_6 \oplus x_1x_7 \oplus x_2x_7 \oplus x_5x_8 \oplus 1$
$y_2 = x_1 \oplus x_0x_3 \oplus x_3x_4 \oplus x_0x_5 \oplus x_2x_6 \oplus x_3x_6 \oplus x_5x_6 \oplus x_4x_7 \oplus x_5x_7 \oplus x_6x_7 \oplus x_8 \oplus x_0x_8 \oplus 1$
$y_3 = x_0 \oplus x_1x_2 \oplus x_0x_3 \oplus x_2x_4 \oplus x_5 \oplus x_0x_6 \oplus x_1x_6 \oplus x_4x_7 \oplus x_0x_8 \oplus x_1x_8 \oplus x_7x_8$
$y_4 = x_0x_1 \oplus x_1x_3 \oplus x_4 \oplus x_0x_5 \oplus x_3x_6 \oplus x_0x_7 \oplus x_6x_7 \oplus x_1x_8 \oplus x_2x_8 \oplus x_3x_8$
$y_5 = x_2 \oplus x_1x_4 \oplus x_4x_5 \oplus x_0x_6 \oplus x_1x_6 \oplus x_3x_7 \oplus x_4x_7 \oplus x_6x_7 \oplus x_5x_8 \oplus x_6x_8 \oplus x_7x_8 \oplus 1$
$y_6 = x_0 \oplus x_2x_3 \oplus x_1x_5 \oplus x_2x_5 \oplus x_4x_5 \oplus x_3x_6 \oplus x_4x_6 \oplus x_5x_6 \oplus x_7 \oplus x_1x_8 \oplus x_3x_8 \oplus x_5x_8 \oplus x_7x_8$
$y_7 = x_0x_1 \oplus x_0x_2 \oplus x_1x_2 \oplus x_3 \oplus x_0x_3 \oplus x_2x_3 \oplus x_4x_5 \oplus x_2x_6 \oplus x_3x_6 \oplus x_2x_7 \oplus x_5x_7 \oplus x_8 \oplus 1$
$y_8 = x_0x_1 \oplus x_2 \oplus x_1x_2 \oplus x_3x_4 \oplus x_1x_5 \oplus x_2x_5 \oplus x_1x_6 \oplus x_4x_6 \oplus x_7 \oplus x_2x_8 \oplus x_3x_8$

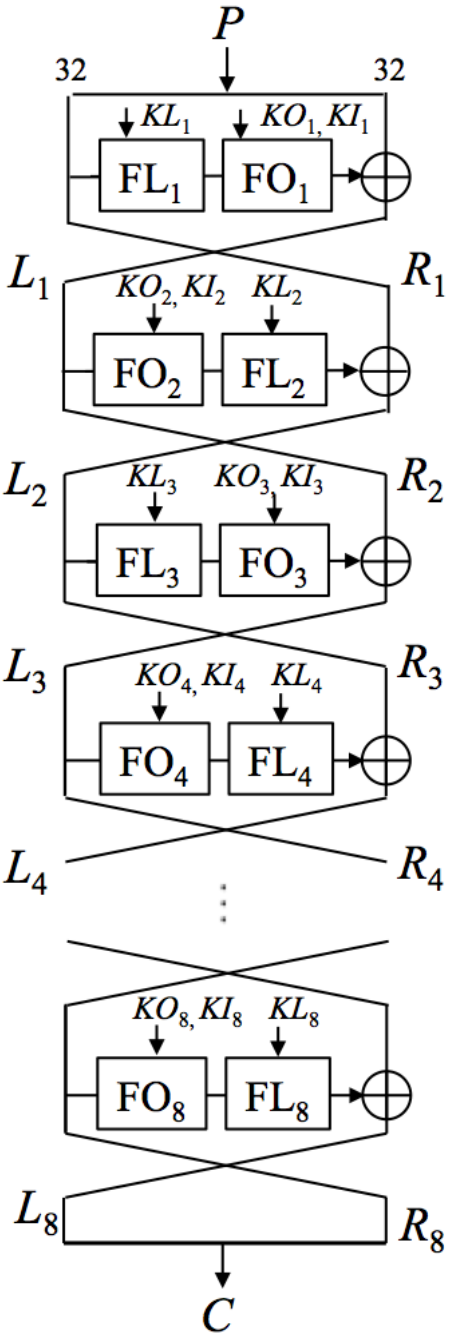

図 2.4: KASUMI の内部構造

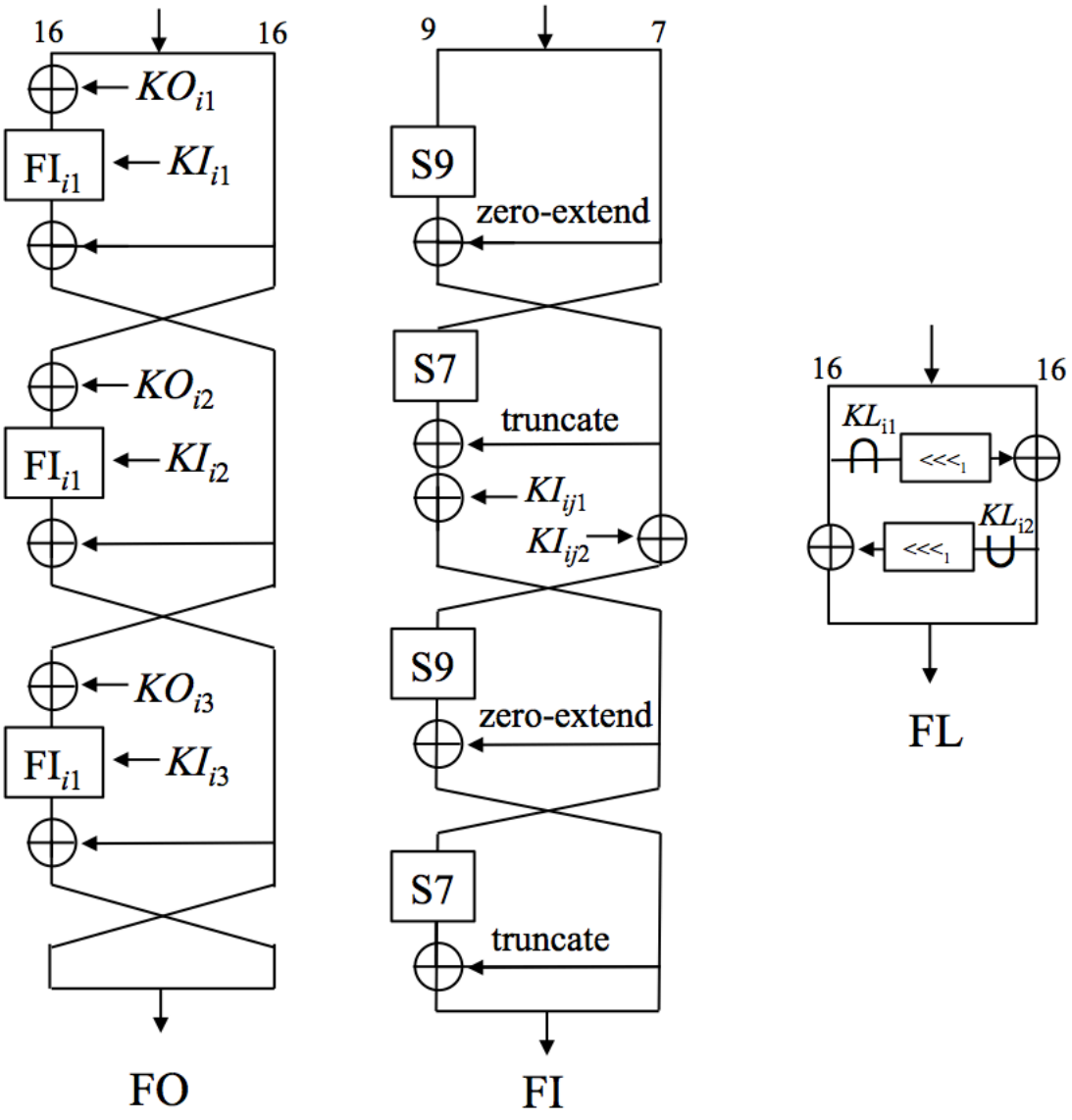

図 2.5: KASUMI の FO 関数，FI 関数，FL 関数

2.2.2 鍵スケジュール部

共通鍵暗号では，秘密鍵から各段に挿入する段鍵を生成し，生成した段鍵で平文の攪拌を行う．本節では，KASUMI の鍵スケジュール部について説明する．KASUMI の鍵スケジュール部は，秘密鍵 128 bit から各々の段で使用する 128 bit の拡大鍵を生成する．まず初めに 128 bit の秘密鍵 K を 16 bit 毎の $Ki, (i = 1, 2, \ldots, 8)$ に分割する．尚，記号 $||$ は結合を表すものとする．

$$K = K1||K2||K3||K4||K5||K6||K7||K8$$

続いて，$K'j, (j = 1, 2, \ldots, 8)$ を下記の通り計算する．この時に用いる定数 $Cj, (j = 1, 2, \ldots, 8)$ は表 2.5 に示す 16 進数の値を用いる．また，生成した Kj と $K'j, (j = 1, 2, \ldots, 8)$ を用いて各段の拡大鍵を表 2.6 の通りに設定する．

$$K'j = Kj \oplus Cj$$

表 2.5: 定数 Cj

$C1$	0x0123
$C2$	0x4567
$C3$	0x89AB
$C4$	0xCDEF
$C5$	0xFEDC
$C6$	0xBA98
$C7$	0x7654
$C8$	0x3210

表 2.6: 拡大鍵 $(i = 1, 2, \ldots, 8)$

	1	2	3	4	5	6	7	8
KL_{i1}	$K1 \lll 1$	$K2 \lll 1$	$K3 \lll 1$	$K4 \lll 1$	$K5 \lll 1$	$K6 \lll 1$	$K7 \lll 1$	$K8 \lll 1$
KL_{i2}	$K'3$	$K'4$	$K'5$	$K'6$	$K'7$	$K'8$	$K'1$	$K'2$
KO_{i1}	$K2 \lll 5$	$K3 \lll 5$	$K4 \lll 5$	$K5 \lll 5$	$K6 \lll 5$	$K7 \lll 5$	$K8 \lll 5$	$K1 \lll 5$
KO_{i2}	$K6 \lll 8$	$K7 \lll 8$	$K8 \lll 8$	$K1 \lll 8$	$K2 \lll 8$	$K3 \lll 8$	$K4 \lll 8$	$K5 \lll 8$
KO_{i3}	$K7 \lll 13$	$K8 \lll 13$	$K1 \lll 13$	$K2 \lll 13$	$K3 \lll 13$	$K4 \lll 13$	$K5 \lll 13$	$K6 \lll 13$
KI_{i1}	$K'5$	$K'6$	$K'7$	$K'8$	$K'1$	$K'2$	$K'3$	$K'4$
KI_{i2}	$K'4$	$K'5$	$K'6$	$K'7$	$K'8$	$K'1$	$K'2$	$K'3$
KI_{i3}	$K'8$	$K'1$	$K'2$	$K'3$	$K'4$	$K'5$	$K'6$	$K'7$

2.3 MISTY

MISTY[34] は松井によって提案された共通鍵ブロック暗号アルゴリズムで，入出力長は 64 bit，秘密鍵長は 128 bit である．MISTY は，構造上の違いから MISTY1 と MISTY2 が存在する．推奨段数は MISTY1 の場合は 8 段，MISTY2 の場合は 12 段で，内部構造は Feistel 構造を有している．MISTY は差分攻撃 [5] と線形攻撃 [33] に対し，数学的に安全性が証明された（証明可能安全性を有する）暗号である．

2.3.1 データ攪拌部

MISTY は，構造上の違いから MISTY1 と MISTY2 が存在する．MISTY1 の内部構造を図 2.6 に，MISTY2 の内部構造を図 2.7 に示す．内部構造は，32 bit の非線形関数である FO 関数と，鍵依存の線形関数である FL 関数を段関数とした 8 段（MISTY2 の場合は 12 段）の Feistel 構造である．FO 関数は 16 bit 入出力の FI 関数を 3 段繰り返す変形 Feistel 構造となっている．FI 関数は入力 16 bit を 9 bit と 7 bit に不等分割し，2 種類の非線形関数（S7, S9）を 3 段繰り返す構造となっている．S7 と S9 のブール代数次数は其々 3 次と 2 次である．非線形関数 S7 と S9 のブール展開式を表 2.7，2.8 に示す．FL 関数は，鍵との AND 演算，OR 演算で構成される．

表 2.7: S7-box のブール展開式 (MISTY)

$y_0 = x_0 \oplus x_1x_3 \oplus x_0x_3x_4 \oplus x_1x_5 \oplus x_0x_2x_5 \oplus x_4x_5 \oplus x_0x_1x_6 \oplus x_2x_6 \oplus x_0x_5x_6 \oplus x_3x_5x_6 \oplus 1$
$y_1 = x_0x_2 \oplus x_0x_4 \oplus x_3x_4 \oplus x_1x_5 \oplus x_2x_4x_5 \oplus x_6 \oplus x_0x_6 \oplus x_3x_6 \oplus x_2x_3x_6 \oplus x_1x_4x_6 \oplus x_0x_5x_6 \oplus 1$
$y_2 = x_1x_2 \oplus x_0x_2x_3 \oplus x_4 \oplus x_1x_4 \oplus x_0x_1x_4 \oplus x_0x_5 \oplus x_0x_4x_5 \oplus x_3x_4x_5 \oplus x_1x_6 \oplus x_3x_6 \oplus x_0x_3x_6 \oplus x_4x_6 \oplus x_2x_4x_6$
$y_3 = x_0 \oplus x_1 \oplus x_0x_1x_2 \oplus x_0x_3 \oplus x_2x_4 \oplus x_1x_4x_5 \oplus x_2x_6 \oplus x_1x_3x_6 \oplus x_0x_4x_6 \oplus x_5x_6 \oplus 1$
$y_4 = x_2x_3 \oplus x_0x_4 \oplus x_1x_3x_4 \oplus x_5 \oplus x_2x_5 \oplus x_1x_2x_5 \oplus x_0x_3x_5 \oplus x_1x_6 \oplus x_1x_5x_6 \oplus x_4x_5x_6 \oplus 1$
$y_5 = x_0 \oplus x_1 \oplus x_2 \oplus x_0x_1x_2 \oplus x_0x_3 \oplus x_1x_2x_3 \oplus x_1x_4 \oplus x_0x_2x_4 \oplus x_0x_5 \oplus x_0x_1x_5 \oplus x_3x_5 \oplus x_0x_6 \oplus x_2x_5x_6$
$y_6 = x_0x_1 \oplus x_3 \oplus x_0x_3 \oplus x_2x_3x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_3x_5 \oplus x_1x_3x_5 \oplus x_1x_6 \oplus x_1x_2x_6 \oplus x_0x_3x_6 \oplus x_4x_6 \oplus x_2x_5x_6$

2.3.2 鍵スケジュール部

MISTY の鍵スケジュール部は，秘密鍵 128 bit から 256 bit の拡大鍵を生成し，各段に挿入している．まず初めに 128 bit の秘密鍵 K を 16 bit 毎の $Ki, (i = 1, 2, \ldots, 8)$ に分割する．尚，記号 $||$ は結合を表すものとする．

$$K = K_1||K_2||K_3||K_4||K_5||K_6||K_7||K_8$$

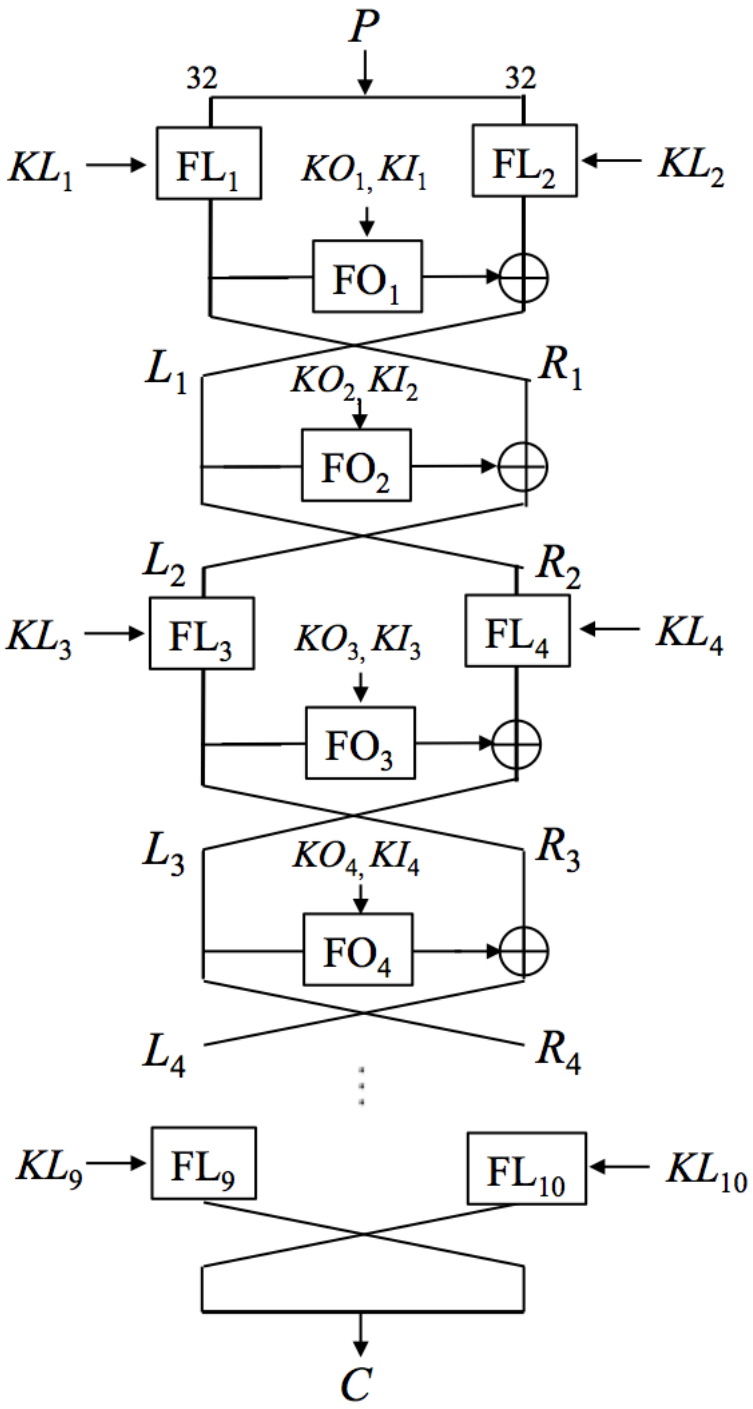

図 2.6: MISTY1 の内部構造

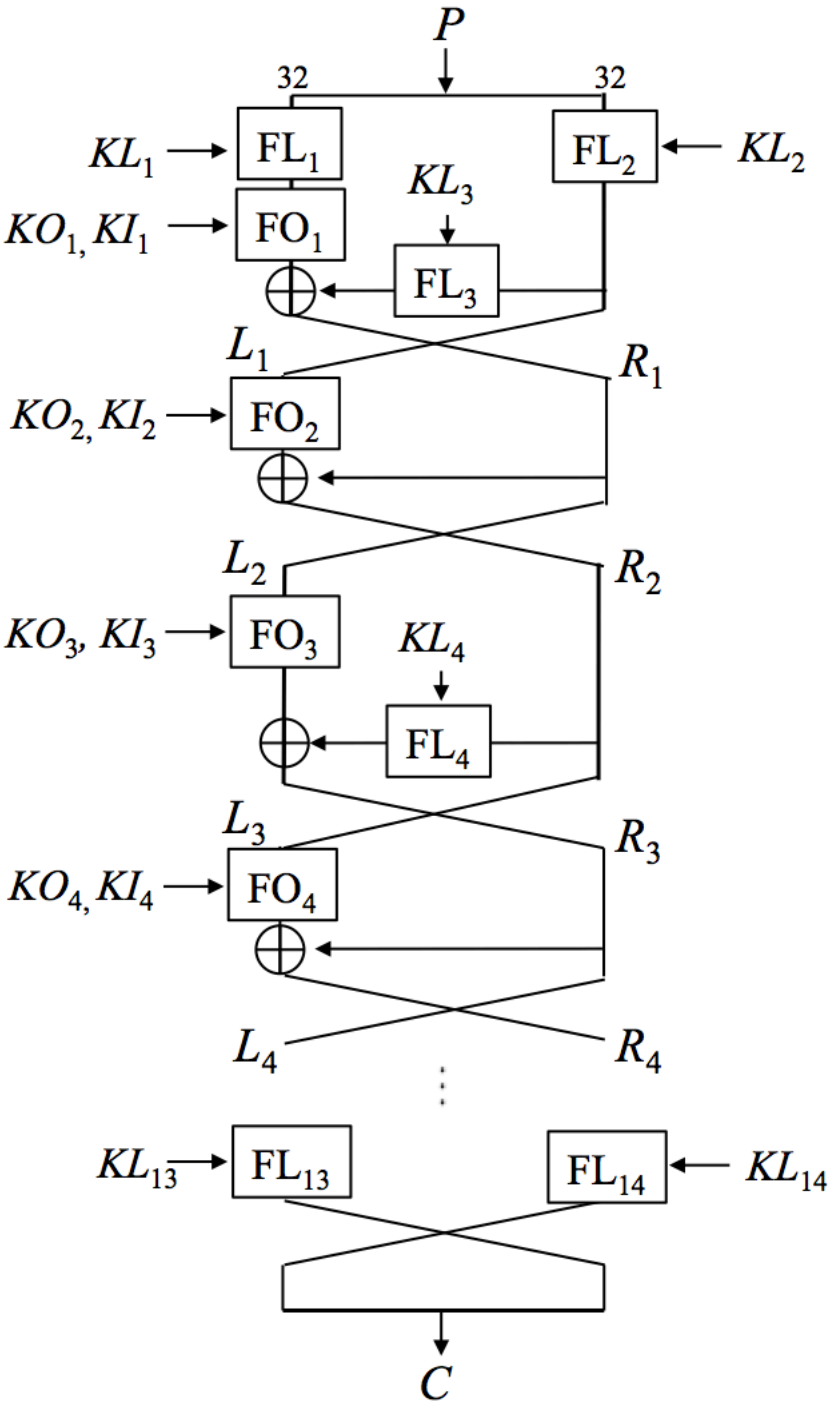

図 2.7: MISTY2 の内部構造

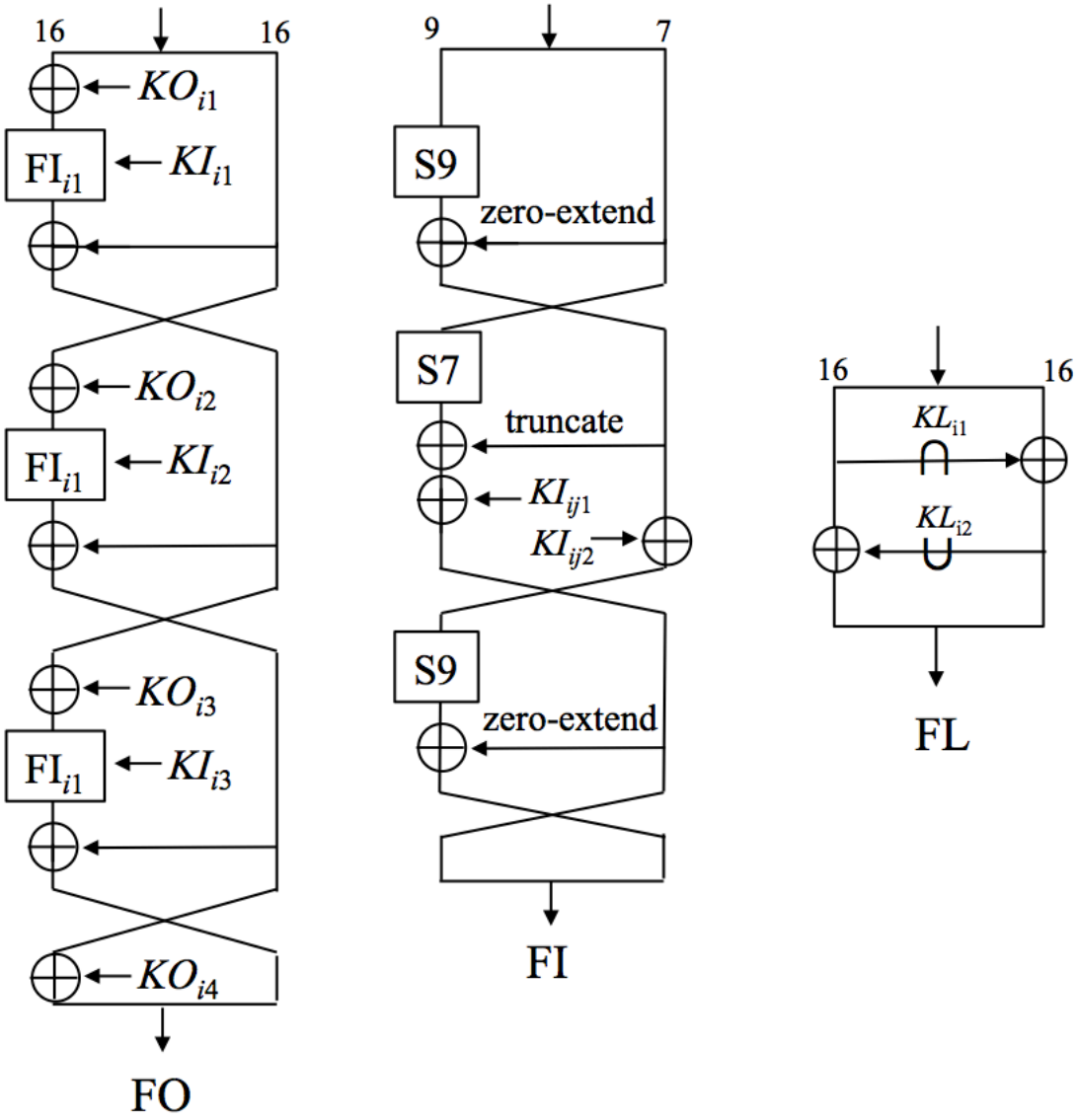

図 2.8: MISTY の FO 関数，FI 関数，FL 関数

表 2.8: S9-box のブール展開式 (MISTY)

$y_0 = x_0x_4 \oplus x_0x_5 \oplus x_1x_6 \oplus x_2x_6 \oplus x_2x_7 \oplus x_3x_7 \oplus x_3x_8 \oplus x_4x_8 \oplus 1$
$y_1 = x_0x_2 \oplus x_3 \oplus x_1x_3 \oplus x_2x_3 \oplus x_3x_4 \oplus x_4x_5 \oplus x_0x_6 \oplus x_2x_6 \oplus x_7 \oplus x_0x_8 \oplus x_3x_8 \oplus x_5x_8 \oplus 1$
$y_2 = x_0x_1 \oplus x_1x_3 \oplus x_4 \oplus x_0x_4 \oplus x_2x_4 \oplus x_3x_4 \oplus x_4x_5 \oplus x_0x_6 \oplus x_5x_6 \oplus x_1x_7 \oplus x_3x_7 \oplus x_8$
$y_3 = x_0 \oplus x_1x_2 \oplus x_2x_4 \oplus x_5 \oplus x_1x_5 \oplus x_3x_5 \oplus x_4x_5 \oplus x_5x_6 \oplus x_1x_7 \oplus x_6x_7 \oplus x_2x_8 \oplus x_4x_8$
$y_4 = x_1 \oplus x_0x_3 \oplus x_2x_3 \oplus x_0x_5 \oplus x_3x_5 \oplus x_6 \oplus x_2x_6 \oplus x_4x_6 \oplus x_5x_6 \oplus x_6x_7 \oplus x_2x_8 \oplus x_7x_8$
$y_5 = x_2 \oplus x_0x_3 \oplus x_1x_4 \oplus x_3x_4 \oplus x_1x_6 \oplus x_4x_6 \oplus x_7 \oplus x_3x_7 \oplus x_5x_7 \oplus x_6x_7 \oplus x_0x_8 \oplus x_7x_8$
$y_6 = x_0x_1 \oplus x_3 \oplus x_1x_4 \oplus x_2x_5 \oplus x_4x_5 \oplus x_2x_7 \oplus x_5x_7 \oplus x_8 \oplus x_0x_8 \oplus x_4x_8 \oplus x_6x_8 \oplus x_7x_8 \oplus 1$
$y_7 = x_1 \oplus x_0x_1 \oplus x_1x_2 \oplus x_2x_3 \oplus x_0x_4 \oplus x_5 \oplus x_1x_6 \oplus x_3x_6 \oplus x_0x_7 \oplus x_4x_7 \oplus x_6x_7 \oplus x_1x_8 \oplus 1$
$y_8 = x_0 \oplus x_0x_1 \oplus x_1x_2 \oplus x_4 \oplus x_0x_5 \oplus x_2x_5 \oplus x_3x_6 \oplus x_5x_6 \oplus x_0x_7 \oplus x_0x_8 \oplus x_3x_8 \oplus x_6x_8 \oplus 1$

FI 関数の入力を K_i，その時の暗号化鍵を K_{i+1} とした時の出力値 16 bit を K'_i, $(i = 1, 2, \ldots, 8)$ とする．生成した拡大鍵を表 2.9 に示す通りに設定する．

$$K'_i = FI(K_i, K_{i+1})$$

表 2.9: 拡大鍵 $(i = 1, 2, \ldots, 8)$

	KO_{i1}	KO_{i2}	KO_{i3}	KO_{i4}	KI_{i1}	KI_{i2}	KI_{i3}	KL_{i1}	KL_{i2}
拡大鍵	K_i	K_{i+2}	K_{i+7}	K_{i+4}	K'_{i+5}	K'_{i+1}	K'_{i+3}	$K_{\frac{i+1}{2}}$ (odd i) $K'_{\frac{i}{2}+2}$ (even i)	$K'_{\frac{i+1}{2}+6}$ (odd i) $K_{\frac{i}{2}+4}$ (even i)

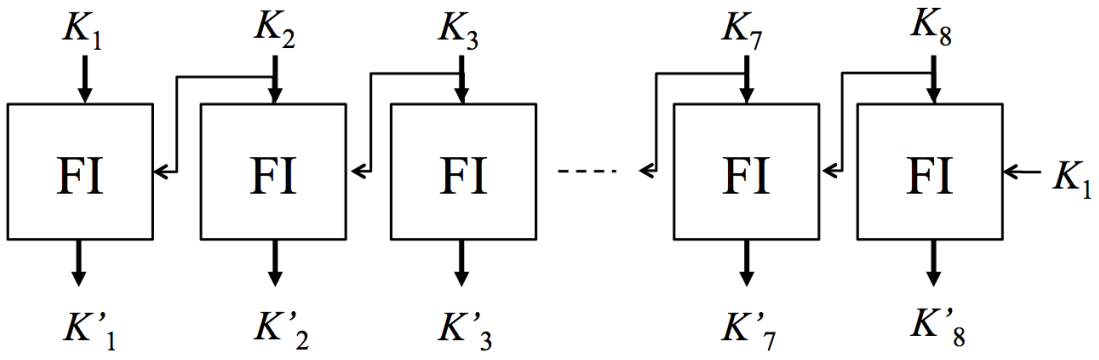

図 2.9: MISTY の鍵スケジュール部

2.4 計算量的安全性による評価

本論文では，暗号の安全性を計算量的安全性によって評価する[1]．計算量的安全性とは，攻撃者が最も効率のよい攻撃手法を用いて未知の情報（秘密鍵等）を求める為に必要な計算量を見積もり，その計算量が現在（及びムーアの法則を考慮した未来）の計算機の能力に比較して膨大であり，現実的な時間では実行不可能である場合に計算量的に安全と考えるものである．

2.4.1 全数探索法

全数探索法は，攻撃者が入手した少数の平文と暗号文のペアを元に，暗号で使用されている秘密鍵を総当たりで試行し，正しい秘密鍵を推定する手法である．暗号を E，平文を P，暗号文を C，秘密鍵を K とする．攻撃者は $C = E(P;K)$ の関係式を元に，鍵 K の候補の総当りで解く．この場合に必要な計算量は，暗号の計算回数で見積もる．例えば，秘密鍵長が 128 bit の場合，計算量は $T = 2^{128}$ 回の暗号処理計算量となる．

実際に全数探索法で秘密鍵を解いた例として，1999 年の DES Challenge III[38] や，2002 年の RC5-64 Challenge[39] がある．前者は，DES 解読用の専用ハードウェアとインターネット上の PC の共同作業で 56 bit 鍵を 22 時間で導出した．後者は，RC5 の 64 bit 鍵をインターネット上の PC の共同作業 4 年で解読した．今後の計算機能力の進歩を考えた場合，現在では，80 bit 未満の鍵長は，全数探索法で破られる可能性がある．

2.4.2 ショートカット法

ショートカット法とは，暗号の内部構造に関する知識を利用し，全数探索法よりも少ない計算量で攻撃 (秘密鍵の回復等) を目指す手法である．図 2.10 に示す m bit 入力，$R+r$ 段の暗号を考える．段関数を F_j $(j = 1, 2, \ldots, R+r)$ とする．また，i 番目 $(i = 1, 2, \ldots)$ の平文を $P(i)$，対応する暗号文を $C(i)$ とし，各段に挿入される段鍵を K_j $(j = 1, 2, \ldots, R+r)$ とする．i 番目 $(i = 1, 2, \ldots)$ の R 段目出力を $X_R(i)$ とする．

攻撃者は，暗号の内部構造に関する知識を利用し，特徴量 $\mathcal{H}(X_R(i))$ を抽出する．特徴量 $\mathcal{H}(X_R(i))$ は以下の条件で導出する．

1. 段鍵 $K_1, K_2, \ldots, K_R$ に依存せず，平文 $P(i)$ から推定可能なもの．段鍵に依存する場合は，その影響が少ないもの．確率的推定でもよいが，理想乱数と区別可能なもの．

[1]安全性を測る指標として，計算量的安全性の他には，シャノンの情報理論に基づく情報理論的安全性や，暗号の出力系列の統計情報を用いたアバランシュ性による評価が存在する．

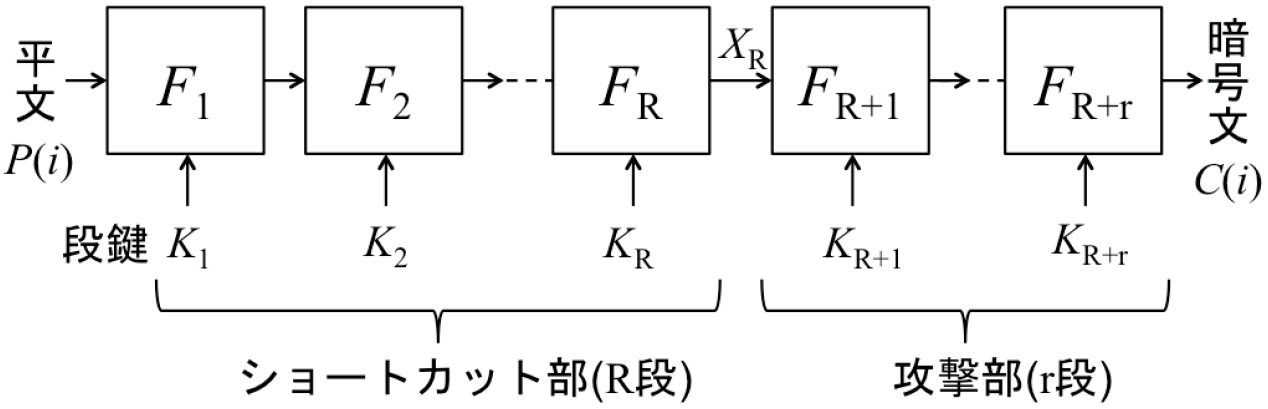

図 2.10: 攻撃対象の暗号 ($R+r$ 段)

2. 段関数 F の構成要素の解析結果を容易に $R+r$ 段の暗号系全体に拡張可能なもの．

3. ショートカット部の段数 R が大きいもの．

特徴量の抽出ができた場合，平文側から推定された特徴量 $\mathcal{H}(X_R(i))$ と，暗号文 $C(i)$ 側から段鍵 $K_{R+1}, \ldots, K_{R+r}$ を使って復号計算で求めた $X_R(i)$ の特徴量を突き合わせることにより，攻撃ができる．復号計算を $F^{-1}(C(i); K_{R+1}, \ldots, K_{R+r})$ とすれば，攻撃に使用する方程式は

$$\mathcal{H}(X_R(i)) = F^{-1}(C(i); K_{R+1}, \ldots, K_{R+r})$$

となる．特徴量の推定が確率的に成立する場合，真の段鍵 $K_{R+1}, \ldots, K_{R+r}$ であれば高い確率で上記の式が成立し，偽の段鍵であれば，$X_R(i)$ を乱数と考えた程度の確率で成立すると期待される．この両者の成立確率の違い (成立回数の違い) で，段鍵候補の真偽が判定できる．特徴量 $\mathcal{H}(X_R(i))$ は攻撃によって変わり，どのような特徴量を選ぶかにより差分攻撃 [5]，線形攻撃 [33]，高階差分攻撃 [28] 等となる．

2.4.3 暗号解読の定義

攻撃対象の暗号に対し，全数探索法よりも少ない計算量と平文数で攻撃できる手法 (ショートカット法) が発見された場合，暗号が解読されたと定義する．例えば，攻撃対象の暗号の入出力長が 64 bit，秘密鍵長が 128 bit の場合，攻撃に必要な平文数が $D < 2^{64}$，及び，計算量が $T < 2^{128}$ 回の暗号処理計算量を満たす効率的な手法が発見された場合，対象の暗号は解読されたことになる．尚，暗号が解読された場合でも，攻撃に必要な計算量が現実的な時間で実行不可能であれば，実社会においては，直ちに暗号が危殆化（十分な安全性を確保できないこと）する訳ではない事に注意が必要である．

2.5 代表的な攻撃方法

2.4.2 節で述べた特徴量 $\mathcal{H}(X_R(i))$ を選ぶ際の代表的な攻撃方法に，差分攻撃 [5]，線形攻撃 [33] が知られている．本節では，これらの攻撃方法を含め，現在知られている代表的な攻撃方法について概説する．各攻撃方法の詳細は，参考文献を参照されたい．以下に，これから述べる攻撃方法の全体像を示す．

表 2.10: 代表的な攻撃方法

	代表的な攻撃方法
入出力間の差分伝搬に着目した攻撃	差分攻撃(Differential Attack)
	切詰差分攻撃(Truncated Differential Attack)
	不可能差分攻撃(Impossible Differential Attack)
	ブーメラン攻撃(Boomerang Attack)
	差分線形攻撃(Differential-Linear Attack)
入出力間の線形相関に着目した攻撃	線形攻撃(Linear Attack)
	線形ふるい攻撃
	差分線形攻撃(Differential-Linear Attack)
	零相関線形攻撃(Zero-correlation Linear Attack)
代数的性質に着目した攻撃	高階差分攻撃(Higher Order Differential Attack)
	補間攻撃(Interpolation Attack)
集合の性質に着目した攻撃	SQUARE攻撃(SQUARE Attack)
	飽和攻撃(Saturation Attack)
	積分攻撃(Integral Attack)

2.5.1 攻撃条件

共通鍵暗号の攻撃に関して，攻撃者に許される前提条件で分類すると以下となる．

- 暗号文単独攻撃：暗号文のみを使う攻撃
- 既知平文攻撃：与えられた平文と，平文に対応する暗号文のみを使う攻撃
- 選択平文攻撃：攻撃者が都合が良いように選んだ平文に対応する暗号文を手に入れて行う攻撃

