Vol.2015-MPS-102 No.13
2015/3/4

PRINCE に対する統計処理を用いた 階層的フォールト解析とその評価

野崎佑典[†1†2]　野原康平[†1†2]　松久僚真[†1†2]　旭健作[†1†2]　吉川雅弥[†1†2]

センサーネットワークで用いられるデバイスでは，利用できるハードウェアが厳しく制限されている．一方で，これらのデバイスから情報が漏えいする危険性が指摘されている．そのため，データの暗号化が必須になってきており，そのようなデバイスでは小面積実装が可能な軽量暗号が注目されている．そのような軽量暗号の 1 つに PRINCE がある．しかし，理論的に安全な暗号であっても，回路実装した場合，フォールト攻撃が可能であることが報告されている．フォールト解析攻撃は，暗号処理中に故意に故障を引き起こし，正常の場合と故障の場合を比較することで秘密鍵を特定する攻撃手法である．そのため，暗号回路では，フォールト攻撃に対する耐性を考慮する必要がある．そこで本研究では，PRINCE を対象とした新しいフォールト解析手法を提案する．提案手法では，階層的な統計処理を利用することで，秘密情報の解析を可能にする．提案手法の有効性について，シミュレーション実験を通して検証する．

Hierarchical Statistical Fault Analysis for PRINCE and its Evaluation

YUSUKE NOZAKI[†1†2]　KOHEI NOHARA[†1†2]
RYOMA MATSUHISA[†1†2]　KENSAKU ASAHI[†1†2]　MASAYA YOSHIKAWA[†1†2]

For devices which are used for sensor networks, the number of available hardware resources is limited. Since the risk of information leakage from these devices is pointed out, data to be handled by these devices must be enciphered. A lightweight block cipher algorithm can be used for these devices with implementation constraints. Prince is such an algorithm that has attracted much attention. However, it was recently reported that when a theoretically safe encryption algorithm was embedded in hardware, secret keys could be illegally revealed by fault analysis attacks. The fault analysis attacks reveal secret keys by intentionally generating a fault during encryption processing and by comparing the fault and normal cases. To guarantee the safety of lightweight block ciphers in the future, the tamper resistance against fault analysis attacks must be examined. This study proposes a new fault analysis method which introduces hierarchical attack model using statistical processing to reveal secret keys. The validity of the proposed method is verified by performing simulations.

1. はじめに

現在，センサーネットワークで用いられる小型デバイスでは，利用できるハードウェアのリソースが厳しく制限されている．一方で，これらの小型デバイスから情報が漏えいする危険性が指摘されている．そのため，データの暗号化が必須となってきている．暗号化では，少ないリソースで実装が可能な軽量暗号が注目されており，そのような軽量暗号の 1 つに PRINCE[1]がある．

しかし，理論的に安全性が保障されている暗号であっても，回路実装した場合，フォールト解析攻撃が可能であることが報告されている[2][3]．フォールト解析攻撃は，暗号処理中に故意に故障（フォールト）を引き起こし，正常の場合と故障の場合を比較することで秘密鍵を特定する攻撃手法である．そのため，暗号回路では，フォールト解析攻撃に対する耐性を考慮する必要がある．

そこで本研究では，PRINCE を対象とした新しいフォールト解析手法を提案する．提案手法では，階層的な統計処理を利用することで，秘密情報の解析を可能にする．また，シミュレーション実験を通して，提案手法の有効性について検証する．

†1 名城大学
Meijo University
†2 （独）科学技術振興機構，CREST
JST, CREST

2. 準備

2.1 PRINCE

PRINCE[1]は Borgrhoff 氏らによって 2012 年に発表された軽量ブロック暗号である．PRINCE は暗号化と復号化を同一の回路で実現することができ，小面積で実装が可能である．PRINCE は平文が 64bit，秘密鍵が 128bit であり，合計 12 ラウンドのラウンド処理を行うことで暗号化する．また，秘密鍵kは部分鍵k_0，k_1に分割して利用する．

