Computer Security Symposium 2021
26 - 29 October 2021

軽量ブロック暗号 Shadow-32 に対する Bit-Based Division Property を用いた Integral 攻撃

苫瓜 剛志[1,*]　五十嵐 保隆[1]　金子 敏信[1]

概要: Shadow-32 は 2021 年に Ying Guo らによって提案された Feistel 構造を持つブロック暗号である.ブロック長は 32 ビット,段数 16 段,鍵長は 64 ビットをサポートしている.Shadow-32 の提案者論文には,不能差分攻撃については定量的な安全性評価が記載されているが,Integral 特性については評価が行われていない.そこで,本稿では,Shadow-32 に対する Integral 特性について報告する.その結果,混合整数線形計画法(Mixed Integer Linear Programming：MILP)を用いた Bit-Based Division Property の解析によって 31 階差分による 10 段の Integral 特性を報告する．そしてこの Integral 特性を利用することにより，暗号文側に 2 段追加した 12 段分の鍵回復において，鍵長 64 ビットの場合，必要な選択平文数が2^{31}および計算量が$2^{29.53}$となることを示す.

キーワード：軽量ブロック暗号, Integral 特性, Bit-Based Division Property

Integral Attack with Bit-Based Division Property of lightweight block cipher Shadow-32

Tsuyoshi Nigauri[1,*]　Yasutaka Igarashi[1]　Toshinobu Kaneko[1]

Abstract: Shadow-32 is a block cipher with a Feistel structure proposed by Ying Guo, Lang Li, and Botao Liu in 2021. Its block length is 32 bits, the number of rounds is 16, and the key length is 64 bits. In their proposed paper, security analysis of impossible differentia is evaluated, but integral characteristics is not evaluated. Therefore, this paper reports on integral characteristics of Shadow-32. As a result, we report the 10-round integral characteristics of the 31st-order differential discovered by the analysis of Bit-Based Division Property using the Mixed Integer Linear Programming (MILP). And by using the integral characteristics, it is shown that the required number of chosen plain texts is 2^{31} and the computational complexity is $2^{29.53}$ encryptions when the key length is 64 bits in the key recovery for 12 rounds, which is added by 2 rounds on the cipher text side.

Keywords: Lightweight Block Cipher, Integral characteristics, Bit-Based Division Property

1. 序論

概要：Shadow-32 は 2021 年に Ying Guo らによって提案された Feistel 構造を持つブロック暗号である [1].ブロック長は 32 ビット,段数 16 段,鍵長は 64 ビットをサポートしている.Shadow-32 の提案者論文には,不能差分攻撃については定量的な安全性評価が記載されているが,Integral 特性については評価が行われていない.そこで,本稿では,Shadow-32 に対する Integral 特性について報告する.その結果,混合整数線形計画法(Mixed Integer Linear Programming：MILP)を用いた Bit-Based Division Property の解析によって 31 階差分による 10 段の Integral 特性を報告する．そしてこの Integral 特性を利用することにより，暗号文側に 2 段追加した 12 段分の鍵回復において，鍵長 64 ビットの場合，必要な選択平文数が2^{31}および計算量が$2^{29.53}$となることを示す.

2. 表記法

表 1 に本稿を通して用いる記号の表記を以下に示す.

表 1. 表記法

記号	説明
$\oplus$	排他的論理和(XOR)
$x[i]$	スカラ値xの 2 進数表記での右からiビット目
$w(x)$	スカラ値xの 2 進数表記でのハミング重み (：xを構成する 1 の個数) ex). $x = 110 \rightarrow w(x) = 2$
$W(\vec{x})$	ベクトル$\vec{x}$のハミング重みベクトル ex). $\vec{x} = (110,001)$ $\rightarrow W(\vec{x}) = (w(110), w(001)) = (2,1)$
$\geq$	$GF(2)^n$ 上の同次元の 2 つのベクトル$x = (x_1, x_2, x_3, \cdots, x_n)$，$y = (y_1, y_2, y_3, \cdots, y_n)$において常に$x_i \geq y_i$ $(i = 1,2,3, \cdots, n)$ならば, $\overrightarrow{x_1} \geq \overrightarrow{x_2}$
$\|\|$	ビット結合

1 東京理科大学理工学研究科電気工学専攻
Department of Electrical Engineering, Faculty of Science and Technology, Tokyo University of Science
* 7320577@ed.tus.ac.jp

© 2021 Information Processing Society of Japan

3. Shadow-32 の構造

図 1 に Shadow-32 のデータ撹拌部を示す。Shadow-32 はブロック長 32bit，段数 16 段，鍵長は 64bit である．また，段関数について，平文（32bit）の MSB（最上位ビット）側から 8bit がL_0，次の 8bit がL_1，その次の 8bit がR_0，LSB（最下位ビット）側の 8bit がR_1となる．$F_j(j=1,2,3,4)$は 8 ビット入出力の非線形関数であり，$RK_{RN}^{j}(j=0,1,2,3)$が段鍵 8bit，$RN(RN=1,2,\cdots,16)$がRN段目の段鍵であることを表す．1 段につき 4 種類のF_j関数がある．$F_j(j=1,2,3,4)$の構造を図 2 に示す．その仕様としては，<<<n は n bit 左巡回シフト，&はビット毎の AND を表す．また，鍵スケジュール部に関して，64bit の内部状態から副鍵を生成し，その副鍵を 8 ビットずつ 4 分割したものを段鍵としている．鍵スケジュール部の内部構造を図 3 に示し，鍵スケジュール部で用いる処理である AddRoundConstant，NX module についてそれぞれ図 4，5 に示す．

