Vol.2013-DPS-154 No.7
Vol.2013-CSEC-60 No.7
2013/3/14

非線形変換を含む単純化ストリーム暗号 KCipher-2 への GD 攻撃

伊藤 竜馬[1,a)]　岩切 宗利[1,b)]

概要：KCipher-2 は，128 ビット秘密鍵と 128 ビット初期ベクトルの計 256 ビットを入力とし，1 サイクル当たり 64 ビットの鍵ストリームを出力するストリーム暗号である．既存研究では，K2 の構成要素である置換関数と転置関数を省略した単純化モデルに対して，GD（Guess and Determine）攻撃に基づくビットスライス解読手法が提案されている．本研究では，従来手法に加えて，非線形変換を含む単純化モデルへの解読手法について検討した．提案手法を用いることで，単純化 KCipher-2 を解読できた．

キーワード：ストリーム暗号，KCipher-2，GD 攻撃

Guess and Determine Attack on Simplified Stream Cipher KCipher-2 with Non Linear Function

Ryoma Ito[1,a)]　Munetoshi Iwakiri[1,b)]

Abstract: KCipher-2 is a stream cipher, outputs a 64-bit keystream on each cycle with a 256-bit input which consists of a 128-bit secret key and a 128-bit initial vector. In the previous study, we suggested the bit slice cryptanalysis based on Guess and Determine Attack without clock control estimate on simplified K2 which made except for Substitution and Permutation Functions. In this study, we improved the previous method, investigated a method of Guess and Determine Attack on simplified KCipher-2 with non linear function, could break the experimental model.

Keywords: Stream Cipher，KCipher-2，Guess and Determine Attack

1. はじめに

KCipher-2 [1–4] は，128 ビット秘密鍵と 128 ビット初期ベクトルの計 256 ビットを入力とし，1 サイクル当たり 64 ビットの鍵ストリームを生成するストリーム暗号 [5–10] である．本報告で示す KCipher-2 とは，文献 [4] に示されたストリーム暗号である．

KCipher-2 の特徴として，線形フィードバックシフトレジスタ（LFSR:Linear Feedback Shift Register）と動的フィードバック制御が挙げられる．KCipher-2 で用いられている動的フィードバック制御は，フィードバック関数を変更する制御を行うため，K2 で用いられているクロック制御 [1,11,12] よりも安全性を向上できる [4]．代表的な動的フィードバック制御型ストリーム暗号として，KCipher-2 のほかに MICKEY [9] がある．

ストリーム暗号への攻撃手法の一種である GD（Guess and Determine）攻撃は，既知平文攻撃を前提として，内部状態の一部を推測（Guess）し，残りの部分を内部状態の更新関数，出力関数，鍵ストリーム等を用いて決定（Determine）する攻撃である [12–16]．ビットスライス解読手法とは，各レジスタが 2 ビット以上の値を持つ暗号に対し，1 ビット毎順に解読する手法である．これらの手法を用いることで，文献 [17] の研究では，K2 の構成要素である置換関数

[1] 防衛大学校情報工学科
National Defense Academy,
Yokosuka, Kanagawa 239-8686, Japan
[a)] f12010@nda.ac.jp
[b)] iwak@nda.ac.jp

と転置関数を省略した単純化モデルの K2 を解読している．文献 [18] の研究では，クロック制御を無効化し，文献 [17] の研究よりも計算量を削減した．

本研究では, 文献 [18] で提案した手法に工夫を加えることで, 一部の非線形変換処理を含む単純化モデルの KCipher-2 を解読した．

2. KCipher-2 の概要

KCipher-2 は，128 ビット秘密鍵と 128 ビット初期ベクトルの計 256 ビットを入力とし，1 サイクル当たり 64 ビットの鍵ストリームを出力するストリーム暗号である．各レジスタは 32 ビットである．KCipher-2 の鍵ストリーム生成部を**図 1** に示す．

KCipher-2 は，2 つの LFSR（LFSR-A，LFSR-B），4 つのメモリ（L1，L2，R1，R2），非線形関数（置換関数，転置関数）で構成され，LFSR-A の出力値を LFSR-B の制御に用いるための動的フィードバック制御部を有する．

