Galaxy：ストリーム暗号ベースSpacehard暗号

小池 祐二[1]　阪本 光星[1]　林 卓也[2]　五十部 孝典[1,2]

概要：ホワイトボックス暗号は攻撃者が実行環境に自由にアクセスができる場合において秘密鍵の安全性を確保する技術である．ACM CCS 2015 において Bogdanov と Isobe により AES テーブルをもとにしたホワイトボックス専用のブロック暗号 SPACE と，共通鍵暗号の新しい安全性指標 Space Hardness が提案された．SPACE ではホワイトボックスモデルにおいてテーブルから秘密鍵を求める問題をブラックボックスモデルの AES の鍵回復問題に帰着させた．しかし AES テーブルの生成において多くの計算量が必要となり，鍵更新を頻繁に行うようなプロトコルにおいては利用できない．本研究ではストリーム暗号ベースのホワイトボックス用ブロック暗号 Galaxy を提案する．この暗号ではソフトウェアで高速なストリーム暗号を用いてテーブルを生成する．これにより鍵を求める困難性は SPACE と同様にストリーム暗号の鍵回復問題に帰着している．また Galaxy においても SPACE と同じ安全性を確保するように設計している．結果としてストリーム暗号 chacha を用いた場合には SPACE と比べて約 20 倍から 35 倍に高速化した．

キーワード：暗号，共通鍵暗号，ホワイトボックス暗号

Galaxy: Spacehard Cipher based on Stream Cipher

Yuji Koike[1]　Kosei Sakamoto[1]　Takuya Hayashi[2]　Takanori Isobe[1,2]

Abstract: Whtie-box Cryptography aims to ensure the security of cryptographic algorithms in untrusted environments where the attacker has full access to their implementations. Bogdanov and Isobe proposed white-box cryptography SPACE, table-based cryptography created with a block cipher such as AES. SPACE is designed to be secure in the way the security against key-extraction in the white-box setting is reduced to the security of block cipher against key-extraction in the black-box setting. However, it costs much time complexity when generating the table. As a result, it is not applicable to protocols which require frequent updates of the secret key. In this paper, we propose a white-box cryptography Galaxy. In Galaxy, we employ stream cipher to generate the table, and just like SPACE, the security of Galaxy against key-extraction in the white-box setting is reduced to the security of stream cipher against key-extraction in the black-box setting. We successfully create the table of Galaxy 25x-35x faster than that of SPACE.

Keywords: Cryptography, Symmetric cryptography, Whitebox cryptography

1. はじめに

技術発展によるソフトウェアの多様化に伴い，信頼できない環境（ホワイトボックスモデル）におけるソフトウェアの安全性を確保する必要性が増してきている．ホワイトボックスモデルでは，攻撃者は暗号化（復号）の中間値にアクセス，改変だけでなく，実行コードのリバース・エンジニアリングなど様々な攻撃が可能である．

ACM CCS 2015 において Bogdanov と Isobe によりホワイトボックスモデルにおいても安全性の高いブロック暗号 SPACE と，共通鍵暗号の新しい性質である Space Hardness が提案された [2]．SPACE は AES などのブロック暗号における入力と出力の一部をテーブルに保存しテーブルそのものを秘密鍵として扱うブロック暗号である．SPACE で

[1] 兵庫県立大学応用情報科学研究科
Graduate School of Applied Informatics, University of Hyogo
[2] 情報通信研究機構
National Institute of Information and Communications Technology

	ホワイトボックス鍵回復問題	Space Hardness	テーブルサイズ	テーブル生成 (cycle)	暗号化 (cpb)
SPACE-8	ブロック暗号（AES など）の鍵回復問題に帰着	$(T/4,\ 128)$	3.75KB	27218	349.56
SPACE-16			896KB	6780452	173.01
SPACE-32			48GB	451234349359	181.39
Galaxy-8	ストリーム暗号（chacha など）の鍵回復問題に帰着		256B	761	78.52
Galaxy-16			128KB	192878	30.26
Galaxy-32			16GB	23215671915	47.63