AddRoundCounstant では，RNラウンド目であるRNの 2 進数表記での各ビットを MSB から順にc_o, c_1, c_2, c_3, c_4に割り当てる．さらに，Permutation での内部状態の各ビットの対応関係を表 2 に示す．内部状態の撹拌では$k_j(j=3,4,5,6,7)$に AddRoundConstant を行い，次に NX module に$k_j(j=56,57,58,59,60,61,62,63)$を通し，最後に Permutation を行う．これにより全てのk_jがk_j'に変換される．変換されたk_j'を内部状態のk_jの場所に格納する．格納後の内部状態の $k_j(j=0,1,2,3,8,9,10,11)$ を RK_{RN}^{0}，$k_j(j=4,5,6,7,12,13,14,15)$ を RK_{RN}^{1}，$k_j(j=16,17,18,19,24,25,26,27)$ を RK_{RN}^{2}，$k_j(j=20,21,22,23,28,29,30,31)$を$RK_{RN}^{3}$とする．以上の操作を$RN$回繰り返し，全ての$RK_{RN}^{j}$を生成する．

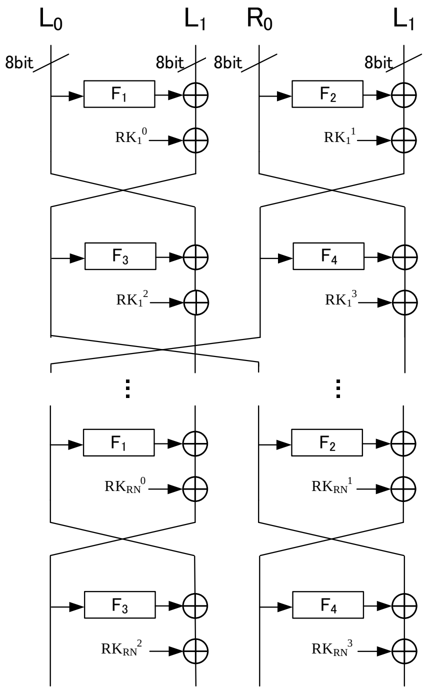

図 1. Shadow-32 のデータ撹拌部

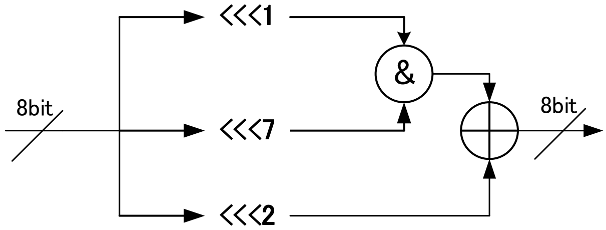

図 2. $F_j(j=1,2,3,4)$関数

k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7
k_8	k_9	k_{10}	k_{11}	k_{12}	k_{13}	k_{14}	k_{15}
k_{16}	k_{17}	k_{18}	k_{19}	k_{20}	k_{21}	k_{22}	k_{23}
k_{24}	k_{25}	k_{26}	k_{27}	k_{28}	k_{29}	k_{30}	k_{31}
k_{32}	k_{33}	k_{34}	k_{35}	k_{36}	k_{37}	k_{38}	k_{39}
k_{40}	k_{41}	k_{42}	k_{43}	k_{44}	k_{45}	k_{46}	k_{47}
k_{48}	k_{49}	k_{50}	k_{51}	k_{52}	k_{53}	k_{54}	k_{55}
k_{56}	k_{57}	k_{58}	k_{59}	k_{60}	k_{61}	k_{62}	k_{63}

図 3. 鍵スケジュール部の内部状態

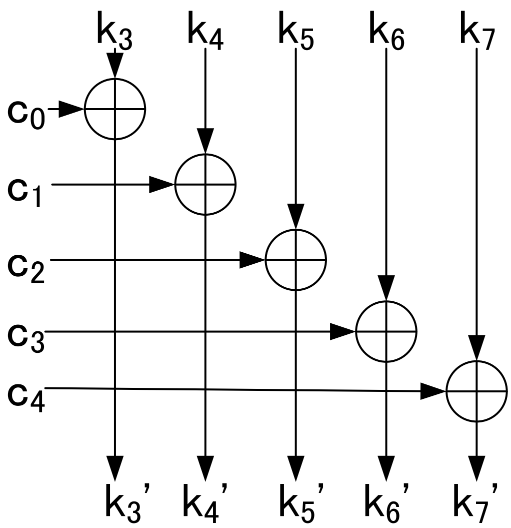

図 4. AddRoundConstant の構造

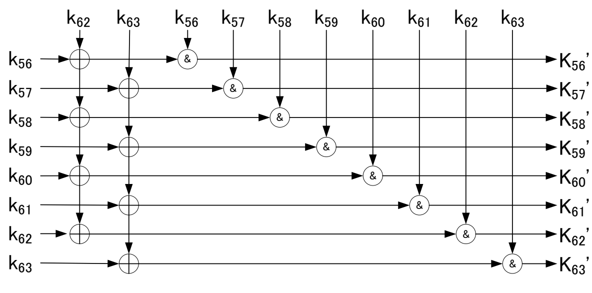

図 5. NX module の構造

表 2. Permutation の対応関係

k_i	k_i'	k_i	k_i'	k_i	k_i'	k_i	k_i'
0	56	16	60	32	40	48	0
1	57	17	61	33	41	49	1
2	58	18	62	34	42	50	2
3	59	19	63	35	43	51	3
4	16	20	28	36	44	52	4
5	17	21	29	37	45	53	5
6	18	22	30	38	46	54	6
7	19	23	31	39	47	55	7
8	20	24	32	40	48	56	8
9	21	25	33	41	49	57	9
10	22	26	34	42	50	58	10
11	23	27	35	43	51	59	11
12	24	28	36	44	52	60	12
13	25	29	37	45	53	61	13
14	26	30	38	46	54	62	14
15	27	31	39	47	55	63	15

4. Integral 攻撃

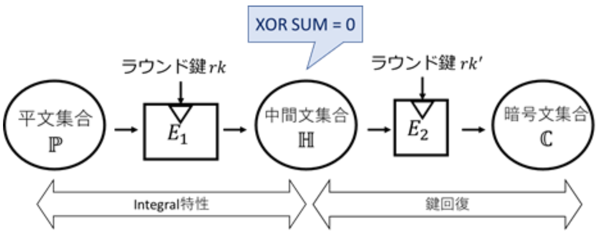

図 6. Integral 攻撃の概要

Integral 攻撃は Kundsen と Wagner が提案した暗号解読手法である [2]. 図 6 に Integral 攻撃の概要を示す．攻撃者は Integral 攻撃を実行する際，初めに Integral 特性の探索を行う．Integral 特性とは，特定の平文集合$\mathbb{P}$に属する全ての平文を複数段暗号化した中間データの集合$\mathbb{H}$の XOR の総和が全ての鍵に対して 0 になるような特性を表す. 以下の 4.1 節に Integral 特性を発見する主要手法の一つを紹介する.

