Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8

鍵長128ビット, 192ビット, 256ビットの軽量暗号CLEFIAに対するスキャンベース攻撃手法

於久 太祐[1,a)]　多和田 雅師[1]　柳澤 政生[1]　戸川 望[1]

概要：小型ハードウェアには機密情報のような暗号化を必要とする情報を扱うものが存在する. ハードウェアで機密情報を扱う場合, 暗号化ができる回路を実装する. 特に小型ハードウェアに実装する暗号として, 軽量暗号が存在する. 軽量暗号の中でも, 国際標準規格に採択されているアルゴリズムとしてブロック暗号CLEFIA がある. CLEFIA の鍵長は 128 ビット, 192 ビット, 256 ビットの 3 種類ある. ハードウェア内の機密情報を保護する仕組みを考える場合, 回路に対する攻撃を考慮する必要がある. 暗号回路に対する攻撃であるサイドチャネル攻撃の 1 種として, スキャンベース攻撃が報告されている. スキャンベース攻撃とは回路のテスト容易化技術の 1 つであるスキャンパステストを悪用した攻撃である. スキャンパステストはLSI のレジスタを直列に繋いだスキャンチェイン利用し, 外部から回路内部のレジスタを観測や制御する手法である. 鍵長が 128 ビットの CLEFIA 暗号回路に対するスキャンベース攻撃が存在するが, 鍵長が 192 ビット, 256 ビットである CLEFIA アーキテクチャに対するスキャンベース攻撃手法は存在しない. 本稿では鍵長 128 ビット, 192 ビット, 256 ビットの軽量暗号 CLEFIA に対するスキャンベース攻撃を提案する. 提案手法では多数の入力メッセージを使い, 内部レジスタとスキャンチェインから得られるスキャンデータの対応付けにより攻撃する. 計算機実験により, CLEFIA 回路とその他制御回路などの周辺回路のレジスタがスキャンチェインに存在している場合でも, 提案手法によりを用いることで CLEFIA 回路のラウンド鍵を特定できることを確認した.

キーワード：サイドチャネル攻撃, スキャンベース攻撃, ブロック暗号, CLEFIA, 軽量暗号

1. はじめに

近年, クレジットカードや公的機関が発行するカード, 医療機器, IoT (Internet of Things) デバイスなどの小型ハードウェアが普及している. これらのデバイスでは個人情報や生体情報のような機密情報を扱うことが考えられる. 機密情報は流出を防ぐために暗号化し扱う. そのため, ハードウェアに暗号回路を組み込む必要がある. 特に, 小型ハードウェアでは十分な計算リソースがなく, 供給可能な電力も少ないため, 搭載する暗号は軽量暗号になると考えられる. 軽量暗号の国際標準規格に採択されている暗号アルゴリズムとして CLEFIA [1] がある. 暗号回路が機密情報を流失させないか評価する必要がある.

暗号回路を攻撃する手法としてサイドチャネル攻撃が存在する. サイドチャネル攻撃は外部から動作中の回路を物理的観測・計測することで得られる情報を使い機密情報を復元する手法である. サイドチャネル攻撃にはタイミング攻撃 [2], 故障解析攻撃 [3,4], 電力差分攻撃 [5] が報告されており, スキャンパステストに用いるスキャンチェインを悪用したサイドチャネル攻撃として, スキャンベース攻撃 [6–10] が報告されている. スキャンパステストは LSI のテスト容易化技術の 1 つである. LSI チップ内のレジスタを直列に接続するスキャンチェインを使ってテストする. スキャンベース攻撃はスキャンチェインから内部レジスタの値を取得できることを利用し, スキャンチェインから得られるスキャンデータを解析することで暗号回路を攻撃する. スキャンチェインにアクセスが可能である LSI チップが存在することが指摘されている [11,12].

スキャンベース攻撃の先行研究としてブロック暗号 AES に対する手法 [6], ストリーム暗号 Trivium に対する手法 [7], 公開鍵暗号 RSA に対する手法 [8], ハッシュ関数を用いたメッセージ認証符号 HMAC-SHA-256 に対する手法 [9] がある. 先行研究はスキャンチェインから暗号化中のレジスタの値を取得できることを前提としている. いずれの手法もシミュレーション実験では, スキャンベース攻撃できることが示されている. 我々は鍵長 128 ビットの CLEFIA 暗

[1] 早稲田大学大学院基幹理工学研究科 情報理工・情報通信専攻
[a)] daisuke.oku@togawa.cs.waseda.ac.jp

Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8

号回路に対しての攻撃手法 [10] を提案した. 提案手法は鍵スケジュール部とデータ処理部が共有されているアーキテクチャを対象としており, 共有していないアーキテクチャに対しては有効でない. また, 軽量暗号 CLEFIA には鍵長が 128 ビット, 192 ビット, 256 ビットの 3 種類あり, 他の鍵長へ対応できない.