表 1 Galaxy と SPACE の比較

はホワイトボックスモデルにおいてテーブルから秘密鍵を求める問題をブラックボックスモデルにおけるブロック暗号の鍵回復問題に帰着させた．また Space Hardness は漏洩したテーブルのデータ量とそれに応じた安全性の関係を表す指標であり，SPACE はテーブルサイズを T としたとき，その 1/4 が漏洩しても 2^{-128} の安全性を保証する $(T/4,\ 128)$-space hardness を実現している．しかし実装性能の観点では最適化されておらず，特にテーブルの生成において多くの計算量が必要となり，SSL/TLS，QUIC [6]，Signal protocol [7] などの鍵更新を頻繁にするプロトコルでは致命的で利用することができない．

本研究ではストリーム暗号ベースのホワイトボックス用ブロック暗号 Galaxy を提案する．この暗号ではソフトウェアで高速なストリーム暗号を用いてテーブルを生成する．これにより鍵を求める困難性は SPACE と同様にストリーム暗号の鍵回復問題に帰着している．また Galaxy では Type-2 Generalized Feistel 構造を採用することで暗号化演算の効率化もはかる．安全性に関しては SPACE と同じ $(T/4,\ 128)$-space hardness を実現している．結果としてストリーム暗号 chacha [1] を用いた場合，SPACE と同じ安全性を保ちながら，暗号化（復号）において約 3.8 倍から 5.7 倍の効率化を実現し，テーブル生成においては約 20 倍から 37 倍の効率化を実現した．表 1 で Galaxy と SPACE を比較したまとめを示す．

2. 準備

2.1 Space Hardness

ACM CCS2015 で Bogdanov と Isobe によって共通鍵暗号の新しい性質として Space Hardness が提案された [2]．Space Hardness の定義は以下の通りである．

定義 1. *((M, Z)-space hardness)*

ブロック暗号 E_K において，サイズが M 以下のコード（テーブル）が与えられた場合，ランダムに選ばれた平文（暗号文）を 2^{-z} 以上の確率で暗号化（復号）することが不可能なとき，ブロック暗号 E_K は *(M, Z)-space hardness* であるとする．

Space Hardness はホワイトボックスモデルにおいて実行環境からプログラムコードの一部を抜き取り，そのまま秘密鍵として利用する攻撃である code lifting 攻撃に対する安全性を定量的に評価するための指標として使われる．例えば，$(T/4, 128)$-space hardness は，コードサイズを T としたとき，その 1/4 以下を攻撃者が入手したとしても，ランダムに選ばれた平文を 2^{-128} 以上の確率で求めることができないことを意味する．

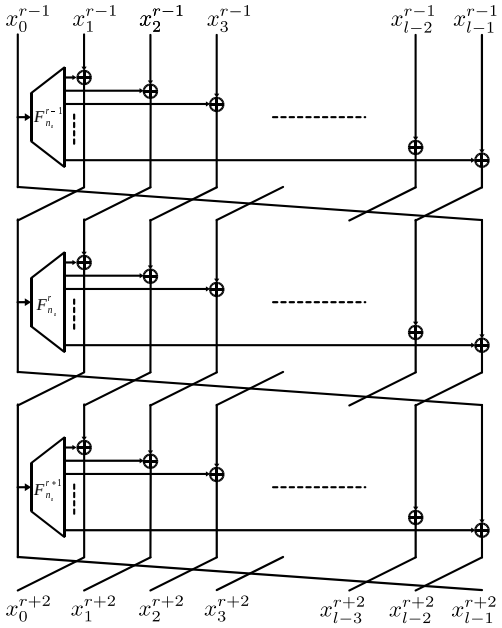

図 1 SPACE のアルゴリズム

2.2 ブロック暗号 SPACE

2.2.1 SPACE の仕様

Space Hardness を持つ暗号として Bogdanov と Isobe は SPACE と呼ばれるブロック暗号を提案した [2]．

SPACE では図 1 で示すように l 個のラインで構成される Target Heavy Generalized Feistel 構造を採用している．SPACE は n ビットの平文と k ビットの秘密鍵 K を入力とし，n ビットの暗号文を出力する．R を総ラウンド数，$n_a = n/l$ としたとき，r ラウンド目の中間値 $X^r = \{x_0^r, x_1^r, ..., x_l^r\}$, $x_i^r \in \{0,1\}^{n_a}$ は以下のように表される．