4.1 Integral Property

表 3 に示すデータの集合の性質の伝搬を評価する．この 4 つの性質を Integral Property と呼ぶ．このうち，A,C,B の 3 性質はデータの XOR 総和が確定的に 0 となる．U の場合は XOR 総和が鍵の値に依存し，一般には攻撃に使用できない．なお，大文字の場合は複数ビット纏め，小文字の場合は 1 ビット単位の Integral Property を表す.

表 3. Integral Property

A(all) a(active)	全値が同数回出現
C, c(Const)	同値が出現
B, b(Balance)	A, C 以外で XOR 総和が 0
U, u(Unknown)	XOR 総和が鍵に依存

5. Bit-Based Division Property

2015 年に藤堂らは Division Property(DP)を考案し [3], Integral Property を拡張し代数次数を利用できるようにした. また，2016 年に藤堂らによって Bit-Based Division Property(BDP)が考案され [4], 1 ビット単位で DP を考えることが可能となった.

5.1 ビット積関数

任意のnビット変数

$$u = u[n] \parallel u[n-1] \parallel \cdots \parallel u[1], u[i] \in \{0.1\}$$

そして，入力のnビット変数

$$x = x[n] \parallel x[n-1] \parallel \cdots \parallel x[1], x[i] \in \{0.1\}$$

に対するビット積関数$\pi_u(x)$は(1)式で定義される. ここで，$\pi_u(x)$は 1 ビットである.

$$\pi_u(x) \coloneqq \prod_{i=1}^{n} x[i]^{u[i]} \tag{1}$$

次に任意のm次元ベクトル

$$\vec{u} = (u_m, u_{m-1}, \cdots, u_1), u_i \in \{0,1\}^n$$

そして入力のm次元ベクトル

$$\vec{x} = (x_m, x_{m-1}, \cdots, x_1), x_i \in \{0,1\}^n$$

に対するビット積関数$\pi_{\vec{u}}(\vec{x})$は(2)式で定義される．ここで出力$\pi_{\vec{u}}(\vec{x})$は 1 ビットである.

$$\pi_{\vec{u}}(\vec{x}) \coloneqq \prod_{i=1}^{m} \pi_{u_i}(x_i) \tag{2}$$

ただし，本稿では Bit-Based Division Property を扱うため，(1),(2)式において$n = 1$となる.

5.2 BDP の定義

m次元ベクトル$\vec{x}$の集合を$\mathbb{X}$と表し，次に示すm次元ベクトル$\vec{u}$について(3)式を満たす時，集合$\mathbb{X}$は BDP $D_{\mathbb{K}}^{1^m}$を持つという.

$$\bigoplus_{\vec{x} \in \mathbb{X}} \pi_{\vec{u}}(\vec{x}) = \begin{cases} unkown, if\ there\ is\ \vec{k} \in \mathbb{K}\ s.t.\ W(\vec{u}) \succcurlyeq \vec{k} \\ 0, otherwise \end{cases} \tag{3}$$

5.3 BDP と Integral 特性の対応

表 4 に BDP と Integral 特性の対応を示す.

表 4. BDP と Integral 特性の対応

Rule 1. 集合$\mathbb{K}$の要素が一つのとき， その要素であるベクトルの 1 のビット位置→"a", 0 のビット位置→"c" ex) $D^{1^4}_{\{(1,1,1,0)\}} \rightarrow aaac$
Rule 2. 集合$\mathbb{K}$が単位ベクトルを含む時， 単位ベクトルの要素で 1 のビット位置→"u" ex) $D^{1^4}_{\mathbb{K}=\{(1,0,0,0),(0,1,0,0),(0,0,1,0),(0,0,0,1)\}} \rightarrow uuuu$

Rule 3.
単位ベクトルの要素で 1 が存在しないビット位置→”b”
ex) $D^{1^4}_{\{(1,0,0,0),\ (0,1,0,0)\}} \rightarrow uubb$

5.4 Division Trail

まず，r段目までnビットの BDP が段関数を通って(4)式のように変化していくとする．

$$D^{1^n}_{\mathbb{K}_0} \rightarrow D^{1^n}_{\mathbb{K}_1} \rightarrow D^{1^n}_{\mathbb{K}_2} \rightarrow \cdots \rightarrow D^{1^n}_{\mathbb{K}_{r-1}} \rightarrow D^{1^n}_{\mathbb{K}_r} \qquad (4)$$

ここで BDP の変遷を集合から集合への伝播で表すのではなく，集合の要素であるベクトルからベクトルへの伝播しうる 1 本道で表すことを考える．それが Division Trail である．それによって 6 章における MILP の考え方を適用できる．

具体的には$i \geq 1$のとき任意の $\overrightarrow{k_i^*} \in \mathbb{K}_i$において伝播元である $\overrightarrow{k_{i-1}^*} \in \mathbb{K}_{i-1}$が存在する．そのため，(5)式のように $\overrightarrow{k_i^*}$を$0 \leq i \leq r$の範囲で順に繋げ，伝播しうる道を表すことができる．

$$\overrightarrow{k_0^*} \rightarrow \overrightarrow{k_1^*} \rightarrow \overrightarrow{k_2^*} \rightarrow \cdots \rightarrow \overrightarrow{k_{r-1}^*} \rightarrow \overrightarrow{k_r^*} \qquad (5)$$