本稿では鍵長 128 ビット, 192 ビット, 256 ビットの CLEFIA 暗号回路に対するスキャンベース攻撃の手法を提案する. 攻撃対象のアーキテクチャは鍵スケジュール部とデータ処理部を共有していない. 提案手法は入力メッセージを使い, 内部レジスタとスキャンチェインから得られるスキャンデータの対応付けする既存手法 [10] に新たにラウンド鍵を特定するステップを加えることで攻撃する. ラウンド鍵を特定することで, 中間鍵と秘密鍵を復元し, 暗号文から平文を復元する. 計算機実験により, CLEFIA 回路とその他制御回路などの周辺回路のレジスタがスキャンチェインに存在している場合でも, 提案手法を用いることで CLEFIA 回路のラウンド鍵を特定することに成功した.

本稿の貢献は以下の通りである.

(1) 鍵スケジュール部とデータ処理部を共有してない鍵長 128 ビット, 192 ビット, 256 ビットの CLEFIA 暗号回路に対するスキャンベース攻撃を提案する. 提案手法はスキャンチェインに周辺回路のレジスタが存在しても攻撃可能である.

(2) 計算機実験により, 提案手法を使い暗号化部動作中のスキャンデータのみでラウンド鍵の特定に成功した.

2. CLEFIA [1]

本章では軽量暗号 CLEFIA を紹介する. CLEFIA は ISO/IEC 29192 軽量暗号の国際標準規格に採択されているブロック暗号アルゴリズムである. 暗号化ブロック長は 128 ビット, 秘密鍵の鍵長は 128 ビット, 192 ビット, 256 ビットがある. 本章では鍵長 128 ビット, 192 ビット, 256 ビットの CLEFIA のアルゴリズムを紹介する. CLEFIA のアルゴリズムは鍵スケジュール部とデータ処理部で構成されている. 鍵スケジュール部では暗号化に使うホワイトニング鍵と中間鍵を生成する. データ処理部ではホワイトニング鍵とラウンド鍵を使い, 平文を暗号化する. ラウンド鍵は中間鍵から生成される. CLEFIA では鍵長によって鍵スケジュール部のアルゴリズムが異なり, また, データ処理部のアルゴリズムではラウンド数が異なる.

2.1 データ処理部

データ処理部では 128 ビットの平文 PT をホワイトニング鍵とラウンド鍵を使って 128 ビットの暗号文 CT を出力する. CLEFIA の暗号アルゴリズムは 4 系列一般化 Feistel 構造をしており, 1 ラウンドで 2 つの F 関数 F_0, F_1 を用いる. CLEFIA のデータ処理部の構造を図 1 に示す.

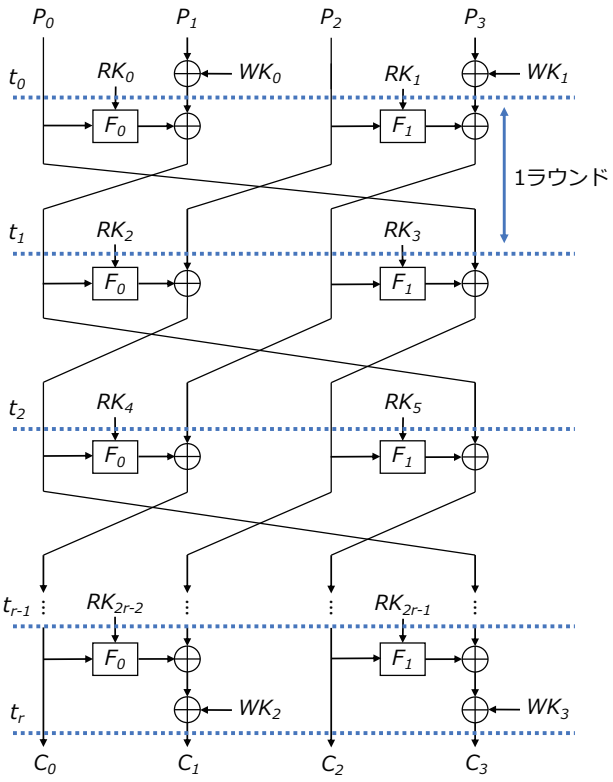


図 1: CLEFIA の 4 系列一般化 Feistel 構造 [1].