$$X^{r+1} = (F_{n_a}^r(x_0^r) \oplus (x_1^r \parallel x_2^r \parallel ... \parallel x_{l-1}^r)) \parallel x_0^r$$

ここで，$\parallel$ は要素の連結を表し，$F_{n_a}^r$ は n_a ビットを入力に $n_b (= n - n_a)$ ビットを出力する関数であり，以下のように

定義される．

$$F^r_{n_a}(x) = (msb_{n_b}(E_K(C_0 \parallel x))) \oplus r$$

ここでの E_K とは n ビットのブロックと k ビットの秘密鍵 K を入力としたブロック暗号であり，SPACE では AES が用いられている．また $msb_{n_b}(x)$ は x の上位 n_b ビット選択を表し，C_0 は n_b ビットの 0 を表す．図 2 に $F^r_{n_a}(x)$ を示す．

ホワイトボックスモデルにおいては，F_{n_a} は 2^{n_a} 通りの $(x, F_{n_a}(x))$ からなるテーブル $F^{'}_{n_a}$ として実装される．

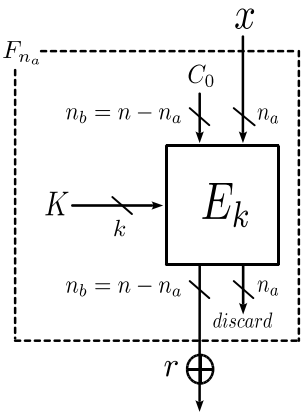

図 2 SPACE の関数 $F^r_{n_a}(x) = (msb_{n_b}(E_K(C_0 \parallel x))) \oplus r$

2.2.2 SPACE の種類

SPACE にはテーブルサイズの異なる以下の 4 つの種類がある．

- SPACE-(8, 300) : $n = 128$, $l = 16$, $n_a = 8$, $F^r_8 : \{0,1\}^8 \to \{0,1\}^{120}$, $R = 300$
- SPACE-(16, 128) : $n = 128$, $l = 8$, $n_a = 16$, $F^r_{16} : \{0,1\}^{16} \to \{0,1\}^{112}$, $R = 128$
- SPACE-(24, 128) : $n = 128$, $l = 16$, $n_a = 24$, $F^r_{24} : \{0,1\}^{24} \to \{0,1\}^{104}$, $R = 128$
- SPACE-(32, 128) : $n = 128$, $l = 4$, $n_a = 32$, $F^r_{32} : \{0,1\}^{32} \to \{0,1\}^{96}$, $R = 128$

この 4 つの種類においてテーブル $F^{'}_{n_a}(x)$ を生成するために AES などのブロック暗号 E_K を SPACE-(8, 300)，SPACE-(16, 128)，SPACE-(24, 128)，SPACE-(32, 128) においてそれぞれ 2^8 回，2^{16} 回，2^{24} 回，2^{32} 回実行する必要がある．

2.3 SPACE のホワイトボックスモデルでの安全性

ホワイトボックスでの SPACE の鍵回復攻撃に対する安全性はブロック暗号 E_K のブラックスボックスモデルにおける鍵回復問題に帰着しており，内部のブロック暗号 E_K が鍵回復攻撃について安全である限り，SPACE の安全性も保障されている．

またすべての種類において SPACE では $(T/4, 128)$-space hardness の安全性が確保されている．つまり，コード（テーブル）の 1/4 以下が盗まれたとしても，その機能はコピーはできない．