- 関連鍵攻撃 : 攻撃者が自由に秘密鍵を操作した上で行う選択平文攻撃．尚，鍵の操作は攻撃が自明とならない範囲に制限される．

後者ほど，攻撃者の自由度は増加し，攻撃は易しくなる．一方，そのような攻撃条件が成立する可能性は小さくなる．

2.5.2 入出力間の差分伝搬を利用した攻撃

差分攻撃 [5] は，入力を変化させた時に，出力の変化の分布に偏りがある場合に行える攻撃である．攻撃者の前提条件は，選択平文攻撃である．付録 A にて差分攻撃について記している為，興味のある読者は適宜参照されたい．

差分攻撃の派生系として，切詰差分攻撃（Truncated Differential Attack）[24]，不可能差分攻撃（Impossible Differential Attack）[6]，ブーメラン攻撃（Boomerang Attack）[56]，差分線形攻撃（Differential-Linear Attack）[29] 等がある．以下に攻撃の概要を記す．

- 切詰差分攻撃 : 差分条件を緩めた差分攻撃．差分の有無を byte 単位等で表し，差分伝搬を解析
- 不可能差分攻撃 : ありえない差分を生じる鍵を排除するという原理に基づく攻撃
- ブーメラン攻撃 : 暗号化と復号の双方向に関して差分攻撃を行う攻撃
- 差分線形攻撃 : 差分攻撃と線形攻撃を組み合わせた攻撃

2.5.3 入出力間の線形相関を利用した攻撃

線形攻撃 [33] は，入力の特定ビットの排他的論理和と出力の特定ビットの排他的論理和の間の相関関係（線形相関）を調べ，その相関関係を用いて最終段に入力される鍵を推定する攻撃である．攻撃者の前提条件は，既知平文攻撃である．付録 A にて線形攻撃について記している為，興味のある読者は適宜参照されたい．

線形攻撃の派生系として，線形ふるい攻撃 [47]，差分線形攻撃（Differential-Linear Attack）[29]，零相関線形攻撃（Zero-correlation Linear Attack）[8] 等がある．以下に攻撃の概要を記す．

- 線形ふるい攻撃 : 線形ふるい条件を満たすものだけを取り出して線形確率を大きくし，通常の線形攻撃に比べ少ない既知平文数で解読する攻撃
- 差分線形攻撃 : 差分攻撃と線形攻撃を組み合わせた攻撃

- 零相関線形攻撃：成立確率が零の線形特性を利用し，偽鍵を排除するという原理に基づく攻撃

2.5.4 代数的性質を利用した攻撃

暗号 E の R 段目出力 X_R は平文入力 m bit を変数とする $GF(2)$ 上のブール式として展開できる．このブール式に存在する最大次数をブール代数次数（又は単に次数）と呼ぶ．高階差分攻撃（Higher Order Differential Attack）[28] や補間攻撃（Interpolation Attack）[21] は暗号の代数的な性質を利用した攻撃である．以下に攻撃の概要を記す．

- 高階差分攻撃：R 段目出力 X_R を平文入力 m bit でブール式であらわしたとき，出力が平文入力の $d\ (< m)$ 次式以下の場合，$(d+1)$ 階差分が常に 0 となる性質が知られている．この性質を特徴量として利用し，最終段の鍵を回復する攻撃
- 補間攻撃：暗号化関数が代数系の上で表現できる時に行える攻撃

高階差分攻撃の詳細は第 3 章を参照されたい．

2.5.5 集合の性質を利用した攻撃

g 関数を n bit 入出力関数 $g : \{0,1\}^n \to \{0,1\}^n$ とする．g 関数は S-box や図 2.10 の F_i 関数 $(1 \leq i \leq R+r)$ 等である．g 関数への入力が全通りとなる様に 2^n 種類の平文 P を与えた時，g 関数が 1 対 1 写像であるならば，その出力は全通り出現する．従って，出力を全て排他的論理和（XOR）した結果は 0 となる．2^n 種類の平文に対応する出力全てを集合と捉えた場合，集合の性質を利用した攻撃が可能となる．その様な攻撃として，SQUARE 攻撃（SQUARE Attack）[12]，飽和攻撃（Saturation Attack）[30]，積分攻撃（Integral Attack）[25] 等が存在する．

- SQUARE 攻撃：S-box に全通り入力される平文集合を用意し，出力の全てを排他的論理和（XOR）した結果が 0 となる事を利用した攻撃
- 飽和攻撃：SQUARE 攻撃と同じく集合の性質を利用した攻撃
- 積分攻撃：SQUARE 攻撃の排他的論理和（XOR）を総和として捉えた攻撃

積分攻撃の詳細は第 4 章を参照されたい．

2.6　本論文で対象とする攻撃方法

共通鍵暗号に対する汎用的な攻撃手法に差分攻撃 [5]，線形攻撃 [33] がある．近年これらの攻撃の対策に関して研究が進み，新たに提案される共通鍵暗号は差分攻撃と線形攻撃に対する証明可能安全性（数学的に安全であること）を有する様に設計がなされている．

表 2.11: 代表的な攻撃方法に対する証明可能安全性の議論

	代表的な攻撃方法	
入出力間の差分伝搬に着目した攻撃	差分攻撃(Differential Attack)	証明可能安全性議論有り
	切詰差分攻撃(Truncated Differential Attack)	
	不可能差分攻撃(Impossible Differential Attack)	
	ブーメラン攻撃(Boomerang Attack)	
	差分線形攻撃(Differential-Linear Attack)	
入出力間の線形相関に着目した攻撃	線形攻撃(Linear Attack)	
	線形ふるい攻撃	
	差分線形攻撃(Differential-Linear Attack)	
	零相関線形攻撃(Zero-correlation Linear Attack)	
代数的性質に着目した攻撃	高階差分攻撃(Higher Order Differential Attack)	証明可能安全性議論無し
	補間攻撃(Interpolation Attack)	
集合の性質に着目した攻撃	SQUARE攻撃(SQUARE Attack)	
	飽和攻撃(Saturation Attack)	
	積分攻撃(Integral Attack)	

そこで本研究では，証明可能安全性に関する議論がなされていない攻撃手法の中から，代数的性質を利用した高階差分攻撃と，高階差分攻撃の一種であり，次数を見積もる簡易な手法が提案された積分攻撃を研究対象として選定した．高階差分攻撃と積分攻撃は，共通鍵暗号に対する強力，且つ汎用的な攻撃手法であり，本研究成果は暗号研究の発展に寄与するものである．

第3章　高階差分攻撃耐性評価

3.1　概要

高階差分攻撃（Higher Order Differential Attack）[21] は，暗号の代数的性質を利用した攻撃の一つである．出力を入力 m bit でブール式に展開し，出力が入力に関する d 次式の場合，$(d+1)$ 階差分が常に 0 となる性質が知られている．本章では，ショートカット法で用いる R 段目出力 X_R の特徴量 $\mathcal{H}(X_R(i))$ に，前述の高階差分の性質を用いる．尚，本章では，高階差分の性質を用いた特徴量を高階差分特性と呼び，ショートカット部の段数を R とした場合，R 段高階差分特性（又は，単に R 段特性）と呼ぶ．

本章では，高階差分攻撃を行う際の特性探索法と鍵回復の両面で高速化が可能な技法の提案と，その適用結果を示す．具体的には，田中らが示した 32 階差分の 4 段に対し，効果的な選択平文の探索を行う事で，16 階差分の 4 段特性が存在する事を示す．この手法を用いる事により，4 段特性を構築する為に必要な選択平文数を 1 組当たり 2^{-16} 削減する事が可能である．また，鍵回復を行う際には，全数探索法よりも効率的な線形化手法 [42] を採用し，更に線形化手法の高速化を行う事で，5 段の KASUMI の鍵回復に必要な計算量を $2^{85.5}$ 倍高速化する事が可能である事を示す．線形化手法の高速化技法は，(1) 事前に方程式を解析し，鍵 (未知項) の係数導出を高速化する技法と，(2) 高次項の鍵（未知項）の係数を 0 に制御し，導出する鍵 (未知項) の数を削減する技法の 2 つから成る．この 2 つの高速化技術は，KASUMI 以外の共通鍵暗号にも適用可能な汎用的な技法である．

以下に，第 3 章で用いる記号の一覧を示す．

3.2　高階差分攻撃

3.2.1　高階差分

高階差分は 1994 年に Lai によって差分の拡張として定義された [28]．暗号化関数 $f(X;K)$: $GF(2)^m \times GF(2)^s \to GF(2)^n$ を以下の様に示す．

$$Y = f(X;K) \tag{3.1}$$

表 3.1: 第 3 章で用いる記号一覧

記号	説明
f	m bit 入力，n bit 出力の暗号化関数
$\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_i\}$	$GF(2)^m$ 上で 1 次独立な i 個のベクトルの集合
$V^{(i)}$	i 個のベクトル $\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_i\}$ で張られる i 次元部分空間
$\Delta^{(i)}_{V^{(i)}} f(X;K)$	暗号化関数 f の X に関する i 階差分
$\bigoplus_{\alpha \in V^{(i)}} f(X \oplus \alpha; K)$	入力 X，差分 $\alpha \in V^{(i)}$ を与えた出力の XOR 総和
D	鍵回復に必要な選択平文数
T	鍵回復に必要な計算量
$\lceil z \rceil$	天井関数を表し，z より大きな最小の整数を表す
L	線形方程式中に存在する段鍵 K_{R+1} に関する未知項数
L'	消去する未知項数
L''	推定する未知項数，$L'' = L - L'$
$\mathbf{A}$	サイズ $L \times L$ の係数行列
$\mathbf{k}$	段鍵 K_{R+1} に関する未知項を並べた L 次元ベクトル
$\mathbf{b}$	定数項ベクトル
$\mathbf{A}_j$	サイズ $n\times$ C の行列，$j = 1, 2, \ldots, L+1$
$\mathbf{c}$	暗号文の一次項から高次項を並べた C 次元ベクトル
$Z[j-k]$	任意の中間値 Z の j から k bit 目

ここで $X \in GF(2)^m$ を入力，$K \in GF(2)^s$ を鍵，$Y \in GF(2)^n$ を出力とする．$\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_i\}$ を $GF(2)^m$ 上で 1 次独立な i 個のベクトルの集合とし，これらのベクトルによって張られる $GF(2)^m$ 上の i 次元部分空間を $V^{(i)}$ と表す．このとき暗号化関数 $f(X;K)$ の X に関する i 階差分 $\Delta^{(i)}_{V^{(i)}} f(X;K)$ は以下の式で定義される．

$$\Delta^{(i)}_{V^{(i)}} f(X;K) = \bigoplus_{\alpha \in V^{(i)}} f(X \oplus \alpha; K) \tag{3.2}$$

以下では，$\Delta^{(i)}_{V^{(i)}}$ を $\Delta^{(i)}$ と省略して記す．

高階差分の性質として以下の性質が知られている．

[**性質 1**] 関数 $f(X;K)$ の X に関する次数が d であるならば，X, K によらず以下が成立する．

$$deg_X\{f(X;K)\} = d \Leftrightarrow \begin{cases} \Delta^{(d+1)} f(X;K) = 0 \\ \Delta^{(d)} f(X;K) = constant \end{cases} \tag{3.3}$$

[**性質 2**] 高階差分は排他的論理和 ($\oplus$ 演算) に関して線形性を有する．

$$\Delta^{(d)}\{f_1(X;K_1) \oplus f_2(X;K_2)\} = \Delta^{(d)} f_1(X;K_1) \oplus \Delta^{(d)} f_2(X;K_2) \tag{3.4}$$

3.2.2 鍵回復攻撃

Jakobsen と Knudsen は高階差分を用いた鍵回復攻撃ができる事を示した [21]．本節では，高階差分を用いた鍵回復攻撃について説明する．m bit 入力，$R+1$ 段の暗号を考える．これは，図 2.10 で $r = 1$ とした暗号である．ここでは平文を $X \in GF(2)^m$，対応する暗号文を $C(X) \in GF(2)^m$，R 段目出力の一部を $X_R \in GF(2)^n$，段鍵を $K_i \in GF(2)^{|K|}$ $(i = 1, 2, \ldots, R+1)$ とする．

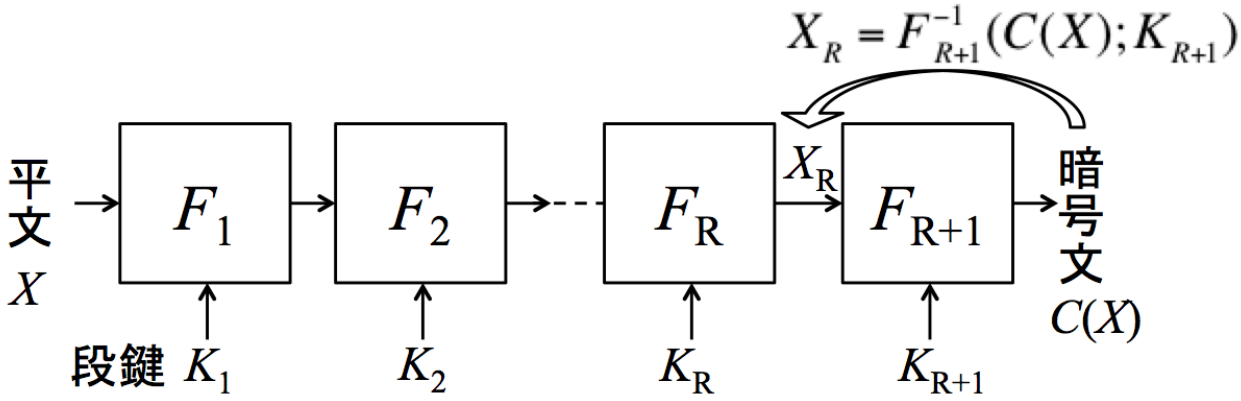

図 3.1: 攻撃対象の暗号 ($R+1$ 段)

平文 $X \in GF(2)^m$ から R 段目出力の一部 $X_R \in GF(2)^n$ に着目する関数を式 (3.1) の $f(X; K)$ とした時，X_R は以下の式で表す事ができる．

$$X_R = f(X; K) = F_R(\cdots F_2(F_1(X; K_1); K_2); \cdots, K_R)[j-k]$$

$[j-k]$ は出力の j bit 目から k bit 目の n bit を示す．ここで，X_R の X に関する次数が d の時，高階差分の性質 1 から $(d+1)$ 階差分は平文 X と段鍵 $K_1, K_2, \ldots, K_R$ に依存せず，常に 0 となる．

$$deg_X\{X_R\} = d \Leftrightarrow \Delta^{(d+1)} X_R = 0 \tag{3.5}$$

暗号文 $C(X)$ を復号して X_R を求める関数を F_{R+1}^{-1} とした場合，式 (3.5) を用いて段鍵 K_{R+1} に関する以下の方程式を導出する事ができる．

$$\Delta^{(d+1)} X_R = \bigoplus_{\alpha \in V^{(d+1)}} F_{R+1}^{-1}(C(X \oplus \alpha); K_{R+1}) = 0 \tag{3.6}$$

式 (3.6) は，段鍵 K_{R+1} の推定が正しい場合に確率 1 で成立し，推定が誤りの場合は確率 2^{-n} で成

立する．攻撃者は段鍵 K_{R+1} を推定し，式 (3.6) が成立するかどうか確認する．式 (3.6) が成立しない (0 にならない) 場合は，推定した段鍵 K_{R+1} を破棄する事で，最終的に真の段鍵 K_{R+1} を一意に求める事ができる．本論文では，式 (3.6) を段鍵 K_{R+1} を求める為の攻撃方程式と呼ぶ．式 (3.6) を解いて K_{R+1} を回復する手法として，(1) 全数探索法や，(2) 線形化手法 [42] 等が存在する．

[全数探索法]

まず，全数探索法を用いた鍵回復攻撃について説明する．一組の $(d+1)$ 階差分を用意する為に必要な選択平文数は 2^{d+1} である．式 (3.6) が n bit で構成される場合，式 (3.6) は段鍵 K_{R+1} の推定が推定が正しい場合は確率 1 で成立し，誤りの場合は確率 2^{-n} で成立する．従って，総当たりを行う段鍵 K_{R+1} を $|K|$ bit とした時，$2^{|K|} \times 2^{-n \times i} \ll 1$ を満たす i 組の異なる $(d+1)$ 階差分を用意すれば，真の段鍵 K_{R+1} を一意に求める事ができる．以上より，段鍵 K_{R+1} を求める為に必要な選択平文数 D は

$$D = 2^{d+1} \times i \tag{3.7}$$

であり，鍵回復に必要な計算量 T は

$$T = 2^{d+1} \times 2^{|K|} \times \sum_{i=0}^{i-1} 2^{-n \times i} \tag{3.8}$$

回の段関数暗号化計算量である．

[線形化手法]

続いて，線形化手法を用いた鍵回復攻撃について説明する．線形化手法は，攻撃方程式に存在する段鍵 K_{R+1} に関する高次項を全て独立な新たな一次項と見なして線形化し，得られた線形方程式を解いて段鍵を求める手法である [42]．線形化手法を用いる場合，全数探索法に比べ，段鍵を求める為に必要となる選択平文数は増加するが，計算量を削減する事ができる場合がある．

式 (3.6) を線形化手法を用いて解く場合について考える．$(d+1)$ 階差分を用いて導出した式 (3.6) をブール展開式で表記し，段鍵 K_{R+1} に関する高次項を全て新たな一次項と見なして線形化し，得られた線形方程式を行列を用いて表現すると以下の式 (3.9) の様に書く事ができる．尚，一組の $(d+1)$ 階差分から得られる線形方程式は n 本であり，線形方程式中に存在する段鍵 K_{R+1} に関する未知項数を L とする．

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,L} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,L} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_L \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \Rightarrow \mathbf{A'k} = \mathbf{b'} \tag{3.9}$$

未知項数が L である為，段鍵 K_{R+1} は $\lceil \frac{L}{n} \rceil$ 組の攻撃方程式から得られる線形方程式を解く事で求める事ができる．尚，$\lceil z \rceil$ は天井関数を表し，z より大きな最小の整数を表す．$\lceil \frac{L}{n} \rceil$ 組の攻撃方程式から得られる線形方程式を行列形式で表すと式 (3.10) となる．

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,L} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{L,1} & a_{L,2} & \cdots & a_{L,L} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_L \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \\ \vdots \\ b_L \end{bmatrix} \Rightarrow \mathbf{Ak} = \mathbf{b} \tag{3.10}$$

式 (3.10) の行列 $\mathbf{A}$ を係数行列，ベクトル $\mathbf{k}, \mathbf{b}$ をそれぞれ鍵ベクトル，定数ベクトルと呼ぶ事にする．式 (3.10) はガウス・ジョルダン法等を用いる事で段鍵 K_{R+1} に関する鍵ベクトル $\mathbf{k}$ を求める事ができる．以上より，線形化手法に必要な選択平文数 D は

$$D = 2^{d+1} \times \left\lceil \frac{L}{n} \right\rceil \tag{3.11}$$

であり，鍵回復に必要な計算量 T は，係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を求める計算量であり，

$$T = 2^{d+1} \times (L+1) \times \left\lceil \frac{L}{n} \right\rceil \tag{3.12}$$

回の段関数暗号化計算量である [1]．

3.3 従来法と問題提起

3.3.1 従来法

図 2.2 に示す Feistel 構造を持つブロック暗号を考える．入力サイズを $m = 2l$ bit とし，1/2 ずつ l bit に分割し，それぞれ上位を L，下位を R とする．この時，i 段目の出力は次式で表せる．

$$L_i = R_{i-1} \oplus F(L_{i-1}; K_i)$$
$$R_i = L_{i-1}$$

[1] 係数行列 $\mathbf{A}$ のサイズが小さい場合，ガウス・ジョルダン法に必要な計算量は，係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を求める為に必要な計算量に比べて小さい為，無視する事ができる．然しながら，係数行列 $\mathbf{A}$ のサイズが大きい場合，ガウス・ジョルダン法に必要な計算量 $O(L^3)$ も考慮する必要がある．

この時，段関数 F が全単射関数である場合，入力の下位 l bit を変数とする 4 段の高階差分特性が存在する事が知られている．

$$\Delta^{(l)} R_4 = 0 \tag{3.13}$$

上記の 4 段特性を用いて，5 段の Feistel 構造を持つブロック暗号に対し，5 段目の段鍵に対する鍵回復攻撃が可能である．5 段目の段鍵 K_5 を線形化手法 [42] を用いて回復する場合，段鍵 K_5 未知項数を L とすれば，鍵回復に必要な選択平文数 D と計算量 T はそれぞれ式 (3.11) と式 (3.12) から次式で表せる．

$$D = 2^l \times \left\lceil \frac{L}{l} \right\rceil$$
$$T = 2^l \times (L+1) \times \left\lceil \frac{L}{l} \right\rceil$$

3.3.2 問題提起

高階差分攻撃は，使用する階数が高い程，必要な選択平文数が増加する ($d+1$ 階差分を構成する場合，2^{d+1} の選択平文が必要になる．)．従来法では，4 段特性を構成する為に下位 l bit を変数とした l 階差分を用いる為，一組の 4 段特性を得る為に必要な選択平文数は 2^l である．これが，攻撃に必要な選択平文数が多くなる原因であった．また，全数探索法に比べ，効率的に段鍵を求める手法として線形化手法 [42] が知られているが，線形化手法で鍵回復を行う場合，段関数を用いて係数行列を導出する必要がある．

3.4 選択平文数の削減法

3.4.1 従来法の問題点

従来法では下位 l bit を変数とし，4 段目出力 R_4 の l bit が 0 になる特性を用いて 5 段目の鍵回復攻撃を行っている．高階差分攻撃は，使用する階数が高い程，必要な選択平文数が増加する (l 階差分を構成する場合，2^l の選択平文が必要になる．)．従って，特徴量を得る為に必要な高階差分の階数を抑える事が出来れば，その分だけ選択平文数を削減する事が可能になる．

3.4.2 本論文の提案法

本論文では，攻撃に必要な選択平文数を少なくする為の，効果的な選択平文の探索法を提案する．

図 2.2 に示す Feistel 構造を持つブロック暗号は入力サイズを $m = 2l$ bit とし，1/2 ずつ l bit に分割し，それぞれ上位を L，下位を R とする．入力サイズが l bit の段関数 F を並列な i 個の S-box s_i からなる関数とする．また，S-box s_i の入力サイズを n_i bit とする．

平文 P の下位 l bit を下記の通り S-box s_i の入力サイズ（n_i bit）に合わせて分割する [2]．

$$P = P_L||p_i||p_{i-1}||\cdots||p_1||p_0, p_i \in GF(2)^{n_i} \tag{3.14}$$

式 (3.25) において，p_i を変数とした 4 段特性が存在するかどうか探索を行う．S-box s_i の入力サイズ n_i bit に合わせて高階差分特性を探索する事で，下位 l bit 全てを変数とした 4 段特性と比べ，少ない階数で高階差分特性が見つかる可能性がある．

本論文の提案法によって期待される効果として，従来法は一組の高階差分特性に用いる選択平文数が 2^l であるが，提案法を用いた場合の選択平文数は 2^{l-n_i} 以下となる．即ち，鍵回復攻撃に必要な選択平文数を削減する事が可能になる．

3.5 攻撃方程式の高速解法

3.5.1 従来法の問題点

段鍵に関する攻撃方程式を解く方法として線形化手法が知られている [42]．この手法は全数探索法に比べ，効率的に段鍵を求める手法であるが，段関数を用いて式 (3.10) の係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を導出する必要がある．

3.5.2 本論文の提案法

本論文では，線形化手法の高速解法に関する技法を提案する．その高速化技法は，下記 2 つである．

1. 高速化技法 1 : 事前に攻撃方程式のブール展開式を解析し，段関数を使わずに係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を導出する (段関数を使わない事による高速化．攻撃方程式中に存在する非線形関数の代数次数が小さく，未知項数が少ない場合に効果的．)

2. 高速化技法 2 : 線形化方程式中に存在する段鍵 K_{R+1} の高次項から成る未知項数 L の係数を事前に 0 に制御する事で，求める未知項数を削減する (段鍵 K_{R+1} の高次項の係数は暗号文の低次多項式である事に着目し，事前に未知項数を減らす事による高速化．未知項数が多い場合に効果的．)

[2] 平文 P の上位 l bit を変数とした場合，1 段目から F 関数を通過し，次数が上昇する．従って，本節では，平文 P の下位 l bit に変数箇所を限定する事で，変数による次数上昇の影響を 2 段目からにする事が可能となる．

以下，２つの高速化技法を説明する．

[高速化技法 1]

高速化技法 1 は，攻撃者にとって既知の情報である攻撃方程式のブール展開式を事前に解析する事で，段関数を使わずに係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を導出する手法である．

攻撃方程式 (3.6) を以下に再掲する．

$$\bigoplus_{\alpha \in V^{(d+1)}} F_{R+1}^{-1}(C(X \oplus \alpha); K_{R+1}) = 0$$

攻撃対象の暗号の内部構造は攻撃者に既知である為，攻撃者は上記の復号化関数 $F_{R+1}^{-1}(\cdot)$ の構造 (ブール展開式) を事前に把握する事が可能である．攻撃方程式 (3.6) をブール展開し，段鍵 K_{R+1} に関する高次項を全て新たな一次項と見なして線形化した方程式 (3.9) を以下に再掲する．

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,L} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,L} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_L \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \Rightarrow \mathbf{A'k} = \mathbf{b'}$$

係数行列 $\mathbf{A'}$ の成分 $a_{i,j}$, $(1 \leq i \leq n, 1 \leq j \leq L)$ と定数ベクトル $\mathbf{b'}$ の成分 b_i, $(1 \leq i \leq n)$ は暗号文 $C = (c_1, c_2, \ldots, c_m)$, $c_i \in GF(2)$ のブール展開式で表現可能である．式 (3.9) の係数行列 $\mathbf{A'}$ の列ベクトルと定数ベクトル $\mathbf{b'}$ を n 次元ベクトル $\mathbf{a}_j$, $(1 \leq j \leq L+1)$ とする．また，ブール展開式で表した暗号文の項数を C とし，暗号文の一次項から高次項を並べたベクトルを $\mathbf{c} = {}^{\mathrm{T}}(c_1, c_2, \ldots, c_1c_2, \ldots, c_1c_2c_3, \ldots)$ する．記号 'T' はベクトルの転置を示す．この時，n 次元ベクトル $\mathbf{a}_j$, $(1 \leq j \leq L+1)$ は $(d+1)$ 階差分入力に対応する暗号文を用いて以下の様に計算できる．

$$\mathbf{a}_j = \bigoplus_{\alpha \in V^{(d+1)}} \mathbf{A}_j \times \mathbf{c}\ , (1 \leq j \leq L+1) \tag{3.15}$$

$\mathbf{A}_j$ は $n \times$ C サイズの行列であり，$\mathbf{c}$ は C 次元ベクトルである．式 (3.15) を以下の様に変形する．

$$\mathbf{a}_j = \mathbf{A}_j \times \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c} = \mathbf{A}_j \times \mathbf{c}_i\ , \left(1 \leq i \leq \left\lceil \frac{L}{n} \right\rceil, 1 \leq j \leq L+1\right) \tag{3.16}$$

$V_i^{(d+1)}$ は i 組目の攻撃方程式で利用する $(d+1)$ 次元部分空間とする．$\mathbf{c}_i$ は i 組目の $(d+1)$ 階差分入力に対応する暗号文を用いて計算した C 次元ベクトルである．式 (3.16) を用いる事により，式 (3.6) 中の復号化関数 $F_{R+1}^{-1}(\cdot)$ を使わずに式 (3.9) の係数行列と定数ベクトルを計算する事ができる．

テーブル G を m bit の暗号文から C 次元ベクトル $\mathbf{c}$ の要素を出力するテーブルとし，ベクトル $\mathbf{c}$ をテーブル G を用いて導出する．テーブル G を図 3.2 に示す．

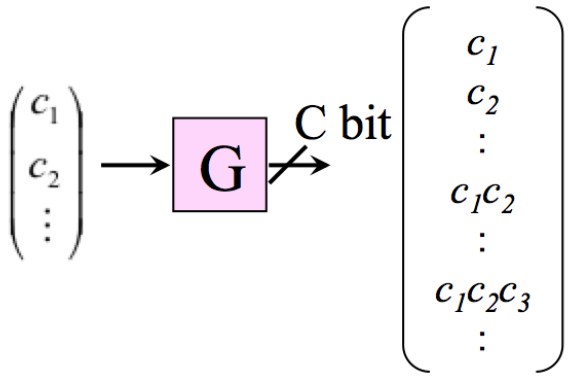

図 3.2: テーブル G

通常，暗号の中で用いる非線形関数 S-box は，テーブル参照で実装されている．本論文も同様に，テーブル G はテーブル参照によって実装するものとする．本論文では，非線形関数 S-box の一回のテーブル参照に必要な計算量と，テーブル G の一回のテーブル参照に必要な計算量を同一とみなす．もし，テーブル G を 32 bit プロセッサー上で実装した場合，C bit の要素を全て得る為には，$\lceil \frac{\mathrm{C}}{32} \rceil$ 回のテーブル参照が必要である．

以下に高速化技法 1 のアルゴリズムと，鍵回復に必要な選択平文数 D と計算量 T を示す．

高速化技法 1 のアルゴリズム

1. $\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分入力に対応する暗号文を用意する．
2. $\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \leq i \leq \left\lceil \frac{L}{n} \right\rceil)$ を計算する．
3. 式 (3.16) を用いて n 次元ベクトル $\mathbf{a}_j, (1 \leq j \leq L+1)$ を計算する．
4. 得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する．

鍵回復に必要な選択平文数 D と計算量 T

$$D = 2^{d+1} \times \left\lceil \frac{L}{n} \right\rceil \tag{3.17}$$

$$T_1 = 2^{d+1} \times \left\lceil \frac{L}{n} \right\rceil$$

$$T_2 = 2^{d+1} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times \left\lceil \frac{L}{n} \right\rceil$$

$$T_3 = 2 \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times n \times (L+1) \times \left\lceil \frac{L}{n} \right\rceil$$

$$T_4 = \left\lceil \frac{L^3}{32} \right\rceil$$

$$T = T_1 + \frac{T_2 + T_3 + T_4}{\#S} \tag{3.18}$$

$\#S$ は対象の暗号に存在する非線形関数 S-box の数を示す．鍵回復に必要な選択平文数は，式 (3.11) で示した線形化手法で必要な選択平文数 D に等しい．計算量は式 (3.18) で見積もられる．尚，計算量の単位は，対象の暗号を一回暗号化 (又は復号化) する為に必要な演算量を示し，[ENC.] で表示する．鍵回復に必要な計算量の見積もり方法は，本論文の付録 B に示す．

高速化技法 1 は，係数行列 $\mathbf{A}$ のサイズが小さい場合に効果的である．しかしながら，線形方程式中に存在する未知項数 L が増えるに連れて，ガウス・ジョルダン法で鍵回復を行う部分の計算量 $\left\lceil \frac{L^3}{32} \right\rceil$ が全体の計算量 T の中で支配的となる．従って，本論文では，計算量を削減する手法を以下に示す．

[高速化技法 2]

高速化技法 2 は，線形化した方程式中に存在する段鍵 K_{R+1} に関する未知項数 L の係数を 0 に制御する事で，求める未知項数を削減する手法である．段鍵 K_{R+1} の高次項の係数は暗号文の低次多項式である事に着目し，事前に未知項数を減らす事による高速化で，未知項数が多い場合に効果的である．

$(d+1)$ 階差分を用いて導出した Q 組の攻撃方程式を考える．$V_i^{(d+1)}$ を i 組目の攻撃方程式で利用する $(d+1)$ 次元部分空間とする．

$$\bigoplus_{\alpha \in V_i^{(d+1)}} F_{R+1}^{-1}(C(X \oplus \alpha); K_{R+1}) = 0 \ , (1 \le i \le Q)$$

$e_i \in \{0, 1\}$ を任意の係数とし，以下の方程式を考える．

$$\bigoplus_{i=1}^{Q} e_i \bigoplus_{\alpha \in V_i^{(d+1)}} F_{R+1}^{-1}(C(X \oplus \alpha); K_{R+1}) = 0 \ , (1 \le i \le Q) \tag{3.19}$$

式 (3.19) の復号化関数部分をブール展開し，段鍵 K_{R+1} に関する高次項を全て新たな一次項と見

なして線形化した方程式を以下に示す.

$$\bigoplus_{i=1}^{Q} e_i \left\{ \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,L} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,L} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,L} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_L \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \right\}$$

高速化技法 1 で示した様に，係数行列の列ベクトルと定数ベクトルを m 次元ベクトル $\mathbf{a}_j, (1 \leq j \leq L+1)$ とすれば，式 (3.16) より，$\mathbf{a}_j$ は以下の通りに計算する事ができる.

$$\mathbf{a}_j = \bigoplus_{i=1}^{Q} e_i \left\{ \mathbf{A}_j \times \mathbf{c}_i \right\} = \mathbf{A}_j \times \bigoplus_{i=1}^{Q} e_i \left\{ \mathbf{c}_i \right\} \ , (1 \leq i \leq \left\lceil \frac{L}{n} \right\rceil , 1 \leq j \leq L+1) \tag{3.20}$$

式 (3.20) において，$Q(>\mathrm{C})$ 組の攻撃方程式を用意すれば，適切な e_i を選ぶ事により $\bigoplus_{i=1}^{Q} e_i \left\{ \mathbf{c}_i \right\} = \mathbf{0}$ とする事ができ，$\mathbf{a}_j = \mathbf{0}$ となる．これは即ち，線形化した方程式において，$\mathbf{a}_j$ に対応する未知項を消す事と同じである． 従って，$\mathbf{a}_j$ に対応する未知項を求める必要が無くなる為，この手法を用いる事により，鍵回復で求める未知項の数を減らす事ができる.

線形方程式中に存在する $L'(< L)$ 個の未知項数を消去する場合を考える．式 (3.10) 中に存在する未知項数 L を L' と $L''(= L - L')$ に分割する．n 次元ベクトル $\mathbf{a}_j, (1 \leq j \leq L')$ を未知項数 L' に関わる係数行列の列ベクトルとし，$\mathbf{a}_j$ の各要素は $\mathrm{C}'(< \mathrm{C})$ 個の暗号文単項式の線形和で構成されているものとする．同様に，n 次元ベクトル $\mathbf{a}_j, (L'+1 \leq j \leq L+1)$ を未知項数 L'' に関わる係数行列の列ベクトルと定数ベクトルとし，$\mathbf{a}_j$ は式 (3.15) にて計算できるものとする．以上より，式 (3.16) は以下の様に表す事ができる.

$$\mathbf{a}_j = \begin{cases} \mathbf{A}'_j \times \mathbf{c}'_i \ , (1 \leq j \leq L') \\ \mathbf{A}_j \times \mathbf{c}_i \ , (L'+1 \leq j \leq L+1), \end{cases} \tag{3.21}$$

$\mathbf{A}'_j$ は $n \times \mathrm{C}'$ サイズの定数行列である．また，$\mathbf{c}'_i$ は $\mathrm{C}'(< \mathrm{C})$ 次元ベクトルであり，C 次元ベクトル $\mathbf{c}_i$ の一部分である．もし，$Q(\mathrm{C}' < Q < \mathrm{C})$ 組の攻撃方程式を用意した場合，式 (3.20) と式 (3.21) から以下の式を導出する事ができる.

$$\mathbf{a}_j = \begin{cases} \mathbf{A}'_j \times \bigoplus_{i=1}^{Q} e_i \left\{ \mathbf{c}'_i \right\} \ , (1 \leq j \leq L') \\ \mathbf{A}_j \times \bigoplus_{i=1}^{Q} e_i \left\{ \mathbf{c}_i \right\} \ , (L'+1 \leq j \leq L+1), \end{cases} \tag{3.22}$$

適切な $e_i \in \{0,1\}$ を選ぶ事により $\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ とする事ができ，式 (3.22) から $\mathbf{a}_j = \mathbf{0}, (1 \leq j \leq L')$ となる．L' 個の未知項に対応する $\mathbf{a}_j, (1 \leq j \leq L')$ が $\mathbf{0}$ である為，L' 個の未知項は推定する必要が無くなる．一方，$\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}_i\} \neq \mathbf{0}$ である為，L'' 個に対応する未知項を計算する事が可能である．

以下に高速化技法 2 のアルゴリズムと，鍵回復に必要な選択平文数 D と計算量 T を示す．

高速化技法 2 のアルゴリズム

1. Q 組の $(d+1)$ 階差分入力に対応する暗号文を用意する．
2. Q 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \leq i \leq Q)$ を計算する．
3. 適切な $e_i \in \{0,1\}$ を選ぶ事により，$\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ とする．
4. 式 (3.22) を用いて n 次元ベクトル $\mathbf{a}_j, (L'+1 \leq j \leq L+1)$ を計算する．
5. 得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する．

鍵回復に必要な選択平文数 D と計算量 T

$$D = 2^{d+1} \times Q \tag{3.23}$$

$$T_1 = 2^{d+1} \times Q$$
$$T_2 = 2^{d+1} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times Q$$
$$T_3 = \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times \frac{Q-1}{2} \times \mathrm{C}'$$
$$T_4 = 2 \times \left\lceil \frac{\mathrm{C}-\mathrm{C}'}{32} \right\rceil \times n \times (L''+1) \times \left\lceil \frac{L''}{n} \right\rceil$$
$$T_5 = \left\lceil \frac{L''^3}{32} \right\rceil$$
$$T = T_1 + \frac{T_2 + T_3 + T_4 + T_5}{\#S} \tag{3.24}$$

$\#S$ は対象の暗号に存在する非線形関数 S-box の数を示す．鍵回復に必要な選択平文数は，Q 組の $(d+1)$ 階差分を構成する為に必要な選択平文数である．計算量は式 (3.24) で見積もられる．尚，計算量の単位は，対象の暗号を一回暗号化 (又は復号化) する為に必要な演算量を示し，[ENC.] で表示する．鍵回復に必要な計算量の見積もり方法は，本論文の付録 B に示す．

3.6 KASUMIの高階差分攻撃への適用

3.6.1 田中らの5段攻撃の問題点

田中らはKASUMIの段関数(FO関数とFL関数)が全単射関数である性質を用いて，下位32 bit(入力64 bit中，右側32 bit)を変数とした32階差分を用いた4段特性が存在する事を示した(図3.3参照．R_4は4段目右出力32 bitを示す．$\Delta^{(32)}$は32階差分を示す．)．田中らは，この4段特性を用いて5段目の段鍵を全数探索法で導出している．攻撃に必要な選択平文数は$D = 2^{39.4}$，計算量は$T = 2^{117}$回の暗号化計算量であると見積もられている[61].