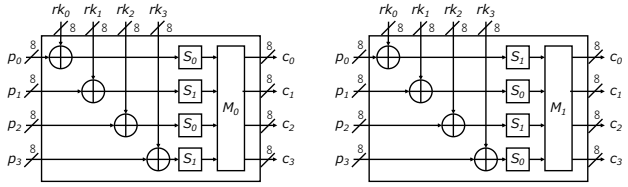
P_i $(0 \le i \le 3)$ は 128 ビットの平文 PT を 4 分割した 32 ビット部分平文である. C_i $(0 \le i \le 3)$ は 128 ビットの暗号文 CT を 4 分割した 32 ビット部分暗号文である. 32 ビットのホワイトニング鍵 WK_i $(0 \le i \le 3)$ と 32 ビットのラウンド鍵 RK_j を使い平文を暗号化する. ラウンド鍵の個数 j は鍵長 128 ビットでは $j = 36$, 鍵長 192 ビットでは $j = 44$, 鍵長 256 ビットでは $j = 52$ である.

F 関数 F_0, F_1 の構造を図 2 に示す. F 関数は 2 つの 32 ビットの入力 P_i, RK_j から 1 つの 32 ビットの出力を得る関数である. rk_i $(0 \le i \le 3)$ はそれぞれ 8 ビットであり, 32 ビットのラウンド鍵 RK_j を 4 分割したものである. p_i $(0 \le i \le 3)$ はそれぞれ 8 ビットであり, 演算途中の 32 ビットの部分平文を 4 分割したものである. F 関数の演算内容としては 2 種類の入力 P_i, RK_j の排他的論理和, 2 種類の S-box S_0, S_1 による変換, 1 つの拡散行列 M_0 もしくは M_1 との乗算の 3 つで構成されている. 2 種類の S-box S_0, S_1 はそれぞれ 8 ビットの入力 w に対し, 対応した 8 ビットの出力 $S_0(w)$, $S_1(w)$ を返す. 2 つの行列 M_0 と M_1 を以下に示す. 行列の各要素は 16 進数表現である.

$$M_0 = \begin{pmatrix} 01 & 02 & 04 & 06 \\ 02 & 01 & 06 & 04 \\ 04 & 06 & 01 & 02 \\ 06 & 04 & 02 & 01 \end{pmatrix}, M_1 = \begin{pmatrix} 01 & 08 & 02 & 0A \\ 08 & 01 & 0A & 02 \\ 02 & 0A & 01 & 08 \\ 0A & 02 & 08 & 01 \end{pmatrix}$$

加算は排他的論理和として計算され, 乗算は辞書的順序で最初となる原始多項式 $z^8 + z^4 + z^3 + z^2 + 1 = 0$ で定義されている GF (2^8) 上の演算として計算される.

Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8



(a) F_0 の概略. (b) F_1 の概略.

図 2: F 関数の概略 [1].

Algorithm 1 ラウンド鍵 RK_j の生成 (128 ビット)

for $j = 0$ to 8 **do**
  $CON \leftarrow B^{128}_{24+4j}|B^{128}_{24+4j+1}|B^{128}_{24+4j+2}|B^{128}_{24+4j+3}$;
  $T \leftarrow L \oplus CON$;
  $L \leftarrow \Sigma(L)$;
  if $j \bmod 2 = 1$ **then**
    $T \leftarrow T \oplus K$;
  end if
  $RK_{4j}|RK_{4j+1}|RK_{4j+2}|RK_{4j+3} \leftarrow T$;
end for

ラウンド鍵は中間鍵 (L もしくは L_L, L_R) から生成される. 鍵長 128 ビットのラウンド鍵 RK_j の生成アルゴリズムを Algorithm 1 に示す. 鍵長 192 ビット, 256 ビットのラウンド鍵 RK_j の生成アルゴリズムを Algorithm 2 に示す. B^k は鍵長 ($k = 128, 192, 256$) ごとに決められた定数である. Algorithm 1, Algorithm 2 中の関数 Σ を式 1 に示す.

$$\Sigma(x) = x_{7:63}|x_{0:6}|x_{121:127}|x_{64:120} \quad (1)$$

$x_{b:c}$ は変数 x の b ビット目から c ビット目を表す. 関数 Σ を使い中間鍵 (L もしくは L_L, L_R) を更新しながらラウンド鍵を生成する. ホワイトニング鍵と中間鍵の生成方法は 2.2 節で示す.

2.2 鍵スケジュール部

鍵スケジュール部では秘密鍵を入力とし, 暗号化で使用する 32 ビットのホワイトニング鍵と中間鍵を生成する. 鍵スケジュール部はホワイトニング鍵生成フェーズと中間鍵生成フェーズの 2 つのフェーズがある.