2.4 SPACE の問題点

SPACE はホワイトボックスモデルでの安全性は優れているが，実装性能の観点では 2 つの問題点がある．

(1) テーブル $F^{'}_{n_a}(x)$ の生成における効率性

(2) SPACE の暗号化（復号）アルゴリズムにおける効率性

(1) については，テーブルの生成の際に，多くの計算量が必要となる．例えば内部の暗号化アルゴリズムを AES としたとき，SPACE-32 の場合は 2^{32} 回 AES の暗号化アルゴリズムを実行する必要がある．このため SPACE は SSL/TLS，QUIC [6]，Signal protocol [7] などの鍵交換を頻繁にするプロトコルでは利用することができない．

(2) においては，SPACE は Target Heavy Generalized Feistel 構造を採用しているため，安全性を確保するために必要なラウンド数が多く，推奨されている必要なラウンド数は最小でも 128 ラウンドである．このため暗号化（復号）において効率的ではない．

3. 設計目標

本論文では安全性を保ったまま，これらの SPACE の実装性能上の問題点を改善した暗号 Galaxy を提案する．

まず (1) を解決するため，内部で利用する暗号として**ストリーム暗号**を利用する．ソフトウェアにおいてストリーム暗号はブロック暗号と比較して長いメッセージの暗号化は非常に高速であるため，SPACE と比べてテーブル生成に必要な計算量を削減することが可能である．

次に (2) を解決するために SPACE の暗号化アルゴリズムは Target Heavy Feistel 構造を採用していたが，このアルゴリズムを **Type-2 Generalized Feistel** 構造に変更することで暗号化に必要なラウンド数を削減する．Type-2 Generalized Feistel 構造では，シャッフルを変更することで，短いラウンドで安全性が達成可能なことが知られている [3][4][9].

またストリーム暗号を利用することでテーブル実装ではないブラックボックス実装は SPACE と比較すると遅くなるが，一般的にはテーブル実装されるため，実用上問題はない．

4. Galaxy の仕様

Galaxy では図 3 で示すように l 通りのラインの Type-2 Generalized Feistel 構造を採用する．Galaxy は SPACE と同様に 128 ビットの平文と k ビットの秘密鍵 K を入力として 128 ビットの暗号文を出力する．ラインごとのサイズは $n_a = n/l$ とする．r ラウンドにおける中間値を $X^r = \{x^r_0, x^r_1, ..., x^r_l\}$ とすると，次のように定義される．

$$X^{r+1} = \Phi(F^r_{n_a}(x^r_0) \oplus x^r_1 \parallel F^r_{n_a}(x^r_2) \oplus x^r_3 \parallel ... \parallel F^r_{n_a}(x^r_{n-1}) \oplus x^r_n)$$

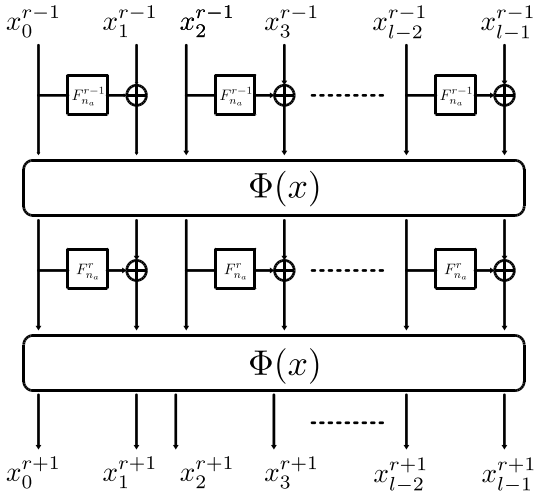

図 3 Galaxy のアルゴリズム

ここで $\Phi(x)$ とは入力 x の 1 バイト単位のシャッフルを意味しており，$\|$ は各値を連結することを意味する．$F_{n_a}^r$ はストリーム暗号から生成される，n_a ビット入出力とした関数で以下のように定義する（図 4）．