暗号処理を図 1 に示す．図 1 に示すように，0 ラウンド目（0R）の定数加算処理，1 ラウンド目（1R）から 5 ラウンド目（5R）のラウンド関数Rによる処理，中間処理，6 ラウンド目（6R）から 10 ラウンド目（10R）までの逆ラウンド関数R^{-1}による処理，11 ラウンド目（11R）の定数加算処理で構成する．

まず，0 ラウンド目の定数加算処理は，ラウンド定数RC_0と部分鍵k_0，k_1による排他的論理和演算を行う．ここで，RCはラウンド毎に変化する定数値である．次に，1 ラウンド目から 5 ラウンド目までのラウンド関数RはS，M'，SRの処理，ラウンド定数RC_iと部分鍵k_1の排他的論理和演算で構成

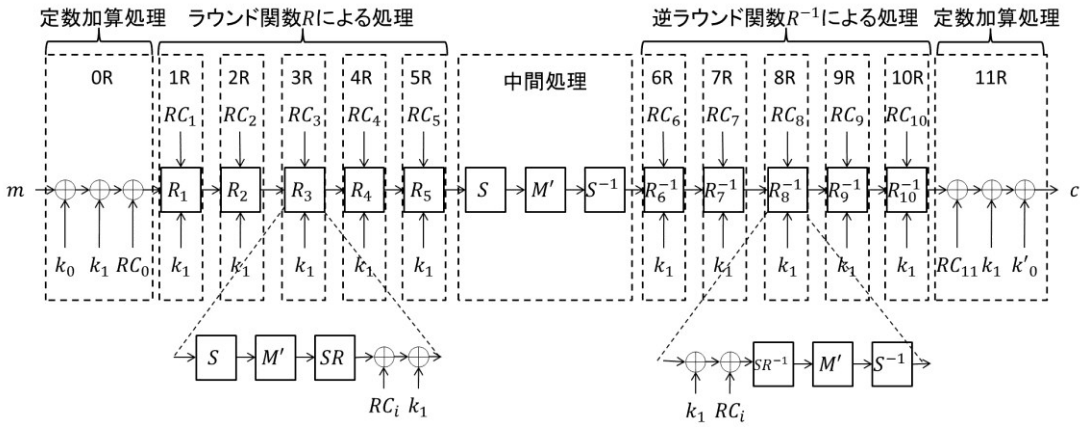

図 1 PRINCE の暗号処理の概要

Figure 1 Outline of encryption processing of PRINCE

する．ここで，RC_iはiラウンド目のRCの値である．ラウンド関数Rの処理を図 2 に示す．Sは表 1 に示す S-Box の置換表による置換処理を行う．M'は行列演算を行う．M'を式(1)に示す．

$$M' = \begin{pmatrix} \hat{M}^{(0)} & 0 & 0 & 0 \\ 0 & \hat{M}^{(1)} & 0 & 0 \\ 0 & 0 & \hat{M}^{(1)} & 0 \\ 0 & 0 & 0 & \hat{M}^{(0)} \end{pmatrix} \quad \cdots\cdots (1)$$

ここで，M'は 64×64 の行列であり，16×16 の行列である$\hat{M}^{(0)}$と$\hat{M}^{(1)}$によって構成する．また，$\hat{M}^{(0)}$, $\hat{M}^{(1)}$には式(2)に示す関係が成り立つ．

$$\hat{M}^{(0)} = \left(\hat{M}^{(0)}\right)^{-1}, \hat{M}^{(1)} = \left(\hat{M}^{(1)}\right)^{-1} \quad \cdots\cdots (2)$$

SRでは，図 2 に示すように，64bit のデータを 4bit ずつの 16 個のデータに分割し，それぞれシフト処理を行う．

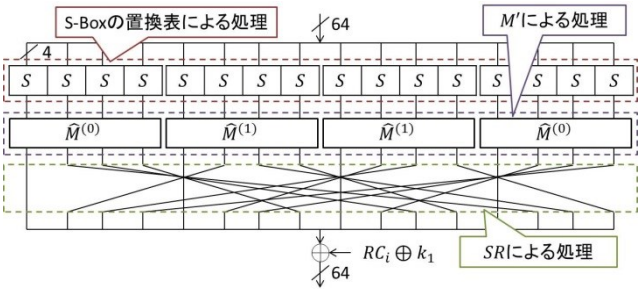

図 2 ラウンド関数R

Figure 2 Round function R

ここで，ラウンド関数RにおけるM'による行列演算は，図 2 に示すように，$\hat{M}^{(0)}$, $\hat{M}^{(1)}$（16×16 の行列）によって分割して計算することが出来る．したがって，64bit のデータを 16bit ずつに分割して計算することが可能である．