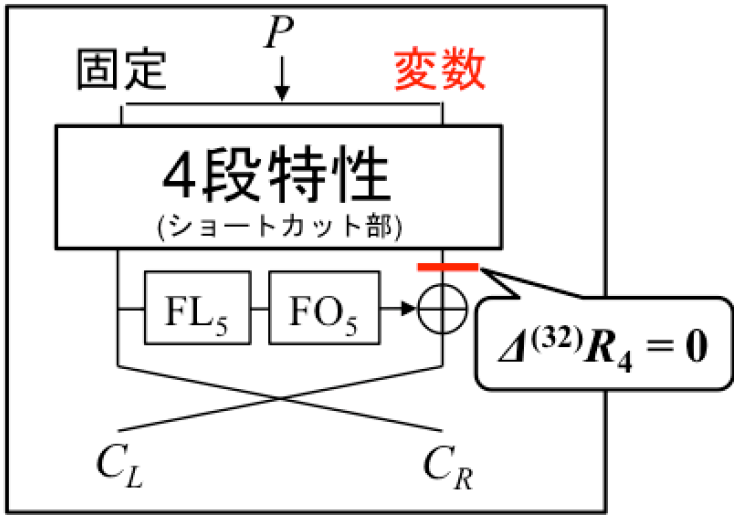

図 3.3: 田中らが示したKASUMIの4段特性

田中らは下位32 bitを変数とした32階差分を攻撃に用いている．この場合，一組の高階差分特性を得る為に必要な選択平文数は2^{32}である．これが，攻撃に必要な選択平文数が多くなる原因であった．また，鍵回復において，田中らは段鍵を全数探索法を用いている為，計算量は段鍵のビット数分の試行が必要となる．田中らの手法を用いた場合，攻撃方程式中に存在する段鍵は82 bitであり，$2^{32} \times 2^{82}$回の暗号化計算量が必要になる．

3.6.2 選択平文数の削減

本節では，5段のKASUMIに対し，3.4節で示した効果的な選択平文の探索法を適用し，提案法の性能評価を行う．平文Pの下位32 bitを下記の通りS-boxの入力サイズ(7 bit，又は9 bit)

に合わせて分割する．

$$P = P_L||p_3||p_2||p_1||p_0, \{p_3, p_1\} \in GF(2)^9, \{p_2, p_0\} \in GF(2)^7 \tag{3.25}$$

式 (3.25) において，$p_i, (i = 0, 1, 2, 3)$ に変数を設定し，4 段特性が存在するかどうか計算機実験により探索を行う．平文 $p_i, (i = 0, 1, 2, 3)$ に変数を設定するパターン数 (探索数) は $\binom{4}{3} + \binom{4}{2} + \binom{4}{1} = 14$ 通りである．図 3.4 に示す通り，4 段目出力 R_4 は S-box の出力サイズ 7 bit と 9 bit を交互に繰り返した 32 bit である．従って，32 bit を纏めて特性を調べるのではなく，S-box の出力サイズに合わせて高階差分特性を探索する事で，32 階差分よりも少ない階数で高階差分特性が見つかる可能性がある．

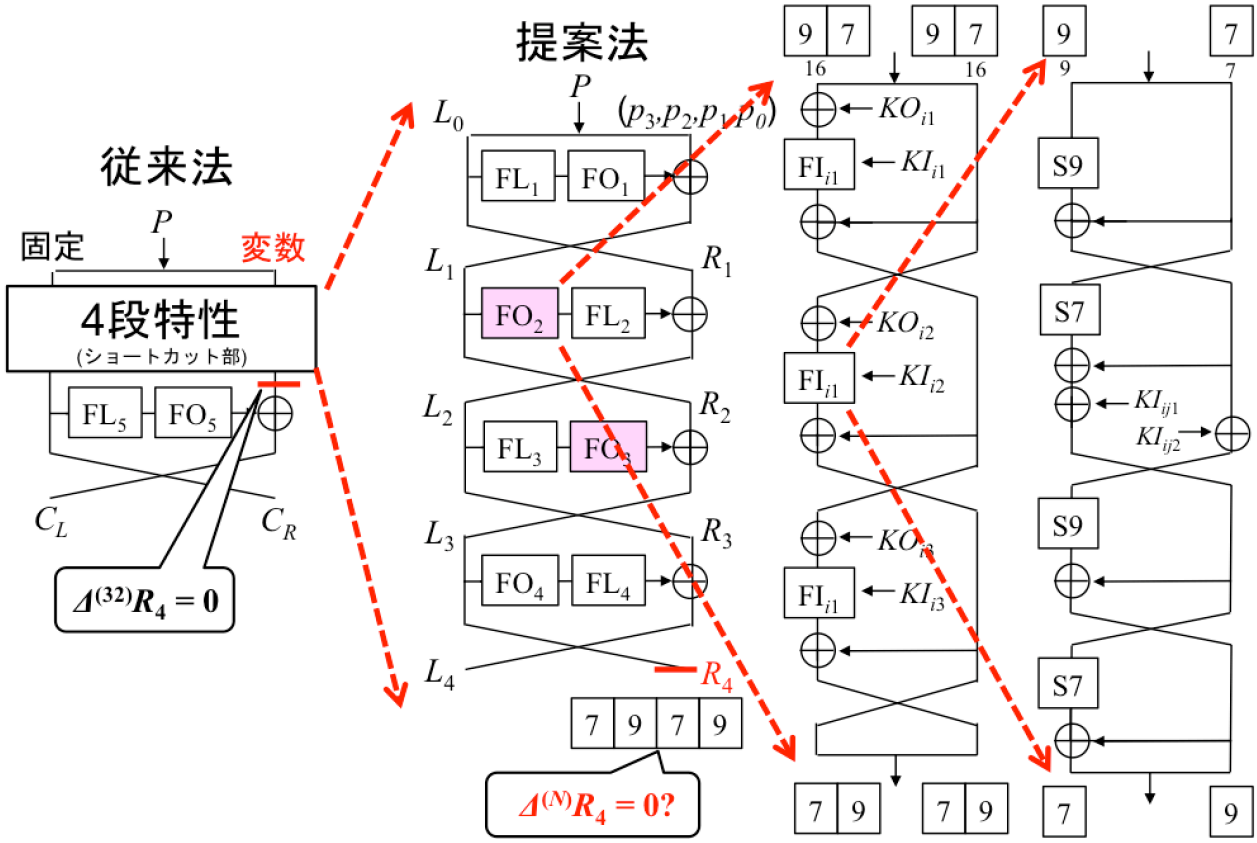

図 3.4: 効果的な選択平文の探索

本論文の提案法によって期待される効果として，従来法は一組の高階差分特性に用いる選択平文数が 2^{32} である事に対し，提案法を用いた場合の一組の高階差分特性に用いる選択平文数は 2^{25} 以下となる．即ち，鍵回復攻撃に必要な選択平文数を削減する事が可能になる．

式 (3.25) の $p_i, (i = 0, 1, 2, 3)$ に対し，高階差分特性を得る為の変数を設定し，計算機実験により 4 段特性を調査した．その結果，表 3.2 に示す 3 通りで，4 段の高階差分特性が存在する事を確認した．

表 3.2 において，$\{v_1, v_3\} \in GF(2)^9$，$\{v_0, v_2\} \in GF(2)^7$ は変数箇所を示し，$p_1 \in GF(2)^9$ と

表 3.2: KASUMI の新たな 4 段特性

変数箇所	選択平文数	高階差分値が 0 となる 4 段目出力のビット位置
(v_3, v_2, p_1, p_0)	2^{16}	$R_4[24-16]$
(v_3, v_2, p_1, v_0)	2^{23}	$R_4[24-16]$
(v_3, v_2, v_1, p_0)	2^{25}	$R_4[24-16]$

$p_0 \in GF(2)^7$ は任意の固定値を示す．$R_4[24-16]$ は，高階差分特性で 0 が得られた 4 段目出力の一部 (16 bit から 24 bit 目迄の 9 bits) である．尚，4 段の高階差分特性を計算機実験で調査する際には，各々のパターンに対し，100 通りの異なる平文固定値と段鍵を設定する事で，偶然性を排除している [3]．16 階差分を用いた新たな 4 段特性を図 3.5 に示す．

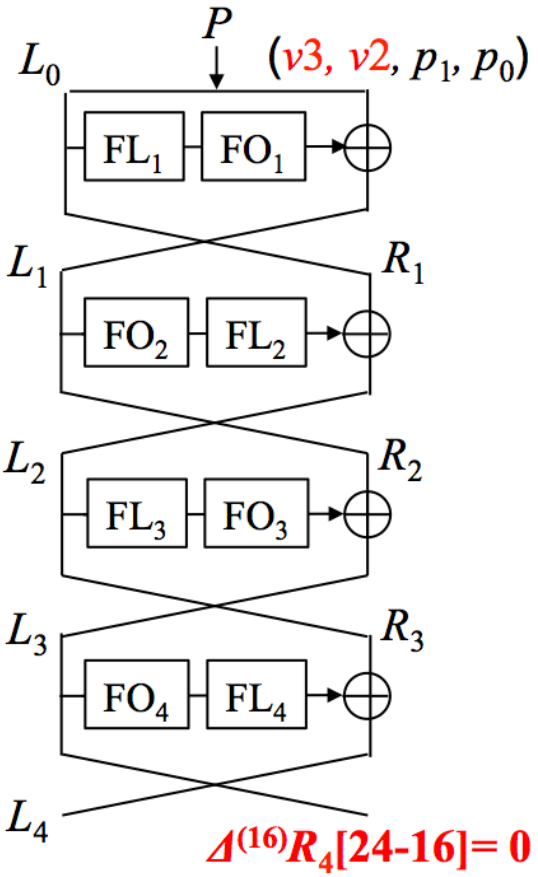

図 3.5: 新たな 4 段特性 (16 階差分)

田中らの示した 4 段の高階差分特性では，一組当たり 2^{32} の選択平文数を必要とする．本論文の提案法によって得られた結果と比較したものを表 3.3 に示す．本論文の提案法を用いた場合，一組

[3]得られた高階差分特性は 9 bit である．100 通りの異なる平文固定値と段鍵を用いて，偶然に 4 段の高階差分特性が成立する確率は $p = (2^{-9})^{100} = 2^{-900}$ である．$2^{-900} \ll 1$ より，本論文では，偶然性は排除されたものと見なす．

の 4 段高階差分特性を得る為に必要な選択平文数は，従来法と比べ最大 2^{16} 程度削減可能である．

表 3.3: 従来法との比較

	変数箇所	選択平文数	従来法からの削減数
従来法	(v_3, v_2, v_1, v_0)	2^{32}	
提案法	(v_3, v_2, p_1, p_0)	2^{16}	2^{16}
	(v_3, v_2, p_1, v_0)	2^{23}	2^{9}
	(v_3, v_2, v_1, p_0)	2^{25}	2^{7}

3.6.3 計算量の削減

本節では，5 段の KASUMI に対し，3.5 節で示した高速化技法 1，及び高速化技法 2 を適用し，提案法の性能評価を行う．具体的には，5 段の KASUMI に対し，3.4 節で示した効果的な選択平文を用いて 5 段目で使用される段鍵を推定する為の攻撃方程式を導出し，その攻撃方程式を解く際に，高速化技法 1，及び高速化技法 2 を適用する．

64 bit の入力平文の内，上位 32 bit(左側 32 bit) の P_L に任意の固定値を設定し，下位 32 bit(右側 32bit) に表 3.2 に示す選択平文を使用する．

$$P = P_L||P_R, P_R = v_3||v_2||p_1||p_0, \tag{3.26}$$

$\{v_3\} \in GF(2)^9$ と $\{v_2\} \in GF(2)^7$ は変数を示し，$P_L \in GF(2)^{32}$，$p_1 \in GF(2)^9$ と $p_0 \in GF(2)^7$ は任意の固定値を示す．上記の選択平文を用いる事で，一次独立な 16 種類のベクトル $\{\boldsymbol{\alpha}_{16}, \boldsymbol{\alpha}_{17}, \ldots, \boldsymbol{\alpha}_{31}\} \in GF(2)^{64}$ によって張られる 16 次元部分空間 $V^{(16)}$ を構成する事ができる．尚，ここで示すベクトル $\boldsymbol{\alpha}_i \in GF(2)^{64}, (16 \leq i \leq 31)$ は i bit 目の成分が 1 で，その他の成分が 0 のベクトルとする．式 (3.26) に示す選択平文を用いた場合，以下に示す 4 段の高階差分特性が得られる (図 3.5 参照)．

$$\bigoplus_{\alpha \in V^{(16)}} R_4[24-16] = 0$$

ここで，$R_4[24-16]$ は 4 段目出力 R_4 32 bit の内，i bit 目 $(16 \leq i \leq 24)$ を示す．上記の 4 段の高階差分特性を用いる事で，5 段目の段鍵 KO_5, KL_5 を推定する為の攻撃方程式を導出する事ができる (図 3.6 参照)．

$$\bigoplus_{\alpha \in V^{(16)}} FO5(FL5(C_R(X \oplus \alpha); KL_5); KO_5)[24-16] \oplus \bigoplus_{\alpha \in V^{(16)}} C_L(X \oplus \alpha)[24-16] = 0, \tag{3.27}$$

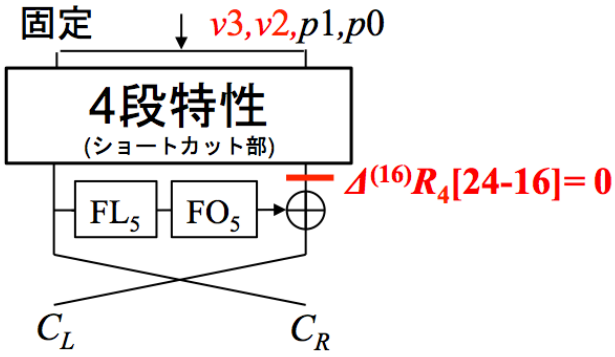

図 3.6: 5 段 KASUMI の鍵回復

C_L, C_R は暗号文の上位 32 bit(左側 32 bit) と下位 32 bit(右側 32 bit) を示す．また，KO_5 は 50 bits $(KO_{51} \in GF(2)^{16}, KO_{52} \in GF(2)^{16}, KI_{512} \in GF(2)^9, KI_{522} \in GF(2)^9)$，及び KL_5 は 32 bits $(KL_{51} \in GF(2)^{16}, KL_{52} \in GF(2)^{16})$ であり，式 (3.27) は合計 82 bits の段鍵に関する攻撃方程式である．$FO5[24-16]$ は FO5 出力 32 bit の内，i bit 目 $(16 \leq i \leq 24)$ を示す．また，同様に，$C_L[24-16]$ は暗号文 C_L 32 bit の内，i bit 目 $(16 \leq i \leq 24)$ を示す．

線形化手法を用いる場合，攻撃方程式 (3.27) 中に存在する 82 bit の段鍵に関する未知項数を調べる必要がある．本論文では，数式解析ソフトウェア REDUCE version 3.6 を用いて未知項数 L を解析した．尚，未知項数 L の解析において，南部らが提案した見かけ上の次数上昇を抑える手法 [62] を採用した．その結果，未知項数 L は $L = 26,693$ である．また，i bit 目 $(16 \leq i \leq 24)$ に存在する未知項数を L_i とした時，各 bit に存在する未知項数は表 3.4 の通りである．

表 3.4: 未知項数 L の解析

bit 位置	未知項数 L_i
16-th bit	13,893
17-th bit	13,899
18-th bit	13,255
19-th bit	13,183
20-th bit	13,403
21-st bit	14,003
22-nd bit	13,605
23-rd bit	13,657
24-th bit	14,447
all	26,693

all は攻撃方程式を 9 bit とみた場合の未知項数を示す．

攻撃方程式 (3.27) を線形化すると，以下の様に変形できる．

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,26693} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,26693} \\ \vdots & \vdots & \ddots & \vdots \\ a_{9,1} & a_{9,2} & \cdots & a_{9,26693} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_{26693} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_9 \end{bmatrix} \Rightarrow \mathbf{A}'\mathbf{k} = \mathbf{b}' \tag{3.28}$$

係数行列 $\mathbf{A}'$ は $9 \times 26,693$ サイズの行列，鍵ベクトル $\mathbf{k}$ は 26,693 次元ベクトル，定数ベクトル $\mathbf{b}'$ は 9 次元ベクトルである．未知項数が $L = 26,693$ である為，段鍵 KO_5, KL_5 は $\lceil \frac{26,693}{9} \rceil$ 組の攻撃方程式から得られる線形方程式を解く事で求める事ができる．$\lceil \frac{26,693}{9} \rceil$ 組の攻撃方程式から得られる線形方程式を行列形式で表すと式 (3.29) となる．

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,26693} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,26693} \\ \vdots & \vdots & \ddots & \vdots \\ a_{9,1} & a_{9,2} & \cdots & a_{9,26693} \\ \vdots & \vdots & \ddots & \vdots \\ a_{26693,1} & a_{26693,2} & \cdots & a_{26693,26693} \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \\ \vdots \\ k_{26693} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_9 \\ \vdots \\ b_{26693} \end{bmatrix} \Rightarrow \mathbf{A}\mathbf{k} = \mathbf{b} \tag{3.29}$$

式 (3.29) はガウス・ジョルダン法等を用いる事で段鍵 KO_5, KL_5 に関する鍵ベクトル $\mathbf{k}$ を求める事ができる．以下では，それぞれ高速化技法 1，高速化技法 2 を用いた場合の選択平文数と計算量の見積もりを行う．

[高速化技法 1 の適用]

高速化技法 1 を用いて係数行列 $\mathbf{A}'$ と定数ベクトル $\mathbf{b}'$ を導出する場合，事前に式 (3.15) に示す行列 $\mathbf{A}_j, (1 \leq j \leq 26,694)$ とベクトル $\mathbf{c}$ を解析する必要がある．本論文では，攻撃方程式 (3.27) を数式解析ソフトウェア REDUCE version 3.6 を用いてブール展開し，行列 $\mathbf{A}_j, (1 \leq j \leq 26,694)$ とベクトル $\mathbf{c}$ を解析した．その結果，$\mathbf{A}_j, (1 \leq j \leq 26,694)$ は $9 \times 9,109$ サイズの行列と，$\mathbf{c}$ は 9,109 次元ベクトルである事が明らかになった．ベクトル $\mathbf{c}$ の要素数を表 3.5 に示す．

行列 $\mathbf{A}_j, (1 \leq j \leq 26,694)$ とベクトル $\mathbf{c}$ を使用する高速化技法 1 に必要な選択平文数と計算量の見積もりを行う．鍵回復に必要な選択平文数 D は式 (3.17) から，

$$D = 2^{16} \times \left\lceil \frac{26,693}{9} \right\rceil \approx 2^{27.5} \tag{3.30}$$

表 3.5: ベクトル $\mathbf{c}$ の要素数

bit 位置	$\mathbf{c}$ の要素数
16-th bit	6,537
17-th bit	6,686
18-th bit	6,237
19-th bit	6,433
20-th bit	6,419
21-st bit	6,713
22-nd bit	6,569
23-rd bit	6,493
24-th bit	6,854
all	9,109

all は攻撃方程式を 9 bit とみた場合の要素数を示す.

となる．また，計算量 T は式 (3.18) から，

$$T_1 = 2^{16} \times \left\lceil \frac{26,693}{9} \right\rceil \approx 2^{27.5}$$

$$T_2 = 2^{16} \times \left\lceil \frac{9,109}{32} \right\rceil \times \left\lceil \frac{26,693}{9} \right\rceil \approx 2^{35.69}$$

$$T_3 = 2 \times \left\lceil \frac{9,109}{32} \right\rceil \times 9 \times (26,694) \times \left\lceil \frac{26,693}{9} \right\rceil \approx 2^{38.56}$$

$$T_4 = \left\lceil \frac{26,693^3}{32} \right\rceil \approx 2^{39.11}$$

$$T = 2^{27.5} + \frac{2^{35.69} + 2^{38.56} + 2^{39.11}}{\#S} = 2^{27.5} + \frac{2^{35.69} + 2^{38.56} + 2^{39.11}}{12 \times 5} \approx 2^{34} \tag{3.31}$$

回の暗号化計算量 [ENC.] である．尚，$\#S$ は対象の暗号に存在する非線形関数 S-box の数を示す．KASUMI の FO 関数一段当たり S-box は 12 個存在し，対象の暗号は 5 段 KASUMI の為，$\#S = 12 \times 5$ である．

高速化技法 1 は，係数行列 $\mathbf{A}$ のサイズが小さい場合に効果的である．しかしながら，線形方程式中に存在する未知項数 L が増えるに連れて，ガウス・ジョルダン法で鍵回復を行う部分の計算量 $\left\lceil \frac{L^3}{32} \right\rceil$ が全体の計算量 T の中で支配的となる．式 (3.31) において，線形化方程式をガウス・ジョルダン法を用いて鍵回復を行う部分の計算量 $\left\lceil \frac{26,693^3}{32} \right\rceil \approx 2^{39.11}$ が支配的である．従って，本論文では，高速化技法 2 を適用する事で，更なる計算量の削減を行う．

[高速化技法 2 の適用]

高速化技法 2 は，線形化した方程式中に存在する段鍵 K_{R+1} に関する未知項数 L の係数を 0 に制御する事で，求める未知項数を削減する手法である．段鍵 K_{R+1} の高次項の係数は暗号文の低次多項式である事に着目し，事前に未知項数を減らす事による高速化で，未知項数が多い場合に効果的である．高速化技法 2 を適用する場合，未知項数 L とベクトル $\mathbf{c}$ の次元数 C に加え，消去する未知項数 $L'(< L)$ と導出する未知項数 $L''(= L - L')$，及び消去する未知項数 L' に関連する暗号文単項式の数 $\mathrm{C}'(< \mathrm{C})$ を明らかにする必要がある．そこで本論文では，攻撃方程式 (3.27) をブール展開し，段鍵に関する未知項と，それらの係数に存在する暗号文多項式を解析した．その上で，計算量を減らす事が可能な L', L'' と C' の探索を行った．以下では，高速化技法 2 の適用に関し，我々が実施したアプローチの概要を説明する．

FL 関数は鍵依存の線形関数であり，鍵の AND 演算と OR 演算，1 bit 左巡回シフトから構成される．FL 関数を GF(2) 上の多項式で表現した場合，その鍵に関する代数次数は 2 次となる [4]．この場合，攻撃方程式に存在する FO 関数の出力 9 bit：FO5[24-16] は，代数次数 2 次の S9-box を 2 回通過する経路が存在する為，段鍵 KL_{51}，KL_{52} に関する代数次数は最大 8 次となる．南部らは FL 関数の出力で暗号文に関する項を纏め，その係数 (段鍵 KL_{51}，KL_{52} に関する最大 2 次項を含む多項式) を一次の等価な鍵に置き換える事で，見かけ上の次数上昇を抑える手法を提案した [62]．この手法を用いる事により，鍵に関する次数上昇を最大 4 次に抑える事が可能に成る．本論文でも南部らの手法を用いるとすれば，線形化方程式中に存在する鍵に関する未知項の次数は最大 4 次となる．高速化技法 2 を適用するに当たり，最大 n 次 $(1 \leq n \leq 4)$ 以上の未知項を消去する未知項数 L' と定める事で，各々のパラメータ $(L', L''(= L - L'), \mathrm{C}')$ を解析した．各パラメータの解析結果を表 3.6 に示す．

表 3.6: パラメータ (L'，L''，C') の解析結果

消去する未知項数 L' の次数	消去する未知項数 L'	推定する未知項数 L''	C'	C''=(C-C')
4 次項	11,970	14,723	2,943	6,166
3 次以上	24,777	1,916	7,205	1,904
2 次以上	26,595	98	8,557	552
1 次以上	26,693	0	8,983	126

表 3.6 に示したパラメータを用いて，高速化技法 2 に必要な選択平文数と計算量の見積もりを行う．尚，全ての未知項数 $L = 26,693$ を消去した場合は段鍵を回復する事が出来なくなる事に

[4]任意の 1 bit 変数 $a, b \in \{0, 1\}$ において，AND 演算は GF(2) 上で a AND $b = a \cdot b$ となる．尚，ドット $(\cdot)$ はビット積を示す．また，OR 演算は GF(2) 上で a OR $b = a \cdot b \oplus a \oplus b$ となる．これを踏まえて FL 関数の入力を一次の暗号文と段鍵 KL_{51} の AND 演算，KL_{52} との OR 演算を行った場合，出力の段鍵 KL_{51}，KL_{52} に関する代数次数は 2 次となる．

注意が必要である．消去する未知項数 L' の次数を 2 次以上，3 次以上，4 次項の 3 つのパターンで鍵回復に必要な計算量を試算した結果，3 次以上の未知項数 $L' = 24,777$ を消去する場合が最も計算量を削減する事が出来た．その際に必要な選択平文数 D は式 (3.23) から，

$$D = 2^{16} \times 24,777 \approx 2^{28.86} \tag{3.32}$$

である．また，計算量 T は式 (3.24) から，

$$T_1 = 2^{16} \times 24,777 \approx 2^{28.86}$$

$$T_2 = 2^{16} \times \left\lceil \frac{9{,}109}{32} \right\rceil \times 24,777 \approx 2^{37.01}$$

$$T_3 = \left\lceil \frac{9{,}109}{32} \right\rceil \times \frac{24,776}{2} \times 7,205 \approx 2^{32.83}$$

$$T_4 = 2 \times \left\lceil \frac{9{,}109 - 7{,}205}{32} \right\rceil \times 9 \times (1,916 + 1) \times \left\lceil \frac{1,916}{9} \right\rceil \approx 2^{28.72}$$

$$T_5 = \left\lceil \frac{1,916^3}{32} \right\rceil \approx 2^{27.71}$$

$$T = 2^{28.86} + \frac{2^{37.01} + 2^{32.83} + 2^{28.72} + 2^{27.71}}{12 \times 5} \approx 2^{31.5} \tag{3.33}$$

回の暗号化計算量 [ENC.] である．高速化技法 2 を用いる事により，高速化技法 1 の計算量で支配的であった線形化方程式をガウス・ジョルダン法を用いて解く部分の計算量 $\left\lceil \frac{26,693^3}{32} \right\rceil \approx 2^{39.11}$ を $\left\lceil \frac{1,916^3}{32} \right\rceil \approx 2^{27.71}$ まで削減可能である．また 5 段 KASUMI の鍵回復に必要な計算量も 2^{34} から $2^{31.5}$ まで削減可能である．

本論文の提案法 (高速化技法 1 と高速化技法 2) の性能評価結果を表 3.7 に纏める．また，田中ら

表 3.7: 提案法の性能評価結果

提案方式	未知項数	関連する暗号文単項式数	選択平文数 D	計算量 T
高速化技法 1	$L = 26,693$	C= 9,109	$2^{27.5}$	2^{34}
高速化技法 2	(推定する未知項数)$L'' = 1,916$ (消去する未知項数)$L' = 24,777$	C= 9,109 C$'$ = 7,205	$2^{28.6}$	$2^{31.5}$

の従来法との比較結果を表 3.8 に纏める．本論文で提案した提案法 (高速化技法 2) は，田中らの従来法と比べて選択平文数は $2^{10.8}$ 程度削減可能である．また，鍵回復に必要な計算量は $2^{85.5}$ 倍高速化が可能である．尚，本論文で示した高速化技法 2 を用いた 5 段 KASUMI の鍵回復攻撃は，2016 年 9 月時点で最も効率的な手法である．

表 3.8: 従来法との比較

提案方式	選択平文数 D	計算量 T[ENC.]
従来法 (田中ら)	$2^{39.4}$	2^{117}
高速化技法 1	$2^{27.5}$	2^{34}
高速化技法 2	$2^{28.6}$	$2^{31.5}$

3.7 纏め

本章では，高階差分攻撃の高速化手法の提案と，5 段 KASUMI への適用による性能評価を行った．まず初めに高階差分特性の探索において，高階差分は使用する階数が高い程選択平文数が増加する為，S-box の入出力サイズに合わせた変数探索法を提案した．また，鍵回復する部分 (攻撃方程式を解く部分) において，線形化手法の高速化技法（高速化技法 1 と高速化技法 2）を提案した．高速化技法 1 は係数行列 $\mathbf{A}$ のサイズが小さい場合に効果的な技法である．未知項数 L が多い場合，線形化方程式をガウス・ジョルダン法で解く際の計算量が支配的となる場合がある．この問題を解決する為，未知項数の係数を 0 に制御し，消去する高速化技法 2 を提案した．

本論文の提案法を 5 段 KASUMI へ適用し，性能評価を実施した．提案法を用いる事で 16 階差分を用いた新たな 4 段特性を発見した．その結果，一組の 4 段特性を構成する為に必要な選択平文数を 2^{32} から 2^{16} まで削減する事に成功した．また，5 段目の段鍵 82 bit を鍵回復する部分 (攻撃方程式を解く部分) において，高速化技法 1 を用いる事で，5 段 KASUMI の鍵回復攻撃を選択平文数 $D = 2^{27.5}$ と計算量 $T = 2^{34}$ 回の暗号化計算量 [ENC.] で行う事が可能である．また，高速化技法 2 を用いる場合，5 段 KASUMI の鍵回復攻撃を選択平文数 $D = 2^{28.6}$ と計算量 $2^{31.5}$ 回の暗号化計算量 [ENC.] で行う事が可能である事を示した．

本節の提案法を用いる事により，田中らの従来法と比較し，選択平文数は $2^{10.8}$ 程度削減可能である．また，鍵回復に必要な計算量は $2^{85.5}$ 倍高速化が可能である．尚，本論文で示した高速化技法 2 を用いた 5 段 KASUMI の鍵回復攻撃は，攻撃に必要な計算量が最小という点で，最良の結果 [5] である．

[5] 2016 年 9 月時点において

第4章　積分攻撃耐性評価

4.1　概要

積分攻撃 [25] は，複数の選択平文に対応する出力の総和が 0 になる性質を利用した攻撃で，共通鍵暗号に対して強力，且つ汎用的な攻撃手法の一つである．複数の選択平文を N 個の要素を持つ入力集合と捉えれば，積分攻撃は入力集合に対応する出力集合が持つの性質を利用した攻撃と見なす事が出来る．本章では，ショートカット法で用いる R 段目出力 X_R の特徴量 $\mathcal{H}(X_R(i))$ に，前述の出力集合の性質を用いる．尚，本章では，集合の性質を用いた特徴量を積分特性と呼び，ショートカット部の段数を R とした場合，R 段積分特性（又は，単に R 段特性）と呼ぶ．

本章では，KASUMI の積分攻撃に対する安全性評価を行う．共通鍵ブロック暗号 MISTY1 に対する先行研究として，藤堂は積分特性の探索を効率的に行う手法として Division 属性を新たに提案し，世界で初めてフルラウンドの MISTY1 が全数探索法よりも効率的に解読可能である事を示した [50]．本論文では，この手法を用いて KASUMI の積分特性探索を行い，発見した特性を用いて 6 段の KASUMI が選択平文数 $D = 2^{57}$ と計算量 $T = 2^{58}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示し，更に 7 段の KASUMI が選択平文数 $D = 2^{63}$ と計算量 $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示す．

以下に，第 4 章で用いる記号の一覧を示す．

4.2　積分攻撃と Division 属性

4.2.1　積分攻撃

積分攻撃（Integral Attack）は Knudsen と Wagner によって提案されたブロック暗号に対する汎用的かつ強力な攻撃手法の一つである [25]．積分攻撃は Daemen が提案した SQUARE 攻撃 [12] で排他的論理和（XOR）を総和と捉えたものである．以下に SQUARE 攻撃と積分攻撃について説明する．

図 4.1 に示す SPN 構造の暗号 E について考える．S 層を構成する各 S-box は s bit 入出力で，1 対 1 写像であるとする．この時，平文に最も近い S 層にある一つの S-box に着目し，その S-box への入力が全通りとなる様に 2^s 種類の平文 P を与えた場合，S-box が 1 対 1 写像であるから，段

表 4.1: 第 4 章で用いる記号一覧

記号	説明
$\{X\}$	平文 P の集合
$\{Y\}$	$\{X\}$ に対応する途中段の S-box 出力集合
ALL ($\mathcal{A}$)	集合 $\{Y\}$ の中で，全ての値が同じ回数出現する
BALANCE ($\mathcal{B}$)	集合 $\{Y\}$ の全ての要素の排他的論理和（XOR）が 0 になる
CONSTANT ($\mathcal{C}$)	集合 $\{Y\}$ の中で，要素がある値に固定されている
UNKNOWN ($\mathcal{U}$)	集合 $\{Y\}$ の全ての要素の排他的論理和（XOR）が予測不能
$\boldsymbol{a}[i]$	任意の m bit ベクトル $\boldsymbol{a} \in GF(2)^m$ の i 番目要素
$H_w(\boldsymbol{a})$	ベクトル $\boldsymbol{a}$ のハミング重み
π_u，$\pi_{\mathbf{u}}$	属性判定関数．集合の Division 属性を評価する為に利用
$\boldsymbol{k}$	m 次元ベクトル．i 番目成分 k_i は 0 から n_i の整数値をとる
$\succeq$	ベクトルの大小を示す．異なるベクトル $\boldsymbol{k}, \boldsymbol{k'}$ に対し， 全ての成分で $k_i \geq k'_i$ を満たす時，$\boldsymbol{k} \succeq \boldsymbol{k'}$ と記す
$\mathcal{D}_k^m$，$\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$	集合の Division 属性
$\xrightarrow{Z}$	Division 属性の伝搬．Z は段関数やショートカット段数等を示す

鍵の値に関わらず，その S-box 出力は全通り出現する．従って，この S-box 出力の値を全て排他的論理和した結果は 0 となる．この様に，一つ以上の S-box の入力ビットサイズの全通りの平文 P を入力し，対象とする暗号 E の途中段の S-box 出力が全通りであるならば，その出力値を全て排他的論理和した結果が 0 になる事を利用した攻撃が可能である．この攻撃法を SQUARE 攻撃 [12] と呼ぶ．SQUARE 攻撃は，主に SPN 構造のブロック暗号に対して広く用いられている．

入力する平文 P の集合を $\{X\}$ とし，$\{X\}$ に対応する途中段の S-box 出力集合を $\{Y\}$ とした時，集合 $\{Y\}$ の性質として以下が知られている．尚，文献 [30] では，これらの集合の性質を飽和特性と呼んでいる．

- ALL ($\mathcal{A}$)：集合 $\{Y\}$ の中で，全ての値が同じ回数出現する．
- BALANCE ($\mathcal{B}$)：集合 $\{Y\}$ の全ての要素の排他的論理和（XOR）が 0 になる．
- CONSTANT ($\mathcal{C}$)：集合 $\{Y\}$ の中で，要素がある値に固定されている．
- UNKNOWN ($\mathcal{U}$)：集合 $\{Y\}$ の全ての要素の排他的論理和（XOR）が鍵の値に依存し，不定となる．

積分攻撃は，SQUARE 攻撃で用いる排他的論理和を総和として考えたものである．GF(2) 上において，積分攻撃の総和 (足し算) は排他的論理和と同じ結果となる．本論文では，文献 [50] に合わせて，上記の集合の性質を積分特性と呼ぶことにする．攻撃者は N 個の選択平文を用意し，

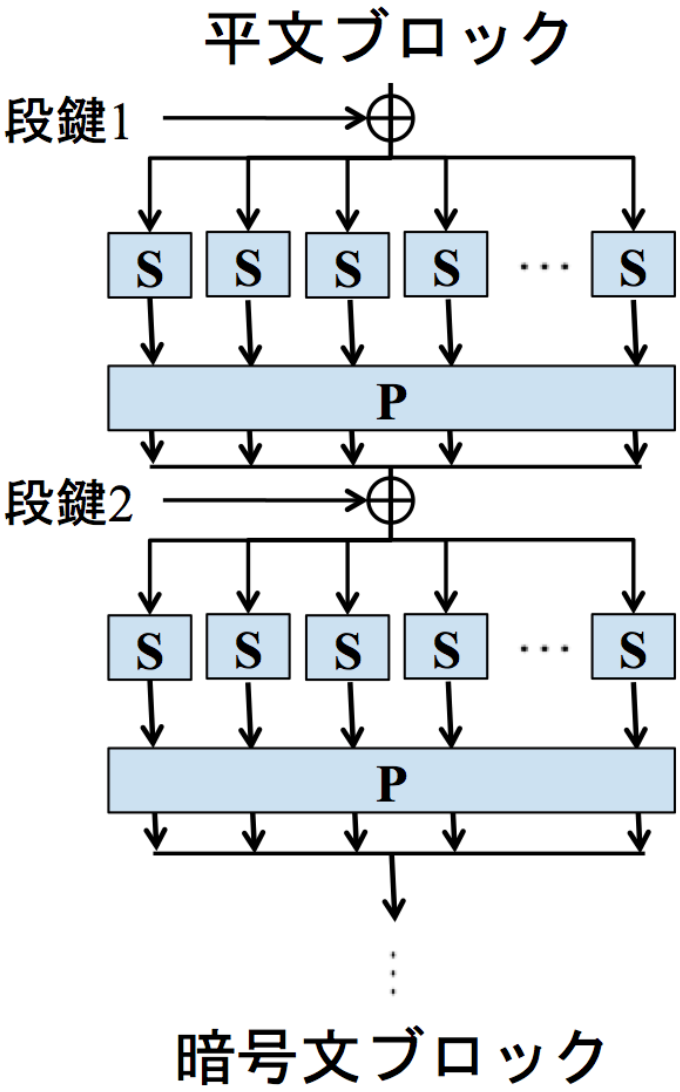

図 4.1: SPN 構造の暗号 E

対応する暗号文を手に入れたとする．もし，N 個全ての暗号文の総和が 0 であるならば，攻撃対象の暗号 E は N 個の選択平文による積分特性を有するという．

積分攻撃では，攻撃者が段数を少なくした対象とする暗号 (例えば，ブロック暗号等) に対して積分特性を有する N 個の選択平文を最初に用意し，対応する暗号文を入手する．その後，攻撃者は解読対象である複数段の段鍵を推定し，暗号文から復号した N 個の中間値の総和を計算する．最後に N 個の中間値の総和が 0 になるかどうか判定し，もし 0 でなければ推定した段鍵は偽の鍵であるとして真の鍵候補から除外する．この様な判定手法を繰り返す事で，攻撃者は真の鍵を推定する事が可能となる．

4.2.2 Division 属性

藤堂は属性判定関数 (bit product function) を用いて積分特性を再定義した [49]，[50]．本論文では，藤堂が新たに定義した Division property を Division 属性と述べる事とする．以下にその概要を記載するが，詳細は文献 [49]，[50] を参照のこと．

本論文では，任意の m bit ベクトル $\boldsymbol{a} \in GF(2)^m$ に対し，$\boldsymbol{a}$ の i 番目の要素を $\boldsymbol{a}[i]$ と示す．また，$\boldsymbol{a}$ のハミング重み H_w を $H_w(\boldsymbol{a}) = \sum_{i=1}^{m} a[i]$ と計算するものとする．

定義 1 (属性判定関数 (Bit product function)[49]) 属性判定関数 π_u と $\pi_{\mathbf{u}}$ は集合の *Division* 属性を評価する為に使用される．$\pi_u : GF(2)^m \to GF(2)$ を任意の $u \in GF(2)^m$ を用いて $1bit$ を出力する関数とする．関数の入力を $\boldsymbol{x} \in GF(2)^m$ とした時，$\pi_u(\boldsymbol{x})$ は $u[i] = 1$ を満たす時の $x[i]$ の積 *(AND)* を計算するものであり，以下の式で定義される．

$$\pi_u(\boldsymbol{x}) := \prod_{i=1}^{m} x[i]^{u[i]} \tag{4.1}$$

$\pi_{\mathbf{u}}$:$GF(2)^{n_1} \times GF(2)^{n_2} \times \cdots \times GF(2)^{n_m} \to GF(2)$ を任意の $\mathbf{u} \in GF(2)^{n_1} \times GF(2)^{n_2} \times \cdots \times GF(2)^{n_m}$ を用いて $1bit$ を出力する関数とする．関数の入力を $\mathbf{x} \in GF(2)^{n_1} \times GF(2)^{n_2} \times \cdots \times GF(2)^{n_m}$ とした時，$\pi_{\mathbf{u}}(\mathbf{x})$ は以下の式で定義される．

$$\pi_{\mathbf{u}}(\mathbf{x}) := \prod_{i=1}^{m} \pi_{u_i}(\boldsymbol{x}_i)$$

定義 2 (Division 属性 $\mathcal{D}_k^m$，$\mathcal{D}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}^{n_1,n_2,\ldots,n_m}$[49]) $\{X\}$ を集合とし，$\{X\}$ の要素が $\boldsymbol{x} \in GF(2)^m$ の値をとるものとする．また，k は 0 から m 迄の値 $(0 \le k \le m)$ を取るものとする．集合 $\{X\}$ が *Division* 属性 $\mathcal{D}_k^m$ を有する時，以下を満たす．

$$\bigoplus_{\boldsymbol{x} \in \{X\}} \pi_u(\boldsymbol{x}) = \begin{cases} 0 \ (H_w(u) \not\succeq k) \\ ? \ (\text{上記以外}) \end{cases} \tag{4.2}$$