LFSR-A と LFSR-B の既約多項式は，

$$f_A(x) = \alpha_0 x^5 + x^2 + 1 \tag{1}$$

$$f_B(x) = (\alpha_1^{c1} + \alpha_2^{1-c1} - 1)x^{11} + x^{10} + x^5 + \alpha_3^{c2} x^3 + 1 \tag{2}$$

である．時刻 t における動的フィードバック制御部からの出力値 $c1$，$c2$ は，

$$c1_t = A_{t+2}[30] \tag{3}$$

$$c2_t = A_{t+2}[31] \tag{4}$$

である．ここで $A_{t+2}[30]$ と $A_{t+2}[31]$ は，時刻 $t+2$ における LFSR-A の 30 ビット目と 31 ビット目の値を表す．31 ビット目はレジスタの最上位ビットである．動的フィードバック制御部からの出力値により，LFSR-B の既約多項式が制御される．出力鍵ストリーム Z_t は，

$$Z_t = (B_{t+10} + L2_t \oplus L1_t \oplus A_t, \quad B_t + R2_t \oplus R1_t \oplus A_{t+4}) \tag{5}$$

である．‘+’ は全加算，‘⊕’ は半加算を表す．Z_t の右辺第 1 項は左側から出力される 32 ビット，第 2 項は右側から出力される 32 ビットである．

動的フィードバック制御部で制御される LFSR-B からの 2 つの出力値と L2，R2 の値を全加算する．その後，L1，R1，LFSR-A の各レジスタから出力を得て，半加算することにより鍵ストリーム Z_t（計 $32 \times 2 = 64$ ビット）が出力される．

L1，R1 は，LFSR-B からの出力値と L2，R2 の全加算，置換と転置（Sub）により更新される．L2，R2 は，L1，R1 の置換と転置により更新される．これが KCipher-2 の 1 サイクル処理の概要である．

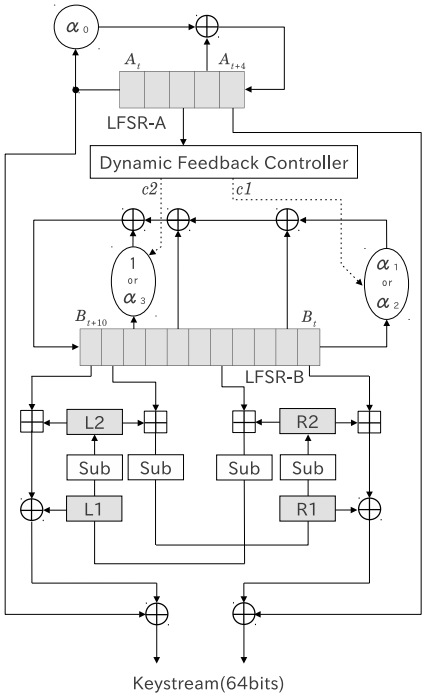

図 1 KCipher-2 の鍵ストリーム生成部

3. 1 ビット単純化 KCipher-2 の解読

3.1 1 ビット単純化モデル

本研究では，文献 [17, 18] に示されているビットスライス解読手法を KCipher-2 に適用するため，各レジスタが 1 ビットの単純化 KCipher-2 を次のとおり定義する．**図 2** は，その単純化モデルの処理フローである．

この単純化モデルでは, 非線形変換を含む単純化モデルの解読手法について検討するため，α_0 の箇所のみ残し，これ以外の置換関数 (α_1,α_2,α_3 を含む) と転置関数を省略した．出力される鍵ストリームは，それぞれの関数に対する入力に依存するため，これらの関数を省略しても KCipher-2 の処理の流れに変化はない．

1 ビット単純化モデルの α_0 に相当する処理を，文献 [18] の研究と同様，乱数生成器（Random Number Generator）を用いて行った．32 ビットの入出力では 1 対 1 対応の関係が成り立つが，1 ビットの入出力ではこの関係が成り立たないため，図 2 のように α_0 の箇所で値をランダムに反転させた．加えて，クロック制御用の 2 ビットも乱数生成器から生成した．各レジスタは 1 ビットであるため，全加算回路を全て半加算回路に置き換えた．このように構成した 1 ビット単純化 KCipher-2 では，各サイクル 2 ビットの鍵ストリームを生成できる．