$$F_{n_a}^r(x) = F_{n_a}(x) \oplus r$$
$$F_{n_a}(x) = \phi_{n_a}(Keygen(K), x)$$

ここで $Keygen(K)$ は k ビットの秘密鍵 K からストリーム暗号により生成されたキーストリーム系列の集合を表し，$\phi_{n_a}(S, x)$ は S の $x \times n_a$ ビット目から n_a ビットを取り出すことを意味する．Galaxy では $F_{n_a}(x)$ を表 2 で示されたテーブルとして持つ．

x	0	1	2	....	n_{a-2}	n_{a-2}
ϕ(x)	S_0	S_1	S_2	....	S_{n_a-2}	S_{n_a-1}

表 2 Galaxy のテーブル F_{n_a}

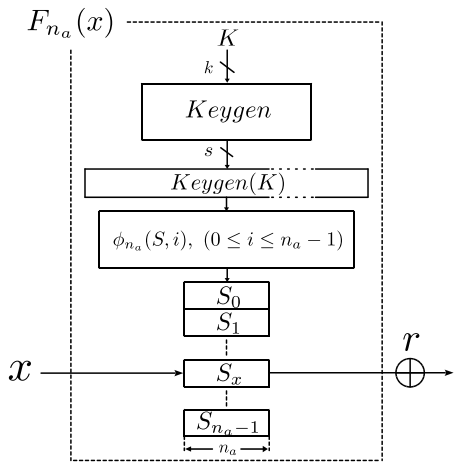

図 4 Galaxy の関数: $F_{n_a}^r(x) = F_{n_a}(x) \oplus r$

Galaxy と Table Generation のアルゴリズムは以下のとおりである．

Algorithm 1 Galaxy Algorithm

$X^0 \Leftarrow$ INPUT
$i, j \Leftarrow 0$
$\{x_0^0, x_1^0, ..., x_{l-1}^0\} \Leftarrow$ SPLIT(X^0) //入力を l 個に分割
while $i < R$ **do**
  while $j < l/2$ **do**
    Update $x_{2*j+1}^i \Leftarrow x_{2*j+1}^i \oplus (F_{n_a}^{'}(x_{2*j}^i) \oplus i)$
  end while
  $\Phi(x_0^i, x_1^i, ..., x_{l-1}^0)$ //要素のシャッフル
  $\{x_0^{i+1}, x_1^{i+1}, ..., x_{l-1}^{i+1}\} \Leftarrow \{x_0^i, x_1^i, ..., x_{l-1}^0\}$
  $i \Leftarrow i + 1$
end while
$X^R \Leftarrow$ COMBINE($\{x_0^R, x_1^R, ..., x_{l-1}^R\}$) //分割した要素を結合
OUTPUT $\Leftarrow X^R$

Algorithm 2 Table Generation

$S \Leftarrow Keygen(K, s)$ //s ビットストリーム鍵を生成
$i \Leftarrow 0$
while $i < n_a$ **do**
  $F_{n_a}^{'}(i) \Leftarrow S[i*n_a : (i+1)*n_a]$ //ストリーム鍵の一部をコピー
end while

4.1 Galaxy の種類

Galaxy はテーブルサイズの異なる以下の 3 つの種類をもつ．

- Galaxy-(8, R) : $n = 128$, $l = 16$, $n_a = 8$, $F_8^r : \{0,1\}^8 \to \{0,1\}^8$
- Galaxy-(16, R) : $n = 128$, $l = 8$, $n_a = 16$, $F_{16}^r : \{0,1\}^{16} \to \{0,1\}^{16}$
- Galaxy-(32, R) : $n = 128$, $l = 4$, $n_a = 32$, $F_{32}^r : \{0,1\}^{32} \to \{0,1\}^{32}$