尚，記号 '?' は不定を示す．

$\{X\}$ を集合とし，$\{X\}$ の要素が $\mathbf{x} \in GF(2)^{n_1} \times GF(2)^{n_2} \times \cdots \times GF(2)^{n_m} \to GF(2)$ の値をとるものとする．$\boldsymbol{k}$ を m 次元ベクトルとし，k_i をベクトル $\boldsymbol{k}$ の i 番目の成分とする．k_i は 0 から n_i の整数値を取るものとする．集合 $\{X\}$ が *Division* 属性 $\mathcal{D}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}^{n_1,n_2,\ldots,n_m}$ を有する時，以下を満たす．

$$\bigoplus_{\mathbf{x} \in \{X\}} \pi_{\mathbf{u}}(\mathbf{x}) = \begin{cases} 0 \ (Hw(\mathbf{u}) \not\succeq \boldsymbol{k}^{(i)}, i = 1, 2, \ldots, q) \\ ? \ (\text{上記以外}) \end{cases} \tag{4.3}$$

$Hw(\mathbf{u}) = (Hw(n_1), Hw(n_2), \cdots, Hw(n_m))$ はベクトル $\mathbf{u}$ のハミング重みを表す．また，異なるベクトル $\boldsymbol{k}, \boldsymbol{k'}$ に対し，全ての成分で $k_i \ge k_i'$ を満たす時，$\boldsymbol{k} \succeq \boldsymbol{k'}$ と記す．

q 個の m 次元ベクトルを $\boldsymbol{k}^{(i)} = (k_1^{(i)}, \ldots, k_m^{(i)}), (i = 1, 2, \ldots, q)$ と示す．集合 $\{X\}$ が Division 属性 $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$ を有する時，積分特性 ALL ($\mathcal{A}$)，BALANCE ($\mathcal{B}$)，UNKNOWN ($\mathcal{U}$) は以下の様に示される [50]．

- $\mathcal{A}$： $k_j^{(i)} = n_j, (0 \le j \le m)$
- $\mathcal{B}$： $2 \le k_j^{(i)} \le n_j - 1, (0 \le j \le m)$
- $\mathcal{U}$： $k_j^{(i)} = 1, (0 \le j \le m)$

属性判定関数 $\pi_u(\boldsymbol{x})$ は GF(2) 上の $\boldsymbol{x}$ に関する d 次 $(0 \le d \le m)$ 単項式を表している．集合 $\{X\}$ が Division 属性 $\mathcal{D}^m_{d+1}$ を満たす時，任意の d 次関数 f に対し，以下が常に成立する．

$$\bigoplus_{\boldsymbol{x} \in \{X\}} f(\boldsymbol{x}) = 0$$

この時，集合 $\{X\}$ は d 次関数 f に対し，積分特性を有するという．

Division 属性を用いて対象の暗号化関数 f の積分特性を探索する場合，平文集合 $\{X\}$ が有する Division 属性がどの様に伝搬していくのか解析する必要がある．以下に，Division 属性の伝搬ルールの概要を示す．尚，伝搬ルールの詳細は文献 [50] を参照のこと．

伝搬ルール 1 (Substitution[50]) 関数 F を並列な m 個の *S-box* からなる関数とする．i 番目の n_i ビット入力 *S-box* の代数次数を d_i とし，各 *S-box* は全単射関数とする．関数 F の入力，出力は $GF(2)^{n_1} \times GF(2)^{n_2} \times \cdots \times GF(2)^{n_m}$ の値を取るものとし，入力集合を $\{X\}$，出力集合を $\{Y\}$ と表すものとする．このとき，入力集合 $\{X\}$ が *Division* 属性 $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$ を有する時，出力集合 $\{Y\}$ の *Division* 属性 $\mathcal{D}^{n_1,n_2,\ldots,n_m}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},\ldots,\boldsymbol{k'}^{(q)}}$ は以下の様に計算される．

$$k_i'^{(j)} = \begin{cases} \left\lceil \dfrac{k_i^{(j)}}{d_i} \right\rceil \ , (1 \le i \le m,\ 1 \le j \le q) \\ k_i^{(j)} \ , (k_i^{(j)} = n_i,\ 1 \le i \le m,\ 1 \le j \le q) \end{cases}$$

伝搬ルール 2 (Copy[50]) 関数 F をコピー関数とする．関数 F の入力を $x \in GF(2)^n$ とし，出力は $(y_1, y_2) = (x, x)$ とする．入力集合 $\{X\}$ が *Division* 属性 $\mathcal{D}^n_k$ を有する時，出力集合 $\{Y\}$ の *Division* 属性 $\mathcal{D}^{n,n}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},\ldots,\boldsymbol{k'}^{(k+1)}}$ は以下の様に計算される．

$$\boldsymbol{k'}^{(i+1)} = (k - i, i) \ , (0 \le i \le k)$$

伝搬ルール 3 (Compression by XOR[50]) 関数 F を排他的論理和を用いた圧縮関数とする．関数 F の入力を $(x_1, x_2) \in GF(2)^n \times GF(2)^n$ とし，出力は $y = x_1 \oplus x_2$ とする．入力集合 $\{X\}$ が

Division 属性 $\mathcal{D}^{n,n}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$ を有する時，出力集合 $\{Y\}$ の *Division* 属性 $\mathcal{D}^{n}_{k'}$ は以下の様に計算される．

$$k' = min\{k_1^{(1)} + k_2^{(1)}, k_1^{(2)} + k_2^{(2)}, \ldots, k_1^{(q)} + k_2^{(q)}\}$$

もし，kの最小値が　n よりも大きいならば，その *Division* 属性の伝搬は中断する．

伝搬ルール 4 (Split[50]) 関数 F を分割関数とする． 関数 F の入力を $x \in GF(2)^n$ とし，出力は $x = y_1||y_2$ とする．$(y_1, y_2) \in F(2)^{n_1} \times GF(2)^{n-n_1}$ とする．入力集合 $\{X\}$ が *Division* 属性 $\mathcal{D}^{n}_{k}$ を有する時，出力集合 $\{Y\}$ の *Division* 属性 $\mathcal{D}^{n_1,n-n_1}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},\ldots,\boldsymbol{k'}^{(k+1)}}$ は以下の様に計算される．

$$\boldsymbol{k'}^{(i+1)} = (k-i, i)\ , (0 \leq i \leq k)$$

$(k-i)$ は n_1 以下とし，i は $n-n_1$ 以下とする．

伝搬ルール 5 (Concatenation[50]) 関数 F を結合関数とする．関数 F の入力を $(x_1, x_2) \in GF(2)^{n_1} \times GF(2)^{n_2}$ とし，出力は $y = x_1||x_2$ とする．入力集合 $\{X\}$ が *Division* 属性 $\mathcal{D}^{n_1,n_2}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$ を有する時，出力集合 $\{Y\}$ の *Division* 属性 $\mathcal{D}^{n_1+n_2}_{k'}$ は以下の様に計算される．

$$k' = min\{k_1^{(1)} + k_2^{(1)}, k_1^{(2)} + k_2^{(2)}, \ldots, k_1^{(q)} + k_2^{(q)}\}$$

4.3　従来法と問題提起

4.3.1　従来法

共通鍵暗号に対する汎用的，且つ強力な攻撃手法に積分攻撃 [25] がある．積分攻撃を行う場合，攻撃者は対象暗号に対する積分特性を事前に調べる必要がある．藤堂は積分特性を発見する高速・効率的な手法として Division 属性 (Division property) を提案した [49]．更に，MISTY1 に対して Division 属性を用いた積分特性探索を行い，6 段の積分特性が選択平文数 2^{63} で存在する事を示し，その 6 段特性を用いてフルラウンド (8 段) の MISTY1 が全数探索法よりも効率的に解読可能である事を示した [50].

4.3.2　問題提起

KASUMI の非線形関数の数 (FI 関数内に存在する S7，S9 の数) は，MISTY1 の $\frac{4}{3}$ 倍である [1]．積分特性は非線形関数の個数に反比例すると仮定した場合，KASUMI は 4.5 段程度の積分特性が

[1]KASUMI の FI 関数内に存在する非線形関数 (S7，S9) の数は 4 である (図 2.4) を参照．一方，MISTY1 の FI 関数内に存在する非線形関数 (S7，S9) の数は 3 である (図 2.6 を参照).

存在すると思われる．しかしながら，KASUMI の非線形関数 (S7，S9) の内部構造，及び鍵依存線形関数である FL 関数の内部構造は MISTY1 と異なる為，単純に MISTY1 の結果を適用する事は不可能である．従って，MISTY1 をベースに開発された KASUMI が，どの程度 Division 属性を用いた積分攻撃に対する耐性を有しているのか不明である．

4.4 KASUMI の Division 属性を用いた積分攻撃評価

Division 属性を用いて対象の KASUMI の積分特性を探索する場合，平文集合 $\{X\}$ が有する Division 属性がどの様に伝搬していくのか解析する必要がある．本論文では，藤堂が示した Division 属性の伝搬ルールを用いて解析を行うものとする．尚，伝搬ルールの詳細は文献 [50] を参照のこと．

4.4.1 S-box の Division 属性の伝搬

Division 属性が KASUMI の S-box(S7，S9) を通過する際，どの様に伝搬するのか解析した結果を表 4.2 と表 4.3 に示す．ここで，S7，S9 の入力集合 $\{X\}$ が有する Division 属性を $\mathcal{D}_k^m$ と示し，出力集合 $\{Y\}$ の Division 属性を $\mathcal{D}_{k'}^m$ $(m = 7, 9)$ と示す．

表 4.2: S7 の Division 属性の伝搬

$\mathcal{D}_k^7$	$\mathcal{D}_1^7$	$\mathcal{D}_2^7$	$\mathcal{D}_3^7$	$\mathcal{D}_4^7$	$\mathcal{D}_5^7$	$\mathcal{D}_6^7$	$\mathcal{D}_7^7$
$\mathcal{D}_{k'}^7$	$\mathcal{D}_1^7$	$\mathcal{D}_1^7$	$\mathcal{D}_1^7$	$\mathcal{D}_2^7$	$\mathcal{D}_2^7$	$\mathcal{D}_4^7$	$\mathcal{D}_7^7$

表 4.3: S9 の Division 属性の伝搬

$\mathcal{D}_k^9$	$\mathcal{D}_1^9$	$\mathcal{D}_2^9$	$\mathcal{D}_3^9$	$\mathcal{D}_4^9$	$\mathcal{D}_5^9$	$\mathcal{D}_6^9$	$\mathcal{D}_7^9$	$\mathcal{D}_8^9$	$\mathcal{D}_9^9$
$\mathcal{D}_{k'}^9$	$\mathcal{D}_1^9$	$\mathcal{D}_1^9$	$\mathcal{D}_2^9$	$\mathcal{D}_2^9$	$\mathcal{D}_3^9$	$\mathcal{D}_3^9$	$\mathcal{D}_4^9$	$\mathcal{D}_4^9$	$\mathcal{D}_9^9$

KASUMI の S-box は 1 対 1 写像である為，S-box への入力が全通りとなる様に集合 $\{X\}$ を与えた場合，その S-box 出力の集合 $\{Y\}$ は全通り出現する．従って，集合 $\{X\}$ が $k = m$ を満たす Division 属性 $\mathcal{D}_k^m$ を有する場合，Division 属性 $\mathcal{D}_k^m$ は劣化せずに，そのまま出力集合 $\{Y\}$ に伝搬する．

S-box の伝搬ルール (Substitution) を用いて評価する場合，S7 の代数次数は 3 次である為，入力集合 $\{X\}$ の Division 属性が $\mathcal{D}_6^7$ の時，出力集合 $\{Y\}$ の Division 属性は $\mathcal{D}_2^7$ となる．しかしながら，実際に S7 を用いて解析した結果は表 4.2 に記載の通り，出力集合 $\{Y\}$ の Division 属性は $\mathcal{D}_4^7$ となった．任意の 3 次関数の Division 属性の伝搬に比べ，S7 の Division 属性の伝搬の劣化は少な

い事がわかる．MISTY の S7 にも同じ現象が報告されている [50]．尚，S9 の Division 属性の伝搬は，任意の 2 次関数と同じ様に劣化する．

4.4.2 FI 関数の Division 属性の伝搬

本論文では，文献 [50] と同様な手法を用いて，Division 属性が FI 関数をどの様に伝搬するのか解析を行った．具体的には，FI 関数の入出力 16 bit を 7 bit，2 bit，7bit に分割し，Division 属性の伝搬を解析している．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}_{\boldsymbol{k}}^{7,2,7}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3), (0 \leq k_1, k_3 \leq 7, 0 \leq k_2 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 3 次元ベクトルである．FI 関数の Division 属性の伝搬の一例を以下に示す．

$$\mathcal{D}_{[3,2,5]}^{7,2,7} \xrightarrow{FI} \mathcal{D}_{[(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)]}^{7,2,7}$$

尚，付録の表 C.1〜表 C.8 に FI 関数の Division 属性の伝搬結果を記す．

4.4.3 FO 関数の Division 属性の伝搬

FO 関数の Division 属性の伝搬は，FI 関数の Division 属性の伝搬と伝搬ルールを組み合わせる事で解析する事ができる．FO 関数の Division 属性は，FO 関数の入出力 32 bit を (7 bit，2 bit，7bit，7 bit，2 bit，7bit) に分割し，Division 属性の伝搬を解析している．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7 \times GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3, k_4, k_5, k_6), (0 \leq k_1, k_3, k_4, k_6 \leq 7, 0 \leq k_2, k_5 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 6 次元ベクトルである．FO 関数の Division 属性の伝搬の一例を以下に示す．ベクトル $\boldsymbol{k'}^{(q)}, (1 \leq q \leq 56)$ を表 4.4 に示す．

$$\mathcal{D}_{[4,2,7,3,1,6]}^{7,2,7,7,2,7} \xrightarrow{FO} \mathcal{D}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},\ldots,\boldsymbol{k'}^{(56)}}^{7,2,7,7,2,7}$$

4.4.4 FL 関数の Division 属性の伝搬

KASUMI の FL 関数は 1 bit 左巡回シフトが存在し，MISTY1 の FL 関数とは構造が異なっている．文献 [50] で説明されている FL 関数の Division 属性を KASUMI に適用する事は出来ない為，本論文では，文献 [50] に記載の FL 関数の Division 属性を元に 1 bit 左巡回シフトの影響を考慮し，解析を行った．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7 \times$

表 4.4: FO 関数の Division 属性の伝搬の一例

$\boldsymbol{k'}^{(1)}$, $\boldsymbol{k'}^{(2)}$,...,$\boldsymbol{k'}^{(56)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},...,\boldsymbol{k'}^{(56)}}$
(0,0,0,0,0,4),(0,0,0,0,1,3),(0,0,0,0,2,2),(0,0,0,1,0,2),
(0,0,0,1,1,1),(0,0,0,1,2,0),(0,0,0,2,0,0),(0,0,1,0,0,3),
(0,0,1,0,1,2),(0,0,1,0,2,1),(0,0,1,1,0,1),(0,0,1,1,1,0),
(0,0,2,0,0,2),(0,0,2,0,1,1),(0,0,2,1,0,0),(0,0,3,0,2,0),
(0,0,4,0,0,1),(0,0,4,0,1,0),(0,1,0,0,0,3),(0,1,0,0,1,2),
(0,1,0,1,0,1),(0,1,0,1,1,0),(0,1,1,0,0,2),(0,1,1,0,1,1),
(0,1,1,1,0,0),(0,1,2,0,2,0),(0,1,3,0,0,1),(0,1,3,0,1,0),
(0,2,0,0,2,1),(0,2,0,1,0,0),(0,2,1,0,2,0),(0,2,2,0,0,1),
(0,2,2,0,1,0),(0,2,7,0,0,0),(1,0,0,0,0,2),(1,0,0,0,1,1),
(1,0,0,0,2,0),(1,0,0,1,0,0),(1,0,1,0,0,1),(1,0,1,0,1,0),
(1,0,6,0,0,0),(1,1,0,0,0,1),(1,1,0,0,1,0),(1,1,5,0,0,0),
(1,2,4,0,0,0),(2,0,0,0,0,1),(2,0,0,0,1,0),(2,0,3,0,0,0),
(2,1,2,0,0,0),(2,2,1,0,0,0),(3,0,2,0,0,0),(3,1,1,0,0,0),
(3,2,0,0,0,0),(4,0,1,0,0,0),(4,1,0,0,0,0),(5,0,0,0,0,0)

$GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3, k_4, k_5, k_6)$, $(0 \leq k_1, k_3, k_4, k_6 \leq 7, 0 \leq k_2, k_5 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 6 次元ベクトルである．FL 関数の Division 属性の伝搬の一例を以下に示す．ベクトル $\boldsymbol{k'}^{(q)}$, $(1 \leq q \leq 101)$ を表 4.5 に示す．

$$\mathcal{D}^{7,2,7,7,2,7}_{[0,1,1,0,1,2]} \xrightarrow{FL} \mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},...,\boldsymbol{k'}^{(101)}}$$

4.4.5 新たな積分特性

前節までに示した Division 属性の伝搬を用いて，KASUMI の積分特性探索を行った．その結果，新たな積分特性が存在する事を発見した．以下に今回発見した 4 段の積分特性示す．尚，この 4 段特性は KASUMI の 1 段目から 4 段目を対象とした特性であり，出力は FO4[31-0] の 32 bit を示している．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} \mathcal{D}^{7,2,7,7,2,7}_{[(0,0,0,0,0,1),(0,0,0,0,1,0),(0,0,0,1,0,0),(0,0,2,0,0,0),(0,1,1,0,0,0),(0,2,0,0,0,0),(1,0,0,0,0,0)]} \quad (4.4)$$

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,1,4,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} \mathcal{D}^{7,2,7,7,2,7}_{[(0,0,0,0,0,1),(0,0,0,0,1,0),(0,0,0,1,0,0),(0,0,2,0,0,0),(0,1,1,0,0,0),(0,2,0,0,0,0),(1,0,0,0,0,0)]} \quad (4.5)$$

表 4.5: FL 関数の Division 属性の伝搬の一例

$\boldsymbol{k'}^{(1)}$, $\boldsymbol{k'}^{(2)}$,...,$\boldsymbol{k'}^{(101)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k'}^{(1)},\boldsymbol{k'}^{(2)},...,\boldsymbol{k'}^{(101)}}$
(0,0,0,0,2,3),(0,0,0,1,1,3),(0,0,0,1,2,2),(0,0,1,0,1,3),(0,0,1,0,2,2),(0,0,1,1,1,2),
(0,0,1,1,2,1),(0,0,2,0,1,2),(0,0,2,0,2,1),(0,0,2,1,1,1),(0,0,2,1,2,0),(0,0,3,0,1,1),
(0,0,3,0,2,0),(0,0,3,1,1,0),(0,0,4,0,1,0),(0,1,0,0,1,3),(0,1,0,0,2,2),(0,1,0,1,0,3),
(0,1,0,1,1,2),(0,1,0,1,2,1),(0,1,1,0,0,3),(0,1,1,0,1,2),(0,1,1,0,2,1),(0,1,1,1,0,2),
(0,1,1,1,1,1),(0,1,1,1,2,0),(0,1,2,0,0,2),(0,1,2,0,1,1),(0,1,2,0,2,0),(0,1,2,1,0,1),
(0,1,2,1,1,0),(0,1,3,0,0,1),(0,1,3,0,1,0),(0,1,3,1,0,0),(0,1,4,0,0,0),(0,2,0,0,1,2),
(0,2,0,0,2,1),(0,2,0,1,0,2),(0,2,0,1,1,1),(0,2,1,0,0,2),(0,2,1,0,1,1),(0,2,1,0,2,0),
(0,2,1,1,0,1),(0,2,1,1,1,0),(0,2,2,0,0,1),(0,2,2,0,1,0),(0,2,2,1,0,0),(0,2,3,0,0,0),
(1,0,0,0,1,3),(1,0,0,0,2,2),(1,0,0,1,0,3),(1,0,0,1,1,2),(1,0,1,0,0,3),(1,0,1,0,1,2),
(1,0,1,0,2,1),(1,0,1,1,0,2),(1,0,1,1,1,1),(1,0,2,0,0,2),(1,0,2,0,1,1),(1,0,2,0,2,0),
(1,0,2,1,0,1),(1,0,2,1,1,0),(1,0,3,0,0,1),(1,0,3,0,1,0),(1,0,3,1,0,0),(1,0,4,0,0,0),
(1,1,0,0,0,3),(1,1,0,0,1,2),(1,1,0,0,2,1),(1,1,0,1,0,2),(1,1,0,1,1,1),(1,1,1,0,0,2),
(1,1,1,0,1,1),(1,1,1,0,2,0),(1,1,1,1,0,1),(1,1,1,1,1,0),(1,1,2,0,0,1),(1,1,2,0,1,0),
(1,1,2,1,0,0),(1,1,3,0,0,0),(1,2,0,0,0,2),(1,2,0,0,1,1),(1,2,0,1,0,1),(1,2,1,0,0,1),
(1,2,1,0,1,0),(1,2,1,1,0,0),(1,2,2,0,0,0),(2,0,0,0,0,3),(2,0,0,0,1,2),(2,0,1,0,0,2),
(2,0,1,0,1,1),(2,0,2,0,0,1),(2,0,2,0,1,0),(2,0,3,0,0,0),(2,1,0,0,0,2),(2,1,0,0,1,1),
(2,1,1,0,0,1),(2,1,1,0,1,0),(2,1,2,0,0,0),(2,2,0,0,0,1),(2,2,1,0,0,0)

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,2,3,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} \mathcal{D}^{7,2,7,7,2,7}_{[(0,0,0,0,0,1),(0,0,0,0,1,0),(0,0,0,1,0,0),(0,0,2,0,0,0),(0,1,1,0,0,0),(0,2,0,0,0,0),(1,0,0,0,0,0)]} \tag{4.6}$$

上記は Division 属性 $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]}$(又は，$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,1,4,7,2,7,7,2,7,7,2,7]}$，$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,2,3,7,2,7,7,2,7,7,2,7]}$) を持つ入力集合 $\{X\}$ を与えた場合，FO4[31-0] の出力集合 $\{Y\}$ の Division 属性が $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\cdots,\boldsymbol{k}^{(7)}}$ となる事を意味する．また，Division 属性において，要素 $k^{(i)}_j$ が 2 以上となるベクトル $\boldsymbol{k}^{(i)}$ が存在する場合，該当箇所の総和が BALANCE する事を意味する．表記 (4.4) から (4.6) において，要素が 2 以上となるベクトル $\boldsymbol{k}^{(4)} = (0,0,2,0,0,0)$ と $\boldsymbol{k}^{(6)} = (0,2,0,0,0,0)$ が存在する為，該当箇所 FO4[24-16] の 9 bit が BALANCE している事が分かる．4 段の積分特性 (4.4)，(4.5)，(4.6) の出力集合の特性を積分特性で表記すると以下の様になる．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U}) \tag{4.7}$$

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,1,4,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U}) \tag{4.8}$$

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,2,3,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U}) \tag{4.9}$$

入力集合 $\{X\}$ の Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]}$ となる為には，上位 11 bit を任意の定数とし，残り 53 bit を変数として全通り出現する入力集合 $\{X\}$ を用意すれば良い．従って，一組の 4 段特性を用意する為に必要な選択平文数は 2^{53} である．その他 ($\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,1,4,7,2,7,7,2,7,7,2,7]}$，$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,2,3,7,2,7,7,2,7,7,2,7]}$) の場合も定数と変数の位置がそれぞれ異なるが，11 bit を任意の定数，53 bit を変数として全通り出現する入力集合 $\{X\}$ を用意すれば良い為，一組の 4 段特性を用意する為に必要な選択平文数は 2^{53} となる．

文献 [27] や [43] では，攻撃者が KASUMI の 2 段目から解析する事を許容している．本論文でも同様に 2 段目から解析した場合，以下に示す 5 段の積分特性が存在する事を発見した．この 5 段特性は KASUMI の 2 段目から 6 段目を対象とした特性であり，出力は FO6[31-0] の 32 bit を示している．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]} \xrightarrow{5R} \mathcal{D}^{7,2,7,7,2,7}_{[(0,0,0,0,0,1),(0,0,0,0,1,0),(0,0,0,1,0,0),(0,0,2,0,0,0),(0,1,1,0,0,0),(0,2,0,0,0,0),(1,0,0,0,0,0)]} \tag{4.10}$$

$$\boldsymbol{k'}^{(1)} = (0,0,0,0,0,1)$$
$$\boldsymbol{k'}^{(2)} = (0,0,0,0,1,0)$$
$$\boldsymbol{k'}^{(3)} = (0,0,0,1,0,0)$$
$$\boldsymbol{k'}^{(4)} = (0,0,2,0,0,0)$$
$$\boldsymbol{k'}^{(5)} = (0,1,1,0,0,0)$$
$$\boldsymbol{k'}^{(6)} = (0,2,0,0,0,0)$$
$$\boldsymbol{k'}^{(7)} = (1,0,0,0,0,0)$$

ベクトル $\boldsymbol{k'}^{(i)}, (i = 1,2,3,7)$ は成分 $k^{(i)}_j, (j = 1,4,5,6)$ が 1 で他の成分が 0 のベクトルである為，UNKNOWN（$\mathcal{U}$）特性を示す．また，ベクトル $\boldsymbol{k'}^{(i)}, (i = 4,6)$ は成分 $k^{(i)}_j, (j = 2,3)$ が 2 で他の成分が 0 のベクトルである為，BALANCE（$\mathcal{B}$）特性を示す．尚，ベクトル $\boldsymbol{k'}^{(5)}$ は成分 $k^{(3)}_j, (j = 2,3)$ が共に 1 で他の成分が 0 のベクトルである為，FO6[24-16] の 9 bit で見れば BALANCE（$\mathcal{B}$）特性を示す．以上から，入力集合に Division 属性 $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$ を与えた時，出力集合の特性を

積分特性（$\mathcal{A},\mathcal{B},\mathcal{U}$）は以下の様になる．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]} \xrightarrow{5R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U}) \tag{4.11}$$

上記はFO6[24-16]の9 bitがBALANCEする事を示す．入力集合にDivision属性$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$を与える為には，平文 P 64 bit中 2 bit（P[56,55]中 1 bitとP[54-48]中 1 bit）を任意の固定値とし，その他 62 bitを変数とする2^{62}の選択平文が必要となる．従って，この5段特性を一組用意する為に必要な選択平文数は2^{62}となる．表4.6に今回発見した積分特性の一覧を示す．

表 4.6: Division属性によるKASUMIの新たな積分特性

段数	入力集合のDivision属性 $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{\boldsymbol{k}}$	出力の積分特性	必要平文数
4*	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]}$	$(\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$	2^{53}
4*	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,1,4,7,2,7,7,2,7,7,2,7]}$	$(\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$	2^{53}
4*	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,2,3,7,2,7,7,2,7,7,2,7]}$	$(\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$	2^{53}
5**	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$	$(\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$	2^{62}

∗4段特性の対象は1段目〜4段目であり，出力はFO4[31-0]の32 bitを示す．
∗∗5段特性は対象は2段目〜6段目であり，出力はFO6[31-0]の32 bitを示す．
記号'$\mathcal{U}$'はUNKNOWNを示し，'$\mathcal{B}$'はBALANCEを示す．

4.4.6 鍵回復攻撃

本節では，新たに発見した4段と5段の積分特性を用いて，6段と7段のKASUMIに対する鍵回復攻撃について説明する．

6 段 KASUMI の攻撃

表 4.6 に示す 4 段の積分特性を用いた 6 段 KASUMI の鍵回復攻撃について説明する．下記に，4 段の積分特性の一つを再掲する．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,7,2,7,7,2,7,7,2,7]} \xrightarrow{4R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$$

上記の 4 段の積分特性は，FO4[24-16] の 9 bit が BALANCE する事を意味する．4 段の積分特性 (4.7) と図 4.2 から，以下に示す段鍵に関する方程式を導出する事ができる．

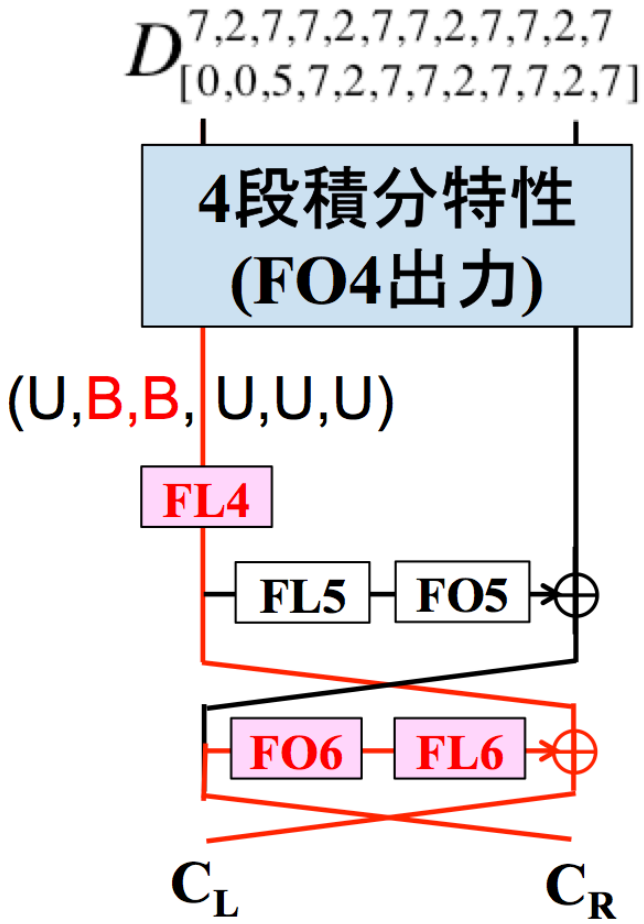

図 4.2: 6 段 KASUMI の鍵回復攻撃

$$\bigoplus_{C_R,C_L\in\{C\}} FL4^{-1}(FL6(FO6(C_R;KO_6,KI_6);KL_6)\oplus C_L;KL_4)[24-16]=0 \qquad (4.12)$$

式 (4.12) において，$\{C\}$ は平文集合に対応する暗号文集合を示す．また，C_L は暗号文 C の上位 (左側) 32 bit，C_R は暗号文 C の下位 (右側) 32 bit をそれぞれ示す．また，$FL4^{-1}$ は FL 関数の逆関数を示す．式 (4.12) には FO6 内に 96 bit，FL6 内の 18 bit，FL4 内の 9 bit，合計 123 bit の段鍵が存在する．ここで，$KL_{62}[15,7-0]$ と $KL_{42}[15,7-0]$ の合計 18 bit が全て 1 だと仮定する

と，式 (4.12) は以下の様にみなす事ができる．

$$\bigoplus_{C_R \in \{C\}} FO6(C_R; KO_6, KI_6)[24-16] \oplus \bigoplus_{C_L \in \{C\}} C_L[24-16] = 0 \tag{4.13}$$

式 (4.13) には 50 bit ($KO_6 = \{KO_{61}, KO_{62}\},\ KI_6 = \{KI_{61}[8-0], KI_{62}[8-0]\}$) の段鍵が存在する．従って，18 bit の段鍵 ($KL_{62}[15, 7-0]$ と $KL_{42}[15, 7-0]$) を全て 1 と仮定する事により，鍵回復攻撃で推定する段鍵数を削減する事ができる．

本論文では，50 bit の段鍵 ($KO_6 = \{KO_{61}, KO_{62}\},\ KI_6 = \{KI_{61}[8-0], KI_{62}[8-0]\}$) を線形化手法 [42] を用いて回復する．線形化手法は，式 (4.13) を GF(2) 上の多項式に展開し，段鍵に関する高次項を新たな独立未知項と見なして線形化し，得られた線形方程式を解いて段鍵を求める手法である．数式解析ソフトウェア REDUCE version 3.6 を用いて，式 (4.13) を線形化した際の未知項数 L を解析した結果，$L = 494$ である．

式 (4.13) は 9 bit から成る方程式である為，一組の式 (4.13) から 9 本の線形方程式を用意する事ができる．線形方程式中に存在する未知項数は $L = 494$ の為，線形方程式を解いて 50 bit の段鍵を回復する為には，$\lceil \frac{494}{9} \rceil = 55$ 組の異なる式 (4.13) が必要である．4 段の積分特性 (4.7)，(4.8)，(4.9) を一組用意する為には，2^{53} の選択平文数が必要になる．ここで，上位 9 bit を任意の固定値とし，残り 55 bit を変数として全通り出現する選択平文 2^{55} を用意すれば，選択平文の使い回しを行う事で $\binom{7}{5} \times 2^2 = 84$ 組の 4 段の積分特性 (4.7)，(4.8)，(4.9) を生成する事ができる．従って，50 bit の段鍵を回復する為に必要な選択平文数は $D = 2^{55}$ である．

6 段 KASUMI の鍵回復攻撃に必要な計算量は以下の 3 ステップの合計で見積もる事ができる．

1. T_E：暗号文の用意
2. T_{MFDT}：mod 2 頻度分布表 (mod 2 frequency distribution table，MFDT) の導出
3. T_{KR}：鍵回復

ステップ 1 は，鍵回復攻撃に用いる暗号文を用意する為に必要な計算量である．6 段 KASUMI の攻撃において，$D = 2^{55}$ の選択平文を暗号化する必要がある為，ステップ 1 で必要な計算量は

$$T_E = 2^{55} \tag{4.14}$$

回の暗号化計算量 [ENC.] である．尚，6 段の KASUMI の暗号化に必要な計算量を単位 (1[ENC.]) として見積もる事とする．

ステップ 2 は，mod 2 頻度分布表 (mod 2 frequency distribution table，MFDT) を導出する為に必要な計算量である．ステップ 1 で得た暗号文をそのまま用いて式 (4.13) を計算する場合，一

組の式 (4.13) を計算する為に 2^{53} 組の暗号文 C を代入し，排他的論理和による総和を計算する必要がある．しかしながら，GF(2) 上で偶数回出現する値を排他的論理和した結果は 0 となる為，奇数回出現する C_R と $C_L[24-16]$ のみ用いて式 (4.13) を計算すれば，2^{53} 組の暗号文 C を用いて計算した結果と同じ結果が得られる．C_R は 32 bit，$C_L[24-16]$ は 9 bit である為，2^{53} 組の暗号文 C の中には，重複する C_R と $C_L[24-16]$ が存在している．重複する C_R を取り除く為には，32 bit のテーブルを用意し，出現頻度の mod 2 を計算すれば良い．同様に，重複する $C_L[24-16]$ を取り除く為には，9 bit のテーブルを用意し，出現頻度の mod 2 を計算すれば良い．2^{53} の暗号文 C を用いて，2 種類のテーブルで C_R と $C_L[24-16]$ の出現回数を数える計算量は $2^{53} \times 2$ である．KASUMI の S-box (S7，S9) テーブルを 1 回参照する計算量と，2 種類のテーブルを 1 回参照する計算量が等しいと仮定すれば，KASUMI の FO 関数には 12 個の S-box が存在し，攻撃対象が 6 段の KASUMI である為，mod 2 頻度分布を計算する計算量は $\frac{2^{53} \times 2}{12 \times 6}$ 回の暗号化計算量 [ENC.] である．6 段の KASUMI の鍵回復攻撃を行う場合，55 組の式 (4.13) を計算する必要がある為，55 組の異なる 2^{53} の暗号文 C に対して C_R と $C_L[24-16]$ の mod 2 頻度分布を計算する必要がある．以上より，ステップ 2 に必要な計算量は

$$T_{MFDT} = 55 \times \frac{2^{53} \times 2}{12 \times 6} \approx 2^{53.7} \tag{4.15}$$

回の暗号化計算量 [ENC.] である．

ステップ 3 は，ステップ 2 で導出した mod 2 頻度分布テーブルを用いて線形化方程式を導出し，鍵回復を行う為に必要な計算量である．式 (3.12) から，線形化手法を用いて鍵回復を行う計算量は

$$T_{KR} = 2^{32} \times (494+1) \times \left\lceil \frac{494}{9} \right\rceil \times \frac{1}{6} \approx 2^{44.2} \tag{4.16}$$

回の暗号化計算量 [ENC.] である．

6 段の KASUMI の鍵回復攻撃を行う為に必要な計算量は，上記ステップ 1 からステップ 3 に必要な計算量の総和であり，

$$T = T_E + T_{MFDT} + T_{KR} \approx 2^{55.5} \tag{4.17}$$

回の暗号化計算量 [ENC.] である．尚，FL 関数内の段鍵 18 bit が全て 1 であると仮定している為，本手法を用いた攻撃の成功確率は 2^{-18} である．

6 段の KASUMI の鍵回復攻撃の成功確率を改善する手法について，以下説明する．式 (4.12) は 9 bit の方程式である為，1 bit 毎に式 (4.12) を分割する．$KL_{62}[j]$ と $KL_{42}[j], (j = 0, 1, \cdots, 7, 15)$

が共に 1 であると仮定すれば，式 (4.12) を以下の様にみなす事ができる．

$$\bigoplus_{C_R \in \{C\}} FO6(C_R; KO_6, KI_6)[i] \oplus \bigoplus_{C_L \in \{C\}} C_L[i] = 0 \ , \ (16 \leq i \leq 24) \tag{4.18}$$

上記の式 (4.18) には，50 bit の段鍵 ($KO_6 = \{KO_{61}, KO_{62}\},\ KI_6 = \{KI_{61}[8-0], KI_{62}[8-0]\}$) が存在する．1 bit の式 (4.18) を別々に解くことにより，50 bit の段鍵 KO_6 を回復する事ができる．先ほどと同様に線形化手法を用いて段鍵を回復するならば，494 組の式 (4.18) が必要となる．ここで，上位 7 bit を任意の固定値とし，残り 57 bit を変数として全通り出現する選択平文 2^{57} を用意すれば，選択平文の使い回しを行う事で $\binom{9}{5} \times 2^4 = 2016$ 組の 4 段の積分特性 (4.7)，(4.8)，(4.9) を生成する事ができる．従って，50 bit の段鍵を回復する為に必要な選択平文数は $D = 2^{57}$ である．

鍵回復攻撃に必要な計算量は前述のステップ 1 からステップ 3 に必要な計算量の総和で見積もる事ができる．ステップ 1 で暗号文を用意する為に必要な計算量は

$$T_E = 2^{57} \tag{4.19}$$

回の暗号化計算量 [ENC.] である．また，ステップ 2 で mod 2 頻度分布テーブルを計算する為に必要な計算量は

$$T_{MFDT} = 494 \times \frac{2^{53} \times 2}{12 \times 6} \approx 2^{56.8} \tag{4.20}$$

回の暗号化計算量 [ENC.] である．また，ステップ 3 で mod 2 頻度分布テーブルを用いて線形化方程式を導出し，鍵回復を行う為に必要な計算量は

$$T_{KR} = 2^{32} \times (494 + 1) \times \left\lceil \frac{494}{1} \right\rceil \times \frac{1}{6} \times 9 \approx 2^{50.5} \tag{4.21}$$

回の暗号化計算量 [ENC.] である．6 段の KASUMI の鍵回復攻撃を行う為に必要な計算量は，上記ステップ 1 からステップ 3 に必要な計算量の総和であり，

$$T = T_E + T_{MFDT} + T_{KR} = 2^{57} + 2^{56.8} + 2^{50.5} \approx 2^{58} \tag{4.22}$$

回の暗号化計算量 [ENC.] である．尚，段鍵 $KL_{62}[j]$ と $KL_{42}[j], (j = 0, 1, \cdots, 7, 15)$ が共に 1 であると仮定している為，本手法を用いた攻撃の成功確率は $1 - (\frac{3}{4})^9 \approx 0.925$ である．

7 段 KASUMI の攻撃

表 4.6 に示す 5 段の積分特性を用いた 7 段 KASUMI の鍵回復攻撃について説明する．下記に，5 段の積分特性を再掲する．

$$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]} \xrightarrow{5R} (\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$$