5.5 Bit-Based Division Property の伝播ルール

本研究で用いた BDP の 3 つの伝播ルールを(6)~(8)式に示す．

Rule 1. Copy（分岐）

$$\begin{cases} D^1_0 \rightarrow D^{1^2}_{(0,0)} \\ D^1_1 \rightarrow D^{1^2}_{\{(0,1),(1,0)\}} \end{cases} \qquad (6)$$

Rule 2. XOR

$$\begin{cases} D^{1^2}_{(0,0)} \rightarrow D^1_0 \\ D^{1^2}_{(0.1)} \rightarrow D^1_1 \\ D^{1^2}_{(1.0)} \rightarrow D^1_1 \\ D^{1^2}_{(1,1)} \rightarrow 破棄 \end{cases} \qquad (7)$$

Rule 3. AND

$$\begin{cases} D^{1^2}_{(0.0)} \rightarrow D^1_0 \\ D^{1^2}_{(0.1)} \rightarrow D^1_1 \\ D^{1^2}_{(1,0)} \rightarrow D^1_1 \\ D^{1^2}_{(1,1)} \rightarrow D^1_1 \end{cases} \qquad (8)$$

6. MILP aided Bit-Based Division Property

目的関数および制約条件は線形であり，変数の全てまたは一部が整数に制限される数理最適化問題の解法を混合数線形計画法(Mixed Integer linear Programing: MILP)と呼ぶ．本稿では，2016 年に Xiang らによって提案された MILP を BDP へ適用する手法 [5]を使用する．その手法では，BDP の伝播を連立線形不等式に置き換え，r段目集合$\mathbb{K}_r$内の単位ベクトル(Integral Property “u”に相当)を探索するという方針でr段目の Integral 特性を調査する．

6.1 目的関数

Division Trail のr段目におけるベクトル $\overrightarrow{k_r^*}$を(9)式のように表せるとする．

$$\overrightarrow{k_r^*} = (a_0^r, a_1^r, \cdots, a_{n-1}^r) \qquad (9)$$

このとき，目的関数は(10)式のように表すことができる．

$$odj = Min\{a_0^r + a_1^r + \cdots + a_{n-1}^r\} \qquad (10)$$

ここで，目的関数の値が 1 になったとき，r段目集合$\mathbb{K}_r$内に単位ベクトルが存在する．つまり Integral Property “u” が存在することを意味する．

6.2 Copy, XOR, AND の制約条件

(6)～(8)式を線形不等式に置き換えると以下の(11)～(13)式のようになる．

Model 1. Copy（分岐）
Division Trail: $(a) \xrightarrow{copy} (b_0, b_1, b_2, b_3)$

$$\begin{cases} a - b_0 - b_1 - b_2 - b_3 = 0 \\ a.\, b_0, b_1.\, b_2.\, b_3\ are\ binaries \end{cases} \qquad (11)$$

Model 2. XOR
Division Trail: $(a_0, a_1, a_2, a_3) \xrightarrow{XOR} (b)$

$$\begin{cases} a_0 + a_1 + a_2 + a_3 - b = 0 \\ a_0, a_1, a_2, a_3, b\ are\ binaries \end{cases} \qquad (12)$$

Model 3. AND
Division Trail: $(a_0, a_1) \xrightarrow{AND} (b)$

$$\begin{cases} b - a_0 \geq 0 \\ b - a_1 \geq 0 \\ b - a_0 - a_1 \leq 0 \\ a_0, a_1, b\ are\ binaries \end{cases} \qquad (13)$$

7. Shadow-32 の Integral 特性

調査の結果，入力の 31 ビットを active(31 階差分)にしたとき，合計 10 段分の Integral 特性が得られた．以下にその詳細を示す．なお，小文字は 1 ビット単位，大文字は 4 ビット単位の Integral 特性を示す．

表 5. 31 階差分での 10 段目特性結果

Input of the 1st round	Output of the 10th round
AAAAAaaacAA	BBUUBBUU (特性①)
AAAAAaacaAA	
AAAAAacaaAA	
AAAAAcaaaAA	
AAAAaaacAAA	
AAAAaacaAAA	
AAAAacaaAAA	
AAAAcaaaAAA	
AaaacAAAAAA	
AaacaAAAAAA	

AacaaAAAAAA	
AcaaaAAAAAA	
aaacAAAAAAA	
aacaAAAAAAA	
acaaAAAAAAA	
caaaAAAAAAA	

特性①は，1~8 ビット目または 17~24 ビット目の内どれかを 1 ビットを定数とし，他 31 ビットについて全通り（パターン数2^{31}）入力としたとき，10 段目出力の 1~8 ビット目と 17~24 ビット目の出力データの XOR 総和が 0 となることを示す．この他の入力では 10 段目出力の XOR 総和が 0 とはならなかった．

なお，active ビットが 30 以下（30 階差分以下）では 10 段特性は発見されなかった．

8. 鍵回復

7 章に示した特性を用いて，暗号文側に 2 段追加した計 12 段の Shadow-32 の鍵回復を行う．

8.1 段鍵の推定と候補数の削減

12 段目の出力の暗号文を 11 段目の入力まで部分的復号する際，7 章で示した特性に関与する経路は図 7 の青線と赤線の通りである．また，得られた特性に関与する段鍵RK_{12}^2（64 ビット中の 8 ビット），RK_{12}^3（64 ビット中の 8 ビット）の計 16 ビットにおいて特性の出現位置で排他的論理和の総和が 0 になるか調査し，鍵候補の削減，推定を行う．鍵候補の削減，推定では Ferguson らが提案した部分和法 [6]を用いる．

8.1.1 青線における鍵候補の推定と削減

図 7 におけるd_1, d_2について，攻撃方程式は(14)式になる．

$$\sum d_1 = \sum d_2 \tag{14}$$

ただし，

$$\sum d_1 = \sum F_4(F_3(L_0^{13}) \oplus L_1^{13} \oplus RK_{12}^2) \tag{15}$$

$$\begin{aligned}\sum d_2 &= \sum F_2(F_4(R_0^{13}) \oplus R_1^{13} \oplus RK_{12}^3) \oplus R_0^{13} \oplus RK_{12}^1 \\ &\quad \oplus RK_{11}^3 \\ &= \sum F_2(F_4(R_0^{13}) \oplus R_1^{13} \oplus RK_{12}^3) \oplus R_0^{13} \end{aligned} \tag{16}$$