LFSR-A と LFSR-B の既約多項式は，

$$f_A(x) = x^5 + r0 + x^2 + 1 \tag{6}$$

$$f_B(x) = c1x^{11} + x^{10} + x^5 + c2x^3 + 1 \tag{7}$$

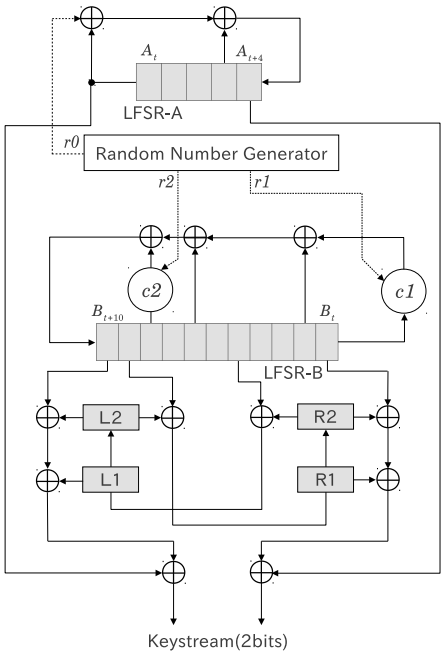

図 2 1 ビット単純化 KCipher-2 モデル

となる．$r0$ は乱数生成器から生成される値である．置換関数 α_1,α_2,α_3 を省略し，クロック制御ビットの処理を単純化した．$c1$，$c2$ の値は，

$$c1_t = r1_t \tag{8}$$

$$c2_t = r2_t \tag{9}$$

となる．$r1_t$，$r2_t$ は，時刻 t において乱数生成器から生成されたクロック制御ビットである．出力鍵ストリーム Z_t は，

$$Z_t = (B_{t+10} \oplus L2_t \oplus L1_t \oplus A_t, \\ B_t \oplus R2_t \oplus R1_t \oplus A_{t+4}) \tag{10}$$

である．この右辺第 1 項は，左側から出力される 1 ビット，第 2 項は右側から出力される 1 ビットである．

3.2 提案手法

3.2.1 GD 攻撃

図 2 に示した乱数生成器を含む 1 ビット単純化 KCipher-2 への GD 攻撃手法を**図 3** に示す．

step1. 図 3（a）の上図（$t = 0$）において，G と表記のある箇所，すなわち $\{A_t, A_{t+3}, A_{t+4}, B_{t+4}, B_{t+9}, L1_t, L2_t, R1_t, R2_t, r0_t\}$ の 10 箇所の値を推測する．その結果，N と表記のある箇所，すなわち $\{B_t, B_{t+10}\}$ の 2 箇所が式（10）より求められる．これを 1 サイクル動かした状態（$t = 1$）が図 3（a）の下図であり，式（6）より A_{t+4} が求められる．D と表記のある箇所が決定したレジスタの内部状態であり，step1 では，10 箇所の