中間処理では，S, M', S^{-1}による処理を行う．このとき，S, M'はラウンド関数Rで使用した処理と同様の処理を行う．また，S^{-1}は表 1 に示す S-Box の逆置換表による置換処理である．

次に，6 ラウンド目から 10 ラウンド目の逆ラウンド関数R^{-1}は部分鍵k_1とラウンド定数RC_iによる排他的論理和演算，SR^{-1}, M', S^{-1}の処理で構成する．逆ラウンド関数の処理を図 3 に示す．SR^{-1}はラウンド関数Rで使用したSRの逆関数であり，図 3 に示すように，シフト処理を行う．M'の行列演算はラウンド関数Rのときと同様に，図 3 に示すように，$\hat{M}^{(0)}$, $\hat{M}^{(1)}$を用いて，16bit 単位で計算が可能である．

最後に，11 ラウンド目の定数加算処理では，ラウンド定数RC_{11}と部分鍵k_1, k'_0による排他的論理和演算を行う．このとき，部分鍵の計算を式(3)に示す．

$$k'_0 = (k_0 \ggg 1) \oplus (k_0 \gg 63) \quad \cdots\cdots (3)$$

ここで，$\ggg$は右ローテーション処理，$\gg$は右シフト処理である．

表 1 S-Box の置換表S・逆置換表S^{-1}

Table I Substitution table and inverse substitution table for S-Box

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S(x)$	B	F	3	2	A	C	9	1	6	7	8	0	E	5	D	4
$S^{-1}(x)$	B	7	2	3	F	D	8	9	A	6	4	0	5	E	C	1

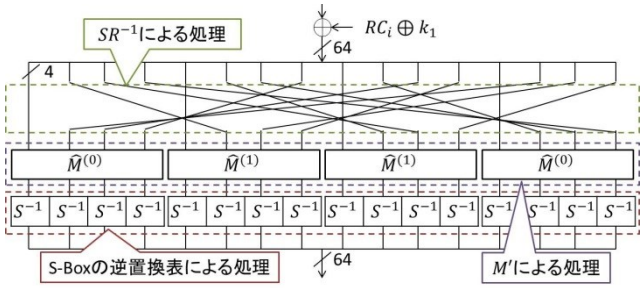

図 3 逆ラウンド関数R^{-1}

Figure 3 Inverse round function R^{-1}

2.2 関連する研究

フォールト解析は，1997 年に RSA に対する解析[2]が提案されて以降，いくつかの手法が提案されている．代表的なフォールト解析には，差分故障解析(Differential Fault Analysis : DFA)[3]がある．DFA は正しい暗号文とフォールト入り暗号文のペアを複数用いて秘密鍵を解析する手法である．また，Advanced Encryption Standard (AES)[4]に対するフォールト解析としては，Piret と Quisquater による DFA[5]など，いくつかの解析手法が提案されている[5][6][7].

軽量暗号に対するフォールト解析としては，これまでに PRESENT[8]，Piccolo[9]，PRINCE[1]などに対するフォールト解析が提案されている[10][11][12]．これらは，全て DFA をベースとしているため，秘密鍵の解析には，フォールトの個数や位置について制約がある．