上記は KASUMI の 2 段目から 6 段目を対象とした 5 段の積分特性であり，FO6[24-16] の 9 bit が BALANCE する事を意味する．5 段の積分特性 (4.11) と図 4.3 から，以下に示す段鍵に関する方程式を導出する事ができる．

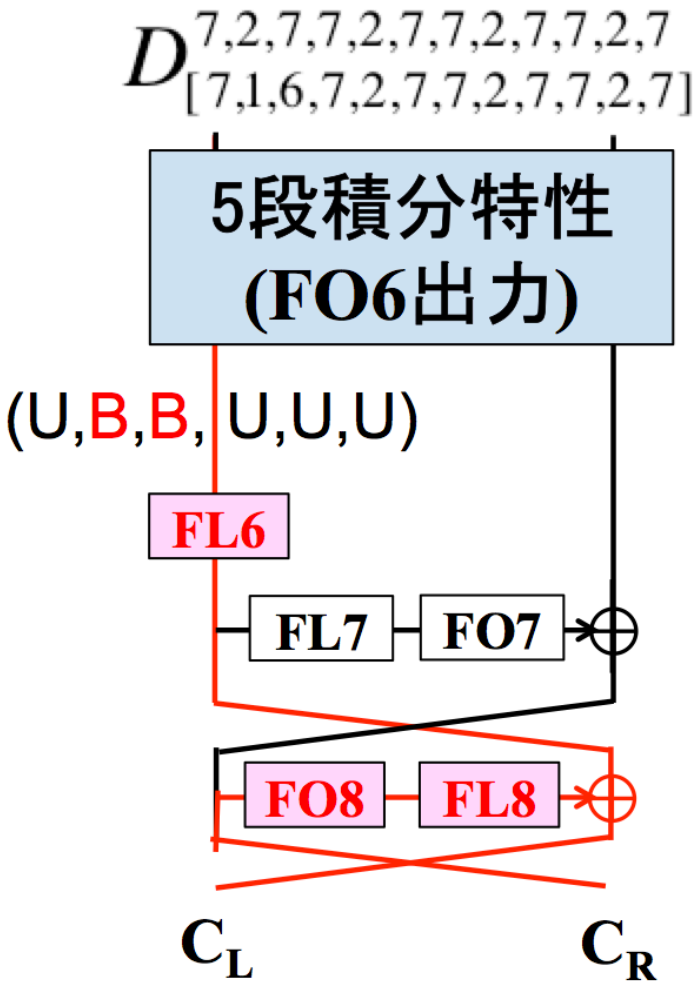

図 4.3: 7 段 KASUMI の鍵回復攻撃

$$\bigoplus_{C_R,C_L\in\{C\}} FL6^{-1}(FL8(FO8(C_R;KO_8,KI_8);KL_8)\oplus C_L;KL_6)[24-16]=0 \qquad (4.23)$$

式 (4.23) において，$\{C\}$ は平文集合に対応する暗号文集合を示す．また，C_L は暗号文 C の上位 (左側) 32 bit，C_R は暗号文 C の下位 (右側) 32 bit をそれぞれ示す．また，$FL6^{-1}$ は FL 関数の

逆関数を示す．式 (4.23) には FO8 内に 96 bit，FL8 内の 18 bit，FL6 内の 9 bit，合計 123 bit の段鍵が存在する．ここで，$KL_{82}[15,7-0]$ と $KL_{62}[15,7-0]$ の合計 18 bit が全て 1 だと仮定すると，式 (4.23) は以下の様にみなす事ができる．

$$\bigoplus_{C_R \in \{C\}} FO8(C_R; KO_8, KI_8)[24-16] \oplus \bigoplus_{C_L \in \{C\}} C_L[24-16] = 0 \tag{4.24}$$

式 (4.24) には 50 bit ($KO_8 = \{KO_{81}, KO_{82}\}$, $KI_8 = \{KI_{81}[8-0], KI_{82}[8-0]\}$) の段鍵が存在する．従って，18 bit の段鍵 ($KL_{82}[15,7-0]$ と $KL_{62}[15,7-0]$) を全て 1 と仮定する事により，式 (4.23) を用いて推定する段鍵数を削減する事ができる．

本論文では，50 bit の段鍵 ($KO_8 = \{KO_{81}, KO_{82}\}$, $KI_8 = \{KI_{81}[8-0], KI_{82}[8-0]\}$) を攻撃方程式 (4.24) の突き合わせを用いて回復する．また，KASUMI の鍵スケジュール部を用いて，1 と仮定した 18 bit の段鍵 ($KL_{82}[15,7-0]$ と $KL_{62}[15,7-0]$) を除く，残りの秘密鍵 $128-18-50=60$ bit を導出する．攻撃方程式 (4.24) の突き合わせ法は，式 (4.24) 中に存在する段鍵が互いに独立になる様に式 (4.24) を二つに分割 (左辺と右辺に分ける) し，左辺に存在する段鍵を推定して得られた結果をメモリに格納後，右辺に存在する段鍵を推定して得られた結果を用いてメモリを参照し，結果が一致する段鍵が正しいものとする手法である [44]．

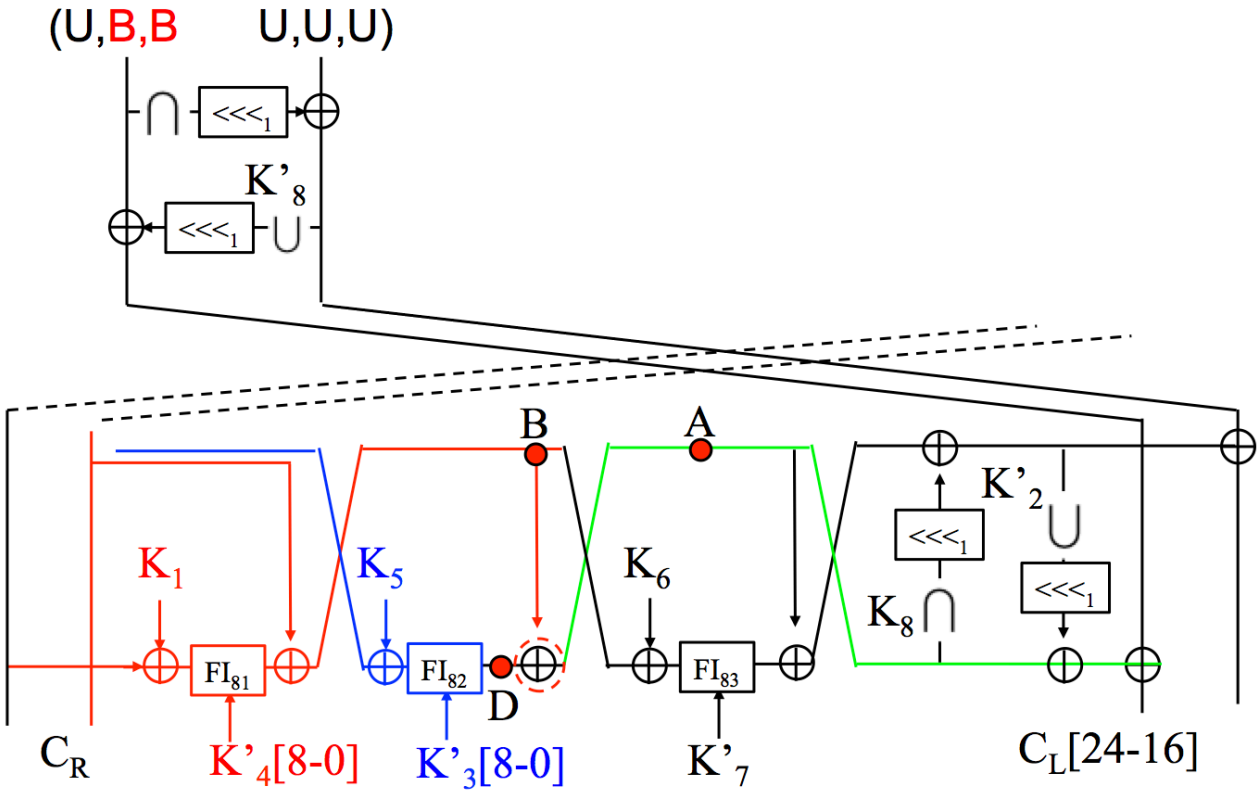

図 4.4: 7 段 KASUMI の鍵回復攻撃 (詳細)

図 4.4 を用いて，7 段 KASUMI の鍵回復攻撃のアルゴリズムを以下に示す．

1. 平文集合 $\{P\}$ の Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,7,7,2,7,7,2,7,7,2,7]}$ となる 2^{63} の選択平文を用意し，対応する暗号文の用意する．尚，平文の使い回しを行う事で，5 段の積分特性を構成する Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$ の平文集合 $\{P\}$ を，7 種類用意する事ができる．

2. メモリアドレスが 0〜511 の 9 bit テーブルを 1 つ用意し，0 で初期化する．

3. 暗号文集合 $\{C\}$ を用いて $C_R[31-0]$ と $C_L[24-16]$ に関する mod 2 頻度分布表をそれぞれ作成する．また，9 bit サイズ $C_L[24-16]$ の mod 2 頻度分布表を用いて，図 4.4 中の A（9 bit）を計算する．

4. 32 bit サイズ $C_R[31-0]$ の mod 2 頻度分布表を用いて $K_1, K'_4[8-0]$ の 25 bit を推定し，図 4.4 中の B（9 bit）を計算する．ステップ 2 で用意した 9 bit テーブルのメモリアドレスが $A \oplus B$ のアドレスに，推定した $K1, K'4[8-0]$ を格納する．

5. 32 bit サイズ $C_R[31-0]$ の mod 2 頻度分布表を用いて 16 bit サイズ $C_R[15-0]$ の mod 2 頻度分布表を作成する．(この時，$C_R[31-16]$ に関する部分は破棄する．)

6. 16 bit サイズ $C_R[15-0]$ の mod 2 頻度分布表を用いて $K5, K'3[8-0]$ を推定し，図 4.4 中の D（9 bit）を計算する．ステップ 2 で用意した 9 bit テーブルのメモリアドレスが D のアドレスに，推定した $K5, K'3[8-0]$ を格納する．

7. $A \oplus B = D$ を満たす $K1, K'4[8-0], K5, K'3[8-0]$ を鍵候補として導出する．異なる暗号文に対し，鍵候補が 1 つになるまでステップ 2 からステップ 7 を繰り返す．

8. ステップ 2 からステップ 7 にて 50 bit の段鍵 $(KO_8 = \{KO_{81}, KO_{82}\},\ KI_8 = \{KI_{81}[8-0], KI_{82}[8-0]\})$ を回復した後，KASUMI の鍵スケジュール部を用いて，1 と仮定した 18 bit の段鍵 $(KL_{82}[15,7-0]$ と $KL_{62}[15,7-0])$ を除く，残りの秘密鍵 60 bit を全数探索法を用いて導出する．

以下に，7 段の KASUMI の鍵回復攻撃に必要な選択平文数と計算量の見積もりを行う．

1. ステップ 1 では，平文集合 $\{P\}$ の Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,7,7,2,7,7,2,7,7,2,7]}$ となる 2^{63} を用意し，対応する暗号文を用意する．Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,7,7,2,7,7,2,7,7,2,7]}$ となる平文集合 $\{P\}$ は，平文 $P[63-0]$ において，$P[56,55]$ の内 1 bit を任意の定数とし，残り 63 bit を変数として全通り出現する入力集合 $\{X\}$ を用意すれば良い．ここでは，$P[56]$ を任意の定数とする．従って，

必要な選択平文数は $D = 2^{63}$ である．また，対応する暗号文を用意する為に必要な計算量 T_E は

$$T_E = 2^{63} \tag{4.25}$$

回の暗号化計算量 [ENC.] である．尚，平文の使い回しを行う事で，Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$ の平文集合 $\{P\}$ を，7 種類用意する事ができる．

2. ステップ 2 に必要な計算量は 2^9 回のテーブル初期化に必要な計算量である．

3. ステップ 3 では，5 段の積分特性を構成する Division 属性が $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,1,6,7,2,7,7,2,7,7,2,7]}$ の平文集合 $\{P\}$ に対応する暗号文集合 $\{C\}$ を用いて $C_R[31-0]$ と $C_L[24-16]$ に関する mod 2 頻度分布表をそれぞれ作成する．従って，必要な計算量は $T_{MFDT1} = 2^{62} \times 2$ 回の mod 2 頻度分布表を参照する計算量であり，前節と同様に S-box（S7，S9）を参照する計算量と等しいと仮定すれば，

$$T_{MFDT1} = \frac{2^{62} \times 2}{12 \times 7} \tag{4.26}$$

回の暗号化計算量 [ENC.] である [2]．尚，A の値は，9 bit サイズ $C_L[24-16]$ の mod 2 頻度分布表を用いて $A = \bigoplus_{i=1}^{2^9} C_L(i)[24-16]$ にて導出可能であり，必要な計算量は 2^9 回の mod 2 頻度分布表を参照する計算量である．

4. ステップ 4 では，32 bit サイズ $C_R[31-0]$ の mod 2 頻度分布表を用いて K_1, $K'_4[8-0]$ の 25 bit を推定し，

$$B = \bigoplus_{i=1}^{2^{32}} FI_{81}(C_R(i)[31-16] \oplus K_1; K'_4[8-0])[8-0] \oplus \bigoplus_{i=1}^{2^{32}} C_R(i)[8-0]$$

を計算する．必要な計算量は $T_{KR1} = 2^{32} \times 2^{25}$ 回の FI 関数計算量であり，

$$T_{KR1} = \frac{2^{32} \times 2^{25}}{3 \times 7} \tag{4.27}$$

回の暗号化計算量 [ENC.] である [3]．また，ステップ 2 で用意した 9 bit テーブルのメモリアドレスが $A \oplus B$ のアドレスに，推定した $K1$, $K'_4[8-0]$ を格納する．

5. ステップ 5 では，32 bit サイズ $C_R[31-0]$ の mod 2 頻度分布表を用いて 16 bit サイズ $C_R[15-0]$ の mod 2 頻度分布表を生成する．この処理に必要な計算量は $T_{MFDT2} = 2^{32}$ 回の mod 2 頻

[2] T_{MFDT1} の参照計算量は S-box を参照する計算量で導出している．KASUMI の FO 関数には S-box（S7，S9）が 12 個存在し，攻撃対象が 7 段 KASUMI であるので，(12×7) で割る事で暗号化計算量 [ENC.] にしている．

[3] T_{KR1} の計算量は FI 関数計算量で導出している．KASUMI の FO 関数には FI 関数が 3 個存在し，攻撃対象が 7 段 KASUMI であるので，(3×7) で割る事で暗号化計算量 [ENC.] にしている．

度分布表を参照する計算量であり，

$$T_{MFDT2} = \frac{2^{32}}{12 \times 7} \tag{4.28}$$

回の暗号化計算量 [ENC.] である．

6. ステップ 6 では，ステップ 5 で生成した 16 bit サイズ $C_R[15-0]$ の mod 2 頻度分布表を用いて $K5, K'3[8-0]$ を推定し，

$$D = \bigoplus_{i=1}^{2^{16}} FI_{82}(C_R(i)[15-0] \oplus K_5; K'_3[8-0])[8-0]$$

を計算する．この処理に必要な計算量は $T_{KR2} = 2^{16} \times 2^{25}$ 回の FI 関数計算量であり，

$$T_{KR2} = \frac{2^{16} \times 2^{25}}{3 \times 7} \tag{4.29}$$

回の暗号化計算量 [ENC.] である．また，ステップ 2 で用意した 9 bit テーブルのメモリアドレスが D のアドレスに，推定した $K5, K'3[8-0]$ を格納する．

7. ステップ 7 では，$A \oplus B = D$ を満たす $K1, K'4[8-0], K5, K'3[8-0]$ を鍵候補として導出する．平文を使い回す事によって作成した 6 種類の 5 段積分特性に対応する暗号文集合 $\{C\}$ に対し，式 (4.24) 中に存在する段鍵 50 bit の候補が 1 つになるまでステップ 2 からステップ 7 を繰り返す．式 (4.24) は 9 bit で構成される為，一組の式 (4.24) で鍵候補を 2^{-9} づつ絞り込む事が可能である．従って，50 bit の鍵候補数は以下の様にステップ 2 からステップ 7 を繰り返す度に減少し，1 つに絞り込まれる．

$$\text{鍵候補数} : 2^{50} \xrightarrow{1^{st}} 2^{41} \xrightarrow{2^{nd}} 2^{32} \xrightarrow{3^{rd}} 2^{23} \xrightarrow{4^{th}} 2^{14} \xrightarrow{5^{th}} 2^{5} \xrightarrow{6^{th}} 1$$

5 段積分特性に対応する一組目の暗号文集合 $\{C\}$ を用いてステップ 2 からステップ 7 を行う為に必要な計算量は

$$\begin{aligned} T_{1^{st}} &= T^{1^{st}}_{MFDT1} + T^{1^{st}}_{KR1} + T^{1^{st}}_{MFDT2} + T^{1^{st}}_{KR2} \\ &= \frac{2^{62} \times 2}{12 \times 7} + \frac{2^{32} \times 2^{25}}{3 \times 7} + \frac{2^{32}}{12 \times 7} + \frac{2^{16} \times 2^{25}}{3 \times 7} \end{aligned} \tag{4.30}$$

回の暗号化計算量 [ENC.] である．$T_{1^{st}}$ の計算量で支配的なのは，$T_{MFDT1} + T_{KR1}$ である．また，2 組目以降は鍵候補数が減少する為，ステップ 3 の mod 2 頻度分布表を作成する計算量 T_{MFDT1} が支配的となる．

8. ステップ 8 では，KASUMI の鍵スケジュール部を用いて，1 と仮定した 18 bit の段鍵 ($KL_{82}[15, 7-$

0] と $KL_{62}[15, 7-0]$) を除く，残りの秘密鍵 60 bit を導出する．60 bit の秘密鍵を全数探索法で導出する為に必要な計算量は

$$T_{KR3} = 2^{60} \tag{4.31}$$

回の暗号化計算量 [ENC.] である．尚，この時に使用する平文・暗号文組はステップ 1 で用意したものを使い回せば良い．

以上より，7 段の KASUMI の鍵回復攻撃に必要な計算量は

$$\begin{aligned} T &= T_E + T_{1^{st}} + T_{2^{nd}} + T_{3^{rd}} + T_{4^{th}} + T_{5^{th}} + T_{6^{th}} + T_{KR3} \\ &= 2^{63} + \frac{2^{62} \times 2}{12 \times 7} + \frac{2^{32} \times 2^{25}}{3 \times 7} + \frac{2^{32}}{12 \times 7} + \frac{2^{16} \times 2^{25}}{3 \times 7} + \frac{2^{62} \times 2}{12 \times 7} \times 5 + 2^{60} \\ &\approx 2^{63.3} \end{aligned} \tag{4.32}$$

回の暗号化計算量 [ENC.] である．KASUMI の既存研究結果との比較を表 4.7 に纏める．

表 4.7: 既存研究との比較

段数	平文数	計算量 [ENC.]	攻撃手法	文献
5	$2^{39.4}$ CP	2^{117}	Higher Order Differential	[61]
5	$2^{27.5}$ CP	2^{34}	Higher Order Differential	**3.5 節**
5	$2^{28.9}$ CP	$2^{31.2}$	Higher Order Differential	**3.5 節**
6(2-7)	$2^{60.8}$ CP	$2^{65.4}$	Higher Order Differential	[43]
6(2-7)	2^{55} CP	2^{100}	Impossible Differential	[27]
6	$2^{62.8}$ KP	2^{85}	Multidimensional Zero-Correlation	[55]
6	2^{57} CP	2^{58}	Integral by Division Property	**4.4 節**
7(2-8)	$2^{52.5}$ CP	$2^{114.3}$	Impossible Differential	[22]
7	2^{62} KP	$2^{115.8}$	Impossible Differential	[22]
7(2-8)	$2^{62.1}$ KP	$2^{110.5}$	Multidimensional Zero-Correlation(Weak Key)	[55]
7(2-8)	2^{63} CP	$2^{63.3}$	Integral by Division Property(Weak Key)	**4.4 節**
8	2^{32} CP	$2^{125.5}$	Meet-in-the-Middle Attack	[23]

KP(Known Plaintext)：既知平文，CP(Chosen Plaintext)：選択平文，ENC.：暗号化関数計算量を示す．

4 段の積分特性を用いた場合，6 段の KASUMI (攻撃対象は 1 段目から 6 段目) が選択平文数 $D = 2^{57}$ と計算量 $T = 2^{58}$ 回の暗号化計算量 [ENC.] で攻撃可能である．6 段 KASUMI の攻撃では，攻撃方程式を bit 単位に分割し，FL 関数内の OR 演算を行う段鍵 (KL_{42}, KL_{62}) を 1 と仮定する事により，推定する段鍵の bit 数を 123 bit から 50 bit に削減した．また，攻撃方程式を解く際には線形方程式を並列に解く事により，6 段の KASUMI の鍵回復攻撃の成功確率をほぼ 1($p = 0.925$) にした．また，5 段の積分特性を用いた場合，7 段の KASUMI(攻撃対象は 2 段目から 8 段目) が選

択平文数 $D = 2^{63}$ と計算量 $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] で攻撃可能である．7 段 KASUMI の攻撃では，KASUMI の鍵スケジュール部を活用し，且つ FL 関数内の OR 演算に着目した発生確率が $p = 2^{-18}$ の弱鍵を用いる事で，秘密鍵 110 bit を回復している．鍵回復攻撃では，攻撃方程式の値に関して中間一致法を適用する事で，計算量を削減した．弱鍵を用いた 7 段の KASUMI に対する攻撃に関して Yi らの Multidimensional Zero-Correlation Attack が知られている [55]．文献 [55] と比較した場合，本論文の手法は攻撃に必要な計算量が最も少ないという点で，最良の結果 [4] である．

[4] 2016 年 9 月時点において

4.5 纏め

本章では，KASUMI に対し Division 属性を用いた積分特性探索を適用した．積分特性の探索において，KASUMI の非線形関数 (S7，S9) の Division 属性伝搬を調査し，その結果と Division 属性の伝搬ルールを組合わせる事により，FI 関数・FO 関数の Division 属性伝搬を解析した．また，鍵依存線形関数である FL 関数内に存在する 1 ビット左巡回シフトの影響を考慮した属性伝搬を構築し，FO 関数・FL 関数の伝搬特性を組合わせる事で，KASUMI の Division 属性伝搬を解析した．その結果，1 段目から開始した場合は 4 段の積分特性が存在し，2 段目から開始した場合は 5 段の積分特性が存在する事を明らかにした．

4 段特性を用いた場合，6 段の KASUMI(攻撃対象は 1 段目から 6 段目) が選択平文数 $D = 2^{57}$ と計算量 $T = 2^{58}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示した．6 段 KASUMI の攻撃では，攻撃方程式を bit 単位に分割し，FL 関数内の OR 演算を行う段鍵 (KL_{42}, KL_{62}) を 1 と仮定する事により，推定する段鍵の bit 数を 123 bit から 50 bit に削減した．また，攻撃方程式を解く際には線形方程式を並列に解く事により，6 段の KASUMI の鍵回復攻撃の成功確率をほぼ $1(p = 0.925)$ にした．また，5 段特性を用いた場合，7 段の KASUMI(攻撃対象は 2 段目から 8 段目) が選択平文数 $D = 2^{63}$ と計算量 $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示した．7 段 KASUMI の攻撃では，KASUMI の鍵スケジュール部を活用し，且つ FL 関数内の OR 演算に着目した発生確率が $p = 2^{-18}$ の弱鍵を用いる事で，秘密鍵 110 bit を回復している．鍵回復攻撃では，攻撃方程式の値に関して中間一致法を適用する事で，計算量を削減した．弱鍵を用いた 7 段の KASUMI に対する攻撃に関して，本論文の手法は攻撃に必要な計算量が最も少ないという点で，最良の結果 [5] である．

[5] 2016 年 9 月時点において

第5章　高階差分攻撃と積分攻撃の関係性

5.1　概要

本章では，高階差分攻撃と積分攻撃の関係性について明らかにする．必要平文数が最小となる最良の積分攻撃は高階差分攻撃と一致する事を示す．上記を証明する手段として，リード・マラー符号 [32] に関する先行研究の成果（最小重みと符号語数）を用いる．具体的には，m bit 入力の d 次関数に対し，高階差分攻撃の $(d+1)$ 階差分に必要な選択平文数は 2^{d+1} であり，その選び方はアフィン変換構造を持つ種類数を数え上げる事により $2^{m-d-1}\prod_{i=0}^{d}\frac{(2^{m-i}-1)}{(2^{d+1-i}-1)}$ 通りである事を示す．また，Division 属性から導出される最良の積分攻撃は，高階差分攻撃の $(d+1)$ 階差分と等しく，2^{d+1} の選択平文数を必要とする事を示す．更に，2^{d+1} の選択平文数を要素に持つ集合の種類数は，リード・マラー符号の既存成果を使用する事により $2^{m-d-1}\prod_{i=0}^{d}\frac{(2^{m-i}-1)}{(2^{d+1-i}-1)}$ 通りであり，必要な選択平文数を最小とする積分攻撃は高階差分攻撃と一致する事を示す．

続いて，理論の実証として KASUMI と MISTY(MISTY1，MISTY2) に対し，Division 属性を用いて探索した積分特性と既存の高階差分特性を比較する．比較の結果，KASUMI に関して既知の高階差分特性と，Division 属性を用いて導出した積分特性に必要な選択平文の変数位置，及び選択平文数は一致した．また，MISTY(MISTY1，MISTY2) の特性に関して Division 属性を用いた積分特性と高階差分特性は概ね一致した．尚，一部の特性比較結果に差がある事から，最良の積分特性は高階差分特性と一致する事を踏まえると，Division 属性の探索法に改良の余地が見込まれる．

以下に，第 5 章で用いる記号の一覧を示す．

5.2　従来法と問題提起

5.2.1　従来法

共通鍵ブロック暗号の安全性は様々な攻撃手法を用いて，どの程度耐性を有しているかで評価を行う．高階差分攻撃と積分攻撃に関する証明可能安全性の議論はなされておらず，共通鍵ブロック暗号の安全性を評価する場合，それぞれの攻撃手法を用いて，どの程度耐性を有するのか評価

表 5.1: 第 5 章で用いる記号一覧

記号	説明
$\{\boldsymbol{v}\}$	任意の $\boldsymbol{v}$ を要素とする集合．$\boldsymbol{v} \in \{\boldsymbol{v}\}$.
$\#\{\boldsymbol{v}\}$	集合 $\{\boldsymbol{v}\}$ の要素の数
$\{\{\boldsymbol{v}\}\}$	$\{\boldsymbol{v}\}$ を要素とする集合
$\#\{\{\boldsymbol{v}\}\}$	集合 $\{\boldsymbol{v}\}$ の種類数
$\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_{d+1}\}$	$GF(2)^m$ 上で 1 次独立な $(d+1)$ 個のベクトルの集合
$\boldsymbol{A}$	i 番目行ベクトルが $\boldsymbol{\alpha}_i$ の $(d+1) \times m$ サイズの行列，$\mathrm{rank}(\boldsymbol{A}) = d+1$
$\boldsymbol{\beta}$	$(d+1)$ 次元ベクトル．$\boldsymbol{\beta} \in GF(2)^{d+1}$
$\boldsymbol{\gamma}$	m 次元ベクトル．任意の定数．$\boldsymbol{\gamma} \in GF(2)^m$
$\mathrm{RM}(r, m)$	リード・マラー符号
d_{min}	$\mathrm{RM}(r, m)$ 符号の最小重み
$A_{2^{m-r}}$	最小重み d_{min} を与える符号語数
$\boldsymbol{w}$	$\mathrm{RM}(r, m)$ 符号の符号語
$\boldsymbol{H}$	パリティ検査行列．パリティ検査方程式 $\boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{0}$ を満たす
$\mathrm{D}_{\{X\}}(\boldsymbol{x})$	m 次元ベクトル $\boldsymbol{x} \in GF(2)^m$ の集合 $\{X\}$ の頻度分布
$\mathrm{D2}_{\{X\}}(\boldsymbol{x})$	$\mathrm{D}_{\{X\}}(\boldsymbol{x})$ の mod 2 を計算した頻度分布
$\boldsymbol{w}_{\{X\}}$	要素が $\mathrm{D2}_{\{X\}}$ の 2^m 次元ベクトル

していた．

5.2.2 問題提起

従来法では，対象とする共通鍵ブロック暗号の安全性を評価する場合，高階差分攻撃と積分攻撃のそれぞれを適用し，耐性の評価をする必要があった．そこで本論文では，高階差分攻撃と積分攻撃の関係性について明らかにする．また，発見的手法で知られている既存の高階差分特性と Division 属性を用いて発見した積分特性との比較を行う．

5.3 高階差分攻撃

5.3.1 最小平文数

本論文で用いる表記について説明する．任意の $\boldsymbol{v}$ において，$\{\boldsymbol{v}\}$ は集合を示し，その要素は $\boldsymbol{v}$ とする．従って，$\boldsymbol{v} \in \{\boldsymbol{v}\}$ である．また，$\{\{\boldsymbol{v}\}\}$ は集合を示し，その要素は $\{\boldsymbol{v}\}$ とする．$\#\{\boldsymbol{v}\}$ は集合 $\{\boldsymbol{v}\}$ の要素の数を示し，$\#\{\{\boldsymbol{v}\}\}$ は集合 $\{\boldsymbol{v}\}$ の種類数を示す．

本節では，高階差分の性質 $\Delta^{(d+1)} f(X; K) = 0$ に着目し，$(d+1)$ 階差分に必要な選択平文数と，$(d+1)$ 階差分を構成する集合の種類数について説明する．

式 (3.1) に示す暗号化関数 f の次数を d とする． $\{\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_{d+1}\}$ を $GF(2)^m$ 上で 1 次独立な $(d+1)$ 個のベクトルの集合とし，これらのベクトルによって張られる $GF(2)^m$ 上の $(d+1)$ 次元部分空間を $V^{(d+1)}$ と表す．$X \in GF(2)^m$ を暗号化関数 f の平文入力とした時，$(d+1)$ 階差分を構成する入力集合 $\{X\}$ は以下の式で表現される．

$$\{X\} = \{\boldsymbol{\beta}\boldsymbol{A} \oplus \boldsymbol{\gamma} | \boldsymbol{\beta} \in GF(2)^{d+1}\}, \tag{5.1}$$

ここで，行列 $\boldsymbol{A}$ は i 番目の行ベクトルを $\boldsymbol{\alpha}_i$ $(i = 1, 2, \cdots, d+1)$ とする $(d+1) \times m$ サイズの行列であり，行列 $\boldsymbol{A}$ の階数（rank）を $(d+1)$ とする．また，$\boldsymbol{\gamma} \in GF(2)^m$ は任意の定数とする．

暗号化関数 f の X に関する $(d+1)$ 階差分は以下の式で表現される．

$$\Delta^{(d+1)} f(X) = \bigoplus_{X \in \{X\}} f(X) \tag{5.2}$$

また，$(d+1)$ 階差分に必要な最小平文数 (選択平文数) は，

$$\#\{X\} = \#\{\boldsymbol{\beta}\} = 2^{d+1} \tag{5.3}$$

である．

5.3.2 高階差分の種類数

式 (5.1) に示す行列 $\boldsymbol{A}$ の種類数を $\#\{\boldsymbol{A}\}$ とする．行列 $\boldsymbol{A}$ の種類数 $\#\{\boldsymbol{A}\}$ は，行列 $\boldsymbol{A}$ の行ベクトル $\boldsymbol{\alpha}_i$ の候補数を数える事で計算できる．行ベクトル $\boldsymbol{\alpha}_1$ は，全ての成分が 0 のゼロベクトル $\boldsymbol{0}$ を除く任意の m 次元ベクトルから選ぶ事ができる．従って，行ベクトル $\boldsymbol{\alpha}_1$ の候補数は $2^m - 1$ である．行ベクトル $\boldsymbol{\alpha}_2$ は，$\boldsymbol{\alpha}_1$ と線形独立な任意のベクトルである．従って，行ベクトル $\boldsymbol{\alpha}_1$ の候補数は $2^m - 2$ である．行ベクトル $\boldsymbol{\alpha}_3$ は，$\boldsymbol{\alpha}_1$，$\boldsymbol{\alpha}_2$ と線形独立な任意のベクトルである．従って，行ベクトル $\boldsymbol{\alpha}_3$ の候補数は $2^m - 2^2$ である．上記を繰り返し行う事で，行ベクトル $\boldsymbol{\alpha}_i$ を選ぶ事ができる．従って，行列 $\boldsymbol{A}$ の種類数 $\#\{\boldsymbol{A}\}$ は以下の式で求める事ができる．

$$\#\{\boldsymbol{A}\} = \prod_{i=0}^{d} (2^m - 2^i) \tag{5.4}$$

行列 $\boldsymbol{A}$ が異なっても，m 次元ベクトルの集合 $\{\boldsymbol{\beta}\boldsymbol{A} | \boldsymbol{\beta} \in GF(2)^{d+1}\}$ が同一となる場合がある．その様な行列 $\boldsymbol{A}$ の種類数は $\prod_{i=0}^{d} (2^{d+1} - 2^i)$ で与えられる．これは，$(d+1) \times (d+1)$ サイズの正

則行列の種類数に等しい．従って，集合 $\{\boldsymbol{\beta A}\}$ の種類数 $\#\{\{\boldsymbol{\beta A}\}\}$ は

$$\#\{\{\boldsymbol{\beta A}\}\} = \prod_{i=0}^{d} \frac{2^m - 2^i}{2^{d+1} - 2^i} \tag{5.5}$$

となる．同一の集合 $\{\boldsymbol{\beta A}\}$ を用いて異なる集合 $\{X\}$ を生成する定数 $\boldsymbol{\gamma}$ の種類数は，2^{m-d-1} である．これは，空間 $V^{(m)}$ を $(d+1)$ 次元部分空間 $V^{(d+1)}$ で割った剰余類数である．

以上を纏めると $(d+1)$ 階差分に必要な最小平文数 (選択平文数) は

$$\#\{X\} = 2^{d+1} \tag{5.6}$$

であり，異なる $(d+1)$ 階差分を構成する集合 $\{X\}$ の種類数$\#\{\{\mathrm{X}\}\}$ は

$$\begin{aligned} \#\{\{X\}\} &= 2^{m-d-1} \prod_{i=0}^{d} \frac{(2^m - 2^i)}{(2^{d+1} - 2^i)} \\ &= 2^{m-d-1} \prod_{i=0}^{d} \frac{(2^{m-i} - 1)}{(2^{d+1-i} - 1)} \end{aligned} \tag{5.7}$$

である．

5.4 リード・マラー符号 (RM 符号)

本論文で使用するリード・マラー符号 (以下，RM 符号と記す) について概要を示す．尚，詳細は文献 [32] を参照のこと．

GF(2) 上の n 次元ベクトル $\boldsymbol{b}_1$ と $\boldsymbol{b}_2$ について考える．

$$\begin{aligned} \boldsymbol{b}_1 &= (b_1[1], b_1[2], \cdots, b_1[n]) \\ \boldsymbol{b}_2 &= (b_2[1], b_2[2], \cdots, b_2[n]) \end{aligned} \tag{5.8}$$

ベクトル $\boldsymbol{b}_1$ と $\boldsymbol{b}_2$ の掛け算，$\boldsymbol{b}_1\boldsymbol{b}_2$，は bit 毎の掛け算で以下の様に定義する．

$$\boldsymbol{b}_1\boldsymbol{b}_2 = (b_1[1]b_2[1], b_1[2]b_2[2], \cdots, b_1[n]b_2[n]) \tag{5.9}$$

ここで，$n = 2^m$ とし，以下の n 次元行ベクトルについて考える．$\boldsymbol{v}_0$ を要素が全て 1 のベクトルとし，0 次行ベクトルと呼ぶ．$g(\boldsymbol{v}) = v_i \ (1 \leq i \leq m)$ を 1 次関数とし，入力を $\boldsymbol{v} = (v_1, v_2, \cdots, v_m) \in GF(2)^m$ とする．ベクトル $\boldsymbol{v}_i \ (1 \leq i \leq m)$ は，全ての入力 $\boldsymbol{v} \in GF(2)^m$ に対し，$g(\boldsymbol{v})$ の出力を 2 進数表記で並べたベクトルとする．ここでは，ベクトル $\boldsymbol{v}_i \ (1 \leq i \leq m)$ を 1 次項行ベクトルと呼ぶ．行ベクトル $\boldsymbol{v}_i$ は $\binom{m}{1}$ 通り存在する．ベクトル $\boldsymbol{v}_i\boldsymbol{v}_j \ (1 \leq i < j \leq m)$ は，1 次関数 $g(\boldsymbol{v})$ を使用

した 2 次関数として表現可能である．ここでは，ベクトル $\boldsymbol{v}_i\boldsymbol{v}_j$ $(1 \leq i < j \leq m)$ を 2 次項行ベクトルと呼ぶ．行ベクトル $\boldsymbol{v}_i\boldsymbol{v}_j$ は，$\binom{m}{2}$ 通り存在する．以下，上記を繰り返す事で r 次 $(1 \leq r \leq m)$ 項行ベクトルを表現可能である．一例として，表 5.2 に $m = 4$ で行ベクトルを示す．

表 5.2: RM 符号生成基底 $(m = 4)$

$\boldsymbol{v_0}$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$\boldsymbol{v_1}$	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
$\boldsymbol{v_2}$	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
$\boldsymbol{v_3}$	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
$\boldsymbol{v_4}$	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
$\boldsymbol{v_1v_2}$	0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
$\boldsymbol{v_1v_3}$	0 0 0 0 0 0 0 0 0 0 1 1 0 0 1 1
$\boldsymbol{v_1v_4}$	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$\boldsymbol{v_2v_3}$	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$\boldsymbol{v_2v_4}$	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$\boldsymbol{v_3v_4}$	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1
$\boldsymbol{v_1v_2v_3}$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
$\boldsymbol{v_1v_2v_4}$	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1
$\boldsymbol{v_1v_3v_4}$	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
$\boldsymbol{v_2v_3v_4}$	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1
$\boldsymbol{v_1v_2v_3v_4}$	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

定義 3 (RM(r, m)) *r 次の RM 符号 (RM(r, m)) は 2 元 (n, k) 線形符号であり，i 次 $(0 \leq i \leq r)$ 項行ベクトル $\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_m, \boldsymbol{v}_1\boldsymbol{v}_2, \cdots$ 迄を生成基底とする．符号長は $n = 2^m$ で，情報ビット数は $k = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}$ である．*

RM 符号に関して，以下の定理が知られている [32]．

定理 1 (最小重みと符号語数) *RM(r, m) 符号の最小重みは $d_{min} = 2^{m-r}$ である．$A_{2^{m-r}}$ を最小重み d_{min} を与える符号語数とした時，符号語数 $A_{2^{m-r}}$ は以下の式で計算される．*

$$A_{2^{m-r}} = 2^r \prod_{i=0}^{m-r-1} \frac{(2^{m-i} - 1)}{(2^{m-r-i} - 1)} \tag{5.10}$$