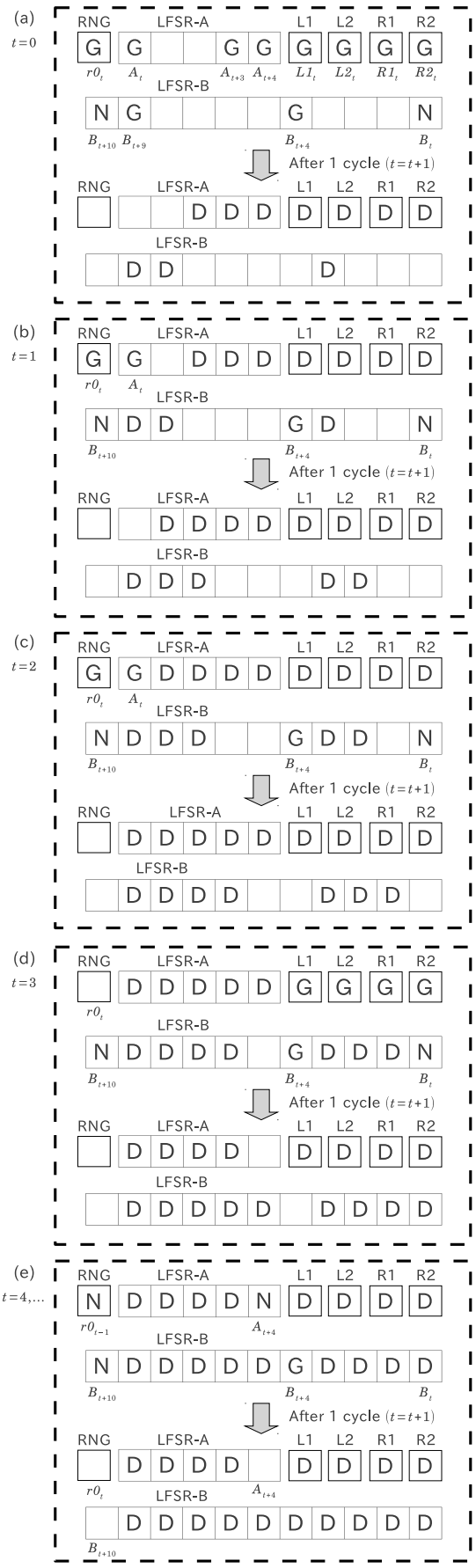

図 3 1 ビット単純化 KCipher-2 への GD 攻撃

推測で 10 箇所の値が定まることを示している．

step2. 図 3（b）のように，$\{A_t, B_{t+4}, r0_t\}$ を推測することで，$\{B_t, B_{t+10}\}$ の 2 箇所が式（10）より求められる．これを 1 サイクル動かす（$t = 2$）と，式（6）より A_{t+4} が求められる．したがって，step2 では，3 箇所の推測で計 13 箇所の値が定まる．

step3. 図 3（c）のように，$\{A_t, B_{t+4}, r0_t\}$ を推測することで，$\{B_t, B_{t+10}\}$ の 2 箇所が式（10）より求められる．これを 1 サイクル動かす（$t = 3$）と，式（6）より A_{t+4} が求められる．したがって，step3 では，3 箇所の推測で計 16 箇所の値が定まる．

step4. 図 3（d）のように，B_{t+4} を推測することで，$\{B_t, B_{t+10}\}$ の 2 箇所が式（10）より求められる．しかし，step4 では $r0_t(= r0_3)$ を推測しない．これを 1 サイクル動かす（$t = 4$）と，$r0_{t-1}(= r0_3)$ が定まらないため A_{t+4} が定まらないが，その理由は後述する．結果として，step4 では，1 箇所の推測で計 17 箇所の値が定まる．

step5. 図 3（e）のように，B_{t+4} を推測することで，B_{t+10} の箇所が式（10）より求められる．これを 1 サイクル動かす（$t = t + 1$）と，step5 では，1 箇所の推測で計 18 箇所の値が定まる．

図 3（e）では，$\{B_t, R2_t, R1_t\}$ の値と鍵ストリームが既知であるため，式（10）より A_{t+4} を求められる．$A_{t+4}(=A_8)$ の値は，式（6）より $A_8 = A_3 \oplus A_6 \oplus r0_3$ で求められる．すなわち，step5 では $\{A_3, A_6, A_8\}$ の値が定まっているため，step4 で推測しなかった $r0_3$ の値が定まる．$r0_3$ の値は，時刻 $t = 4$ で求められるため，図 3（e）における RNG の値を $r0_{t-1}$ と表記している．

この手順を繰り返すことにより，乱数生成器の値は，最初の 3 サイクル分を推測することで，4 サイクル以降の値を一意に決定できる．

既知平文攻撃により得た鍵ストリームと，推測した内部状態から出力した鍵ストリームを比較することで，推測した内部状態の判定を行なう（以後，判定フェイズ）．全ての鍵ストリームが一致すれば解読成功となる．その後，推測した値と決定した値を用いて，初期内部状態を復元できる．提案手法では，20 ビット中 18 ビットを推測することで，残りの 2 ビット及び 4 サイクル目以降の乱数生成器の値を決定した．

3.2.2 擬似クロック制御ビットの生成

提案手法により，理論上では乱数生成器を含む 1 ビット単純化 KCipher-2 を解読できた．しかし，シミュレーション実験では，復元できた初期内部状態のパターンが 2^{18} 通りとなった．この原因は，不確定要素である乱数生成器の値 $r0$ を 3 サイクル分のみ推測し，4 サイクル目以降の $r0$ が LFSR-A の値に依存して定められるためである．$r0$ の値を 4 サイクル目以降も推測することで解決できると考えられるが，その分計算量が増加する．