ホワイトニング鍵生成フェーズでは鍵長 128 ビットの場合, 秘密鍵 K^{128} を 32 ビット毎に 4 分割する.

$$WK_1|WK_2|WK_3|WK_4 = K_0|K_1|K_2|K_3 = K^{128}$$

鍵長 192 ビットの場合, 秘密鍵 K^{192} を 32 ビット毎に 6 分割し, 以下の式で生成する.

$$K^{192} = K_0|K_1|K_2|K_3|K_4|K_5$$
$$K_L \leftarrow K_0|K_1|K_2|K_3, K_R \leftarrow K_4|K_5|\overline{K_0}|\overline{K_1}$$
$$WK_1|WK_2|WK_3|WK_4 = K_L \oplus K_R$$

Algorithm 2 ラウンド鍵 RK_j の生成 (192, 256 ビット)

for $j = 0$ to 10 ($k = 192$) or 12 ($k = 256$) **do**
  $CON \leftarrow B^k_{40+4j}|B^k_{40+4j+1}|B^k_{40+4j+2}|B^k_{40+4j+3}$;
  if $j \bmod 4 = 0$ or 1 **then**
    $T \leftarrow L_L \oplus CON$;
    $L_L \leftarrow \Sigma(L_L)$;
    if $j \bmod 2 = 1$ **then**
      $T \leftarrow T \oplus K_R$;
    end if
  else
    $T \leftarrow L_R \oplus CON$;
    $L_R \leftarrow \Sigma(L_R)$;
    if $j \bmod 2 = 1$ **then**
      $T \leftarrow T \oplus K_L$;
    end if
  end if
  $RK_{4j}|RK_{4j+1}|RK_{4j+2}|RK_{4j+3} \leftarrow T$;
end for

鍵長 256 ビットの場合, 秘密鍵 K^{256} を 32 ビット毎に 8 分割し, 以下の式で生成する.

$$K^{256} = K_0|K_1|K_2|K_3|K_4|K_5|K_6|K_7$$
$$K_L \leftarrow K_0|K_1|K_2|K_3, K_R \leftarrow K_4|K_5|K_6|K_7$$
$$WK_1|WK_2|WK_3|WK_4 = K_L \oplus K_R$$

中間鍵生成フェーズでは鍵長 128 ビットの場合, 図 1 のような 4 系列一般化 Feistel 構造を使い 128 ビットの中間鍵 L を生成する. ラウンド数は $r = 12$ である. ラウンド鍵として 32 ビットの定数 B^{128}_a ($0 \le a \le 23$) を入力する. 鍵長 192 ビットと 256 ビットの場合, 8 系列一般化 Feistel 構造を使い 128 ビットの 2 つの中間鍵 L_L, L_R を生成する. ラウンド数は $r = 10$ である. 8 系列一般化 Feistel 構造を図 3 に示す. ラウンド鍵として 32 ビットの定数 B^{192}_a, B^{256}_a ($0 \le a \le 39$) を入力する.

3. 攻撃の前提条件

本章では攻撃となる CLEFIA 暗号回路のアーキテクチャを示し, 攻撃の前提条件を明示する.

3.1 対象となるアーキテクチャ

攻撃の対象となるアーキテクチャは図 4 の構造を実現する回路である. 図 4 はデータ処理部と鍵スケジュール部で構成されている. データ処理部と鍵スケジュール部はレジスタを含んでおり, スキャンチェインが接続されている. データ処理部は暗号化部とラウンド鍵生成で構成されており, 暗号化部は図 1 を実現する回路である. 暗号化部は図 1 の破線のタイミング t_n ($0 \le n \le r$) で値がレジスタに保存

Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8

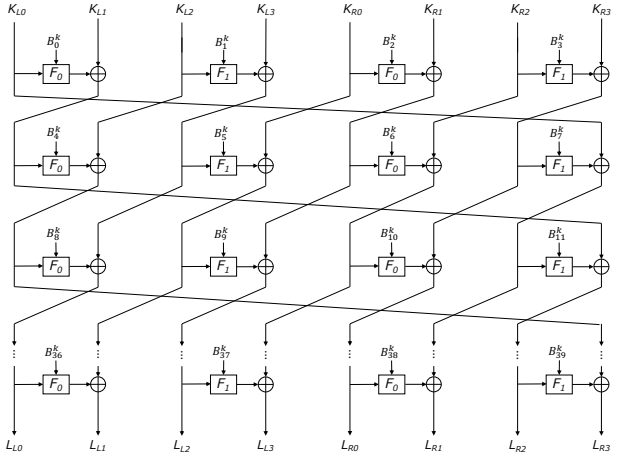


図 3: CLEFIA の 8 系列一般化 Feistel 構造 [1].