3. 提案する解析手法

3.1 フォールトモデル

まず，提案手法で利用するフォールトモデルについて説明する．提案手法では，2 つのフォールトモデル（フォールトモデルⅠとフォールトモデルⅡ）を使用する．

はじめに，フォールトモデルⅠについて，図 4 を用いて説明する．フォールトモデルⅠでは，図 4 に示すように 9 ラウンド目計算終了後のレジスタに対して，フォールトを混入させる．このとき，レジスタの特定ビット列にはフォールトは混入しないものとする．この特定ビット列は，図 4 に示すように，暗号中間値 64bit を 16bit ずつの 4 つのゾーン（ゾーン 1 からゾーン 4）に分割したうちの任意の 1 つのゾーンとする．図 4 の例では，ゾーン 1 が対応する部分が特定ビット列となる．また，このゾーン 1 以外の部分では，位置や個数などに制限を設けず，全てランダムにフォールトが混入出来るものとする．先行研究[12]では，フォールト解析を行うために，特定の位置に特定の個数のフォールトを混入させる必要があるが，提案手法ではそのような制限は無く，特定ゾーン以外であれば，フォールトの位置や個数は自由である．

フォールトモデルⅠでは，9 ラウンド目計算終了後のレジスタにフォールトが混入しているため，10 ラウンド目の入力値のゾーン 1 以外の値はフォールト入り中間値となる．

したがって，10 ラウンド目の計算により，フォールトは伝搬し，最終的にフォールト入り暗号文を取得する．

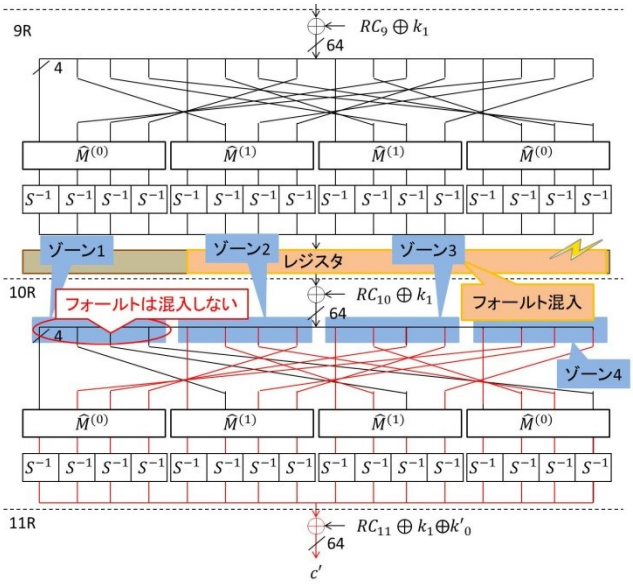

図 4 フォールトモデルⅠ

Figure 4 Fault model I

次に，フォールトモデルⅡについて説明する．フォールトモデルⅠでは，9 ラウンド目計算終了後を対象としていたのに対して，フォールトモデルⅡでは，8 ラウンド目計算終了後を対象とする．したがって，8 ラウンド目計算終了後のレジスタにフォールトが混入しているため，9 ラウンド目の入力値のゾーン 1 以外の値はフォールト入り中間値となる．そして，9 ラウンド目と 10 ラウンド目の計算によりフォールトは伝搬し，最終的にフォールト入り暗号文を取得することが出来る．

3.2 統計処理を用いた攻撃モデル

提案手法では，複数の正しい暗号文とフォールト入り暗号文の組（暗号文ペア）を用意する．そして，正しい暗号文とフォールト入り暗号文からある注目値を算出する．この注目値は，3.1 節のフォールトモデルでフォールトの混入しない任意のゾーンに対応するビット列とする．この注目値の例を図 5 に示す．図 5 に示す例では，ゾーン 1 に対応するビット列が 10 ラウンド目の注目値(フォールトモデルⅠの注目値）となる．この注目値をそれぞれα_1, α_2, α_3, α_4（正しい暗号文から算出した注目値），α'_1, α'_2, α'_3, α'_4（フォールト入り暗号文から算出した注目値）とする．このとき，ゾーン 1 にはフォールトは混入していないため，α'_1, α'_2, α'_3, α'_4とα_1, α_2, α_3, α_4は同じ値となる．

そして，正しい暗号文から計算した注目値と，フォールト入り暗号文から計算した注目値のハミング距離を利用し，解析を行う．注目値は既知の暗号文と部分鍵の予測値を利用した逆算処理により求める．ここで，部分鍵$k_1 \oplus k'_0$をK'