これを解決する手段として，次の手順による擬似クロック制御ビットの生成法を提案する．

step1. 時刻 $t = 0$ から $t = 10 + cycle$ まで（$cycle$ は既知平文攻撃で得た鍵ストリームのサイクル数）における LFSR-B の各レジスタ（B_t）の値を GD 攻撃で定める．すなわち，図 3（e）で示した処理を $t = cycle$ まで繰り返し実行する．

step2. 式（7）より，LFSR-B を更新するために必要な $\{B_t, B_{t+1}, B_{t+6}, B_{t+8}, B_{t+11}\}$ の値から，クロック制御ビット $c1$，$c2$ の取りうる値を表にまとめたものが **表 1**，**表 2** である．表 1 は $B_t = B_{t+8}$ の場合，表 2 は $B_t \neq B_{t+8}$ の場合を示している．これらの表に記されている '∗' は {0,1} どちらの値も取りうるため，ここでは仮に 0 とする．$B_t = B_{t+8} = 1$ の場合は $c1 = 0$ として，$c2$ の値を定める．

step3. 全ての $cycle$ で表 1，表 2 を参照して $(c1, c2)$ を決定し，これを擬似クロック制御ビットとする．このように生成された擬似クロック制御ビットを，判定フェイズ時の LFSR-B の更新に利用する．

擬似クロック制御ビットを利用したシミュレーション実験の結果，$cycle \geq 128$ の時，復元できる初期内部状態を 8 パターンに絞り込むことができた．

3.3 考察

擬似クロック制御ビットを用いて判定フェイズを行うことで，復元できる初期内部状態のパターン数を削減できた．この理由は，擬似クロック制御ビットが真のクロック制御ビットと同じように機能するためである．例えば，表 1 における $B_t = B_{t+8} = 0$ の $(c1, c2) = (*, *)$ の箇所で

表 1 $B_t = B_{t+8}$ の場合

B_t	B_{t+1}	B_{t+6}	B_{t+8}	B_{t+11}	$(c1, c2)$
0	0	0	0	0	(*,*)
0	0	1	0	0	-
0	1	0	0	0	-
0	1	1	0	0	(*,*)
0	0	0	0	1	-
0	0	1	0	1	(*,*)
0	1	0	0	1	(*,*)
0	1	1	0	1	-
1	0	0	1	0	(0,0),(1,1)
1	0	1	1	0	(0,1),(1,0)
1	1	0	1	0	(0,1),(1,0)
1	1	1	1	0	(0,0),(1,1)
1	0	0	1	1	(0,1),(1,0)
1	0	1	1	1	(0,0),(1,1)
1	1	0	1	1	(0,0),(1,1)
1	1	1	1	1	(0,1),(1,0)

表 2 $B_t \neq B_{t+8}$ の場合

B_t	B_{t+1}	B_{t+6}	B_{t+8}	B_{t+11}	(c_1, c_2)
0	0	0	1	0	(*,0)
0	0	1	1	0	(*,1)
0	1	0	1	0	(*,1)
0	1	1	1	0	(*,0)
0	0	0	1	1	(*,1)
0	0	1	1	1	(*,0)
0	1	0	1	1	(*,0)
0	1	1	1	1	(*,1)
1	0	0	0	0	(0,*)
1	0	1	0	0	(1,*)
1	1	0	0	0	(1,*)
1	1	1	0	0	(0,*)
1	0	0	0	1	(1,*)
1	0	1	0	1	(0,*)
1	1	0	0	1	(0,*)
1	1	1	0	1	(1,*)

は，真のクロック制御ビットが $(c1, c2) = (1, 0)$ だとしても，$(c1, c2) = (0, 0)$ と仮定した処理は同じ結果となる．すなわち，擬似クロック制御ビットを用いることで判定フェイズを強化でき，値の絞り込みができたと考えられる．