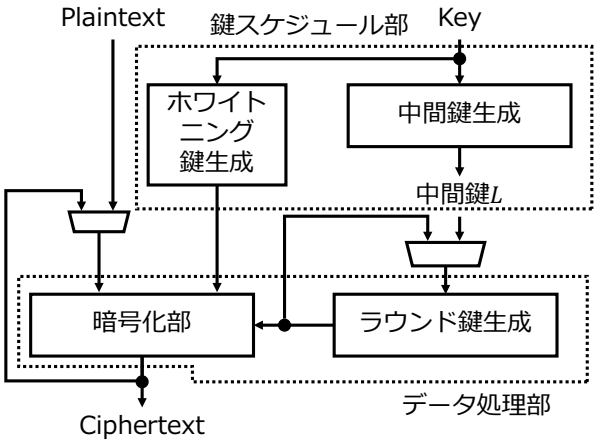


図 4: 攻撃対象のアーキテクチャ.

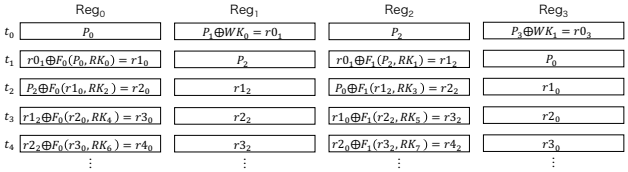


図 5: 暗号化部でレジスタに保存されている値.

される. 各ラウンドにおいて, 暗号化部で使われるレジスタに保存されている値を図 5 に示す. 図 5 の Reg_i $(0 \leq i \leq 3)$ はそれぞれ 32 ビットのレジスタを表している. t_0 では部分平文 P_1, P_3 はホワイトニング鍵の上位部 $(WK_0|WK_1)$ と排他的論理和される. 暗号化中はラウンド鍵生成で中間鍵 L を更新しながら暗号化する. 鍵長に応じたラウンド数後に暗号文を出力する.

鍵スケジュール部はホワイトニング鍵生成と中間鍵生成から構成されており, 中間鍵生成は図 1 もしくは図 3 を実現する回路である. 鍵長 128 ビットの場合, 中間鍵生成の回路と暗号化部の回路は共有できる. しかし, 鍵長 192 ビットと 256 ビットの場合, 中間鍵生成の回路と暗号化部の回路は共有できない. そのため, 提案する手法では, データ処理部動作中に得られるスキャンデータのみを使い暗号文を復号化することを目標とする.

3.2 攻撃の前提条件

攻撃の目的は暗号文から平文を復元することである. 復元するために暗号化に使われるラウンド鍵を特定する. スキャンベース攻撃の既存手法 [7,9,10] と同様に攻撃の前提条件を示す. 攻撃者はスキャンチェインはフルスキャン設計であり, 反転, 圧縮, 動的に変化しないことと暗号化のタイミングがわかる. 攻撃者は LSI に任意の平文を入力して暗号化でき, 任意のタイミングで LSI のスキャンチェインにアクセスでき, スキャンデータを取得できる. 攻撃者はスキャンチェインに含まれるレジスタの接続順と個数がわからないとする.

4. 提案するスキャンベース攻撃手法

本章では鍵長 128 ビット, 192 ビット, 256 ビットの CLEFIA 回路に対するスキャンベース攻撃手法を提案する. 我々は鍵長 128 ビットの CLEFIA 回路に対するスキャンベース攻撃 [10] を提案したが, データ処理部と鍵スケジュール部を共有しているアーキテクチャを対象としていた. 鍵長が 192 ビット, 256 ビットの場合, データ処理部と鍵スケジュール部を共有できないと考えられる.

攻撃の目的は秘密鍵を特定し, 暗号文から平文を復元することである. 秘密鍵の特定のために, ラウンド鍵を特定する. 提案攻撃手法はスキャンデータから CLEFIA 暗号回路の内部レジスタの特定 [10], 内部レジスタからラウンド鍵の特定の 2 ステップで構成されている.

4.1 暗号化部の内部レジスタ特定 [10]

3.2 節より, スキャンチェインに含まれるレジスタの接続順がわからないので, スキャンチェインから得られるスキャンデータのある 1 ビットと回路内部のある 1 ビットレジスタとの対応関係は不明である. 攻撃者から見るとスキャンデータは意味のないデータ列に見える. 本節では多数の平文を入力し, 得られるスキャンデータを使い, スキャンデータと暗号化部の内部レジスタを対応付けを説明する.