これら 3 つで用いられるシャッフル演算 $\Phi(x)$ はそれぞれ表 3, 4, 5 に示す．またそれぞれのテーブルサイズ T は 256B，128KB, 16GB である．

5. 安全性

5.1 ホワイトボックスモデルにおける安全性

5.1.1 鍵回復攻撃

ホワイトボックスモデルでは，攻撃者はすべてのラウンドにおけるテーブルの入力と出力にアクセスすることができる．テーブル $F_{n_a}^{'}(x) = \phi_{n_a}(Keygen(K), x)$ から秘密鍵 K を回復するためには，ブラックボックスモデルの状況でテーブルを生成する $Keygen(K)$ から秘密鍵を回復する必要がある．よってホワイトボックスモデルにおける Galaxy の鍵回復攻撃の安全性はストリーム暗号の鍵回復問題に帰着する．よって内部のストリーム鍵生成アルゴリズムがブラックボックスモデルにおいて安全である限り，ホワイトボックスモデルにおける Galaxy の安全性は保証される．

x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\Phi(x)$	5	0	1	4	7	12	3	8	13	6	9	2	15	10	11	14

表 3 Galaxy-8 のシャッフル

x	0	1	2	3	4	5	6	7
$\Phi(x)$	3	0	1	4	7	2	5	6

表 4 Galaxy-16 のシャッフル

x	0	1	2	3
$\Phi(x)$	3	0	1	2

表 5 Galaxy-32 のシャッフル

	F	D	L	ID	I
Galaxy-(8, R)	8	11	15	14	14
Galaxy-(16, R)	6	8	8	10	11
Galaxy-(32, R)	4	5	6	7	7

F : 完全拡散, D : 差分攻撃, L : 線形攻撃,
ID: 不可能差分攻撃, I : Integral 攻撃

表 6 Galaxy の解析結果

5.1.2 Code Lifting 攻撃

$S = Keygen(K)$ から作られたテーブルのサイズ T は $2^{n_a} \times n_a$ ビットである．このときテーブルの一部である $i \leq 2^{n_a}$ が攻撃者に漏洩したとする．この場合，漏洩した情報がテーブルの中にある確率は $i/2^{n_a}$ である．よって攻撃者が 1 ラウンドで任意の入力から正しい中間値を得られる確率は $(i/2^{n_a})^{(l/2)}$ であり，任意の平文 (暗号文) から正しく暗号化 (復号) できる確率は $((i/2^{n_a})^{((l/2)\times R)})$ となる．この確率が実際の Space Hardness となる．よって Galaxy-8, Galaxy-16, Galaxy-32 における (M, Z)-sapce hardness はそれぞれ $(i,\ (i/2^{n_a})^{8\times R})$, $(i,\ (i/2^{n_a})^{4\times R})$, $(i,\ (i/2^{n_a})^{2\times R})$-space hardness となる．よって SPACE と同じ $(T/4,\ 128)$-space hardness を達成するために必要なラウンド数 R はそれぞれ 8，16，32 である．

5.2 ブラックボックスモデルにおける安全性

5.2.1 鍵回復攻撃

ブラックボックスボデルにおいて，攻撃者は中間値であるテーブル $F'_{n_a}(x) = \phi_{n_a}(Keygen(K), x)$ の入力と出力にアクセスすることができない．よってブラックボックスでの Galaxy の鍵回復攻撃は内部のストリーム鍵生成アルゴリズムから秘密鍵を回復する問題より困難である．

5.2.2 差分攻撃

差分攻撃は 2 つの入力の間にある差分 Δx から生じる出力の差分 Δy の関係性を利用して攻撃する手法である．差分攻撃では入力差分 Δx_i からその出力差分 Δy_j が生じる差分確率と呼ばれる確率 DP_f を導出し，その差分確率が最大である最大差分確率 DP_{fmax} から差分特性確率 DCP を導出，利用して攻撃する．入力を n ビットとしたとき差分特性確率 DCP が $DCP \leq 2^n$ となるとき，識別攻撃として成り立たない．差分特性確率 DCP は DP_{fmax} の積で導出される．安全性評価においては入力差分 $\Delta x_i \neq 0$ を入力として取るテーブルの個数 m と最大差分確率 DP_{fmax} から $(DP_{max})^m$ で評価を行う．