である．

以下の手順で鍵候補の削減，推定を行う．

1. 12 段目の出力が$(L_o^{13}, L_1^{13}, R_0^{13}, R_1^{13})$である暗号文データが$2^{31}$個ある．
2. 2^{31}個の(15)式中の 8 ビットデータ$F_3(L_0^{13}) \oplus L_{13}^1$の値$0 \sim 2^8 - 1$までの出現回数の奇偶を調べ，奇数回出現したデータのみを残す．
3. 2^{31}個の(16)式中の 8 ビットデータ$F_4(R_0^{13}) \oplus R_1^{13}$の値$0 \sim 2^8 - 1$までの出現回数の奇偶を調べ，奇数回出現したデータのみを残す．この過程で(16)式中の$\sum R_0^{13}$も導出する．

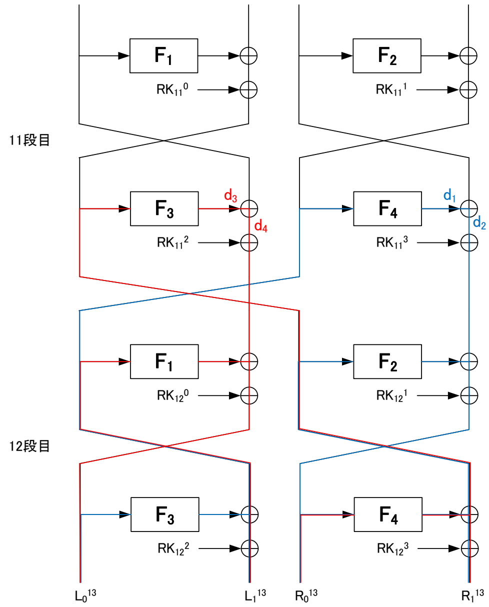

図 7. 特性に関与する経路

4. $RK_{12}^2 = i (i = 0,1,\cdots,2^8)$と仮定し，$\sum d_1$を計算し，$RK_{12}^2$と$\sum d_1$の値の組を表にする．
5. $RK_{12}^3 = j (j = 0,1,\cdots,2^8)$と仮定し，$\sum d_2$を計算する．更に表を参照し，$\sum d_1 = \sum d_2$が成立するか確認する．この時，成立した時の$i$と$j$を正しい鍵の候補として残す．

手順 2 から 5 までのF_j関数の計算回数は，

$$2^{31} + 2^{31} + 2^8 \times 2^8 + 2^8 + 2^8 \approx 2^{32} \tag{17}$$

である．データ撹拌部 12 段分には 48 個のF_j関数が存在するので，暗号化計算回数は，$\frac{2^{32}}{48}$となる．

また，この処理では段鍵RK_{12}^2，RK_{12}^3が真鍵ならば常に攻撃方程式は成立する．一方で偽鍵の場合2^{-8}の確率で成立する．よって，この処理により(RK_{12}^2, RK_{12}^3)の候補数は

$$2^{16} \times 2^{-8} = 2^8$$

へ削減される．

8.1.2 赤線における鍵候補の推定と削減

図 7 におけるd_3, d_4について攻撃方程式は(16)式になる．

$$\sum d_3 = \sum d_4 \tag{18}$$

ただし，

$$\sum d_3 = \sum F_3(F_4(R_0^{13}) \oplus R_1^{13} \oplus RK_{12}^3) \tag{19}$$

$$\begin{aligned}\sum d_4 &= \sum F_1(F_3(L_0^{13}) \oplus L_1^{13} \oplus RK_{12}^2) \oplus L_0^{13} \oplus RK_{12}^0 \oplus RK_{11}^2 \\ &= \sum F_1(F_3(L_0^{13}) \oplus L_1^{13} \oplus RK_{12}^2) \oplus L_0^{13} \end{aligned} \tag{20}$$

である．赤線においても 8.1.1 項と同様の処理を行う．この際に 8.1.1 の手順 2 と手順 3 は再利用する．8.1.1 の処理によって(RK_{12}^2, RK_{12}^3)の候補数が2^8に絞られていることに留意すると，F_j関数の計算回数は，

$$2^8(2 \times 2^8) = 2^{17} \tag{21}$$

である．データ撹拌部 12 段分には 48 個のF_j関数が存在するので，暗号化計算回数は，$\frac{2^{17}}{48}$となる．

また，この処理では段鍵RK_{12}^2，RK_{12}^3が真鍵ならば常に攻

撃方程式は成立する．一方で偽鍵の場合2^{-8}の確率で成立する．よって，この処理により(RK_{12}^2, RK_{12}^3)の候補数は

$$2^8 \times 2^{-8} = 2^0$$

へ削減される．この結果として，秘密鍵 64 ビット中の 16 ビット分の値が一つに絞られた．したがって，残りの不明なビットは$64 - 16 = 48$ビットとなる．

以上より 8.1 節における全体の暗号化計算回数は

$$\frac{2^{32}}{48} + \frac{2^{17}}{48} \approx 2^{26.42} \tag{22}$$

となる．

8.2 NX module の逆算

NX module は次の(a1)~(a8)式で表せる．ただし，$k_i, k_i'(i = 56,57,58,59,60,61,62,63)$は 0 か 1 の値を取る．

$$\begin{cases} k'_{56} = k_{56}(k_{56} + k_{62}) & (a1) \\ k'_{57} = k_{57}(k_{57} + k_{63}) & (a2) \\ k'_{58} = k_{58}(k_{58} + k_{56} + k_{62}) & (a3) \\ k'_{59} = k_{59}(k_{59} + k_{57} + k_{63}) & (a4) \\ k'_{60} = k_{60}(k_{60} + k_{58} + k_{56} + k_{62}) & (a5) \\ k'_{61} = k_{61}(k_{61} + k_{59} + k_{57} + k_{63}) & (a6) \\ k'_{62} = k_{62}(k_{60} + k_{58} + k_{56}) & (a7) \\ k'_{63} = k_{63}(k_{61} + k_{59} + k_{57}) & (a8) \end{cases}$$