とおき，$i$bit 目から$j$bit 目までのK'を$K'_{(i-j)}$と表記する．統計処理を用いた攻撃モデルの概要を図 6 に示す．図 6 の①に示す例では，4bit の注目値α'_1は既知の暗号文c'と 16bit の部分鍵$K'_{(1-16)}$の予測値を利用した逆算処理により求める．

次に，図 6 の②に示すように，図 6 の①に示す処理を暗号文ペアn組の数繰り返し，ハミング距離の平均値を計算する．このとき，部分鍵の予測値が正しい場合，正しい暗号文から算出した注目値とフォールト入り暗号文から算出した注目値のハミング距離の平均値は 0 となる．一方で，部分鍵の予測値が間違っていた場合，正しい暗号文から算出した注目値とフォールト入り暗号文から算出した注目値のハミング距離の平均値は 0 よりも大きい値となる．

したがって，提案手法では，正しい暗号文から算出した注目値とフォールト入り暗号文から算出した注目値のハミング距離の平均値が 0 となる部分鍵の予測値を正解鍵と推定する．以上が，統計処理を用いた攻撃モデルの概要である．

次に，注目値を求めるための逆算処理について説明する．ここでは，フォールト入り暗号文を用いた場合のみについて説明する．まず，図 7 に示すように，10 ラウンド目の逆ラウンド関数R^{-1}の$\widehat{M}^{(0)}$, $\widehat{M}^{(1)}$の 16bit の入力値をそれぞれp'_1, p'_2, p'_3, p'_4とする．そして，この入力値p'_1, p'_2, p'_3, p'_4を求める．このとき，S^{-1}での処理は，S^{-1}の逆関数，すなわち，$(S^{-1})^{-1}=S$より，Sを利用する．また，$\widehat{M}^{(0)}$の計算は，式(2)に示す関係より，$\widehat{M}^{(0)}$をそのまま適用する．したがって，$\widehat{M}^{(0)}$の入力値p'_1は，式(4)で計算出来る．