1 ビット単純化 KCipher-2 では，18 ビットの値を推測したため，解読に要する計算量 O_1 は，

$$O_1 = 2^{18} \tag{11}$$

である．

4. 32 ビット単純化 KCipher-2 の解読

4.1 32 ビット単純化モデル

各レジスタを 1 ビットから 32 ビットに拡張した単純化モデルを次のとおり定義する．**図 4** はその単純化モデルの処理フローである．

1 ビット単純化 KCipher-2 との相違点は，LFSR-B からの出力と L2，R2 を半加算していた箇所が全加算に置き換わった点，α_0 の箇所とクロック制御ビットの生成を図 1 で示した KCipher-2 の鍵ストリーム生成部と同じ構成とした点である．LFSR-A の既約多項式，クロック制御ビット $c1$，$c2$，出力鍵ストリーム Z_t は，KCipher-2 と同じである．動的フィードバック制御部ではなく，クロック制御部を用いた単純化モデルとしたため，LFSR-B の既約多項式は式（7）のとおりである．

KCipher-2 の全加算回路では，桁上がりが発生するが，文献 [18] の研究と同様の手順を適用することにより，32 ビット単純化モデルに拡張しても対応可能である．

4.2 提案手法による GD 攻撃

1 ビット単純化 KCipher-2 の GD 攻撃では，擬似クロック制御ビットを用いる攻撃手法を示した．32 ビット単純化 KCipher-2 の GD 攻撃では，‘∗’ の箇所を 0 と仮定したため，下位ビットスライスで生成した擬似クロック制御ビットを上位ビットスライスにそのまま適用できない．そこで，擬似クロック制御ビットの生成法を次のように工夫した．

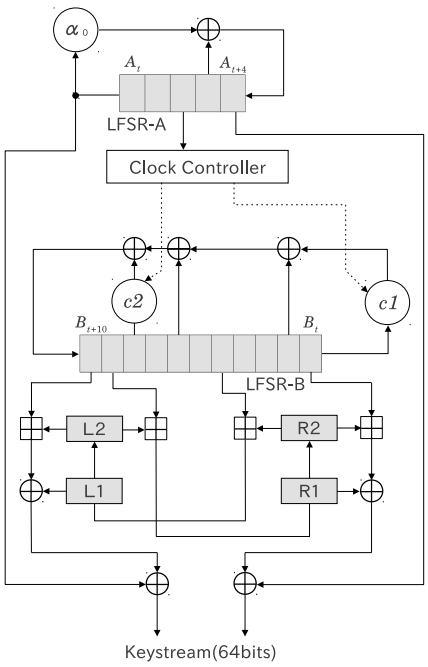

図 4 32 ビット単純化 KCipher-2 モデル

表 2 より，$(B_t, B_{t+8}) = (0, 1)$ のとき $c2$，$(B_t, B_{t+8}) = (1, 0)$ のとき $c1$ が決定する．この値は，真のクロック制御ビットと一致する．このため，決定したクロック制御ビットを，上位ビットスライスにおける擬似クロック制御ビットの生成過程に反映させることで，クロック制御ビットを順に決定できる．シミュレーション実験では，16 ビットスライス目で全 *cycle* のクロック制御ビットが決定できた．

擬似クロック制御ビットの生成法から真のクロック制御ビットを生成でき，これを判定フェイズで利用することで，32 ビット単純化 KCipher-2 を解読できた．シミュレーション実験の結果，2 パターンの初期内部状態を復元できた．この 2 パターンの初期内部状態から得られる出力鍵ストリームは同じ値である．

4.3 考察

提案手法における 32 ビット単純化 KCipher-2 に対し，解読するための計算量 O_2 について示す．

1 ビット単純化 KCipher-2 の GD 攻撃で復元できた初期内部状態が 8 パターン，32 ビット単純化 KCipher-2 では 2 パターンであることに注意する．GD 攻撃のフローチャートを**図 5** に示した．

各ビットスライスで 2^{18} 通りの推測を行うが，判定フェイズを通過した時点で上位ビットスライスの GD 攻撃に移行する．ただし，2^{18} 通りの推測で判定フェイズを通過で