本評価では最大差分確率 DP_{fmax} は [2] から引用して Galaxy-8，Galaxy-16，Galaxy-32 においてそれぞれ 2^7，2^{15}，$2^{30.4}$ とする．このとき安全性を確保するために必要なテーブルの個数は Galaxy-8，Galaxy-16，Galaxy-32 においてそれぞれ 17 個，9 個，5 個である．また，これらのテーブルの個数を得るために必要ラウンド数は混合整数線形計画法 (Mixed Integer Linear Programming) [8] から求め，Galaxy-8，Galaxy-16,Galaxy-32 でそれぞれ 15，8，6 ラウンドである．

5.2.3 線形攻撃

線形攻撃は暗号アルゴリズムにおけるテーブルなどの非線形な部分を確率的に線形な部分に近似させる攻撃手法である．線形攻撃では線形マスクと呼ばれるベクトルを利用して線形確率という暗号アルゴリズムを線形的に近似されることができる確率 LP を導出して攻撃を行う．安全性評価においては線形確率 LP が最大となる確率 LP_{max} と LP_{max} を導出するマスクを使ったときに非 0 のマスクの値を入力とするテーブルの個数 m から評価を行い，${LP_{max}}^m$ を求める．この ${LP_{max}}^m$ が $(LP_{max})^m \leq 2^n$ となるとき線形攻撃は識別攻撃として成り立たない．

本評価では最大線形確率 LP_{max} は [2] から引用して Galaxy-8，Galaxy-16，Galaxy-32 において 2^{-4}，2^{-12}，2^{-28} とする．このとき Galaxy-8，Galaxy-16，Galaxy-32 における安全性を確保するために必要なテーブルの個数はそれぞれ 32 個，11 個，5 個である．また，これらのテーブルの個数を得るために必要ラウンド数は混合整数線形計画法 (Mixed Integer Linear Programming) [8] から求め，Galaxy-8，Galaxy-16，Galaxy-32 においてそれぞれ 15，8，6 ラウンド必要である．

5.2.4 その他の攻撃

差分攻撃や線形攻撃の他に不可能差分攻撃や Integral 攻撃に関する解析も混合整数線形計画法 (Mixed Integer Linear Programming) [8] により行った．これらをまとめた結果は表 6 のとおりである．

5.2.5 推奨ラウンド数

$(T/4)$-space hardness の安全性を確保しつつ，ブラックボックスモデルにおける安全性評価の結果から galaxy 群の推奨ラウンドは Galaxy-(8, 25), Galaxy-(16, 20), Galaxy-(32,32) となる．ホワイトボックスモデルにおける安全性は $(T/4, 128)$-space hardness であり，ブラックボックスに

おいても十分な安全なラウンド数である．

6. 実装評価

6.1 実装比較

Galaxy と SPACE のソフトウェア実装上比較を行う．実行環境として OS は CENT OS 7.6，CPU は Xeon E5-2630 v4 2.2GHz である．また Galaxy の内部ストリームキー生成アルゴリズムはソフトウェアで非常に高速な chacha [1] とする[*1]．SPACE の内部暗号化アルゴリズムは AES と AES NI を用いて実装した [5]．コンパイラは gcc 4.8.5 である．

6.1.1 暗号化速度

表 7 では Galaxy-8 と SPACE-8，Galaxy-16 と SPACE-16，Galaxy-32 と SPACE-32 における暗号化（復号）の効率性を比較している．表 7 から暗号化（復号）において Galaxy-8 は SPACE-8 の約 4.5 倍，Galaxy-16 は SPACE-16 の約 5.7 倍，Galaxy-32 は SPACE-32 の約 3.8 倍の早さで暗号化（復号）可能である．