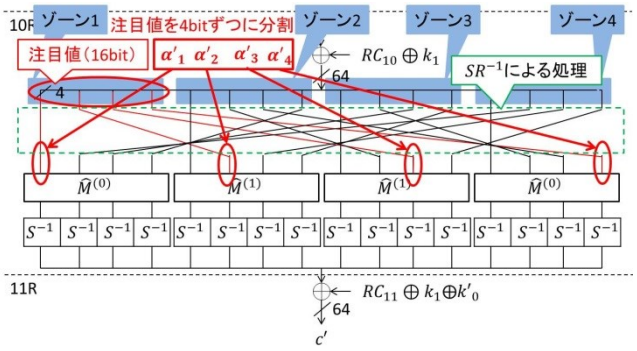

図 5 解析で使用する注目値
Figure 5 Interesting points for analysis method

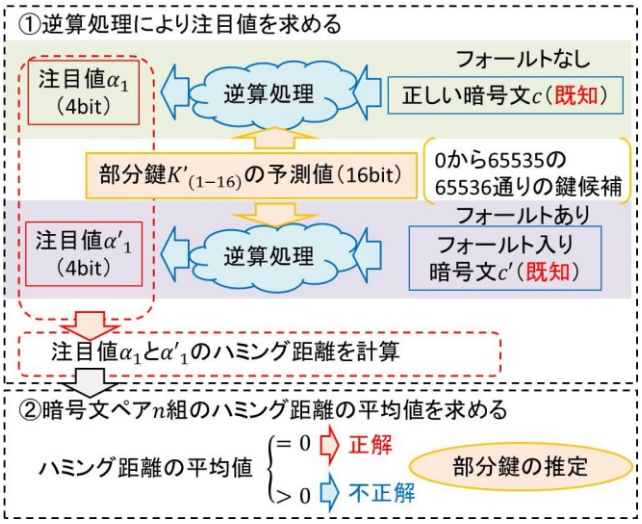

図 6 統計処理を用いた攻撃モデルの概要
Figure 6 Outline of statistical attack model

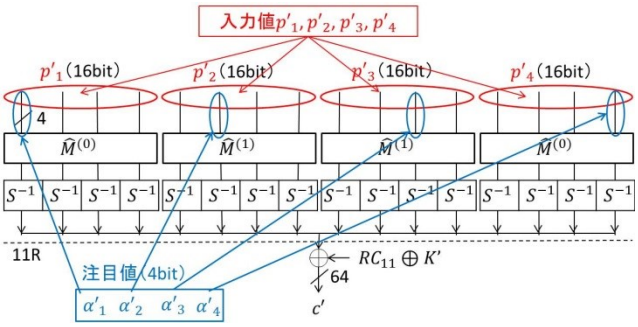

図 7 注目値の計算で利用する値
Figure 7 Value for calculation of interesting points

$$p'_1 = S\left(c'_{(1-16)} \oplus RC_{11(1-16)} \oplus K'_{(1-16)}\right) \cdot \widehat{M}^{(0)} \quad \text{.....(4)}$$

残りのp'_2, p'_3, p'_4についても同様に計算する．また，図 7 より入力値p'_1, p'_2, p'_3, p'_4と注目値α'_1, α'_2, α'_3, α'_4には，式(5)に示す関係が成り立つ．

$$\begin{cases} \alpha'_1 = p'_{1(1-4)}, \alpha'_2 = p'_{2(5-8)} \\ \alpha'_3 = p'_{3(9-12)}, \alpha'_4 = p'_{4(13-16)} \end{cases} \quad \text{.....(5)}$$

したがって，式(4)と式(5)より，注目値を求めることが出来る．また，正しい暗号文においても同様に計算して，注目値を求める．

ここで，式(4)において，フォールト入り暗号文$c'_{(1-16)}$，ラウンド定数$RC_{11(1-16)}$は既知の値であるが，部分鍵$K'_{(1-16)}$は未知の値である．このとき，16bit の部分鍵$K'_{(1-16)}$に予測値として$TK(0 \le TK \le 65535)$を与え，式(4)を計算し，p'_1求める．この結果をTx'とする．正しい暗号文においても同様に計算し，p_1を求め，これをTxとする．

このとき，Tx', Txはそれぞれ，p'_1, p_1と対応しているので，式(5)の関係を利用して，注目値を求めることが出来る．この注目値をそれぞれ$Tx'_{(1-4)}$, $Tx_{(1-4)}$とする．さらに，xとyのハミング距離を計算する関数を$HD(x,y)$とし，$Tx'_{(1-4)}$と$Tx_{(1-4)}$，α'_1とα_1のハミング距離をそれぞれ計算すると，式(6)が成り立つ．

$$\begin{cases} HD\left(Tx_{(1-4)}, Tx'_{(1-4)}\right) = HD(\alpha_1, \alpha'_1)\left(TK = K'_{(1-16)}\right) \\ HD\left(Tx_{(1-4)}, Tx'_{(1-4)}\right) \neq HD(\alpha_1, \alpha'_1)\left(TK \neq K'_{(1-16)}\right) \end{cases} \quad \text{.....(6)}$$

このとき，部分鍵の予測値TKが正解値でない場合，すなわち，$TK \neq K'_{(1-16)}$の場合，注目値$Tx_{(1-4)}$と$Tx'_{(1-4)}$の間には相関は無く，そのハミング距離の平均値は 0 よりも大きい値となる．

一方で，部分鍵の予測値TKが正解値の場合，すなわち，$TK = K'_{(1-16)}$の場合，注目値$Tx_{(1-4)}$と$Tx'_{(1-4)}$は等しくなるため，そのハミング距離の平均値は 0 となる．

したがって，n組の正しい暗号文(c)とフォールト入り暗号文(c')のペア$(c_1, c'_1), (c_2, c'_2), \cdots, (c_n, c'_n)$を準備し，式(3)(4)より$(Tx_{1(1-4)}, Tx'_{1(1-4)}), (Tx_{2(1-4)}, Tx'_{2(1-4)}), \cdots, (Tx_{n(1-4)}, Tx'_{n(1-4)})$を計算すると，注目値$(Tx_{(1-4)}, Tx'_{(1-4)})$のハミング距離の平均値$\overline{HD}$は式(7)となる．

$$\overline{HD} = \frac{1}{n}\sum_{i=1}^{n} HD\left(Tx_{i(1-4)}, Tx'_{i(1-4)}\right) \begin{cases} = 0 \left(TK = K'_{(1-16)}\right) \\ > 0 \left(TK \neq K'_{(1-16)}\right) \end{cases} \quad \cdots\cdots (7)$$