暗号化部は図 1 の破線のタイミング t_n $(0 \leq n \leq r)$ で値がレジスタに保存される. 図 5 の t_0 では部分平文 P_0, P_2 が挿入されるレジスタ Reg_0, Reg_2 があり, t_1 では部分平文 P_0, P_2 が挿入されるレジスタ Reg_1, Reg_3 がある. 3.2 節より, 攻撃者は平文を自由に回路に入力できるので, t_0 で Reg_0 と Reg_2 には自由に入力できる. 同様に, t_1 で Reg_1 と Reg_3 にも自由に入力できる. しかし, 周辺回路は制御できない. 1 つの平文だけでは周辺回路のレジスタと Reg_i $(0 \leq i \leq 3)$ の値は区別できない. そのため, 多数の平文を入力し, 得られるスキャンデータ中のある 1 ビットに注目する. 入力した順に平文を縦に並べ, 取得したスキャンデータも縦に並べる. 平文とスキャンデータを縦に見ると平文とスキャンデータそれぞれで, ある 1 ビットの変化が読み取れる. ある 1 ビットの変化列をスキャンシグネチャと呼ぶ. スキャ

Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8

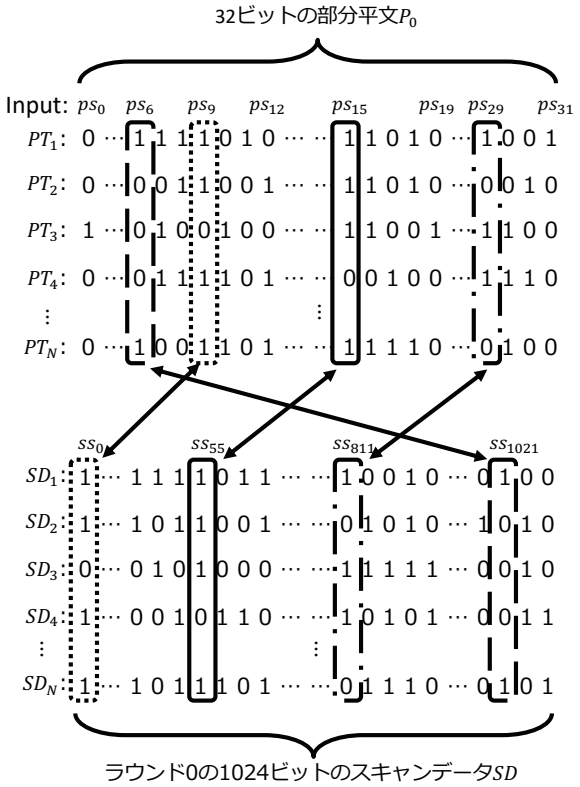


図 6: スキャンシグネチャを用いたビット位置の特定 [10].

ンシグネチャを使用して, 得られるスキャンデータと内部レジスタを対応付けする.

平文 PT を 4 分割した 1 つの部分平文 P_0 とスキャンデータの対応付けを図 6 に示す. 図 6 では例としてスキャンデータを 1024 ビットとした. 図 6 の上部では平文 PT を N 個用意し, 縦に並べる. 図 6 の下部では平文に対応した N 個の t_0 時のスキャンデータ SD を縦に並べる. 図 5 より, t_0 時には Reg_0 に P_0 が保存されている. そのため, 平文のスキャンシグネチャ ps_l $(0 \leq l \leq 31)$ はスキャンデータのスキャンシグネチャ ss_m $(0 \leq m \leq 1023)$ 中のいずれかと一致する. 図 6 では, 平文のスキャンシグネチャをスキャンデータのスキャンシグネチャと比較した結果, $ps_6 = ss_{1021}$, $ps_9 = ss_0$, $ps_{15} = ss_{55}$, $ps_{29} = ss_{811}$ となっている. 比較を繰り返すことでスキャンデータから部分平文 P_0 が求まると考えられる. つまり, スキャンデータから Reg_0 のビット位置が特定できる. 同様に部分平文 P_2 を使うことで, Reg_2 のビット位置を求めることができる. Reg_1, Reg_3 のビット位置を求めるためには, 部分平文 P_2 と P_0 を使い, t_1 時のスキャンデータ中を探索する. 平文の数 N を多く取ることで特定できる可能性が高まる.