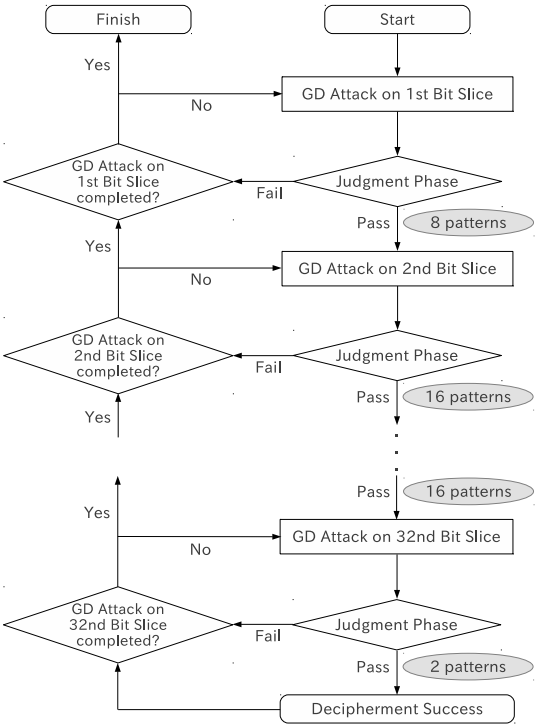

図 5 GD 攻撃のフローチャート

きなかった場合，下位ビットスライスへ戻り，GD 攻撃を継続する．なお，各ビットスライス間で判定フェイズを通過できるパターン数を図 5 の楕円内に記した．この処理を繰り返すことにより，最上位ビットスライスの判定フェイズで，2 パターンの初期内部状態を通過させることができた．

以上のように，下位ビットスライスの 16 パターンの結果を手がかりとして，上位ビットスライスでこの 16 パターンを全て GD 攻撃する必要があるため，その分の計算量が増加する．したがって，32 ビット単純化 KCipher-2 に対する計算量 O_2 は，

$$O_2 = O_1 \times 2^5 \times 2^4 = 2^{18} \times 2^9 = 2^{27} \qquad (12)$$

となる．O_1 は 1 ビット単純化 KCipher-2 に対して解読するための計算量，2^5 はビットスライス数，2^4 は各ビットスライス間で判定フェイズを通過できるパターン数を表す．

KCipher-2 の入力長が 256 ビットであるため，全数探索 2^{256} よりも少ない計算量で解読できた．

5. 実験結果

シミュレーション実験における実験環境を**表 3** に示す．1 ビット単純化 KCipher-2 における内部状態の初期値は，rand 関数を用いてランダムに決定した．32 ビット単純化 KCipher-2 においては，128 ビット秘密鍵と 128 ビット初期ベクトルを rand 関数を用いてランダムに決定し，文献 [4] に記載されている初期化処理を実行した．

表 3 実験環境

計算機	EPSON Endeavor MR7000E
OS	ubuntu 12.04.1 LTS (64 ビット) Linux
メモリ	15.6GB
CPU	Intel Core i7-3770K 3.5GHz × 8
実装言語	C 言語
コンパイラ	gcc 4.6.3

表 4 実験結果

試行回数	1 ビット単純化モデル	32 ビット単純化モデル
1	0.372s	4m30.033s
2	0.376s	4m32.929s
3	0.368s	4m31.449s
4	0.372s	4m30.729s
5	0.372s	4m28.989s
6	0.368s	4m29.357s
7	0.372s	4m31.589s
8	0.376s	4m30.333s
9	0.372s	4m32.437s
10	0.372s	4m32.601s
平均	0.372s	4m31.045s

決定した初期内部状態から鍵ストリームを生成し，提案手法を適用する．1 ビット単純化モデルと 32 ビット単純化モデルに対する GD 攻撃の試行回数を 10 回ずつとした．実行時間とその平均値を示したものが**表 4** である．

実行時間は time コマンドで取得した．time コマンドでは，‘real’，‘user’，‘sys’ 時間を取得できる．プログラム実行中に CPU が処理した時間を測定するため，‘user’ 時間を取得した．なお，本実験における実行時間とは，初期内部状態を決定し，鍵ストリームを出力してから GD 攻撃を開始し，$2^{27}(= O_2)$ 通りの全数探索が終了するまでの時間が含まれる．

式 (12) より，32 ビット単純化モデルに対する計算量 O_2 は，1 ビット単純化モデルに対する計算量 O_1 の $2^9(= 512)$ 倍である．本実験結果では，約 728 倍の解読時間がかかっていることがわかる．この原因は，プログラム開発における実装面の問題であると考える．