定理 2 (双対符号：RM(m-r-1, m)) *RM$(m - r - 1, m)$ 符号は RM(r, m) の双対符号である．*

$\boldsymbol{w}$ を RM(r,m) の符号語とする．符号語 $\boldsymbol{w}$ は 0 次項から r 次項行ベクトル $(\boldsymbol{v}_0, \boldsymbol{v}_1, \cdots, \boldsymbol{v}_m, \boldsymbol{v}_1\boldsymbol{v}_2, \cdots)$ までの線形結合で表される．0 次項行ベクトルから $(m-r-1)$ 次項行ベクトルまでを一列に並べて生成した $(1+\binom{m}{1}+\cdots+\binom{m}{m-r-1}) \times n$ サイズの行列を $\boldsymbol{H}$ とする．行列 $\boldsymbol{H}$ はパリティ検査行列と呼ばれ，パリティ検査方程式 $\boldsymbol{H}\boldsymbol{w}^{\mathrm{T}} = \boldsymbol{0}$ を満たす．ここで，'T' は転置を表す．従って，符号語 $\boldsymbol{w}$ は $(m-r-1)$ 次以下の行ベクトル $\boldsymbol{v}_i\boldsymbol{v}_j\cdots\boldsymbol{v}_k$ と直行する．即ち，$(\boldsymbol{v}_i\boldsymbol{v}_j\cdots\boldsymbol{v}_k) \bullet \boldsymbol{w} = 0$ である．記号'$\bullet$' はベクトルの内積を示す．

5.5 積分攻撃

5.5.1 Division 属性と mod 2 頻度分布

式 (3.1) に示す暗号化関数 f の次数を d とする．m 次元ベクトル $\boldsymbol{x} \in GF(2)^m$ の集合 $\{X\}$ の頻度分布を $\mathrm{D}_{\{X\}}(\boldsymbol{x})$ とする． $\boldsymbol{x} = \boldsymbol{a}$ を満たすベクトルの数は以下の式で表される．

$$\mathrm{D}_{\{X\}}(\boldsymbol{a}) = \#\{\boldsymbol{x} | \boldsymbol{x} = \boldsymbol{a} \in \{X\}\} \tag{5.11}$$

$\mathrm{D}_{\{X\}}(\boldsymbol{a})$ が偶数の場合，集合 $\{X\}$ の中にベクトル $\boldsymbol{a}$ は偶数回出現し，排他的論理和 (XOR) による総和 $\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x})$ に影響を与えない．一方，$\mathrm{D}_{\{X\}}(\boldsymbol{a})$ が奇数の場合，集合 $\{X\}$ の中にベクトル $\boldsymbol{a}$ が一回出現したものと同じ効果を XOR 総和 $\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x})$ に与える．従って，XOR 総和 $\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x})$ は頻度分布 $\mathrm{D}_{\{X\}}(\boldsymbol{a})$ の mod 2 に依存する．ここで，mod 2 頻度分布を $\mathrm{D2}_{\{X\}}(\boldsymbol{a})$ で表すとすれば，$\mathrm{D2}_{\{X\}}(\boldsymbol{a})$ は以下の式で計算される．

$$\mathrm{D2}_{\{X\}}(\boldsymbol{a}) = \mathrm{D}_{\{X\}}(\boldsymbol{a}) \bmod 2 \tag{5.12}$$

最小の必要平文数を議論する為，$\mathrm{D}_{\{X_e\}}(\boldsymbol{a}) = \mathrm{D2}_{\{X\}}(\boldsymbol{a})$ を満たす集合 $\{X_e\}$ を本質的な集合と呼ぶ．任意の d 次関数 $f(\boldsymbol{x})$ に対し，以下の式が成り立つ．

$$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x}) = \bigoplus_{\boldsymbol{x}\in\{X_e\}} f(\boldsymbol{x}) \tag{5.13}$$

上記の議論から，以下の定理が導出できる．

定理 3 *m 次元ベクトルの集合 $\{X\}$ と $\{X_e\}$ に対し，集合 $\{X\}$ と $\{X_e\}$ の mod 2 頻度分布が等しい場合，集合 $\{X\}$ の Division 属性は，集合 $\{X_e\}$ の Division 属性に等しい．*

最小の必要平文数を議論する為，集合 $\{X_e\}$ を用いた場合，明らかに $\#\{X_e\} \leq \#\{X\}$ である．

文献 [49] 中に例示されている以下の 4 ビットベクトルの集合 $\{X\}$ について示す．

$$\{X\} = \{\texttt{0x0}, \texttt{0x3}, \texttt{0x3}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x8}, \texttt{0xB}, \texttt{0xD}, \texttt{0xE}\}$$

上記の集合 $\{X\}$ は Division 属性 $\mathcal{D}_3^4$ を与える集合である．集合 $\{X\}$ の中に，0x3 が 3 回出現しており，$\#\{X\} = 10$ である．従って，以下の集合 $\{X_e\}$

$$\{X_e\} = \{\texttt{0x0}, \texttt{0x3}, \texttt{0x5}, \texttt{0x6}, \texttt{0x8}, \texttt{0xB}, \texttt{0xD}, \texttt{0xE}\}$$

を考えれば，$\mathrm{D}_{\{X_e\}}(\boldsymbol{a}) = \mathrm{D2}_{\{X\}}(\boldsymbol{a})$ を満たし，$\#\{X_e\} = 8$ となる．従って，$\#\{X_e\} \leq \#\{X\}$ を満たす．また，集合 $\{X_e\}$ は Division 属性 $\mathcal{D}_3^4$ を与える集合である．

例として，$(u[1], u[2], u[3], u[4]) = (1, 1, 0, 0)$ とする 2 次項 $\pi_u(\boldsymbol{x})$ に 4 ビットベクトル $\boldsymbol{x} \in GF(2)^4$ を 2 進数順に入力し，以下の様に並べる．

$$(\pi_{1100}(0), \pi_{1100}(1), \cdots, \pi_{1100}(i), \cdots, \pi_{1100}(15)) \tag{5.14}$$

ここで，i $(0 \leq i \leq 15)$ は 4 ビットベクトル $\boldsymbol{x} \in GF(2)^4$ を 2 進数順に入力した値を示す．表記 (5.14) を 2^4 次元ベクトルとみなした場合，表記 (5.14) は表 5.2 の 2 次項行ベクトル $\boldsymbol{v_1 v_2}$ に等しい．一般に，d 次項 $\pi_u(\boldsymbol{x}) = x[j]x[k]\cdots x[l]$ に m ビットベクトル $\boldsymbol{x} \in GF(2)^m$ を 2 進数順に入力して並べた 2^m 次元ベクトルは，RM 符号の d 次項行ベクトル $\boldsymbol{v}_j \boldsymbol{v}_k \cdots \boldsymbol{v}_l$ である．

$\boldsymbol{w}_{\{X\}}$ は要素が集合 $\{X\}$ の mod 2 頻度分布 $\mathrm{D2}_{\{X\}}$ である 2^m 次元ベクトルとし，以下の様に示す．

$$\boldsymbol{w}_{\{X\}} = (\mathrm{D2}_{\{X\}}(0), \mathrm{D2}_{\{X\}}(1), \cdots, \mathrm{D2}_{\{X\}}(i), \cdots, \mathrm{D2}_{\{X\}}(2^m - 1)) \tag{5.15}$$

ここで，i $(0 \leq i \leq 2^m - 1)$ は m ビットベクトル $\boldsymbol{x} \in GF(2)^m$ を 2 進数順に入力した値を示す．この時，XOR 総和 $\bigoplus_{\boldsymbol{x} \in \{X\}} f(\boldsymbol{x})$ は以下の様に計算できる．

$$\begin{aligned} \bigoplus_{\boldsymbol{x} \in \{X\}} \pi_u(\boldsymbol{x}) &= \bigoplus_{i=0}^{2^m-1} \pi_u(i) \mathrm{D2}_{\{X\}}(i) \\ &= (\boldsymbol{v}_j \boldsymbol{v}_k \cdots \boldsymbol{v}_l) \bullet \boldsymbol{w}_{\{X\}} \end{aligned} \tag{5.16}$$

ここで，i $(0 \leq i \leq 2^m - 1)$ は m ビットベクトル $\boldsymbol{x} \in GF(2)^m$ を 2 進数順に入力した値を示し，記号'$\bullet$' はベクトルの内積を示す．

以下の定理は，5.4 節で示した $\mathrm{RM}(r, m)$ 符号の r 次項行ベクトルの定義と，属性判定関数 $\pi_u(\boldsymbol{x})$ の定義から導出できる．

定理 4 i次単項式$\pi_u(\boldsymbol{x}) = x[j]x[k]\cdots x[l]$に対し，$XOR$総和$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x}) = 0$が成立することは，$i$次項行ベクトル$\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l$が$mod\ 2$頻度分布$\boldsymbol{w}_{\{X\}}$と直交することに等しい．従って，$(\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l)\bullet\boldsymbol{w}_{\{X\}} = 0$が成立する．記号'$\bullet$'はベクトルの内積を示す．

上記の定理を以下証明する．式(5.16)から，$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x})$はベクトル$(\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l)$とベクトル$\boldsymbol{w}_{\{X\}}$の内積に等しい．XOR総和$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x}) = 0$と仮定すれば，$i$次項行ベクトル$\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l$は$\bmod 2$頻度分布$\boldsymbol{w}_{\{X\}}$と直交する．

(証明終了)

5.5.2 Division属性とRM符号

Division属性とRM符号の関係について，以下の定理に纏める．

定理 5 m次元ベクトル$\boldsymbol{x}\in GF(2)^m$の集合を$\{X\}$とする．$d$次関数$f$に対し，積分特性$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x}) = 0$が成立することは，$2^m$次元ベクトル$\boldsymbol{w}_{\{X\}}$が$RM(m-d-1,m)$符号の符号語であることと等価である．

証明を以下に示す．集合$\{X\}$がd次関数fに対し積分特性を持つと仮定すれば，$\bigoplus_{\boldsymbol{x}\in\{X\}} f(\boldsymbol{x}) = 0$が成立し，集合$\{X\}$はDivision属性$\mathcal{D}^m_{d+1}$を与える．集合$\{X\}$のDivision属性が$\mathcal{D}^m_{d+1}$であるため，$(d+1)$次未満の任意の$\pi_u(\boldsymbol{x})$に対し，$\bigoplus_{\boldsymbol{x}\in\{X\}} \pi_u(\boldsymbol{x}) = 0$が成立する．式(5.16)から，集合$\{X\}$の$\bmod 2$頻度分布$\boldsymbol{w}_{\{X\}}$は，任意の$d$次以下の行ベクトル$\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l$と直交する．その様な$\boldsymbol{w}_{\{X\}}$はRM符号の符号語であり，$d$次以下の全ての行ベクトル$\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l$を並べた行列をパリティ検査行列$\boldsymbol{H}$とするRM符号の符号語である．$\boldsymbol{w}_{\{X\}}$は$(\boldsymbol{v}_j\boldsymbol{v}_k\cdots\boldsymbol{v}_l)\bullet\boldsymbol{w}_{\{X\}} = 0$を満たすので，$\boldsymbol{w}_{\{X\}}$はRM$(d,m)$符号の双対符号の符号語であり，RM$(m-d-1,m)$符号の符号語である．

(証明終了)

5.5.3 Division属性と最小平文数

平文入力m bitのd次関数fに対し，積分特性を持つ集合$\{X\}$について考える．最小な必要平文数$\#\{X\}$に関して$\#\{X_e\}\le\#\{X\}$が成立し，$\#\{X_e\}$は$\boldsymbol{w}_{\{X_e\}}$のハミング重みH_wで与えられる．

$$\#\{X\}\ge\#\{X_e\} = H_w(\boldsymbol{w}_{\{X_e\}}) \tag{5.17}$$

定理 5 から，$\boldsymbol{w}_{\{X_e\}}$ は RM$(m-d-1,m)$ の符号語である．定理 1 から，RM(r,m) 符号の最小重みは $d_{min}=2^{m-r}$ である．最小の必要平文数は 2^{d+1} である．これは，最小重み $d_{min}=2^{m-r}$ において，r を $m-d-1$ で置き換えたものである．平文入力 m bit の d 次関数 f に対し，積分特性を持つ平文集合 $\{X_e\}$ の種類数 $\#\{\{X_e\}\}$ は，定理 1 の最小重み d_{min} の符号語数で与えられる．上記の内容を以下に定理としてまとめる．

定理 6 平文入力 m *bit* の d 次関数 f に対し，積分特性を与える集合 $\{X\}$ において，必要な平文数の最小値は

$$min\ \#\{X\}=2^{d+1} \tag{5.18}$$

であり，本質的な集合 $\{X_e\}$ の種類数 $\#\{\{X_e\}\}$ は，以下の式で与えられる．

$$\#\{\{X_e\}\}=2^{m-d-1}\prod_{i=0}^{d}\frac{(2^{m-i}-1)}{(2^{d+1-i}-1)} \tag{5.19}$$

平文数の観点で，必要となる平文数が最小となる積分特性を最良の積分特性であると考えると，以下の定理が言える．

定理 7 最良の積分特性は，高階差分特性に等しい．

証明は以下の通りである．任意の集合 $\{X\}$ は，本質的な集合 $\{X_e\}$ で置き換えても，同じ積分特性を与える．式 (5.3) と式 (5.18) から，積分特性において，必要平文数の最小値は，高階差分の必要平文数の最小値に等しい．また，式 (5.7) と (5.19) から，積分特性において，最小の平文数を与える集合の種類数は，高階差分の最小必要平文数の種類数に等しいこと．これらにより成立する．

(証明終了)

5.6 Division 属性を用いた積分特性と高階差分特性の比較

Division 属性は暗号 E のブール代数次数を見積もる簡易な手法であるが，Division 属性を用いて発見した積分特性が最良か不明である．従って，KASUMI と MISTY を用いて，既存の高階差分特性と Division 属性を用いて発見した積分特性の比較を行う．

5.6.1 KASUMI の特性比較

KASUMI に対する高階差分特性として，田中らが示した 32 階差分特性 [61]，3.4 節で示す発見的手法により見つけた 16 階差分特性，及びその 16 階差分特性を平文側に 1 段拡張した 48 階差

分特性 [43] が存在する事が知られている．これらの高階差分特性と Division 属性を用いて導出した積分特性の比較結果を表 5.3 に示す．

表 5.3: KASUMI の積分特性と高階差分特性の比較

段数	入力特性 (積分)	入力特性 (高階差分)	出力特性
	$\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7,7,2,7,7,2,7}$		
4	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,0,0,0,0,7,2,7,0,0,0]}$	16 階 $(0,0,0,0,0,0,7,2,7,0,0,0)$, 3.4 節	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$
4	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,0,0,0,0,7,2,7,7,2,7]}$	32 階 $(0,0,0,0,0,0,7,2,7,7,2,7)$, [61]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$
5(2-6)	$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[7,2,7,0,0,0,7,2,7,7,2,7]}$	48 階 $(7,2,7,0,0,0,7,2,7,7,2,7)$, [43]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$

5 段特性の対象は 2 段目〜6 段目である．高階差分特性の入力特性で'0' は任意の定数を示し，'0' 以外は変数箇所と変数ビット数を示す．

表 5.3 において，4 段の出力特性 $(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$ を得る為に，入力特性 (積分) は Division 属性 $\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,0,0,0,0,7,2,7,0,0,0]}$ を有する平文集合 $\{P\}$ を用意する必要があり，その選択平文数は 2^{16} である．同様の 4 段特性を高階差分特性で得る為には，入力特性 (高階差分) は平文 $P[63-0]$ の内，$P[31-16]$ の 16 bit を変数とする 16 階差分を与える必要があり，その選択平文数は 2^{16} である．従って，Division 属性を用いて導出した積分特性と高階差分特性において，同一の 4 段の出力特性を得る為に必要な選択平文数は，一致している事が分かる．その他，32 階差分，48 階差分特性においても，同一の出力特性を得る為に必要な選択平文数は，一致している事が分かる．

5.6.2 MISTY(MISTY1，MISTY2) の特性比較

前節と同様に，MISTY1 と MISTY2 に関する既知の高階差分特性と，Division 属性を用いて導出した積分特性の比較結果を表 5.4 と表 5.5 に示す．

表 5.4: MISTY1 の積分特性と高階差分特性の比較

段数	FL	入力特性 (積分) $\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7,7,2,7,7,2,7}$	入力特性 (高階差分)	出力特性
4	-	検出不可	7 階 $(0,0,0,0,0,0,0,0,0,0,0,0,7)$, [48]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},$0x6d$,\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$
4	✓	$\mathcal{D}_{[0,0,0,0,0,0,0,0,0,7,0,0,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	14 階 $(0,0,0,0,0,0,0,0,0,7,0,0,7)$, [16]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$
4	✓	$\mathcal{D}_{[0,0,5,0,0,5,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	38 階 $(0,0,3,0,0,3,7,2,7,7,2,7)$, [63]	$(\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$
4	✓	$\mathcal{D}_{[0,0,7,0,0,7,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	44 階 $(0,0,6,0,0,6,7,2,7,7,2,7)$, [63]	$(\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$
5	-	$\mathcal{D}_{[0,0,0,0,0,1,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	32 階 $(0,0,0,0,0,0,7,2,7,7,2,7)$, [18]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$
5	-	$\mathcal{D}_{[0,0,0,0,0,4,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	36 階 $(0,0,0,0,0,4,7,2,7,7,2,7)$, [40]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U})$
5	-	$\mathcal{D}_{[7,2,6,0,0,0,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	47 階 $(7,2,6,0,0,0,7,2,7,7,2,7)$, [40]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U})$
5	-	$\mathcal{D}_{[7,2,7,0,0,0,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	48 階 $(7,2,7,0,0,0,7,2,7,7,2,7)$, [40]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$
5	✓	$\mathcal{D}_{[7,2,0,7,2,0,7,2,7,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	50 階 $(7,2,0,7,2,0,7,2,7,7,2,7)$, [40]	$(\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{U},\mathcal{U})$

記号 ✓ は FL 関数が有る場合を示す.

表 5.5: MISTY2 の積分特性と高階差分特性の比較

段数	FL	入力特性 (積分) $\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7,7,2,7,7,2,7}$	入力特性 (高階差分)	出力特性
6	-	検出不可	7 階 $(0,0,0,0,0,0,0,0,0,0,0,0,7)$, [17]	$($0x6d$,\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$
7	-	$\mathcal{D}_{[7,2,7,7,2,7,0,0,0,7,2,7]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	32 階 $(7,2,7,7,2,7,0,0,0,0,0,0)$, [19]	$(\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$
7	✓	$\mathcal{D}_{[7,2,7,7,2,7,0,0,0,0,0,0]}^{7,2,7,7,2,7,7,2,7,7,2,7}$	32 階 $(7,2,7,7,2,7,0,0,0,0,0,0)$, [57]	$(\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U})$

記号 ✓ は FL 関数が有る場合を示す.

MISTY1，及び MISTY2 の高階差分特性は，発見的手法で見つけた特性である．KASUMI の場合と同様に，Division 属性を用いて導出した積分特性と高階差分特性において，同一の出力特性を得る為に必要な選択平文数は，概ね一致している．尚，一部の結果に差が存在し，同一の出力特性を得る為に必要な選択平文数で比較した時，高階差分特性の方が Division 属性を用いて導出した積分特性に比べ，必要な選択平文数が少ない場合がある．以下に一例を示す．FL 関数の有る MISTY1 に対

し，4段の出力特性の上位7 bitがBALANCEとなる$(\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$を得る為に必要な入力特性(積分)は，Division属性$\mathcal{D}^{7,2,7,7,2,7,7,2,7,7,2,7}_{[0,0,5,0,0,5,7,2,7,7,2,7]}$を有する平文集合$\{P\}$を用意する必要があり，その選択平文数は$2^{42}$である．一方，同じ4段の出力特性$(\mathcal{B},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{U},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B},\mathcal{B})$を得る為に必要な入力特性(高階差分)は，38階差分で十分であり，必要な選択平文数は2^{38}である．

KASUMIとMISTY(MISTY1，MISTY2)に対してDivision属性を用いた積分特性と高階差分特性の比較検証を実施した．その結果，KASUMIに対してはDivision属性を用いた積分特性と高階差分特性が一致する事を明らかにした．また，MISTY(MISTY1，MISTY2)に対しては，Division属性を用いた積分特性と高階差分特性が概ね一致する事を明らかにした．尚，一部の特性比較結果において差がある事から，最良の積分特性は高階差分特性と一致する事を踏まえると，Division属性の探索法に改良の余地が見込まれる．

5.7 纏め

本節では，高階差分攻撃と積分攻撃の関係性を明らかにした．具体的には，m bit入力のd次関数に対し，Division属性から導出される最良の積分攻撃は，高階差分攻撃の$(d+1)$階差分と等しく，2^{d+1}の選択平文数を必要とすることを示した．また，2^{d+1}の選択平文数の選び方は$\#\{\{X_e\}\} = 2^{m-d-1}\prod_{i=0}^{d}\frac{(2^{m-i}-1)}{(2^{d+1-i}-1)}$通りであり，必要な選択平文数を最小とする積分攻撃は高階差分攻撃と一致する事を示した．

5.6節では，KASUMIとMISTY(MISTY1，MISTY2)に対してDivision属性を用いた積分特性と高階差分特性の比較検証を実施した．その結果，KASUMIに対してはDivision属性を用いた積分特性と高階差分特性が一致する事を明らかにした．また，MISTY(MISTY1，MISTY2)に対しては，両者の特性は発見的手法で見つけた特性であるが，概ね一致する事を明らかにした．また，両者の特性比較結果に一部差がある事から，最良の積分特性は高階差分特性と一致する事を踏まえると，Division属性の探索法に改良の余地が見込まれる．

第6章　KASUMIの安全性評価

6.1　概要

本章では，これまで本論文で示したKASUMIの鍵回復攻撃に必要な計算量に対し，現在，及び今後の計算機能力の向上を見込んだ実行可能性に基づく評価を提案し，5段のKASUMIは汎用PCで実行可能である事を示す．また，7段のKASUMIは，現在の世界Top500位のスーパーコンピューターが有する計算能力で実行可能な計算量である事を示す．

6.2　従来法と問題提起

6.2.1　従来法

従来，共通鍵暗号における暗号解読の定義は「全数探索法よりも少ない計算量と平文数で攻撃できる効率的な手法 (ショートカット法) が発見された状態」である．例えば，秘密鍵長が 128 bit でブロック長が 64 bit の共通鍵ブロック暗号を想定した場合，フルラウンドの暗号に対して計算量 $T < 2^{128}$ であり，且つ，平文数 $D < 2^{64}$ で実行可能なショートカット法が発見された場合，その共通鍵ブロック暗号は解読された事になる．

1990 年代から 2000 年の初頭にかけて，共通鍵ブロック暗号の秘密鍵に対し，全数探索法の解読コンテストが開催された．その結果，DES Challenge (III) では，DES 解読用の専用ハードウェアとインターネット上の約 10 万台の PC の共同作業により，DES の秘密鍵 56 bit を約 22 時間で解読できる事が示された [38]．また，RC5 - 64 Challenge では，インターネット上の数万台の PC の共同作業により，RC5 の秘密鍵 64 bit が 4 年で解読できる事が示された [39]．

6.2.2　問題提起

秘密鍵長が 128 bit でブロック長が 64 bit の共通鍵ブロック暗号に対し，フルラウンドの暗号に対して計算量 $T < 2^{128}$ であり，且つ，平文数 $D < 2^{64}$ で実行可能なショートカット法が見つかった場合，その暗号は解読された事になる．しかしながら，当該暗号を搭載した実際のシステムに対する影響を考慮する為には，上記のショートカット法の計算機による実行可能性を踏まえ，当該暗号の利用可能期間を見積もる必要がある．

6.3 NISTの推奨鍵長

NISTは異なる暗号アルゴリズムと鍵長に対し，同程度の安全性(等価安全性)を提示している[35]．また，将来に向けて，各種暗号アルゴリズムがどの程度の鍵長を有すれば安全に利用できるか公表している(表6.1参照)．表6.1において，等価安全性を用いると，共通鍵暗号アルゴリズムにおける秘密鍵長80 bitで実現される安全性は，公開鍵暗号アルゴリズム(素因数分解問題の困難性に基づく方式，代表例としてRSA方式がある．)の鍵長1,024 bitで実現される安全性と等価である事がわかる．また，表6.1から，共通鍵暗号アルゴリズムに対し，2030年迄の使用を想定する場合，秘密鍵長は112 bit程度あれば良い事が分かる．

表 6.1: NISTの推奨鍵長

使用可能期限	等価安全性	共通鍵暗号	公開鍵暗号
			素因数分解問題に基づく方式(RSA等)
2010年迄	80bit	80bit	1,024bit
2011～2030年	112bit	112bit	2,048bit
2031年以降	128bit以上	128bit以上	3,072bit以上

6.4 KASUMIの安全性評価

6.4.1 従来法の問題点

従来の暗号評価では，暗号解読が解読された時点で，当該暗号は危険と判断していた．安全性の指標となるのは，全数探索法よりも少ない計算量，例えば，秘密鍵長が128 bitでブロック長が64 bitの共通鍵ブロック暗号を想定した場合，フルラウンドの暗号に対して計算量$T < 2^{128}$であり，且つ，平文数$D < 2^{64}$で実行可能なショートカット法が発見された場合，その共通鍵ブロック暗号は解読可能という事になる．ここで，ショートカット攻撃に必要な計算量と平文数の条件は，計算量$T < 2^{128}$と平文数$D < 2^{64}$であり，その値の大小(例えば，計算機で実行可能かどうか，実行した場合に解読に何年かかるか)は重要視されていない事が多い．

移動体通信システム・サービスを提供する立場で実システムに暗号アルゴリズムを組み込む場合，当該暗号がどの程度の期間，安全に利用可能であるかを把握する事が重要である．

6.4.2 本論文の提案法

本論文では，上記の従来法の問題点を鑑み，当該暗号の攻撃に必要な計算量に対し，現在，及び今後の計算機能力の向上を見込んだ実行可能性に基づく評価の提案を行う．具体的には，今後の計算機能力の向上については，スーパーコンピューターのベンチマーク結果の 1 位から 500 位を 1993 年から半年毎に集計している Web サイト TOP500.Org[51] と CRYPTREC が公表しているレポート [11] を参考にする．以下に，CRYPTREC Report 2015 で示されている公開鍵暗号アルゴリズム (素因数分解問題の困難性に基づく方式) で,「1 年間でふるい処理を完了するのに要求される処理能力の予測 (2016 年 2 月更新)」を抜粋し，図 6.1 示す．

表 6.1 に示す NIST の等価安全性から，共通鍵暗号の秘密鍵 80 bit，112 bit，128 bit 以上の安全性は，公開鍵暗号（RSA）の 1024 bit，2048 bit，3072 bit 以上が提供する安全性と等価である．そこで本論文では，公開鍵暗号（RSA）の 公開鍵 $pk = 1024, 2048, 3072$ bit に対して 1 年間でふるい処理を完了するのに要求される計算機処理能力を，共通鍵暗号の秘密鍵 $K = 80, 112, 128$ bit に対して全数探索法を完了するのに要求される計算機処理能力と同じものと見なす．本論文の提案法を用いる事により，実際の計算機を用いた解読可能時間に基づき，対象暗号の安全な利用可能期間の見積もりが可能となる事が期待される．

6.4.3 提案法による評価

本論文で示した KASUMI の鍵回復攻撃に必要な計算量に対し，上記の提案法を用いて，現在，及び今後の計算機能力の向上を見込んだ実行可能性に基づく KASUMI の安全性に関する評価を行う．

第 3 章で示した 5 段 KASUMI の鍵回復攻撃に必要な計算量は $T = 2^{31.5}$ であり，この程度であれば汎用 PC の計算機能力で実行可能である．また，図 6.1 から，2016 年時点で世界 Top500 の上位スーパーコンピューターは RSA 1024 bit を一年間で解読可能な処理能力を有している事が分かる．本論文では，NIST の等価安全性を用いて RSA 1024 bit を 1 年間でふるい処理を完了するのに要求される計算機処理能力を，共通鍵暗号の秘密鍵 $K = 80$ bit に対して全数探索法を完了するのに要求される計算機処理能力（$T = 2^{80}$）と同じものと見なしている．第 4 章で示した 7 段 KASUMI の鍵回復攻撃に必要な計算量は $T = 2^{63.3} \ll 2^{80}$ である為，現在のスーパーコンピューターが有する計算能力で実行可能な計算量である．従って，実際のシステムにおいて，7 段以下の KASUMI を用いる事は，秘密鍵に対する鍵回復攻撃の影響を受ける可能性がある為，危険である．KASUMI の既存研究結果と本論文の結果の比較を表 4.7 に示す．尚，暗号解読技術は日々進化している為，今後も新たな攻撃手法に対する耐性を評価してく事が重要である．

(注意) 下記は **CRYPTREC Report 2015 [11]，頁 26 の図 3.1 を抜粋したものである.**

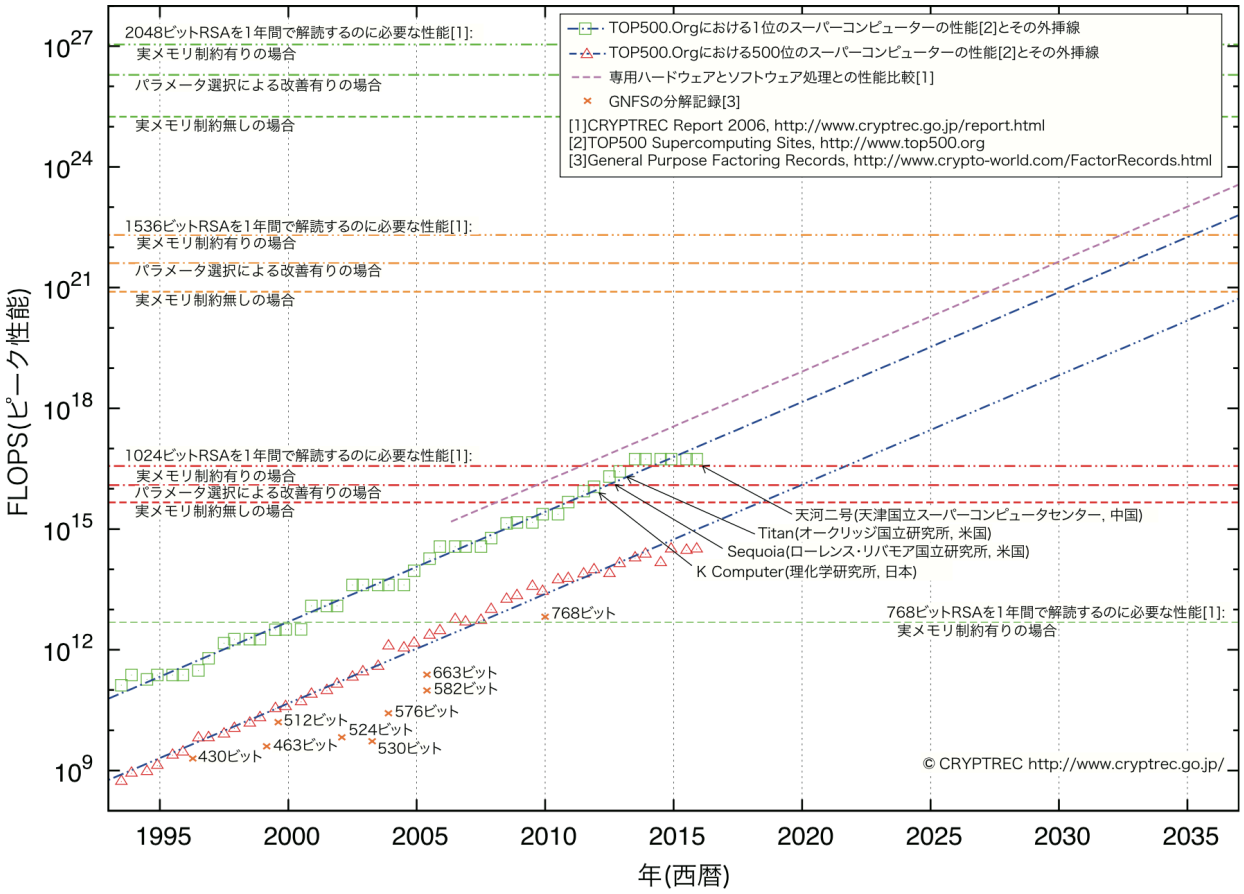

図 6.1: 1 年間でふるい処理を完了するのに要求される処理能力の予測 (2016 年 2 月更新)

6.5 纏め

本章では，これまで本論文で示したKASUMIの鍵回復攻撃に必要な計算量に対し，現在，及び今後の計算機能力の向上を見込んだ実行可能性に基づく評価を提案した．具体的には，今後の計算機能力の向上については，スーパーコンピューターのベンチマーク結果の 1 位から 500 位を 1993 年から半年毎に集計しているWebサイトTOP500.Org[51]とCRYPTRECが公表しているレポート [11] を参考にした．5段のKASUMIを攻撃する為に必要な計算量は$T = 2^{31.5}$回の暗号化計算量 [ENC.] であり，この計算量は汎用PCの計算機能力で実行可能である．また，7段のKASUMIを攻撃する為に必要な計算量は$T = 2^{63.3}$回の暗号化計算量 [ENC.] であり，この計算量は現在の世界Top500位のスーパーコンピューターが有する計算能力で実行可能な計算量である．従って，実際のシステムにおいて，7段以下のKASUMIを用いる事は，秘密鍵に対する鍵回復攻撃の影響を受ける可能性がある為，危険である．尚，暗号解読技術は日々進化している為，今後も新たな攻撃手法に対する耐性を評価してく事が重要である．

第7章　結論

本論文の結論を以下に纏める．第 3 章では，高階差分攻撃の高速化技法の提案と，その性能評価について纏めた．まず初めに，高階差分特性の探索において，高階差分は使用する階数が高い程選択平文数が増加する為，非線形関数 S-box(S7，S9) の入出力サイズに合わせた変数探索法を提案した．計算機による探索の結果，16 階差分を用いた新たな 4 段特性が存在する事を発見した．16 階差分を用いる事で，4 段特性を一組構成する為に必要な選択平文数を，従来の 2^{32} から 2^{16} まで削減する事に成功した．

続いて，5 段 KASUMI の段鍵 82 bit を鍵回復する部分 (攻撃方程式を解く部分) において，線形化手法の高速化技法を提案した．従来の線形化手法は全数探索法よりも効果的であるが，段関数を用いて係数行列を導出する必要あった．そこで，係数行列が暗号文の多項式で表現可能な事に着目し，事前にブール展開式を解析する事で，段関数を使わず係数行列を導出する高速化技法 1 を提案した．高速化技法 1 は，係数行列 $\mathbf{A}$ のサイズが小さい場合に効果的な技法である．高速化技法 1 を用いる事で，5 段 KASUMI の鍵回復攻撃を選択平文数 $D = 2^{27.5}$ と計算量 $T = 2^{34}$ 回の暗号化計算量 [ENC.] で行う事が可能である事を示した．また，未知項数 L が多い場合，線形化方程式をガウス・ジョルダン法で解く際の計算量が支配的となる場合がある．この問題を解決する為，未知項数の係数を 0 に制御し，消去する高速化技法 2 を提案した．高速化技法 2 を用いる場合，5 段 KASUMI の鍵回復攻撃を選択平文数 $D = 2^{28.6}$ と計算量 $2^{31.5}$ 回の暗号化計算量 [ENC.] で行う事が可能である事を示した．

高速化技法 1，及び高速化技法 2 を用いる事により，田中らの従来法と比較し，選択平文数は $2^{10.8}$ 程度削減可能である．また，鍵回復に必要な計算量は $2^{85.5}$ 倍高速化が可能である．尚，本論文で示した高速化技法 2 を用いた 5 段 KASUMI の鍵回復攻撃は，攻撃に必要な計算量が最小という点で，最良の結果 [1] である．

第 4 章では，2015 年に新たに提案された積分特性を効率的に発見する手法（Division 属性）を用いた積分特性探索と，新たに発見した特性を用いた積分攻撃に対する評価を実施した．Division 属性を用いた積分特性の探索において，KASUMI の非線形関数（S7，S9）の Division 属性伝搬を調査し，その結果と Division 属性の伝搬ルールを組合わせる事により，FI 関数・FO 関数の Division 属性伝搬を解析した．また，鍵依存線形関数である FL 関数内に存在する 1 ビット左巡回シフト

[1] 2016 年 9 月時点において

の影響を考慮した属性伝搬を構築し，FO 関数・FL 関数の伝搬特性を組合わせる事で，KASUMI の Division 属性伝搬を解析した．その結果，1 段目から開始した場合は 4 段の積分特性が存在し，2 段目から開始した場合は 5 段の積分特性が存在する事を明らかにした．

前述の 4 段積分特性を用いた場合，6 段の KASUMI(攻撃対象は 1 段目から 6 段目) が選択平文数 $D = 2^{57}$ と計算量 $T = 2^{58}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示した．6 段 KASUMI の攻撃では，攻撃方程式を bit 単位に分割し，FL 関数内の OR 演算を行う段鍵 (KL_{42}, KL_{62}) を 1 と仮定する事により，推定する段鍵の bit 数を 123 bit から 50 bit に削減する事で，攻撃に必要となる選択平文数と計算量の両方を削減した．また，攻撃方程式を解く際には線形方程式を並列に解く事により，6 段の KASUMI の鍵回復攻撃の成功確率をほぼ $1(p = 0.925)$ にした．また，前述の 5 段積分特性を用いた場合，7 段の KASUMI(攻撃対象は 2 段目から 8 段目) が選択平文数 $D = 2^{63}$ と計算量 $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] で攻撃可能である事を示した．7 段 KASUMI の攻撃では，KASUMI の鍵スケジュール部を活用し，且つ FL 関数内の OR 演算に着目した発生確率が $p = 2^{-18}$ の弱鍵を用いる事で，秘密鍵 110 bit を回復した．鍵回復攻撃では，鍵を推定して偽鍵を篩にかける際に，攻撃方程式の値に関して中間一致法を適用する事で，計算量を削減した．弱鍵を用いた 7 段の KASUMI に対する攻撃に関して，本論文の手法は攻撃に必要な計算量が最も少ないという点で，最良の結果 [2] である．

第 5 章では，リード・マラー符号に関する先行研究の成果を活用し，最良の積分特性は高階差分特性と一致する事を証明した．また，KASUMI と MISTY（MISTY1，MISTY2）に対して，Division 属性を用いた積分特性と高階差分特性の比較検証を実施した．その結果，KASUMI に対しては Division 属性を用いた積分特性と高階差分特性が一致する事を明らかにした．また，MISTY（MISTY1，MISTY2）に対しては，両者の特性は発見的手法で見つけた特性であるが，概ね一致する事を明らかにした．また，両者の特性比較結果に一部差がある事から，最良の積分特性は高階差分特性と一致する事を踏まえると，Division 属性の探索法に改良の余地が見込まれる事を明らかにした．

第 6 章では，本論文の評価結果を加味し，共通鍵ブロック暗号 KASUMI の安全性を評価した．具体的には，今後の計算機能力の向上については，スーパーコンピューターのベンチマーク結果の 1 位から 500 位を 1993 年から半年毎に集計している Web サイト TOP500.Org[51] と CRYPTREC が公表しているレポート [11] を参考にした．5 段の KASUMI を攻撃する為に必要な計算量は $T = 2^{31.5}$ 回の暗号化計算量 [ENC.] であり，この計算量は汎用 PC の計算機程度で実行可能である．また，7 段の KASUMI を攻撃する為に必要な計算量は $T = 2^{63.3}$ 回の暗号化計算量 [ENC.] であり，この計算量は現在の世界 Top500 位のスーパーコンピューターが有する計算能力で実行可能な計算量

[2] 2016 年 9 月時点において

である事を示した．従って，実際のシステムにおいて，7段以下のKASUMIを用いる事は，秘密鍵に対する鍵回復攻撃の影響を受ける可能性がある為，危険である．尚，暗号解読技術は日々進化している為，今後も新たな攻撃手法に対する耐性を評価してく事が重要である．

参考文献

[1] 3GPP. “Specification of the 3GPP confidentiality and integrity algorithms; Document 2: Kasumi specification”, http://www.3gpp.org/specifications/60-confidentiality-algorithms

[2] S. Babbage and L. Frisch, “On MISTY1 higher order differential cryptanalysis”, Proc. Third International Conference Information Security and Cryptology (ICISC 2000), LNCS 2015, pp.22-36, Seoul, Korea, December, 2000.