4.2 ラウンド鍵の特定

鍵長 128 ビットの CLEFIA 回路に対するスキャンベース攻撃手法 [10] では, データ処理部と鍵スケジュール部を共有しているアーキテクチャを対象としていたため, 4.1 節だけで秘密鍵が復元できた. 本稿では, データ処理部と鍵スケジュール部を共有していないアーキテクチャを対象としているため, 提案されている手法では秘密鍵を復元できない. そのため, データ処理部の内部レジスタ特定後にラウンド鍵を特定する. ラウンド鍵 RK_j を特定できれば, Algorithm 1, Algorithm 2 により, 2 つの中間鍵と秘密鍵が復元できると考えられる.

図 5 中の値を使ってラウンド鍵の特定を説明する. t_1 時の Reg_0 に保存されている値 $r1_0$ からラウンド鍵 RK_0 を特定する. M_0 の逆行列 M_0^{-1} と t_0 時の Reg_1 に保存されている値 $r0_1$ を用いて RK_0 を求める式を式 2 に示す.

$$\begin{aligned} M_0^{-1}(r1_0 \oplus r0_1) &= M_0^{-1}(r0_1 \oplus F_0(P_0, RK_0) \oplus r0_1) \\ &= M_0^{-1} F_0(P_0, RK_j) \\ &= \begin{pmatrix} S_0(p_0 \oplus rk_0) \\ S_1(p_1 \oplus rk_1) \\ S_0(p_2 \oplus rk_2) \\ S_1(p_3 \oplus rk_3) \end{pmatrix} \end{aligned} \tag{2}$$

rk_i $(0 \leq i \leq 3)$ はそれぞれ 8 ビットであり, 32 ビットのラウンド鍵 RK_0 を 4 分割したものである. p_i $(0 \leq i \leq 3)$ はそれぞれ 8 ビットであり, 32 ビットの部分平文 P_0 を 4 分割したものである. 2 種類の S-box S_0, S_1 はそれぞれ 8 ビットの入力 w に対し, 対応した 8 ビットの出力 $S_0(w)$, $S_1(w)$ を返す. そのため, 2 種類の S-box の逆関数である S_0^{-1}, S_1^{-1} は既知である. 4.1 節よりスキャンデータから Reg_1 の値も特定できているため, $r0_1$ の値も求めることができる. ラウンド鍵 RK_j を特定するためには M_0^{-1} を使えばよい. M_0 の逆行列は M_0 自身であるから, 式 2 の $M_0^{-1} = M_0$ とすれば, RK_0 を特定できる. また, M_1 の逆行列も M_1 自身となるので, F_1 関数に関しても式 2 と同様の計算ができる.

Algorithm 1 より鍵長 $k = 128$ の場合,

$$RK_0|RK_1|RK_2|RK_3 \leftarrow L \oplus (B_{24}^k|B_{25}^k|B_{26}^k|B_{27}^k)$$
$$RK_4|RK_5|RK_6|RK_7 \leftarrow \Sigma(L) \oplus K^k \oplus (B_{24}^k|B_{25}^k|B_{26}^k|B_{27}^k)$$

となる. RK_0 から RK_7 まで特定すれば, 中間鍵と秘密鍵を復元できる. Algorithm 2 より鍵長 $k = 192, 256$ の場合,

$$\begin{aligned} RK_0|RK_1|RK_2|RK_3 &\leftarrow L_L \oplus (B_{40}^k|B_{41}^k|B_{42}^k|B_{43}^k) \\ RK_4|RK_5|RK_6|RK_7 &\leftarrow \Sigma(L_L) \oplus K_R^k \\ &\quad \oplus (B_{44}^k|B_{45}^k|B_{46}^k|B_{47}^k) \\ RK_8|RK_9|RK_{10}|RK_{11} &\leftarrow L_R \oplus (B_{48}^k|B_{49}^k|B_{50}^k|B_{51}^k) \\ RK_{12}|RK_{13}|RK_{14}|RK_{15} &\leftarrow \Sigma(L_R) \oplus K_L^k \\ &\quad \oplus (B_{52}^k|B_{53}^k|B_{54}^k|B_{55}^k) \end{aligned}$$

となる. RK_0 から RK_{15} まで特定すれば, 中間鍵と秘密鍵を復元できる. 中間鍵からはラウンド鍵を全て生成でき, 秘密鍵からはホワイトニング鍵が生成できる. 暗号文から平文の復元が可能であると考えられる.

Vol.2018-ARC-230 No.43
Vol.2018-SLDM-183 No.43
Vol.2018-EMB-47 No.43
2018/3/8

表 1: 鍵長ごとの特定時間と平均時間.