アルゴリズム	ラウンド数	パフォーマンス (cpb)	相対比較
Galaxy-8	25	78.52	0.224
Galaxy-16	20	30.26	0.174
Galaxy-32	32	47.63	0.262
SPACE-8	300	349.56	1
SPACE-16	128	173.01	1
SPACE-32	128	181.39	1

表 7 Galaxy と SPACE の暗号化の実装評価と比較

6.1.2 テーブル生成速度

表 8 では Galaxy-8 と SPACE-8，Galaxy-16 と SPACE-16, Galaxy-32 と SPACE-32 におけるテーブル生成の効率性を比較している．表 8 からテーブル生成において Galaxy-8 は SPACE-8 の約 37 倍，Galaxy-16 は SPACE-16 の約 35 倍，Galaxy-32 は SPACE-32 の約 20 倍の早さで暗号化（復号）に必要なテーブルを生成可能である．

アルゴリズム	テーブルサイズ T	パフォーマンス (cycle)	相対比較
Galaxy-8	256B	761	0.027
Galaxy-16	128KB	192878	0.032
Galaxy-32	16GB	23215671915	0.051
SPACE-8	3.75KB	27218	1
SPACE-16	896KB	6780452	1
SPACE-32	48GB	451234349359	1

表 8 Galaxy と SPACE のテーブル生成の実装評価と比較

表 7，8 から SPACE と同じ (1/4, 128)-space hardness を維持したまま暗号化とテーブル生成に関して大幅な効率化に成功している．

[*1] https://github.com/floodyberry/chacha-opt

7. まとめ

本研究では SPACE のテーブル生成と暗号化の効率性を改善した Galaxy を提案した．SPACE では Target Heavy Feistel 構造を採用していたのに対して，Galaxy では Type-2 Generalized Feistel 構造を採用した．結果として暗号化（復号）において約 3.8 倍から 5.7 倍の効率化を実現した．またテーブル生成においては 20 倍から約 37 倍の効率化を実現した．

8. 謝辞

本研究は JSPS 科研費 19H02141 の助成を受けたものです．

参考文献

[1] J Bernstein and Daniel. Chacha, a variant of salsa20. 01 2008.

[2] Andrey Bogdanov and Takanori Isobe. White-box cryptography revisited: Space-hard ciphers. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 1058–1069, 2015.

[3] Victor Cauchois, Clément Gomez, and Gaël Thomas. General diffusion analysis: How to find optimal permutations for generalized type-ii feistel schemes. *IACR Trans. Symmetric Cryptol.*, 2019(1):264–301, 2019.

[4] Patrick Derbez, Pierre-Alain Fouque, Baptiste Lambin, and Victor Mollimard. Efficient search for optimal diffusion layers of generalized feistel networks. *IACR Trans. Symmetric Cryptol.*, 2019(2):218–240, 2019.

[5] Shay Gueron. Intel® advanced encryption standard (aes) new instructions set. 5 2010.

[6] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan R. Iyengar, Jeff Bailey, Jeremy Dorfman, Jim Roskind, Joanna Kulik, Patrik Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, Wan-Teh Chang, and Zhongyi Shi. The QUIC transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*, pages 183–196, 2017.

[7] Moxie Marlinspike and Trevor Perrin. The double ratchet algorithm. 11 2016.

[8] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In Chuankun Wu, Moti Yung, and Dongdai Lin, editors, *Information Security and Cryptology - 7th International Conference, Inscrypt 2011, Beijing, China, November 30 - December 3, 2011. Revised Selected Papers*, volume 7537 of *Lecture Notes in Computer Science*, pages 57–76. Springer, 2011.

[9] Tomoyasu Suzaki and Kazuhiko Minematsu. Improving the generalized Feistel. In Seokhie Hong and Tetsu Iwata, editors, *FSE 2010*, volume 6147 of *LNCS*, pages 19–39. Springer, Heidelberg, February 2010.