32 ビット単純化 KCipher-2 への GD 攻撃における全数探索では，図 5 のようにビットスライス間を上下に行き来しながら解読する．このため，提案した擬似クロック制御ビットの生成法では，下位ビットスライスの GD 攻撃に戻る際，擬似クロック制御ビットの状態を元に戻す処理が必要となる．したがって，1 ビット単純化 KCipher-2 の解読よりも処理が増えたことが，計算量の理論値よりもプログラム実行にかかる時間が増えた原因の一つと考える．

6. まとめ

本研究では，従来手法 [18] に加えて，非線形変換を含む単純化 KCipher-2 への解読手法について検討した．α_0 は最初の 3 サイクル分を推測することで，その後の値を決定

でき，非線形関数はビットスライス解読手法において乱数生成器に置き換え可能であることを示した．

また，擬似クロック制御ビットの生成法を示し，判定フェイズに適用することで効率的に解読できた．擬似クロック制御ビットの生成に関しては，推測することなく決定できるため，クロック制御を無効化できた．

参考文献

[1] 清本晋作 , 田中俊昭, 櫻井幸一: 効率的なクロック制御を用いたストリーム暗号の設計, ISEC2005-166，pp85-90(2006).
[2] Kiyomoto, S., Tanaka, T., Sakurai, K.: *A Word-Oriented Stream Cipher Using Clock Control*, Proc. SASC2007, pp260-273(2007).
[3] Kiyomoto, S., Tanaka, T., Sakurai,K.: *K2:A stream cipher algorithm using dynamic feedback control*, Proc. SECRYPT2007, pp204-213(2007).
[4] KDDI 株式会社: ストリーム暗号 KCipher-2, available from ⟨http://www.cryptrec.go.jp/⟩ (accessed 2013-2-8).
[5] Ekdahl, P., Johansson, T.: *A new vesion of the stream cipher SNOW*, Proc. of SAC2002, LNCS.2595, pp.47-61,Springer-Verlag(2002).
[6] Shah, J., Mahalanobis, A.: *A New Guess-and-Determine Attack on the A5/1 Stream Cipher*, available from ⟨http://eprint.iacr.org/2012/208.pdf⟩ (accessed 2013-2-8).
[7] Watanabe, D., Furuya, S., Yoshida, H., Preneel, B.: *A new key stream generator MUGI*, Proc. FSE2002, LNCS2365, pp.179-194(2002).
[8] Boesgaard, M., Vesterager, M., Pedersen, T., Christiansen, J., Scavenius, O.: *A new high-performance stream cipher*, FSE2003, LNCS2887, pp.307-329(2003).
[9] Babbage, S., Dodd, M.: *The stream cipher MICKEY 2.0*, The eSTREAM Project(2006).
[10] Ekdahl, P.: *On LFSR based Stream Ciphers -Anarysis and Design-*, LUND UNIVERSITY(2003).
[11] 清本晋作 , 田中俊昭, 櫻井幸一: クロック制御型ストリーム暗号におけるクロック制御の効果について, ISEC2005-112，pp11-16(2005).
[12] 清本晋作 , 田中俊昭, 櫻井幸一: クロック制御型ストリーム暗号に対する推測決定攻撃とその評価, 2004-CSEC-26，pp247-254(2004).
[13] 井手口恒太, 渡辺大: 推測決定攻撃に対する安全性評価の一手法, SCIS2008，pp1-6(2008)
[14] Ahmadi, H., Eghlidos, T., Khazaei, S.: *Improver Guess and Determine Attack on SOSEMANUK*, Tehran,Iran(2006)
[15] Feng, X., Liu, J., Zhou, Z., Wu, C., Feng, D.: *A Byte-Based Guess and Determine Attack on SOSEMANUK*, ASIACRYPT2010, LNCS6477, pp.146-157(2010).
[16] Hawkes, P., Rose, G.: *Guess-and-Determine on SNOW*, SAC2002, LNCS2595, pp.37-46(2003).
[17] 大嶋崇士, 岩切宗利: 単純化ストリーム暗号 K2 のビットスライス解読手法, ISEC2008-141, p253-258(2009).
[18] 伊藤竜馬，岩切宗利，単純化ストリーム暗号 K2 のクロック制御を無効化した GD 攻撃, 2012-CSEC-60(2013)