Trials	特定時間 [s] 128 ビット	192 ビット	256 ビット
1	0.002879	0.002302	0.001798
2	0.000792	0.000962	0.001138
3	0.000747	0.001153	0.001041
4	0.000742	0.001003	0.001096
5	0.000751	0.000971	0.001049
6	0.000751	0.000991	0.001038
7	0.000736	0.000962	0.001048
8	0.001064	0.001025	0.001092
9	0.000746	0.000962	0.001047
10	0.000719	0.001482	0.002185
Average	0.000993	0.001181	0.013432

5. 評価実験

本章では 4 章で提案した手法を用いて, スキャンデータから内部レジスタを特定し, ラウンド鍵を特定する実験の結果を示す. 評価実験では, 提案手法と CLEFIA のシミュレータを python を用いて実装した. ホスト PC の OS は macOS High Sierra, CPU は Intel Core i5 (2.9GHz), メモリは 8GB を用いた.

5.1 実験方法

攻撃対象の回路は図 4 を実現する回路とした. 回路から得られるスキャンデータは図 1 の破線のタイミングで保存される値とした. 4.2 節で提案したラウンド鍵の特定する実験の結果を示す. 4.1 節の暗号化部の内部レジスタの特定は終了してるものとする.

5.2 実験結果

内部レジスタの特定後のスキャンデータからラウンド鍵の特定に必要な時間を計測した. 特定するラウンド鍵は鍵長が 128 ビットの場合, ラウンド鍵 RK_0 から RK_7 とし, 鍵長 192 と 256 ビットの場合, ラウンド鍵 RK_0 から RK_{15} とした. それぞれの鍵長で 10 個の秘密鍵を用意し, 10 個全てのラウンド鍵の特定に成功した. ラウンド鍵の特定に必要な時間と平均時間を表 1 に示す.

6. おわりに

本稿では軽量暗号 CLEFIA を紹介し, 攻撃対象アーキテクチャを説明し, CLEFIA に対するスキャンベース攻撃手法を提案し, 提案手法に対して評価実験した. 提案した鍵長 128 ビット, 192 ビット, 256 ビットの CLEFIA に対するスキャンベース攻撃手法は, 平文を入力することで, スキャンデータと内部レジスタを対応付けし, 逆行列を用いることでラウンド鍵を特定し秘密鍵を復元する. 計算機実験より, CLEFIA 暗号回路とその他周辺回路のレジスタがスキャンチェインに接続されていてもラウンド鍵を特定できることを確認した.

今後の課題としては, スキャンベース攻撃に対する防御手法の提案がある.

謝辞 本研究開発は一部, 総務省 SCOPE (受付番号 171503005) の委託を受けた.

参考文献

[1] Sony Croporation, “The 128-bit blockcipher CLEFIA algorithm specification,” `https://www.sony.co.jp/Products/cryptography/clefia/download/data/clefia-spec-1.0.pdf`.

[2] P. C. Kocher, “Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems,” *Lecture Notes in Computer Science*, vol. 1109, pp. 104–113, 1996.

[3] E. Biham and A. Shamir, “Differential fault analysis of secret key cryptosystems,” *Lecture Notes in Computer Science*, vol. 1294, pp. 513–525, 1997.

[4] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults,” *Lecture Notes in Computer Science*, vol. 1233, pp. 37–51, 1997.

[5] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” in *Proc. CRYPTO ‘99*, Springer-Verlag, pp. 388–397, 1999.

[6] R. Nara, N. Togawa, M. Yanagisawa, and T.Ohtsuki, “A scan-based attack based on discriminators for AES cryptosystems,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E92-A. no. 12, pp. 3229–3237, 2009.

[7] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based attack against Trivium stream cipher using scan signatures,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A no. 7 pp. 1444–1451, 2014.

[8] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki, and N. Togawa, “Scan-based side-channel attack against RSA cryptosystems using scan signatures,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E93-A, no. 12, pp. 2481–2489, Dec. 2010.

[9] D. Oku, M. Yanagisawa, N. Togawa, “A robust scan-based side-channel attack method against HMAC-SHA-256 circuits,” in *Proc. International Conference on Consumer Electronics-Berlin*, pp. 91–96, Sep. 2017.

[10] 於久太祐, 多和田雅師, 柳澤政生, 戸川望, “スキャンシグネチャを用いた周辺回路を含む軽量暗号 CLEFIA に対するスキャンベース攻撃,” DA シンポジウム 2017 論文集, pp. 116–121, 2017.

[11] A. Das, J. Da Rolt, S.Ghosh, S. Seys, S. Dupuis, G. Di Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, “Secure JTAG implementation using Schnorr protocol,” *Journal of Electronic Testing,* vol. 29, no. 2, pp. 193–209, Apr. 2013.

[12] E. DeBusschere, and M. McCambridge, “Modern game console exploitation,” Technical Report, Department of Computer Science, University of Arizona, 2012.