[3] A. Bar On, “Improved Higher-Order Differential Attacks on MISTY1”, Proc. 22nd International Workshop on Fast Software Encryption (FSE 2015), LNCS 9814, pp.28-47, Istanbul, Turkey, March, 2015.

[4] A. Bar On and N. Keller, “A 2^{70} Attack on the Full MISTY1”, Proc. 36th Annual International Cryptology Conference (CRYPTO 2016), pp.435-456, Santa Barbara, California, USA, August, 2016.

[5] E. Biham and A. Shamir, “Differential Cryptanalysis of the Data Encryption Standard”, Springer-Verlag New York, pp.79-88, 1993.

[6] E. Biham, A. Biryukov, and A. Shamir, “Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differentials”, Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT ’99), LNCS 1592, pp.12-23, Prague, Czech Republic, May, 1999.

[7] E. Biham, O. Dunkelman, and N. Keller, “A Related-Key Rectangle Attack on the Full KASUMI”, Proc. 11th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2005), LNCS 3788, pp.443-461, Chennai, India, December, 2005.

[8] A. Bogdanov and M. Wang, “Zero correlation linear cryptanalysis with reduced data complexity”, Proc. 19th International Workshop on Fast Software Encryption (FSE 2012), LNCS 7549, pp.29-48, Washington DC, USA, March, 2012.

[9] A. Bogdanov, G. Leander, K. Nyberg, and M. Wang, “Integral and multidimensional linear distinguishers with correlation zero”, Proc. 18th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2012), LNCS 7658, pp.244-261, Beijing, China, December, 2012.

[10] Cryptography Research and Evaluation Committees, “電子政府における調達のために推奨すべき暗号リスト（CRYPTREC 暗号リスト）”, http://www.cryptrec.go.jp/list.html

[11] Cryptography Research and Evaluation Committees, “CRYPTREC Report 2015 ”, https://www.cryptrec.go.jp

[12] J. Daemen, L. Knudsen, and V. Rijmen, “The block cipher Square”, Proc. 4th International Workshop on Fast Software Encryption (FSE ’97), LNCS 1267, pp.149-165, Haifa, Israel, January, 1997.

[13] O. Dunkelman and N. Keller, “An Improved Impossible Differential Attack on MISTY1”, Proc. 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT 2008), LNCS 5350, pp.441-454, Melbourne, Australia, December, 2008.

[14] O. Dunkelman, N. Keller, and A. Shamir, “A Practical-Time Related-Key Attack on the KASUMI Cryptosystem Used in GSM and 3G Telephony”, Proc. 30th Annual Cryptology Conference (CRYPTO 2010), LNCS 6223, pp.393-410, Santa Barbara, California, USA, August, 2010.

[15] N. Ferguson, J. Kelsey, S. Luck, B. Schneier, M. Stay, D. Wagner, and D. Whiting, “Improved Cryptanalysis of Rijndael”, Proc. 7th International Workshop on Fast Software Encryption (FSE 2000), LNCS 1978, pp.213-230, New York, USA, April, 2000.

[16] Y. Hatano, H. Tanaka, and T. Kaneko, “Optimization for the algebraic method and its application to an attack of MISTY1”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E87-A, no.1, pp.18-27, January, 2004.

[17] Y. Hatano, H. Tanaka, and T. Kaneko, “Higher Oder Differential Attack of MISTY2 without FL Functions”, Proc. International Conference on Fundamentals of Electronics, Communications and Computer Sciences, section 17, pp.6-10, Japan, March, 2002.

[18] Y. Igarashi and T. Kaneko, "The 32nd-Order Differential Attack on MISTY1 without FL Functions", Proceedings, International Symposium on Information Theory and its Applications (ISITA 2008), No.W-TI-4-4, pp.1503-1508, December, Aucland, New Zealand, 2008.

[19] Y. Igarashi, T. Kaneko, Y. Eguchi, T. Murai, R. Sueyoshi, Y. Hashiguchi, S. Fukushima, and T. Hachino, "The Improved 32nd-Order Differential Attack on 8 Rounds of MISTY2 without FL Functions", Proc. International Journal of Cyber-Security and Digital Forensics, pp.27-34, 2013.

[20] International Organization for Standardization : ISO/IEC 18033-3 Information technology - Security techniques - Encryption algorithms - Part 3: Block ciphers. 2010.

[21] T. Jakobsen and L. R. Knudsen, "The Interpolation Attack on Block Ciphers", Proc. 4th International Workshop on Fast Software Encryption (FSE '97), LNCS 1267, pp.28-40, Haifa, Israel, January, 1997.

[22] K. Jia, L. Li, C. Rechberger, J. Chen, and X. Wang, "Improved Cryptanalysis of the Block Cipher KASUMI", Proc. 19th International Conference on Selected Areas in Cryptography (SAC 2012), LNCS 7707, pp.222-233, Windsor, ON, Canada, August, 2012.

[23] K. Jia, C. Rechberger, and X. Wang, "Green Cryptanalysis: Meet-in-the-Middle Key-Recovery for the Full KASUMI Cipher". International Association for Cryptologic Research (IACR), Cryptology ePrint Archive: Report 2011/466.

[24] L.R. Knudsen and T.A. Berson, "Truncated Differentials of SAFER". Proc. 3rd International Workshop on Fast Software Encryption (FSE '96), LNCS 1039, pp.15-25, Cambridge, UK, February, 1996.

[25] L. R. Knudsen and D. Wagner, "Integral cryptanalysis". Proc. 9th International Workshop on Fast Software Encryption (FSE 2002), LNCS 2365, pp.112-127, Leuven, Belgium, February, 2002.

[26] H. Krawczyk, M. Bellare, and R. Canetti, "HMAC : Keyed-Hashing for Message Authentication", Request for Comments: 2104, 1997, `https://tools.ietf.org/html/rfc2104`

[27] U. Kühn. "Cryptanalysis of Reduced-Round MISTY". Proc. International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT 2001), LNCS 2045, pp.325-339, Innsbruck, Austria, May, 2001.

[28] X. Lai, "Higher Order Derivatives and Differential Cryptanalysis", Proc. Communications and Cryptography, pp.227-233, 1994.

[29] S.K. Langford and M.E. Hellman, "Differential-Linear Cryptanalysis", Proc. 14th Annual International Cryptology Conference (CRYPTO ' 94), LNCS 839, pp.17-25, Santa Barbara, California, USA, August, 1994.

[30] S. Lucks, "The Saturation Attack A Bait for Twofish", Proc. 8th International Workshop on Fast Software Encryption (FSE 2001), LNCS 2355, pp.1-15, Yokohama, Japan, April, 2001.

[31] J. Lu, W. Yap, and Y. Wei, "Weak keys of the full misty1 block cipher for related-key differential cryptanalysis". Proc. The Cryptographers ' Track at the RSA Conference 2013 (CT-RSA ' 13), LNCS 7779, pp.389-404, San Francisco, California, USA, 2013.

[32] F. J. MacWilliams and N. J. A. Slone, "The theory of Error-Correcting Codes", North-Holland, 1977.

[33] M. Matsui, "Linear Cryptanalysis Method for DES Cipher", Proc. Workshop on the Theory and Application of Cryptographic Techniques (EUROCRYPT '93), LNCS 765, pp.386-397, Lofthus, Norway, May, 1993.

[34] M. Matsui, "New Block Encryption Algorithm MISTY", Proc. 4th International Workshop on Fast Software Encryption (FSE '97), LNCS 1267, pp.54-67, Haifa, Israel, January, 1997.

[35] NIST, "Recommendation for Key March, 2007 Management Part 1: General (Revised)", National Institute of Standards and Technology (NIST), Special Publication 800-57 Part 1 Revision 4, 2007.

[36] NIST, "Advanced Encryption Standard (AES)", National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS) Publication 197, 2001.

[37] NIST. "Data Encryption Standard", National Institute of Standards and Technology (NIST), Federal Information Processing Standard (FIPS) Publication 46-3. 1977

[38] RSA Laboratories, "DES challenge III", `http://www. rsa.com/rsalabs/`

[39] RSA Laboratories, "The RSA laboratories secret-key challenge", `http://www.rsa.com/rsalabs/`

[40] R. Sato and T. Kaneko, "New higher order differential properties on block cipher MISTY1", Thesis for master's degree of the Tokyo University of Science, pp.8-21, Chiba, Japan, 2015.

[41] N. Shibayama and T. Kaneko, "A New Higher Order Differential of CLEFIA". IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E97-A, pp.118-126, January, 2014.

[42] T. Shimoyama, S. Moriai, T. Kaneko, and S. Tsujii, "Improving Higher Order Differential Attack and Its Application to Nyberg-Knudesen's Designed Block Cipher", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E82-A, no.9, pp.1971-1980, September, 1999.

[43] T. Saito, "A Single-Key Attack on 6-Round KASUMI". International Association for Cryptologic Research (IACR), Cryptology ePrint Archive: Report 2011/584.

[44] Y. Sasaki and L. Wang, "Meet-in-the-Middle Technique for Integral Attacks against Feistel Ciphers", Proc. 19th International Conference, Selected Areas in Cryptography (SAC 2012), LNCS 7707, pp.234-251, Windsor, ON, Canada, August, 2012.

[45] N. Sugio, H. Aono, S. Hongo, and T. Kaneko, "A Study on Higher Order Differential Attack of KASUMI". IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E90-A, no.1, pp.14-21, January, 2007.

[46] X. Sun and X. Lai, "Improved Integral Attacks on MISTY1", Proc. 16th Annual International Workshop, Selected Areas in Cryptography (SAC 2009), LNCS 5867, pp.266-280, Calgary, Alberta, Canada, August, 2009.

[47] M. Takeda, T. Hamaide, K. Hisamatsu and T. Kaneko, "Linear cryptanalysis by Linear Sieve Method", IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E81-A, no.1, pp.82-87, January, 1998.

[48] H. Tanaka, K. Hisamatsu, and T. Kaneko, "Strength of MISTY1 without FL Function for Higher Order Differential Attack", Proc. 13th International Symposium, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC 1999), LNCS 1719, pp.221-230, Honolulu, Hawaii, USA, November, 1999.

[49] Y. Todo, "Structural Evaluation by Generalized Integral Property", Proc. 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT 2015), LNCS 9056, part1, pp.287-314, Sofia, Bulgaria, April, 2015.

[50] Y. Todo, “Integral Cryptanalysis on Full MISTY1”, Proc. 35th Annual Cryptology Conference (CRYPTO 2015), LNCS 9215 Part1, pp.413-432, Santa Barbara, California, USA, August, 2015.

[51] TOP500, “TOP500 Supercomputer Sites ”, http://www.top500.org/

[52] Y. Todo and M. Morii, “Bit-Based Division Property and Application to Simon Family”, Proc. 23rd International Conference on Fast Software Encryption (FSE 2016), pp.357-377, Bochum, Germany, March, 2016.

[53] Y. Tsunoo, T. Saito, T. Kawabata, and H. Nakagawa, “Finding higher order differentials of misty1”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.95-A, no.6, pp.1049-1055, June, 2012.

[54] Y. Tsunoo, T. Saito, M. Shigeri, and T. Kawabata, “Security Analysis of 7-Round MISTY1 against Higher Order Differential Attacks”, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E93-A, no.1, pp.144-152, January, 2010.

[55] W. Yi and S. Chen, “Multidimensional zero-correlation linear cryptanalysis of the block cipher KASUMI”, Journals of IET Information Security, vol.10, Issue 4, pp.1-7, 2015.

[56] D. Wagner, “The Boomerang Attack”, Proc. 6th International Workshop on Fast Software Encryption (FSE ’99), LNCS 1636, pp.156-170, Rome, Italy, March, 1999.

[57] 岡田渓太, 角尾幸保, 峯松一彦, 洲崎智保, 下山武司, “8 ラウンド MISTY2 への高階差分攻撃”, 2010 年 暗号と情報セキュリティシンポジウム（SCIS2010）予稿集, 1D1-4-b, 高松市, 香川県, 1 月, 2010.

[58] 金子敏信, “[招待講演] 高階差分攻撃と積分攻撃”, 電子情報通信学会, 信学技報, vol. 115, no. 376, ISEC2015-58, pp.23-29, 東京都, 12 月, 2015.

[59] 総務省．“電気通信サービスの契約数及びシェアに関する四半期データの公表 （平成 28 年度第 2 四半期（9 月末））”, 2016, http://www.soumu.go.jp/menu_news/s-news/01kiban04_02000117.html

[60] “数式処理システム REDUCE”, http://sourceforge.net/projects/reduce-algebra/files/

[61] 田中秀磨，石井周志，金子敏信，“霞と MISTY の強度評価に関する一考察”, 2001 年 暗号と情報セキュリティシンポジウム 予稿集, 12A-1, pp.647-652, 大磯町, 神奈川県, 1 月, 2001.

[62] 南部俊一，金子敏信，“KASUMI の高階差分攻撃に関する一考察 (III)”, 第 27 回 情報理論とその応用シンポジウム（SITA2004）予稿集, pp.45-48, 下呂市, 岐阜県, 12 月, 2004.

[63] 望月拓良，金子敏信，“MISTY1 の新しい高階差分特性”, 第 29 回 暗号と情報セキュリティシンポジウム（SCIS2012）予稿集, 2C1-4, 金沢市, 石川県, 1 月, 2012.

付 録A　共通鍵暗号に対する代表的な攻撃方法

本論文の 2.4.2 節で述べた特徴量 $\mathcal{H}(X_R(i))$ を選ぶ際の代表的な攻撃方法に，差分攻撃 [5]，線形攻撃 [33] が知られている．本節では，これらの攻撃方法について説明する．

A.1　差分攻撃

差分攻撃 [5] は，入力を変化させた時に，出力の変化の分布に偏りがある場合に行える攻撃である．攻撃者の前提条件は，選択平文攻撃である．

g 関数を n bit 入出力関数 $g : \{0,1\}^n \rightarrow \{0,1\}^n$ とする．g 関数は S-box や図 2.10 の F_i 関数 $(1 \leq i \leq R+r)$ 等である．このとき，差分確率 DP_g と最大差分確率 DP_{max} は以下の式で定義される．

定義 4 (差分確率 DP_g，最大差分確率 DP_{max}) 入力差分 Δx と出力差分 Δy が与えられたとき，差分確率 DP_g と最大差分確率 DP_{max} は以下の式で定義される．

$$DP_g(\Delta x, \Delta y) = \frac{\#\{x \in \{0,1\}^n | g(x) \oplus g(x \oplus \Delta x) = \Delta y\}}{2^n} \tag{A.1}$$

$$DP_{max} = max_{\Delta x \neq 0, \Delta y} DP_g(\Delta x, \Delta y) \tag{A.2}$$

n が小さい場合は最大差分確率 DP_{max} を求めることができるが，n が大きい場合は計算量的に最大差分確率 DP_{max} を求めることが困難である．この様な場合，下記で定義する最大差分特性確率 DCP を用いる．

定義 5 (最大差分特性確率 DCP) i 段目の関数を g_i: $\{0,1\}^n \rightarrow \{0,1\}^n$ とし，合成関数を $g(x) = g_R \circ g_{R-1} \circ \cdots \circ g_2 \circ g_1(x)$ とする．最大差分特性確率 DCP は，1 段目の入力差分が Δx_0 で，且つ 1 段目の出力差分が Δx_1，$\cdots$，$(R-1)$ 段目の出力差分が Δx_{R-1} のとき，R 段目の出力差分が Δx_R となる確率の最大値である．

$$DCP = max_{\Delta x_0 \neq 0, \Delta x_1, \ldots, \Delta x_R} \prod_{i=1}^{R} DP_{g_i}(\Delta x_{i-1}, \Delta x_i) \tag{A.3}$$

図 2.10 に示す m bit 入力の暗号 E に対し，差分攻撃を用いて鍵回復を行う場合について考える．鍵回復は以下の 3 ステップで行う．

1. 事前処理：ショートカット部（1〜R段）の差分特性確率 DCP を計算する．事前に求めた差分特性確率 DCP のうち，入力差分 $\Delta P = P \oplus P^*$ と R 段目の出力差分 $\Delta X_R = X_R \oplus X_R^*$ が大きくなる場合を選び，この時の確率を ρ とする．

2. 暗号文の入手：選択平文 P を ρ^{-1} 個用意し，対応する暗号文 C を入手する．同様に，入力差分 ΔP を加えた平文 $P^*(= P \oplus \Delta P)$ に対応する暗号文 C^* を入手する．

3. 鍵回復：段鍵 $K_{R+1}, \ldots, K_{R+r}$ を仮定し，逆関数 F^{-1} を用いて下記が成立するか確認する．

$$\Delta X_R = F^{-1}(C; K_{R+1}, \ldots, K_{R+r}) \oplus F^{-1}(C^*; K_{R+1}, \ldots, K_{R+r}) \tag{A.4}$$

   仮定した段鍵 $K_{R+1}, \ldots, K_{R+r}$ の中で，式 (A.4) の成立回数が最も多い鍵を，正しい鍵候補として抽出する．

A.2 線形攻撃

線形攻撃 [33] は，入力の特定ビットの排他的論理和と出力の特定ビットの排他的論理和の間の相関関係（線形相関）を調べ，その相関関係を用いて最終段に入力される鍵を推定する攻撃である．攻撃者の前提条件は，既知平文攻撃である．

g 関数を n bit 入出力関数 $g : \{0,1\}^n \rightarrow \{0,1\}^n$ とする．g 関数は S-box や図 2.10 の F_i 関数 $(1 \leq i \leq R+r)$ 等である．このとき，線形確率 LP_g と最大線形確率 LP_{max} は以下の式で定義される．尚，記号 $\bullet$ は n bit ベクトル同士の内積演算を表す．

定義 6 (線形確率 LP_g，最大線形確率 LP_{max}) 入力マスク Γx と出力マスク Γy が与えられたとき，線形確率 LP_g と最大線形確率 LP_{max} は以下の式で定義される．

$$LP_g(\Gamma x, \Gamma y) = (2\frac{\#\{x \in \{0,1\}^n | x \bullet \Gamma x = g(x) \bullet \Gamma y\}}{2^n} - 1)^2 \tag{A.5}$$

$$LP_{max} = max_{\Gamma x, \Gamma y \neq 0} LP_g(\Gamma x, \Gamma y) \tag{A.6}$$

差分の場合と同様に，線形攻撃で用いる最大線形特性確率 LCP を以下の様に定義する．

定義 7 (最大線形特性確率 LCP) i 段目の関数を g_i: $\{0,1\}^n \rightarrow \{0,1\}^n$ とし，合成関数を $g(x) = g_R \circ g_{R-1} \circ \cdots \circ g_2 \circ g_1(x)$ とする．最大線形特性確率 LCP は，R 段目の出力マスクが Γx_R で，

且つ $(R-1)$ 段目の出力マスクが Γx_{R-1}，$\cdots$，*1*段目の出力マスクが Γx_1 のとき，*1*段目の入力マスクが Γx_0 となる確率の最大値である．

$$LCP = max_{\Gamma x_0,\ldots,\Gamma x_{R-1},\Gamma x_R \neq 0} \prod_{i=1}^{R} DP_{g_i}(\Gamma x_{i-1}, \Gamma x_i) \tag{A.7}$$

図 2.10 に示す m bit 入力の暗号 E に対し，線形攻撃を用いて鍵回復を行う場合について考える．鍵回復は以下の 3 ステップで行う．

1. 事前処理：ショートカット部（1〜R段）の線形特性確率 LCP を計算する．事前に求めた線形特性確率 LCP のうち，入力マスク ΓP と R 段目の出力マスク ΓX_R が大きくなる場合を選び，この時の確率を μ とする．

2. 暗号文の入手：既知平文 P と対応する暗号文 C を μ^{-1} 個用意する．

3. 鍵回復：段鍵 $K_{R+1}, \ldots, K_{R+r}$ を仮定し，逆関数 F^{-1} を用いて下記が成立するか確認する．

$$\Gamma X_R \bullet F^{-1}(C; K_{R+1}, \ldots, K_{R+r}) = \Gamma P \bullet P \tag{A.8}$$

偏差 | 式 $(A.8)$ の成立回数 $- 2^{m-1}$| が最大となる鍵を，正しい鍵候補として抽出する．

付 録B　高速化技法に必要な選択平文数と計算量

本節では，3.5 節で示した高速化技法 1 と高速化技法 2 で必要な選択平文数 D と計算量 T の導出方法について説明する．$(d+1)$ 階差分を用いて導出した攻撃方程式を線形化する際の未知項数を L とし，一組の攻撃方程式から n 本の線形方程式が得られるものとする．

B.1　高速化技法 1

高速化技法 1 は，攻撃者にとって既知の情報である攻撃方程式のブール展開式を事前に解析する事で，段関数を使わずに係数行列 $\mathbf{A}$ と定数ベクトル $\mathbf{b}$ を導出する手法である．高速化技法 1 は攻撃方程式中に存在する非線形関数の代数次数が小さく，未知項数が少ない場合に効果的である．高速化技法 1 のアルゴリズムを以下に再掲する．

1. $\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分入力に対応する暗号文を用意する．
2. $\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \leq i \leq \left\lceil \frac{L}{n} \right\rceil)$ を計算する．
3. 式 (3.16) を用いて n 次元ベクトル $\mathbf{a}_j, (1 \leq j \leq L+1)$ を計算する．
4. 得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する．

鍵回復攻撃に必要な選択平文数 D は，従来の線形化手法で必要な選択平文数と同じであり，以下の式で見積もる事ができる．

$$D = 2^{d+1} \times \left\lceil \frac{L}{n} \right\rceil$$

以下，それぞれのステップで必要な計算量の見積もりについて説明する．

ステップ 1 は，$\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分入力を用意し，平文入力に対応する暗号文を入手する為の計算量である．暗号文を得る為に必要な計算量は，$\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分入力を暗号化す

る計算量である．以上から，ステップ 1 に必要な計算量 T_1 は

$$T_1 = 2^{d+1} \times \left\lceil \frac{L}{n} \right\rceil$$

である．計算量の単位は T_1 回の暗号化計算量 [ENC.] である．

ステップ 2 は，$\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \leq i \leq \left\lceil \frac{L}{n} \right\rceil)$ を計算する為の計算量である．ここでベクトル $\mathbf{c}$ は，要素に暗号文の一次項から高次項を要素に持つベクトルであり，C 次元ベクトルとする．テーブル G を n bit の暗号文から C 次元ベクトル $\mathbf{c}$ の要素を出力するテーブルとし，ベクトル $\mathbf{c}$ をテーブル G を用いて導出する (図 3.2 参照)．テーブル G の実装に 32 bit プロセッサーを用いた場合，ベクトル $\mathbf{c}$ の値を全て得る為には $\left\lceil \frac{\mathrm{C}}{32} \right\rceil$ 回のテーブル G の参照が必要である．一組の $(d+1)$ 差分から C 次元ベクトル $\mathbf{c}$ のデータを得る為に必要な計算量は $2^{(d+1)} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil$ 回のテーブル G の参照である．以上から，ステップ 2 で必要な計算量 T_2 は

$$T_2 = 2^{d+1} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times \left\lceil \frac{L}{n} \right\rceil$$

回のテーブル G の参照に必要な計算量である．尚，本論文では，一回のテーブル G の参照に必要な計算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 3 は，式 (3.16) を用いて n 次元ベクトル $\mathbf{a}_j, (1 \leq j \leq L+1)$ を計算する為の計算量である．式 (3.16) に存在する $\mathbf{A}_j$ は $n \times \mathrm{C}$ サイズの行列であり，式 (3.16) は行列 $\mathbf{A}_j$ の n 個の行ベクトルと C 次元ベクトル $\mathbf{c}_i$ の内積演算である．内積演算に必要な計算量は $2 \times \mathrm{C} \times n$ 回のビット演算である．実装に 32 bit プロセッサーを用いて $(L+1)$ 種類の異なる $\mathbf{A}_j, (1 \leq j \leq L+1)$ に対して同様のビット演算を行う場合，計算量は $2 \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times n \times (L+1)$ 回の 32 bit プロセッサー演算量である．$\left\lceil \frac{L}{n} \right\rceil$ 組の $(d+1)$ 階差分に対し上記を繰り返す為，ステップ 3 で必要な計算量 T_3 は

$$T_3 = 2 \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times n \times (L+1) \times \left\lceil \frac{L}{n} \right\rceil$$

回の 32 bit プロセッサー演算量である．尚，本論文では，一回の 32 bit プロセッサー演算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 4 は得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する為に必要な計算量である．通常，ガウス・ジョルダン法の計算量は $O(L^3)$ のビット演算量で見積もられ

る．仮に 32 bit プロセッサーを用いた場合，ガウス・ジョルダン法に必要な計算量は

$$T_4 = \left\lceil \frac{L^3}{32} \right\rceil$$

回の 32 bit プロセッサー演算量である．尚，本論文では，一回の 32 bit プロセッサー演算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 2 からステップ 4 に必要な計算量の単位は，一回の S-box テーブル参照に必要な計算量で見積もっている．従って，暗号化計算量 [ENC.] で評価する際には，暗号に存在する S-box の個数 $\#S$ で割る必要がある事に注意が必要である．以上より，高速化技法 1 で必要な全体の計算量 T は

$$T = T_1 + \frac{T_2 + T_3 + T_4}{\#S}$$

回の暗号化計算量 [ENC.] で見積もる事ができる．

B.2 高速化技法 2

高速化技法 2 は，線形化した方程式中に存在する段鍵 K_{R+1} に関する未知項数 L の係数を 0 に制御する事で，求める未知項数を削減する手法である．段鍵 K_{R+1} の高次項の係数は暗号文の低次である事に着目し，事前に未知項数を減らす事で高速化が可能であり，未知項数が多い場合に効果的である．高速化技法 2 のアルゴリズムを以下に再掲する．

1. Q 組の $(d+1)$ 階差分入力に対応する暗号文を用意する．
2. Q 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \le i \le Q)$ を計算する．
3. 適切な $e_i \in \{0, 1\}$ を選ぶ事により，$\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ とする．
4. 式 (3.22) を用いて n 次元ベクトル $\mathbf{a}_j, (L' + 1 \le j \le L + 1)$ を計算する．
5. 得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する．

高速化技法 2 では，Q 組の $(d+1)$ 階差分を用意する必要がある．以上から，鍵回復攻撃に必要な選択平文数 D は，以下の式で見積もる事ができる．

$$D = 2^{d+1} \times Q, \ (Q = \mathrm{C}' + \left\lceil \frac{L''}{n} \right\rceil)$$

Q は $\mathrm{C}' < Q < \mathrm{C}$ を満たす上記の組数である．尚，$Q = \mathrm{C}' + \left\lceil \frac{L''}{n} \right\rceil$ の内，C' は未知項数 $L'(< L)$ を消去する為の組数であり，$\left\lceil \frac{L''}{n} \right\rceil$ は未知項数 $L''(= L - L')$ を導出する為の組数である．以下，それぞれのステップで必要な計算量の見積もりについて説明する．

ステップ 1 は，Q 組の $(d+1)$ 階差分入力を用意し，平文入力に対応する暗号文を入手する為の計算量である．暗号文を得る為に必要な計算量は，Q 組の $(d+1)$ 階差分入力を暗号化する計算量である．以上から，ステップ 1 に必要な計算量 T_1 は

$$T_1 = 2^{d+1} \times Q, \ (Q = \mathrm{C}' + \left\lceil \frac{L''}{n} \right\rceil)$$

である．計算量の単位は $2^{d+1} \times Q$ 回の暗号化計算量 [ENC.] である．

ステップ 2 は，Q 組の $(d+1)$ 階差分に対し，$\mathbf{c}_i = \bigoplus_{\alpha \in V_i^{(d+1)}} \mathbf{c}, (1 \le i \le Q)$ を計算する為の計算量である．高速化技法 1 と同様に，テーブル G(図 3.2 参照) を用いてベクトル $\mathbf{c}_i, (1 \le i \le Q)$ を計算する．一組のベクトル $\mathbf{c}_i, (1 \le i \le Q)$ の計算に $2^{(d+1)} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil$ 回のテーブル G の参照が必要である．以上から，ステップ 2 に必要な計算量 T_2 は

$$T_2 = 2^{d+1} \times \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times Q, \ (Q = \mathrm{C}' + \left\lceil \frac{L''}{n} \right\rceil)$$

回のテーブル G の参照に必要な計算量である．尚，本論文では，一回のテーブル G の参照に必要な計算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 3 は，適切な $e_i \in \{0, 1\}$ を選ぶ事により，$\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ とする為に必要な計算量である．ステップ 2 で計算したベクトル $\mathbf{c}_i, (1 \le i \le Q)$ を行ベクトルと見なし，Q 組のベクトル $\mathbf{c}_i, (1 \le i \le Q)$ を一列に並べて $Q \times \mathrm{C}$ サイズの行列 $\mathbf{C}$ を生成する (図 B.1 の (1) を参照)．式 (3.21) において，消去する L' 個の未知項に対応するベクトル $\mathbf{a}_j, (1 \le i \le L')$ を計算する為に用いるベクトル $\mathbf{c}'_i$ の要素 $\mathrm{C}'(< \mathrm{C})$ を，行列 $\mathbf{C}$ の左側に集める (図 B.1 の (2) を参照)．ガウス・ジョルダン法を用いて行列 $\mathbf{C}$ の C' 個の列ベクトルを単位行列 $\mathbf{I}$ と $\mathbf{0}$ 行列に変形する．この変形により，任意の $\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ とする事ができる．以上より，ステップ 3 に必要な計算量は

$$T_3 = \left\lceil \frac{\mathrm{C}}{32} \right\rceil \times \frac{Q-1}{2} \times \mathrm{C}'$$

回の 32 bit プロセッサー演算量である [1]．尚，本論文では，一回の 32 bit プロセッサー演算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

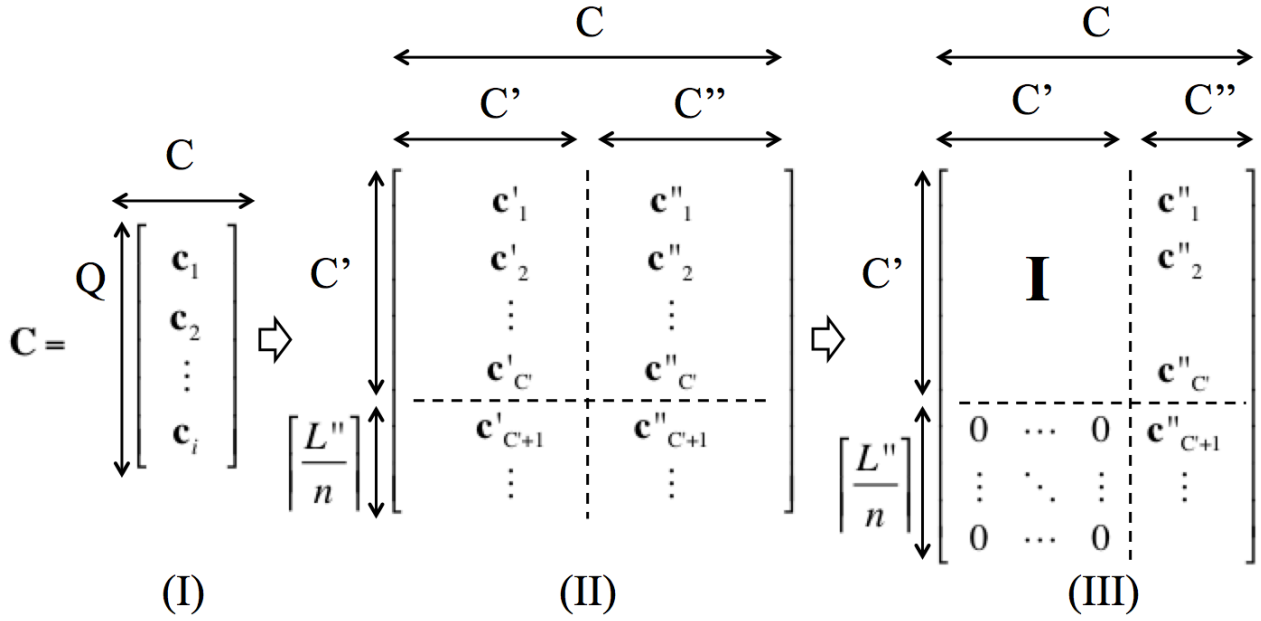

図 B.1: $\bigoplus_{i=1}^{Q} e_i \{\mathbf{c}'_i\} = \mathbf{0}$ の計算方法

ステップ 4 は，式 (3.22) を用いて n 次元ベクトル $\mathbf{a}_j, (L'+1 \leq j \leq L+1)$ を計算する為の計算量である．ステップ 3 の計算で $\mathbf{c}'_i = \mathbf{0}$ となっている為，式 (3.22) に存在する $\mathbf{c}_i$ は $(\mathrm{C}-\mathrm{C}')$ 次元ベクトル，及び $\mathbf{A}_j$ は $n \times (\mathrm{C}-\mathrm{C}')$ サイズの行列と見なす事ができる．式 (3.22) は行列 $\mathbf{A}_j$ の n 個の行ベクトルと $(\mathrm{C}-\mathrm{C})'$ 次元ベクトル $\mathbf{c}_i$ の内積演算である．内積演算に必要な計算量は $2 \times (\mathrm{C}-\mathrm{C}') \times n$ 回のビット演算である．実装に 32 bit プロセッサーを用いて $\mathbf{A}_j, (L'+1 \leq j \leq L+1)$ に対して同様のビット演算を行う場合，計算量は $2 \times \left\lceil \dfrac{\mathrm{C}-\mathrm{C}'}{32} \right\rceil \times n \times (L''+1)$ 回の 32 bit プロセッサー演算量である．$\left\lceil \dfrac{L''}{n} \right\rceil$ 組の $(d+1)$ 階差分に対し上記を繰り返す為，ステップ 4 で必要な計算量 T_4 は

$$T_4 = 2 \times \left\lceil \frac{\mathrm{C}-\mathrm{C}'}{32} \right\rceil \times n \times (L''+1) \times \left\lceil \frac{L''}{n} \right\rceil$$

[1] ガウス・ジョルダン法を用いる場合，計算量は $O(\mathrm{C}^3)$ である．ステップ 3 では，ガウス・ジョルダン法を用いて行列 $\mathbf{C}$ の C' 個の列ベクトルを単位行列 $\mathbf{I}$ と $\mathbf{0}$ 行列に変形している．この部分に必要な計算量は $\mathrm{C} \times (Q-1) \times \mathrm{C}'$ で見積もる事ができる．対象の行列成分は $\{0,1\}$ である為，0 と 1 の出現確率 $p = \frac{1}{2}$ を考慮し，$Q-1$ の部分を $\frac{Q-1}{2}$ と評価している．

回の 32 bit プロセッサー演算量である．尚，本論文では，一回の 32 bit プロセッサー演算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 5 は，得られた線形方程式をガウス・ジョルダン法を用いて解き，鍵を回復する為の計算量である．導出する未知項数が L'' である為，32 bit プロセッサーを用いた場合，ガウス・ジョルダン法に必要な計算量は

$$T_5 = \left\lceil \frac{L''^3}{32} \right\rceil$$

回の 32 bit プロセッサー演算量である．尚，本論文では，一回の 32 bit プロセッサー演算量と，一回の S-box テーブル参照に必要な計算量が同じと見なしている．

ステップ 2 からステップ 5 に必要な計算量の単位は，一回の S-box テーブル参照に必要な計算量で見積もっている．従って，暗号化計算量 [ENC.] で評価する際には，暗号に存在する S-box の個数 $\#S$ で割る必要がある事に注意が必要である．以上より，高速化技法 2 で必要な全体の計算量 T は

$$T = T_1 + \frac{T_2 + T_3 + T_4 + T_5}{\#S}$$

回の暗号化計算量 [ENC.] で見積もる事ができる．

付 録C　KASUMIのDivision属性の伝搬結果

本論文では，文献 [50] と同様な手法を用いて，Division 属性が KASUMI をどの様に伝搬するのか解析している．Division 属性の S-box（S7，S9）伝搬結果は表 4.2，表 4.3 を参照されたい．付録では，FI 関数，FO 関数の Division 属性の伝搬結果を示す．

C.1　FI関数のDivision属性の伝搬

FI 関数の入出力 16 bit を 7 bit，2 bit，7bit に分割し，Division 属性の伝搬を解析する．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}_{\boldsymbol{k}}^{7,2,7}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3)$, $(0 \leq k_1, k_3 \leq 7, 0 \leq k_2 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 3 次元ベクトルである．FI 関数の Division 属性の伝搬を表 C.1, 〜, 表 C.8 以下に示す．

C.2　FO関数のDivision属性の伝搬

FO 関数の Division 属性の伝搬は，FI 関数の Division 属性の伝搬と伝搬ルールを組み合わせる事で解析する事ができる．FO 関数の Division 属性は，FO 関数の入出力 32 bit を (7 bit，2 bit，7bit，7 bit，2 bit，7bit) に分割し，Division 属性の伝搬を解析している．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7 \times GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3, k_4, k_5, k_6)$, $(0 \leq k_1, k_3, k_4, k_6 \leq 7, 0 \leq k_2, k_5 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 6 次元ベクトルである．FO 関数における Division 属性の伝搬を全入力分記載する事は紙面上難しい為，一例（本文中の記載例と異なる）を表 C.9，〜，表 C.11 に示す．

C.3　FL関数のDivision属性の伝搬

FL 関数の Division 属性の伝搬は，1 bit 左巡回シフトの影響を考慮し，伝搬ルールを組み合わせる事で解析する事ができる．FL 関数の Division 属性は，FL 関数の入出力 32 bit を (7 bit，2 bit，

7bit，7 bit，2 bit，7bit) に分割し，Division 属性の伝搬を解析している．入力集合 $\{X\}$ と出力集合 $\{Y\}$ の要素は，各々$GF(2)^7 \times GF(2)^2 \times GF(2)^7 \times GF(2)^7 \times GF(2)^2 \times GF(2)^7$ の値を有する．入力集合 $\{X\}$ の Division 属性は $\mathcal{D}_{\boldsymbol{k}}^{7,2,7,7,2,7}$ で，ベクトル $\boldsymbol{k} = (k_1, k_2, k_3, k_4, k_5, k_6)$, $(0 \leq k_1, k_3, k_4, k_6 \leq 7$, $0 \leq k_2, k_5 \leq 2)$ はゼロベクトル $\mathbf{0}$ を除く 6 次元ベクトルである．FL 関数の Division 属性の伝搬の一例（本文中の記載例と異なる）を表 C.12，〜，表 C.14 に示す．

表 C.1: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[0,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$
(0,0,0)	(0,0,0)
(0,0,1)	(0,0,1),(0,1,0),(1,0,0)
(0,0,2)	(0,0,1),(0,1,0),(1,0,0)
(0,0,3)	(0,0,1),(0,2,0),(1,0,0)
(0,0,4)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(0,0,5)	(0,0,2),(0,1,1),(1,0,0)
(0,0,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(0,0,7)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(0,1,0)	(0,0,1),(0,1,0),(1,0,0)
(0,1,1)	(0,0,1),(0,1,0),(1,0,0)
(0,1,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(0,1,3)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(0,1,4)	(0,0,2),(0,1,1),(1,0,0)
(0,1,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(0,1,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(0,1,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(0,2,0)	(0,0,1),(0,1,0),(1,0,0)
(0,2,1)	(0,0,1),(0,1,0),(1,0,0)
(0,2,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(0,2,3)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(0,2,4)	(0,0,2),(0,1,1),(1,0,0)
(0,2,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(0,2,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(0,2,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)