NX module を逆算する問題は，既知のk_i'から未知のk_iを求める問題である．以下に全てのk_i'が 1 の時と全てのk_i'が 0 の時の式を示す．その後，実際に NX module を逆算する際の解の個数を推察する．

8.2.1 全てのk_i'が 1 である時

全てのk_i'が 1 である時，(a1)~(a8)式は次の(b1)~(b8)式と(c1)~(c8)式で書き直せる．

$$\begin{cases} k_{56} = 1 & (b1) \\ k_{57} = 1 & (b2) \\ k_{58} = 1 & (b3) \\ k_{59} = 1 & (b4) \\ k_{60} = 1 & (b5) \\ k_{61} = 1 & (b6) \\ k_{62} = 1 & (b7) \\ k_{63} = 1 & (b8) \end{cases}$$

$$\begin{cases} (k_{56} + k_{62}) = 1 & (c1) \\ (k_{57} + k_{63}) = 1 & (c2) \\ (k_{58} + k_{56} + k_{62}) = 1 & (c3) \\ (k_{59} + k_{57} + k_{63}) = 1 & (c4) \\ (k_{60} + k_{58} + k_{56} + k_{62}) = 1 & (c5) \\ (k_{61} + k_{59} + k_{57} + k_{63}) = 1 & (c6) \\ (k_{60} + k_{58} + k_{56}) = 1 & (c7) \\ (k_{61} + k_{59} + k_{57}) = 1 & (c8) \end{cases}$$

(b1)~(b8)式を行表現にすると，その係数行列のランクは 8 である．同様に(c1)~(c8)式を行列表現すると，その係数行列のランクは 8 である．したがって，(b1)~(b8)式より解が一意に定まり，(c1)~(c8)より解が一意に定まる．そして 2 つの解が一致していれば真の解である．

8.2.2 全てのk_i'が 0 である時

全てのk_i'が 0 である時，(a1)~(a8)式は次の 2 つの場合分け(d1)~(d8)式,(e1)~(e8)式と(f1)~(f8)式,(g1)~(g8)式で書き直せる．ただし，$x_i(i = 56,57,58,59,60,61,62,63)$は 0 か 1 の値を取る．

$$\begin{cases} k_{56} = 0 & (d1) \\ k_{57} = 0 & (d2) \\ k_{58} = 0 & (d3) \\ k_{59} = 0 & (d4) \\ k_{60} = 0 & (d5) \\ k_{61} = 0 & (d6) \\ k_{62} = 0 & (d7) \\ k_{63} = 0 & (d8) \end{cases}$$

$$\begin{cases} (k_{56} + k_{62}) = 0 & (e1) \\ (k_{57} + k_{63}) = 0 & (e2) \\ (k_{58} + k_{56} + k_{62}) = 0 & (e3) \\ (k_{59} + k_{57} + k_{63}) = 0 & (e4) \\ (k_{60} + k_{58} + k_{56} + k_{62}) = 0 & (e5) \\ (k_{61} + k_{59} + k_{57} + k_{63}) = 0 & (e6) \\ (k_{60} + k_{58} + k_{56}) = 0 & (e7) \\ (k_{61} + k_{59} + k_{57}) = 0 & (e8) \end{cases}$$

$$\begin{cases} k_{56} = x_{56} & (f1) \\ k_{57} = x_{57} & (f2) \\ k_{58} = x_{58} & (f3) \\ k_{59} = x_{59} & (f4) \\ k_{60} = x_{60} & (f5) \\ k_{61} = x_{61} & (f6) \\ k_{62} = x_{62} & (f7) \\ k_{63} = x_{63} & (f8) \end{cases}$$

$$\begin{cases} (k_{56} + k_{62}) = x_{56} + 1 & (g1) \\ (k_{57} + k_{63}) = x_{57} + 1 & (g2) \\ (k_{58} + k_{56} + k_{62}) = x_{58} + 1 & (g3) \\ (k_{59} + k_{57} + k_{63}) = x_{59} + 1 & (g4) \\ (k_{60} + k_{58} + k_{56} + k_{62}) = x_{60} + 1 & (g5) \\ (k_{61} + k_{59} + k_{57} + k_{63}) = x_{61} + 1 & (g6) \\ (k_{60} + k_{58} + k_{56}) = x_{62} + 1 & (g7) \\ (k_{61} + k_{59} + k_{57}) = x_{63} + 1 & (g8) \end{cases}$$

(d1)~(d8)式，(e1)~(e8)式の場合，$k_i = 0$となる．

(f1)~(f8)式，(g1)~(g8)式の場合，(f1)~(f8)式を用いて解を定め，その定めた解の正しさを(g1)~(g8)式を用いて確認する．(g1)~(g8)式を行列表現すると，その係数行列のランクは 8 であるため，(f1)~(f8)式で定めた解が(g1)~(g8)式をも満たす確率は2^{-8}である．(f1)~(f8)式より解は2^8通り得られるので，その中から平均的に 1 つ程度が(g1)~(g8)式を満たすと考えられる．

8.2.3 実際に NX module を逆算する際の解の個数

NX module を逆算する際には，全てのk_i'が 1 である時や全てのk_i'が 0 である時は稀であり，多くの場合はk_i'は 0 と 1 の値を取っている．つまり(b1)から(f8)式の中から方程式を選択することになる．この場合においても解の個数は平均的には 1 個程度と考えられる．また，実際に NX module を逆算する際は，NX module の入出力関係を事前に計算してリストとして保存し，そのリストを参照することで逆算できる．

8.3 29 階差分で得られた Integral 特性を用いた鍵候補の推定

入力の 29 ビットを active(29 階差分)にしたとき，合計 9

段分の Integral 特性が得られた．表 6 に今回の攻撃で用いる特性と，その特性が得られた入力の一部を示す．

表 6. 攻撃で用いる 29 階差分での 9 段特性結果

Input of the 1st round(一部)	Output of the 9th round
acaaAAAacaaaaacAA acaaAAAacacAAA acaaaaacAAacaaAAA	bbbuBUUbbbuBUU (特性②)

図 8 に鍵候補の推定で行うために必要な$d_i(i=1,2,\cdots,10)$の位置を示す．

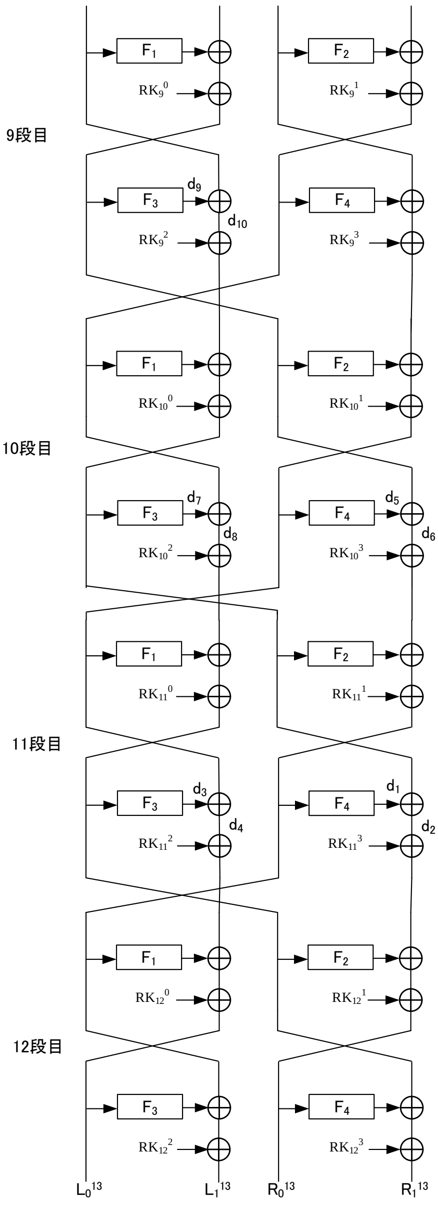

図 8. $d_i(i=1,2,\cdots,10)$の位置

8.3.1 d_7,d_8についての攻撃方程式

攻撃方程式は(23)式になる．

$$\sum d_7=\sum d_8 \tag{23}$$

ただし，

$$\sum d_7=\sum F_3(d_1\oplus d_2) \tag{24}$$

$$\sum d_8=\sum(d_3\oplus d_4)\oplus R_4(R_0^{13})\oplus R_1^{13}\oplus RK_{12}^3 \oplus RK_{11}^0\oplus RK_{10}^2 \tag{25}$$

$$=\sum F_1(d_3\oplus d_4)\oplus F_4(R_0^{13})\oplus R_1^{13} \tag{26}$$

である．

ここで，(15),(16)式中の値が既知である部分をe_1とし，(16)式中の値が未知である部分$RK_{12}^1\oplus RK_{11}^3$を$ek_1$とする．また，(19),(20)式中の値が既知である部分をe_2とし，(20)式中の値が未知である部分$RK_{12}^0\oplus RK_{11}^2$を$ek_2$とする．この時，(24),(26)式はそれぞれ次のように書き換えられる．

$$\sum d_7=F_3(e_1\oplus ek_1)$$

$$\sum d_8=\sum F_1(e_2\oplus ek_2)\oplus F_4(R_0^{13})\oplus R_1^{13}$$

以下の手順で鍵候補の推定を行う．

1. 12 段目の出力$(L_0^{13},L_1^{13},R_0^{13},R_1^{13})$である暗号文から，$e_1$と$e_2$の出現頻度の奇偶を検査し，奇数回出現したデータのみを残す．尚，$\sum F_4(R_0^{13})\oplus R_1^{13}$は 8.1.1 項の手順 3 で得られたデータから計算する．
2. $\sum d_7$をek_1の全通りに対し計算し，ek_1と$\sum d_7$の値の組の表を作成する．
3. $\sum d_8$をek_2の全通りに対し計算し，表を参照して$\sum d_7=\sum d_8$となる時の(ek_1,ek_2)の組を真鍵の候補とする．

この手順を 1 回行うと，(ek_1,ek_2)の候補数が2^{-7}になる．そのため，この手順を異なる 3 種類の 29 階差分入力について 3 回行うと

$$2^{16}\times 2^{-7}\times 2^{-7}\times 2^{-7}=2^{-5}$$

となり一意に定まる． 8.3.1 項で推定した鍵の内部状態と 8.1 節で推定した鍵の内部状態のビットは 4 ビット重複するため，ここで推定できる鍵の内部状態は$16-4=12$ビットとなり，8.1 節で特定した鍵のビット数と合計すると$16+12=28$ビットの鍵の内部状態を特定したこととなる．また，この手順 1 から 3 での暗号化回数は，

$$3\times\frac{4\times 2^{29}+4\times 2^{29}+2^8\times 2^8+2^8\times 2^8}{48}=2^{28} \tag{27}$$

となる．

8.3.2 d_5,d_6についての攻撃方程式

図 8 における$d_5.d_6$についての攻撃方程式は(28)式になる．

$$\sum d_5=\sum d_6 \tag{28}$$

ただし，

$$\sum d_5=\sum F_4(d_3\oplus d_4) \tag{29}$$

$$\sum d_6=\sum F_2(d_1\oplus d_2)\oplus F_3(L_0^{13})\oplus L_1^{13}\oplus RK_{12}^2 \oplus RK_{10}^3\oplus RK_{11}^1$$

$$=\sum F_2(d_1\oplus d_2)\oplus F_3(L_0^{13})\oplus L_1^{13} \tag{30}$$

8.3.1 項にて$RK_{12}^1\oplus RK_{11}^3$と$RK_{12}^0\oplus RK_{11}^2$を推定したため，$d_1\oplus d_2$と$d_3\oplus d_4$が計算できる． また，$RK_{11}^1$は鍵スケジュール部の構造より 8.1 節で推定したRK_{12}^2,RK_{12}^3から求められるので一意に定まる．したがって，(28)式には未知の鍵変数を含まないので攻撃には利用できない．