表 C.2: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[1,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,...,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$
(1,0,0)	(0,0,1),(0,1,0),(1,0,0)
(1,0,1)	(0,0,1),(0,1,0),(1,0,0)
(1,0,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,0,3)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,0,4)	(0,0,2),(0,1,1),(1,0,0)
(1,0,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(1,0,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(1,0,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(1,1,0)	(0,0,1),(0,1,0),(1,0,0)
(1,1,1)	(0,0,1),(0,1,0),(1,0,0)
(1,1,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,1,3)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,1,4)	(0,0,2),(0,1,1),(1,0,0)
(1,1,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(1,1,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(1,1,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(1,2,0)	(0,0,1),(0,1,0),(1,0,0)
(1,2,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,2,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(1,2,3)	(0,0,2),(0,1,1),(1,0,0)
(1,2,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(1,2,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(1,2,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(1,2,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)

表 C.3: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[2,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$
(2,0,0)	(0,0,1),(0,1,0),(1,0,0)
(2,0,1)	(0,0,1),(0,1,0),(1,0,0)
(2,0,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,0,3)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,0,4)	(0,0,2),(0,1,1),(1,0,0)
(2,0,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,0)
(2,0,6)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(2,0,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(2,1,0)	(0,0,1),(0,1,0),(1,0,0)
(2,1,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,1,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,1,3)	(0,0,2),(0,1,1),(1,0,0)
(2,1,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(2,1,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(2,1,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(2,1,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(2,2,0)	(0,0,1),(0,1,0),(1,0,0)
(2,2,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,2,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(2,2,3)	(0,0,2),(0,1,1),(1,0,0)
(2,2,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(2,2,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(2,2,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(2,2,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)

表 C.4: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[3,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$
(3,0,0)	(0,0,1),(0,1,0),(1,0,0)
(3,0,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(3,0,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(3,0,3)	(0,0,2),(0,1,1),(1,0,0)
(3,0,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,0,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,0,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(3,0,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(3,1,0)	(0,0,1),(0,1,0),(1,0,0)
(3,1,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(3,1,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(3,1,3)	(0,0,2),(0,1,1),(1,0,0)
(3,1,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,1,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,1,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(3,1,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(3,2,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(3,2,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(3,2,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,2,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,2,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(3,2,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(3,2,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(3,2,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)

表 C.5: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[4,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,...,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(q)}}$
(4,0,0)	(0,0,1),(0,1,0),(1,0,0)
(4,0,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(4,0,2)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(4,0,3)	(0,0,2),(0,1,1),(1,0,0)
(4,0,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,0,5)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,0,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(4,0,7)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(4,1,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(4,1,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(4,1,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,1,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,1,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,1,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(4,1,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(4,1,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(4,2,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(4,2,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(4,2,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,2,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,2,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(4,2,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(4,2,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(4,2,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)

表 C.6: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[5,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,...,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},...,\boldsymbol{k}^{(q)}}$
(5,0,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(5,0,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(5,0,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,0,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,0,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,0,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(5,0,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,0,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,1,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(5,1,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(5,1,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,1,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,1,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,1,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(5,1,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,1,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,2,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(5,2,1)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,2,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,2,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(5,2,4)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,2,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,2,6)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(5,2,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)

表 C.7: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[6,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,...,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},...,\boldsymbol{k}^{(q)}}$
(6,0,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,0)
(6,0,1)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(6,0,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,0,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,0,4)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,0,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,1),(1,1,0),(2,0,0)
(6,0,6)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,0,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,1,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(6,1,1)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,1,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,1,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,1,4)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,1,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,1,6)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,1,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)
(6,2,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(6,2,1)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,2,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,2,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(6,2,4)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,2,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,2,6)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(6,2,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)

表 C.8: FI 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7}_{[7,*,*]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,...,$\boldsymbol{k}^{(q)}$ of $\mathcal{D}^{7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},...,\boldsymbol{k}^{(q)}}$
(7,0,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(7,0,1)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,0,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,0,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,0,4)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,0,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,0,6)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,0,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)
(7,1,0)	(0,0,2),(0,1,1),(0,2,0),(1,0,1),(1,1,0),(2,0,0)
(7,1,1)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,1,2)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,1,3)	(0,0,3),(0,1,2),(0,2,1),(1,0,1),(1,1,0),(2,0,0)
(7,1,4)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,1,5)	(0,0,4),(0,1,3),(0,2,2),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,1,6)	(0,0,5),(0,1,4),(0,2,3),(1,0,2),(1,1,1),(1,2,0),(2,0,1),(2,1,0),(3,0,0)
(7,1,7)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)
(7,2,0)	(0,0,5),(0,1,4),(0,2,3),(1,0,3),(1,1,2),(1,2,1),(2,0,2),(2,1,1),(2,2,0), (3,0,1),(3,1,0),(4,0,0)
(7,2,1)	(0,0,6),(0,1,5),(0,2,4),(1,0,4),(1,1,3),(1,2,2),(2,0,3),(2,1,2),(2,2,1), (3,0,2),(3,1,1),(3,2,0),(4,0,1),(4,1,0),(5,0,0)
(7,2,2)	(0,0,6),(0,1,5),(0,2,4),(1,0,4),(1,1,3),(1,2,2),(2,0,3),(2,1,2),(2,2,1), (3,0,2),(3,1,1),(3,2,0),(4,0,1),(4,1,0),(5,0,0)
(7,2,3)	(0,0,6),(0,1,5),(0,2,4),(1,0,4),(1,1,3),(1,2,2),(2,0,3),(2,1,2),(2,2,1), (3,0,2),(3,1,1),(3,2,0),(4,0,1),(4,1,0),(5,0,0)
(7,2,4)	(0,0,7),(0,1,6),(0,2,5),(1,0,4),(1,1,3),(1,2,2),(2,0,3),(2,1,2),(2,2,1), (3,0,2),(3,1,1),(3,2,0),(4,0,1),(4,1,0),(5,0,0)
(7,2,5)	(0,0,7),(0,1,6),(0,2,5),(1,0,4),(1,1,3),(1,2,2),(2,0,3),(2,1,2),(2,2,1), (3,0,2),(3,1,1),(3,2,0),(4,0,1),(4,1,0),(5,0,0)
(7,2,6)	(0,2,7),(1,0,6),(1,1,5),(1,2,4),(2,0,4),(2,1,3),(2,2,2),(3,0,3),(3,1,2), (3,2,1),(4,0,2),(4,1,1),(4,2,0),(5,0,1),(5,1,0),(6,0,0)
(7,2,7)	(7,2,7)

表 C.9: FO 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[1,2,3,4,2,5]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(53)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(53)}}$
(1,2,3,4,2,5)	(0,0,0,0,0,4),(0,0,0,0,1,3),(0,0,0,0,2,2),(0,0,0,1,0,2),(0,0,0,1,1,1), (0,0,0,1,2,0),(0,0,0,2,0,0),(0,0,1,0,0,3),(0,0,1,0,1,2),(0,0,1,0,2,1), (0,0,1,1,0,1),(0,0,1,1,1,0),(0,0,2,0,0,2),(0,0,2,0,1,1),(0,0,2,0,2,0), (0,0,2,1,0,0),(0,0,3,0,0,1),(0,0,3,0,1,0),(0,0,6,0,0,0),(0,1,0,0,0,3), (0,1,0,0,1,2),(0,1,0,0,2,1),(0,1,0,1,0,1),(0,1,0,1,1,0),(0,1,1,0,0,2), (0,1,1,0,1,1),(0,1,1,0,2,0),(0,1,1,1,0,0),(0,1,2,0,0,1),(0,1,2,0,1,0), (0,1,5,0,0,0),(0,2,0,0,0,2),(0,2,0,0,1,1),(0,2,0,1,0,0),(0,2,1,0,0,1), (0,2,1,0,1,0),(0,2,4,0,0,0),(1,0,0,0,0,2),(1,0,0,0,1,1),(1,0,0,0,2,0), (1,0,0,1,0,0),(1,0,1,0,0,1),(1,0,1,0,1,0),(1,0,3,0,0,0),(1,1,0,0,0,1), (1,1,0,0,1,0),(1,1,2,0,0,0),(1,2,1,0,0,0),(2,0,0,0,0,1),(2,0,0,0,1,0), (2,0,1,0,0,0),(2,1,0,0,0,0),(3,0,0,0,0,0)

表 C.10: FO 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[5,1,3,4,2,7]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(100)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(100)}}$
(5,1,3,4,2,7)	(0,0,0,0,0,5),(0,0,0,0,1,4),(0,0,0,0,2,3),(0,0,0,1,0,3),(0,0,0,1,1,2), (0,0,0,1,2,1),(0,0,0,2,0,1),(0,0,0,2,1,0),(0,0,0,3,0,0),(0,0,1,0,0,4), (0,0,1,0,1,3),(0,0,1,0,2,2),(0,0,1,1,0,2),(0,0,1,1,1,1),(0,0,1,1,2,0), (0,0,1,2,0,0),(0,0,2,0,0,3),(0,0,2,0,1,2),(0,0,2,0,2,1),(0,0,2,1,0,1), (0,0,2,1,1,0),(0,0,3,0,0,2),(0,0,3,0,1,1),(0,0,3,0,2,0),(0,0,3,1,0,0), (0,0,4,0,0,1),(0,0,4,0,1,0),(0,1,0,0,0,4),(0,1,0,0,1,3),(0,1,0,0,2,2), (0,1,0,1,0,2),(0,1,0,1,1,1),(0,1,0,1,2,0),(0,1,0,2,0,0),(0,1,1,0,0,3), (0,1,1,0,1,2),(0,1,1,0,2,1),(0,1,1,1,0,1),(0,1,1,1,1,0),(0,1,2,0,0,2), (0,1,2,0,1,1),(0,1,2,0,2,0),(0,1,2,1,0,0),(0,1,3,0,0,1),(0,1,3,0,1,0), (0,1,7,0,0,0),(0,2,0,0,0,3),(0,2,0,0,1,2),(0,2,0,0,2,1),(0,2,0,1,0,1), (0,2,0,1,1,0),(0,2,1,0,0,2),(0,2,1,0,1,1),(0,2,1,0,2,0),(0,2,1,1,0,0), (0,2,2,0,0,1),(0,2,2,0,1,0),(0,2,6,0,0,0),(1,0,0,0,0,3),(1,0,0,0,1,2), (1,0,0,0,2,1),(1,0,0,1,0,1),(1,0,0,1,1,0),(1,0,0,2,0,0),(1,0,1,0,0,2), (1,0,1,0,1,1),(1,0,1,0,2,0),(1,0,1,1,0,0),(1,0,2,0,0,1),(1,0,2,0,1,0), (1,0,5,0,0,0),(1,1,0,0,0,2),(1,1,0,0,1,1),(1,1,0,0,2,0),(1,1,0,1,0,0), (1,1,1,0,0,1),(1,1,1,0,1,0),(1,1,4,0,0,0),(1,2,0,0,0,1),(1,2,0,0,1,0), (1,2,3,0,0,0),(2,0,0,0,0,2),(2,0,0,0,1,1),(2,0,0,0,2,0),(2,0,0,1,0,0), (2,0,1,0,0,1),(2,0,1,0,1,0),(2,0,3,0,0,0),(2,1,0,0,0,1),(2,1,0,0,1,0), (2,1,2,0,0,0),(2,2,1,0,0,0),(3,0,0,0,0,1),(3,0,0,0,1,0),(3,0,2,0,0,0), (3,1,1,0,0,0),(3,2,0,0,0,0),(4,0,1,0,0,0),(4,1,0,0,0,0),(5,0,0,0,0,0)

表 C.11: FO 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[6,0,5,7,1,6]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(99)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(99)}}$
(6,0,5,7,1,6)	(0,0,0,0,0,5),(0,0,0,0,1,4),(0,0,0,0,2,3),(0,0,0,1,0,3),(0,0,0,1,1,2), (0,0,0,1,2,1),(0,0,0,2,0,1),(0,0,0,2,1,0),(0,0,0,3,0,0),(0,0,1,0,0,4), (0,0,1,0,1,3),(0,0,1,0,2,2),(0,0,1,1,0,2),(0,0,1,1,1,1),(0,0,1,1,2,0), (0,0,1,2,0,0),(0,0,2,0,0,3),(0,0,2,0,1,2),(0,0,2,0,2,1),(0,0,2,1,0,1), (0,0,2,1,1,0),(0,0,3,0,0,2),(0,0,3,0,1,1),(0,0,3,0,2,0),(0,0,3,1,0,0), (0,0,4,0,0,1),(0,0,4,0,1,0),(0,1,0,0,0,4),(0,1,0,0,1,3),(0,1,0,0,2,2), (0,1,0,1,0,2),(0,1,0,1,1,1),(0,1,0,1,2,0),(0,1,0,2,0,0),(0,1,1,0,0,3), (0,1,1,0,1,2),(0,1,1,0,2,1),(0,1,1,1,0,1),(0,1,1,1,1,0),(0,1,2,0,0,2), (0,1,2,0,1,1),(0,1,2,0,2,0),(0,1,2,1,0,0),(0,1,3,0,0,1),(0,1,3,0,1,0), (0,2,0,0,0,3),(0,2,0,0,1,2),(0,2,0,0,2,1),(0,2,0,1,0,1),(0,2,0,1,1,0), (0,2,1,0,0,2),(0,2,1,0,1,1),(0,2,1,0,2,0),(0,2,1,1,0,0),(0,2,2,0,0,1), (0,2,2,0,1,0),(0,2,7,0,0,0),(1,0,0,0,0,3),(1,0,0,0,1,2),(1,0,0,0,2,1), (1,0,0,1,0,1),(1,0,0,1,1,0),(1,0,0,2,0,0),(1,0,1,0,0,2),(1,0,1,0,1,1), (1,0,1,0,2,0),(1,0,1,1,0,0),(1,0,2,0,0,1),(1,0,2,0,1,0),(1,0,6,0,0,0), (1,1,0,0,0,2),(1,1,0,0,1,1),(1,1,0,0,2,0),(1,1,0,1,0,0),(1,1,1,0,0,1), (1,1,1,0,1,0),(1,1,5,0,0,0),(1,2,0,0,0,1),(1,2,0,0,1,0),(1,2,4,0,0,0), (2,0,0,0,0,2),(2,0,0,0,1,1),(2,0,0,0,2,0),(2,0,0,1,0,0),(2,0,1,0,0,1), (2,0,1,0,1,0),(2,0,3,0,0,0),(2,1,0,0,0,1),(2,1,0,0,1,0),(2,1,2,0,0,0), (2,2,1,0,0,0),(3,0,0,0,0,1),(3,0,0,0,1,0),(3,0,2,0,0,0),(3,1,1,0,0,0), (3,2,0,0,0,0),(4,0,1,0,0,0),(4,1,0,0,0,0),(5,0,0,0,0,0)

表 C.12: FL 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[1,1,0,1,2,1]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(152)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(152)}}$
(1,1,0,1,2,1)	(0,0,0,2,2,2),(0,0,0,3,2,1),(0,0,1,1,2,2),(0,0,1,2,2,1),(0,0,1,3,2,0), (0,0,2,1,2,1),(0,0,2,2,2,0),(0,0,3,1,2,0),(0,1,0,1,2,2),(0,1,0,2,1,2), (0,1,0,2,2,1),(0,1,0,3,1,1),(0,1,0,3,2,0),(0,1,1,0,2,2),(0,1,1,1,1,2), (0,1,1,1,2,1),(0,1,1,2,1,1),(0,1,1,2,2,0),(0,1,1,3,1,0),(0,1,2,0,2,1), (0,1,2,1,1,1),(0,1,2,1,2,0),(0,1,2,2,1,0),(0,1,3,0,2,0),(0,1,3,1,1,0), (0,2,0,1,2,1),(0,2,0,2,1,1),(0,2,0,2,2,0),(0,2,0,3,1,0),(0,2,1,0,2,1), (0,2,1,1,1,1),(0,2,1,1,2,0),(0,2,1,2,1,0),(0,2,2,0,2,0),(0,2,2,1,1,0), (1,0,0,1,2,2),(1,0,0,2,1,2),(1,0,0,2,2,1),(1,0,0,3,1,1),(1,0,1,0,2,2), (1,0,1,1,1,2),(1,0,1,1,2,1),(1,0,1,2,1,1),(1,0,1,2,2,0),(1,0,1,3,1,0), (1,0,2,0,2,1),(1,0,2,1,1,1),(1,0,2,1,2,0),(1,0,2,2,1,0),(1,0,3,0,2,0), (1,0,3,1,1,0),(1,1,0,0,2,2),(1,1,0,1,1,2),(1,1,0,1,2,1),(1,1,0,2,0,2), (1,1,0,2,1,1),(1,1,0,2,2,0),(1,1,0,3,0,1),(1,1,0,3,1,0),(1,1,1,0,1,2), (1,1,1,0,2,1),(1,1,1,1,0,2),(1,1,1,1,1,1),(1,1,1,1,2,0),(1,1,1,2,0,1), (1,1,1,2,1,0),(1,1,1,3,0,0),(1,1,2,0,1,1),(1,1,2,0,2,0),(1,1,2,1,0,1), (1,1,2,1,1,0),(1,1,2,2,0,0),(1,1,3,0,1,0),(1,1,3,1,0,0),(1,2,0,0,2,1), (1,2,0,1,1,1),(1,2,0,1,2,0),(1,2,0,2,0,1),(1,2,0,2,1,0),(1,2,0,3,0,0), (1,2,1,0,1,1),(1,2,1,0,2,0),(1,2,1,1,0,1),(1,2,1,1,1,0),(1,2,1,2,0,0), (1,2,2,0,1,0),(1,2,2,1,0,0),(2,0,0,0,2,2),(2,0,0,1,1,2),(2,0,0,1,2,1), (2,0,0,2,1,1),(2,0,1,0,1,2),(2,0,1,0,2,1),(2,0,1,1,1,1),(2,0,1,1,2,0), (2,0,1,2,1,0),(2,0,2,0,1,1),(2,0,2,0,2,0),(2,0,2,1,1,0),(2,0,3,0,1,0), (2,1,0,0,1,2),(2,1,0,0,2,1),(2,1,0,1,0,2),(2,1,0,1,1,1),(2,1,0,1,2,0), (2,1,0,2,0,1),(2,1,0,2,1,0),(2,1,1,0,0,2),(2,1,1,0,1,1),(2,1,1,0,2,0), (2,1,1,1,0,1),(2,1,1,1,1,0),(2,1,1,2,0,0),(2,1,2,0,0,1),(2,1,2,0,1,0), (2,1,2,1,0,0),(2,1,3,0,0,0),(2,2,0,0,1,1),(2,2,0,0,2,0),(2,2,0,1,0,1), (2,2,0,1,1,0),(2,2,0,2,0,0),(2,2,1,0,0,1),(2,2,1,0,1,0),(2,2,1,1,0,0), (2,2,2,0,0,0),(3,0,0,0,1,2),(3,0,0,0,2,1),(3,0,0,1,1,1),(3,0,1,0,1,1), (3,0,1,0,2,0),(3,0,1,1,1,0),(3,0,2,0,1,0),(3,1,0,0,0,2),(3,1,0,0,1,1), (3,1,0,0,2,0),(3,1,0,1,0,1),(3,1,0,1,1,0),(3,1,1,0,0,1),(3,1,1,0,1,0), (3,1,1,1,0,0),(3,1,2,0,0,0),(3,2,0,0,0,1),(3,2,0,0,1,0),(3,2,0,1,0,0), (3,2,1,0,0,0),(4,0,0,0,1,1),(4,0,1,0,1,0),(4,1,0,0,0,1),(4,1,0,0,1,0), (4,1,1,0,0,0),(4,2,0,0,0,0)

表 C.13: FL 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[0,1,0,1,2,4]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(191)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(191)}}$
(0,1,0,1,2,4)	(0,0,0,2,2,4),(0,0,1,1,2,4),(0,0,1,2,2,3),(0,0,2,1,2,3),(0,0,2,2,2,2), (0,0,3,1,2,2),(0,0,3,2,2,1),(0,0,4,1,2,1),(0,0,4,2,2,0),(0,0,5,1,2,0), (0,1,0,1,2,4),(0,1,0,2,1,4),(0,1,0,2,2,3),(0,1,1,0,2,4),(0,1,1,1,1,4), (0,1,1,1,2,3),(0,1,1,2,1,3),(0,1,1,2,2,2),(0,1,2,0,2,3),(0,1,2,1,1,3), (0,1,2,1,2,2),(0,1,2,2,1,2),(0,1,2,2,2,1),(0,1,3,0,2,2),(0,1,3,1,1,2), (0,1,3,1,2,1),(0,1,3,2,1,1),(0,1,3,2,2,0),(0,1,4,0,2,1),(0,1,4,1,1,1), (0,1,4,1,2,0),(0,1,4,2,1,0),(0,1,5,0,2,0),(0,1,5,1,1,0),(0,2,0,1,2,3), (0,2,0,2,1,3),(0,2,1,0,2,3),(0,2,1,1,1,3),(0,2,1,1,2,2),(0,2,1,2,1,2), (0,2,2,0,2,2),(0,2,2,1,1,2),(0,2,2,1,2,1),(0,2,2,2,1,1),(0,2,3,0,2,1), (0,2,3,1,1,1),(0,2,3,1,2,0),(0,2,3,2,1,0),(0,2,4,0,2,0),(0,2,4,1,1,0), (1,0,0,1,2,4),(1,0,0,2,1,4),(1,0,1,0,2,4),(1,0,1,1,1,4),(1,0,1,1,2,3), (1,0,1,2,1,3),(1,0,2,0,2,3),(1,0,2,1,1,3),(1,0,2,1,2,2),(1,0,2,2,1,2), (1,0,3,0,2,2),(1,0,3,1,1,2),(1,0,3,1,2,1),(1,0,3,2,1,1),(1,0,4,0,2,1), (1,0,4,1,1,1),(1,0,4,1,2,0),(1,0,4,2,1,0),(1,0,5,0,2,0),(1,0,5,1,1,0), (1,1,0,0,2,4),(1,1,0,1,1,4),(1,1,0,1,2,3),(1,1,0,2,0,4),(1,1,0,2,1,3), (1,1,1,0,1,4),(1,1,1,0,2,3),(1,1,1,1,0,4),(1,1,1,1,1,3),(1,1,1,1,2,2), (1,1,1,2,0,3),(1,1,1,2,1,2),(1,1,2,0,1,3),(1,1,2,0,2,2),(1,1,2,1,0,3), (1,1,2,1,1,2),(1,1,2,1,2,1),(1,1,2,2,0,2),(1,1,2,2,1,1),(1,1,3,0,1,2), (1,1,3,0,2,1),(1,1,3,1,0,2),(1,1,3,1,1,1),(1,1,3,1,2,0),(1,1,3,2,0,1), (1,1,3,2,1,0),(1,1,4,0,1,1),(1,1,4,0,2,0),(1,1,4,1,0,1),(1,1,4,1,1,0), (1,1,4,2,0,0),(1,1,5,0,1,0),(1,1,5,1,0,0),(1,2,0,0,2,3),(1,2,0,1,1,3), (1,2,0,2,0,3),(1,2,1,0,1,3),(1,2,1,0,2,2),(1,2,1,1,0,3),(1,2,1,1,1,2), (1,2,1,2,0,2),(1,2,2,0,1,2),(1,2,2,0,2,1),(1,2,2,1,0,2),(1,2,2,1,1,1), (1,2,2,2,0,1),(1,2,3,0,1,1),(1,2,3,0,2,0),(1,2,3,1,0,1),(1,2,3,1,1,0), (1,2,3,2,0,0),(1,2,4,0,1,0),(1,2,4,1,0,0),(2,0,0,0,2,4),(2,0,0,1,1,4), (2,0,1,0,1,4),(2,0,1,0,2,3),(2,0,1,1,1,3),(2,0,2,0,1,3),(2,0,2,0,2,2), (2,0,2,1,1,2),(2,0,3,0,1,2),(2,0,3,0,2,1),(2,0,3,1,1,1),(2,0,4,0,1,1), (2,0,4,0,2,0),(2,0,4,1,1,0),(2,0,5,0,1,0),(2,1,0,0,1,4),(2,1,0,0,2,3), (2,1,0,1,0,4),(2,1,0,1,1,3),(2,1,1,0,0,4),(2,1,1,0,1,3),(2,1,1,0,2,2), (2,1,1,1,0,3),(2,1,1,1,1,2),(2,1,2,0,0,3),(2,1,2,0,1,2),(2,1,2,0,2,1), (2,1,2,1,0,2),(2,1,2,1,1,1),(2,1,3,0,0,2),(2,1,3,0,1,1),(2,1,3,0,2,0), (2,1,3,1,0,1),(2,1,3,1,1,0),(2,1,4,0,0,1),(2,1,4,0,1,0),(2,1,4,1,0,0), (2,1,5,0,0,0),(2,2,0,0,1,3),(2,2,0,1,0,3),(2,2,1,0,0,3),(2,2,1,0,1,2), (2,2,1,1,0,2),(2,2,2,0,0,2),(2,2,2,0,1,1),(2,2,2,1,0,1),(2,2,3,0,0,1), (2,2,3,0,1,0),(2,2,3,1,0,0),(2,2,4,0,0,0),(3,0,0,0,1,4),(3,0,1,0,1,3), (3,0,2,0,1,2),(3,0,3,0,1,1),(3,0,4,0,1,0),(3,1,0,0,0,4),(3,1,0,0,1,3), (3,1,1,0,0,3),(3,1,1,0,1,2),(3,1,2,0,0,2),(3,1,2,0,1,1),(3,1,3,0,0,1), (3,1,3,0,1,0),(3,1,4,0,0,0),(3,2,0,0,0,3),(3,2,1,0,0,2),(3,2,2,0,0,1), (3,2,3,0,0,0)

表 C.14: FL 関数の Division 属性の伝搬（入力 $\mathcal{D}^{7,2,7,7,2,7}_{[7,2,6,5,2,3]}$）

$\boldsymbol{k}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}}$	$\boldsymbol{k}^{(1)}$, $\boldsymbol{k}^{(2)}$,. . . ,$\boldsymbol{k}^{(147)}$ of $\mathcal{D}^{7,2,7,7,2,7}_{\boldsymbol{k}^{(1)},\boldsymbol{k}^{(2)},\ldots,\boldsymbol{k}^{(147)}}$
(7,2,6,5,2,3)	(4,1,4,7,2,7),(4,1,5,7,2,6),(4,1,6,7,2,5),(4,1,7,7,2,4),(4,2,3,7,2,7), (4,2,4,7,2,6),(4,2,5,7,2,5),(4,2,6,7,2,4),(4,2,7,7,2,3),(5,0,4,7,2,7), (5,0,5,7,2,6),(5,0,6,7,2,5),(5,0,7,7,2,4),(5,1,3,7,2,7),(5,1,4,6,2,7), (5,1,4,7,1,7),(5,1,4,7,2,6),(5,1,5,6,2,6),(5,1,5,7,1,6),(5,1,5,7,2,5), (5,1,6,6,2,5),(5,1,6,7,1,5),(5,1,6,7,2,4),(5,1,7,6,2,4),(5,1,7,7,1,4), (5,1,7,7,2,3),(5,2,2,7,2,7),(5,2,3,6,2,7),(5,2,3,7,1,7),(5,2,3,7,2,6), (5,2,4,6,2,6),(5,2,4,7,1,6),(5,2,4,7,2,5),(5,2,5,6,2,5),(5,2,5,7,1,5), (5,2,5,7,2,4),(5,2,6,6,2,4),(5,2,6,7,1,4),(5,2,6,7,2,3),(5,2,7,6,2,3), (5,2,7,7,1,3),(5,2,7,7,2,2),(6,0,3,7,2,7),(6,0,4,6,2,7),(6,0,4,7,2,6), (6,0,5,6,2,6),(6,0,5,7,2,5),(6,0,6,6,2,5),(6,0,6,7,2,4),(6,0,7,6,2,4), (6,1,2,7,2,7),(6,1,3,6,2,7),(6,1,3,7,1,7),(6,1,3,7,2,6),(6,1,4,5,2,7), (6,1,4,6,1,7),(6,1,4,6,2,6),(6,1,4,7,1,6),(6,1,4,7,2,5),(6,1,5,5,2,6), (6,1,5,6,1,6),(6,1,5,6,2,5),(6,1,5,7,1,5),(6,1,5,7,2,4),(6,1,6,5,2,5), (6,1,6,6,1,5),(6,1,6,6,2,4),(6,1,6,7,1,4),(6,1,6,7,2,3),(6,1,7,5,2,4), (6,1,7,6,1,4),(6,1,7,6,2,3),(6,2,1,7,2,7),(6,2,2,6,2,7),(6,2,2,7,1,7), (6,2,2,7,2,6),(6,2,3,5,2,7),(6,2,3,6,1,7),(6,2,3,6,2,6),(6,2,3,7,1,6), (6,2,3,7,2,5),(6,2,4,5,2,6),(6,2,4,6,1,6),(6,2,4,6,2,5),(6,2,4,7,1,5), (6,2,4,7,2,4),(6,2,5,5,2,5),(6,2,5,6,1,5),(6,2,5,6,2,4),(6,2,5,7,1,4), (6,2,5,7,2,3),(6,2,6,5,2,4),(6,2,6,6,1,4),(6,2,6,6,2,3),(6,2,6,7,1,3), (6,2,6,7,2,2),(6,2,7,5,2,3),(6,2,7,6,1,3),(6,2,7,6,2,2),(7,0,3,6,2,7), (7,0,4,5,2,7),(7,0,4,6,2,6),(7,0,5,5,2,6),(7,0,5,6,2,5),(7,0,6,5,2,5), (7,0,6,6,2,4),(7,0,7,5,2,4),(7,1,2,6,2,7),(7,1,3,5,2,7),(7,1,3,6,1,7), (7,1,3,6,2,6),(7,1,4,5,1,7),(7,1,4,5,2,6),(7,1,4,6,1,6),(7,1,4,6,2,5), (7,1,5,5,1,6),(7,1,5,5,2,5),(7,1,5,6,1,5),(7,1,5,6,2,4),(7,1,6,5,1,5), (7,1,6,5,2,4),(7,1,6,6,1,4),(7,1,6,6,2,3),(7,1,7,5,1,4),(7,1,7,5,2,3), (7,2,1,6,2,7),(7,2,2,5,2,7),(7,2,2,6,1,7),(7,2,2,6,2,6),(7,2,3,5,1,7), (7,2,3,5,2,6),(7,2,3,6,1,6),(7,2,3,6,2,5),(7,2,4,5,1,6),(7,2,4,5,2,5), (7,2,4,6,1,5),(7,2,4,6,2,4),(7,2,5,5,1,5),(7,2,5,5,2,4),(7,2,5,6,1,4), (7,2,5,6,2,3),(7,2,6,5,1,4),(7,2,6,5,2,3),(7,2,6,6,1,3),(7,2,6,6,2,2), (7,2,7,5,1,3),(7,2,7,5,2,2)

発表論文

本研究を構成する論文

学術論文

1. A Study on Higher Order Differential Attack of KASUMI
   （KASUMI の高階差分攻撃に関する研究）
   Nobuyuki SUGIO, Hiroshi AONO, Sadayuki HONGO, and Toshinobu KANEKO
   IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, vol.E90-A, no.1, pp.14-21, January, 2007.

国際会議

1. A Study on Higher Order Differential Attack of KASUMI
   （KASUMI の高階差分攻撃に関する研究）
   Nobuyuki SUGIO, Hidema TANAKA, Toshinobu KANEKO
   Proc. International Symposium on Information Theory and Its Applications (ISITA 2002), pp.755-758, Xi ' an, China, October, 2002.

2. A Fast Calculus for the Linearizing Attack and Its Application to an Attack on KASUMI
   （線形化攻撃の高速化と KASUMI への適用）
   Nobuyuki SUGIO, Shunichi NAMBU, and Toshinobu KANEKO
   Proc. 16th International Symposium, Applied Algebra, Algebraic Algorithms and Error-Correcting Codes (AAECC 2006), LNCS 3857, pp.163-172, Las Vegas, NV, USA, February, 2006.

3. A Practical-time Attack on Reduced-round MISTY1
   （段数を減らした MISTY1 への現実的な時間での攻撃）
   Nobuyuki SUGIO, Yasutaka IGARASHI, Toshinobu KANEKO, and Kenichi HIGUCHI
   Proc. 2nd International Conference on Information Systems Security and Privacy (ICISSP 2016), pp.235-242, Rome, Italy February, 2016.

4. New Integral Characteristics of KASUMI Derived by Division Property
   （Division 属性を用いて導出した KASUMI の新たな積分特性）
   Nobuyuki SUGIO, Yasutaka IGARASHI, and Toshinobu KANEKO
   Proc. 17th World Conference on Information Security Applications (WISA 2016), pp.15-16, Jeju, Korea, August, 2016.

5. Integral Characteristics of MISTY2 Derived by Division Property
   （Division 属性を用いて導出した MISTY2 の積分特性）
   Nobuyuki SUGIO, Yasutaka IGARASHI, and Toshinobu KANEKO
   Proc. International Symposium on Information Theory and Its Applications (ISITA 2016), pp.151-155, Monterey, California, USA, November, 2016.

その他の発表論文

国際会議

1. A Study on Higher Order Differential Attack of MISTY1
   （MISTY1 に対する高階差分攻撃に関する研究）
   Hidema TANAKA, Nobuyuki SUGIO, and Toshinobu KANEKO
   Proc. 4th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT 2001), pp.354-358, Kathmandu, Nepal, November, 2001.

2. A Higher Order Differential Attack of 7-round MISTY1 without FL function
   （FL 関数の無い 7 段 MISTY1 に対する高階差分攻撃）
   Makiko SHIROTA, Nobuyuki SUGIO, Yasuo HATANO, Hidema TANAKA, and Toshinobu KANEKO
   Proc. 5th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT 2003), pp.203-208, November 25-26, Noumea, New Caledonia, 2003.

3. Security Analysis of MISTY1
   （MISTY1 の安全性解析）
   Hidema TANAKA, Yasuo HATANO, Nobuyuki SUGIO, and Toshinobu KANEKO
   Proc. 8th International Workshop, Information Security Applications (WISA 2007), pp.215-226, Jeju, Korea, August, 2007.

4. A New Higher Order Differential of Camellia
   （Camellia の新たな高階差分特性）
   Nobuyuki SUGIO, Hiroshi AONO, Kimihiko SEKINO, Toshinobu KANEKO
   Proc. International Symposium on Information Theory and its Applications (ISITA 2014), pp.491-495, Melbourne, Australia, October, 2014.

研究会

1. MISTY1 の強度評価に関する一考察
   杉尾 信行，田中 秀磨，金子 敏信
   第 24 回 情報理論とその応用シンポジウム（SITA 2001）予稿集, pp.659-662, 神戸市, 兵庫県, 12 月, 2001.

2. KASUMI の高階差分攻撃に関する一考察
   杉尾 信行，田中 秀磨，金子 敏信
   2002 年 暗号と情報セキュリティシンポジウム（SCIS2002）予稿集, 13A-3, 西牟婁郡白浜町, 和歌山県, 1 月, 2002.

3. 鍵スケジュールを考慮したブロック暗号に対する攻撃に関する一考察
   田中 秀磨，杉尾 信行，金子 敏信
   2003 年 暗号と情報セキュリティシンポジウム（SCIS2003）予稿集, 5D-3, 浜松市, 静岡県, 1 月, 2003.

4. MISTY1 における上一段消去法の適用に関する一考察
   白田 麻貴子，杉尾 信行，秦野 康生，田中 秀磨，金子 敏信
   2003 年 暗号と情報セキュリティシンポジウム（SCIS2003）予稿集, 6D-3, 浜松市, 静岡県, 1 月, 2003.

5. KASUMI の高階差分攻撃に関する一考察 (II)
   杉尾 信行，金子 敏信
   2004 年 暗号と情報セキュリティシンポジウム（SCIS2004）予稿集, 3A5-3, 仙台市, 宮城県, 1 月, 2004.

6. KASUMI に対する攻撃方程式の効率的な解法
   南部 俊一，杉尾 信行，金子 敏信
   2005 年 暗号と情報セキュリティシンポジウム（SCIS2005）予稿集, 4D2-5, 神戸市, 兵庫県, 1 月, 2005.

7. MISTY1 及び KASUMI における上一段消去法の適用に関する一考察
   杉尾 信行，青野 博，本郷 節之，金子敏信
   第 28 回情報理論とその応用シンポジウム（SITA 2005）予稿集, pp.387-390, 那覇市, 沖縄県, 11 月, 2005.

8. KASUMI に対する攻撃方程式の効率的な解法 (II)
   杉尾 信行，青野 博，本郷 節之，金子 敏信
   2006 年 暗号と情報セキュリティシンポジウム（SCIS2006）予稿集, 3E2-4, 広島市, 広島県, 1 月, 2006.

9. MISTY1 及び KASUMI の Integral-interpolation attack に関する一考察
   杉尾 信行，青野 博，本郷 節之，金子 敏信
   コンピュータセキュリティシンポジウム 2006（CSS2006）予稿集, 3B-3, 京都市, 京都府, 10 月, 2006.

10. 移動体通信システムの暗号危殆化に対する企業活動の主要課題-社内外既存システムの永続性、国際標準との整合性対応-
    杉尾 信行，関野 公彦，松田 卓也，岡田 浩，水田 成仁
    2012 年 暗号と情報セキュリティシンポジウム（SCIS2012）予稿集, 2F2-1, 金沢市, 石川県, 1 月, 2012.

11. Camellia の新たな高階差分特性
    杉尾 信行，青野 博，関野 公彦，金子 敏信
    第 36 回 情報理論とその応用シンポジウム（SITA 2013）予稿集, 3-4-3, pp.226-231, 伊東市, 静岡県, 11 月, 2013.

12. Camellia の新たな高階差分特性 (II)
    杉尾 信行，青野 博，関野 公彦，金子 敏信
    2014 年 暗号と情報セキュリティシンポジウム（SCIS2014）予稿集, 1F3-2, 鹿児島市, 鹿児島県, 1 月, 2014.

13. ブロック暗号アルゴリズム MISTY2 の積分特性探索
    杉尾 信行，五十嵐 保隆，金子 敏信
    2016 年 暗号と情報セキュリティシンポジウム（SCIS2016）予稿集, 3D2-3, 熊本市, 熊本県, 1 月, 2016.

14. 共通鍵ブロック暗号アルゴリズム KASUMI の積分攻撃
    杉尾 信行，五十嵐 保隆，金子 敏信

2017 年 暗号と情報セキュリティシンポジウム（SCIS2017）予稿集, 2B1-4, 那覇市, 沖縄県, 1 月, 2017.

謝辞

本論文の執筆にあたり，御懇切なご指導，ご鞭撻，ならびに様々なご配慮を賜った東京理科大学 理工学部 電気電子情報工学科の金子敏信教授，樋口健一教授，五十嵐保隆講師に深く感謝致します．先生方の多大なるご指導，ご支援のお陰で共通鍵ブロック暗号に関する研究に邁進する事ができたと確信しております．また，研究成果を国際会議で発表する機会を数多く与えて頂き，海外の研究動向にいち早く触れる事ができたと共に，世界で活躍する一流の暗号研究者の方々と意見交換をする事で，本論文の研究成果に生かす事ができたと思っております．

また，学位審査を通じて，東京理科大学 理工学部 電気電子情報工学科 前田譲治教授，松田一朗教授，東京理科大学 理工学部 情報科学科 明石重男教授，東京理科大学 基礎工学部 電子応用工学科 伊丹誠教授には，様々な視点から学位論文の内容についてご指導，ご鞭撻を賜り，厚く御礼申し上げます．

また，本研究の機会を与えて頂き，ご指導，ご支援頂いた株式会社 NTT ドコモ サービスイノベーション部の大野友義部長，関野公彦担当部長，青野博担当課長，ならびに北海道科学大学 工学部 情報工学科 本郷節之教授 (旧株式会社 NTT ドコモ) に深く感謝致します．

また，学生時代に共通鍵ブロック暗号の研究に関するご指導，ご鞭撻を頂いた防衛大学校 情報工学科 田中秀磨准教授 (旧東京理科大学) に深く感謝致します．

最後に，長年私を支えて下さった友人，家族，特に妻の文華に心より感謝申し上げます．