8.3.3 8 段特性を用いた鍵候補の推定

29 階差分で得られた 8 段特性の結果を用いて鍵候補の推

定を行う．表 7 に 29 階差分で得られた攻撃に用いる 8 段特性と，その特性が得られた入力の一部を示す．

表 7. 攻撃で用いる 29 階差分での 8 段特性結果

Input of the 1st round(一部)	Output of the 8th round
AAAAAaaacAaacc AAAAAaaacAacca	BBUUBBUU (特性③)

図 8 におけるd_9, d_{10}に対する攻撃方程式は次のようになる．

$$\sum d_9 = \sum d_{10} \tag{31}$$

ただし，

$$\begin{aligned}\sum d_9 &= \sum F_3(d_5 \oplus d_6)\\ &= \sum F_3(e_3 \oplus RK_{11}^1 \oplus RK_{10}^3)\\ &= \sum F_3(e_2 \oplus ek_3)\end{aligned} \tag{32}$$

$$\begin{aligned}\sum d_{10} &= \sum F_1(d_7 \oplus d_8) \oplus (d_1 \oplus d_2) \oplus RK_{10}^0 \oplus RK_9^2\\ &= \sum F_1(d_7 \oplus d_8) \oplus (d_1 \oplus d_2)\\ &= \sum F_7(e_5 \oplus RK_{11}^0 \oplus RK_{10}^2)\\ &= \sum F_1(e_4 \oplus ek_4)\end{aligned} \tag{33}$$

(32)式中で値が既知である部分をe_3とし，未知である部分$RK_{11}^1 \oplus RK_{10}^3$を$ek_3$とする．また，(33)式で値が既知である部分を$e_4$とし，未知である部分$RK_{11}^0 \oplus RK_{10}^2$を$ek_4$としている．$ek_3$と$ek_4$の計 16 ビットを推定する．手順は次の通りである．

1. 12 段目の出力$(L_0^{13}, L_1^{13}, R_0^{13}, R_1^{13})$である暗号文から，$e_3$と$e_4$の出現頻度の奇偶を検査し，奇数回出現したデータのみを残す．(F_j関数計算回数:$9 \times 2^{29} + 9 \times 2^{29}$)
2. 残ったデータe_3を用いてek_3の全通りに対して(32)式の$\sum d_9$を計算し，ek_3と$\sum d_9$の値の組を表にする．(F_j関数計算回数:$2^8 \times 2^8$)
3. 残ったデータe_4を用いて，ek_4の全通りに対して(33)式の$\sum d_{10}$を計算し，(31)式が成立する時の(ek_3, ek_4)の組を真鍵の候補とする．(F_j関数計算回数:$2^8 \times 2^8$)

手順 1 から 3 までを 1 回行うことで，(ek_3, ek_4)の候補が2^{-8}になる．よって，2 種類の 29 階差分入力を用いることで，真鍵の候補数は

$$2^{16} \times 2^{-8} \times 2^{-8} = 2^0$$

となり，一意に定まる．

ここで，これまで推定した鍵の内部状態との重複に留意すると，ここで 4 ビットが重複しているので新たに$16 - 4$ビットが特定できたことになり，これまでと合わせて推定した鍵の内部状態は$28 + 16 - 4 = 40$ビットとなる．また，8.3.3 項での暗号化計算回数は，

$$2 \times \frac{9 \times 2^{29} + 9 \times 2^{29} + 2^8 \times 2^8 + 2^8 \times 2^8}{48} \approx 2^{28.58} \tag{34}$$

となる．

8.4 総当り

ここで, 秘密鍵まで辿って残り2^{24}の候補に対して総当りを行う．この時の暗号化計算回数は

$$2^{24} \tag{35}$$

である．

8.5 全体の計算量

(22), (27), (34), (35)式より鍵回復に必要な計算回数は,

$$2^{26.42} + 2^{28} + 2^{28.58} + 2^{24} \approx 2^{29.53}$$

である．

9. まとめ

本稿では Shadow-32 に対し, MILP aided Bit-Based Division Property を用いて 10 段分の Integral 特性を発見した．加えて，以下のような数値で鍵回復を行うことができると示した．また，合わせて提案者評価も記載する．表 8 での提案者評価の欄でのNとは，選択した平文構造が2^N個という意味である．

表 8. 鍵回復における各値

	攻撃可能段数	暗号化計算回数	選択平文数
本稿	12	$2^{29.53}$	2^{31}
提案者評価	7	評価なし	2^{N+25}

参考文献

[1] Y. Guo, L. Li , B. Liu, “Shadow A Lightweight Block Cipher for IoT Nodes,” IEEE Internet of Things Jornal ,doi: 10.1109/JIOT.2021.3064203, https://ieeexplore.ieee.org/document/9372286,(cited 2021-06-24), 2021

[2] L. Knudsen , D. Wagner, “Integral cryptanalysis,” FSE 2002. LNCS, vol. 2365, pp. 112‐127. Springer, Heidelberg, 2002.

[3] Y. Todo, “Integral cryptanalysis on full MISTY1,” CRYPTO 2015. LNCS, vol. 9215, pp. 413‐432. Springer, Heidelberg, 2015.

[4] Y. Todo , M. Morii, “Bit-based division property and application to SIMON family,” Cryptology ePrint Archive, Report 2016/285, 2016.

[5] Z. Xiang, W. Zhang, Z. Bao , D. Lin, “Applying MILP Method to Searching IntegralDistinguishers Based on Division Property for 6Lightweight Block Ciphers,” Advances in Cryptology ‐ASIACRYPT 2016, 2016.

[6] N. Ferguson, J. Kelsey, S. Lucks, B. Schneier, M. Stay, D. Wagner , D. Whiting, “Improved Cryptanalysis of Rijndael,” FSE2000, LNCS1978, pp.213‐230, Springer, Heidelberg, 2002.