以上のように，十分な数nのとき，あるハミング距離の平均値が 0 となる部分鍵の予測値TKを正解鍵と推定する．この操作により部分鍵$K'_{(1-16)}$，すなわち，64bit の部分鍵K'のうち 16bit の部分鍵を推定することが出来る．

統計処理を用いた攻撃モデルでは，以上の操作を残りの注目値α'_2，α'_3，α'_4とα_2，α_3，α_4の計算にも適用する．したがって，残りの 48bit の部分鍵を推定することが可能であるため，64bit の部分鍵K'を全て推定することが出来る．

3.3 階層的な攻撃モデル

ここでは，3.2 節の統計処理を用いた解析手法を階層的に適用することで，秘密鍵の解析を行う．

まず，3.2 節に示すように，フォールトモデル I に対して統計処理を用いた攻撃モデルを適用して，部分鍵$k_1 \oplus k'_0$を推定する．次に，フォールトモデルⅡに対して統計処理を用いた攻撃モデルを適用して，部分鍵k_1を推定する．ここで，10 ラウンド目の入力値とラウンド定数RC_{10}と部分鍵k_1で排他的論理和をとった値を暗号中間値Xとする．このとき，暗号中間値Xは推定した部分鍵$k_1 \oplus k'_0$と既知の暗号文を用いて，計算することが出来る．そして，この暗号中間値Xを利用して，3.2 節の計算を行うことで，部分鍵k_1を推定する．

次に，推定した部分鍵$k_1 \oplus k'_0$とk_1を用いた排他的論理和演算を行うことで，部分鍵k'_0を求める．このとき，部分鍵k'_0は式(3)より求めることが出来るため，$k_{0(1)}, \cdots, k_{0(62)}$，$k_{0(64)}$，$k_{0(63)} \oplus k_{0(1)}$は既知の値，$k_{0(63)}$は未知の値である．ここで，$k_{0(63)}$は$k_{0(63)} \oplus k_{0(1)}$と$k_{0(1)}$を用いた排他的論理和演算を行うことで求めることが出来る．したがって，部分鍵k_0を全て求めることが出来る．

以上のように，部分鍵k_0とk_1を求めることが出来るため，128bit の秘密鍵kを全て導出することが出来る．

3.4 枝刈りアルゴリズム

3.2 節の統計処理を用いた攻撃モデルでは，部分鍵を 16bit ずつ計算しているため，65536 通りの鍵候補を暗号文ペアの数だけ計算する必要がある．そこで提案手法では，計算量を削減するための枝刈りアルゴリズムを導入する．枝切りアルゴリズムでは，部分鍵の推定において，注目値のハミング距離が 0 以外の鍵候補を切り捨てることで，計算量を削減する．

枝切りアルゴリズムについて，注目値α_1とα'_1を計算する場合について図 8 を用いて説明する．ここでは，鍵候補リストを作成し，鍵候補リストにある鍵候補を用いて，統計処理を用いた攻撃モデルを適用する．まず，図 8 に示すように，1 組目の暗号文ペアを用いた計算では，65536 通りの鍵候補が存在する鍵候補リストから 1 つずつ鍵候補を試していく．そして，注目値α_1とα'_1のハミング距離を計算し，このハミング距離が 0 の場合，この鍵候補を鍵候補リストA_1に加える．

一方で，注目値α_1とα'_1のハミング距離が 0 でない場合，試行した鍵候補は正解鍵ではないため，鍵候補リストA_1に鍵候補を加えない．この処理を十分な数n回繰り返すことで，鍵候補を 1 つに絞り込むことが出来る．

したがって，この処理で最後まで残った鍵候補は正解鍵となる．

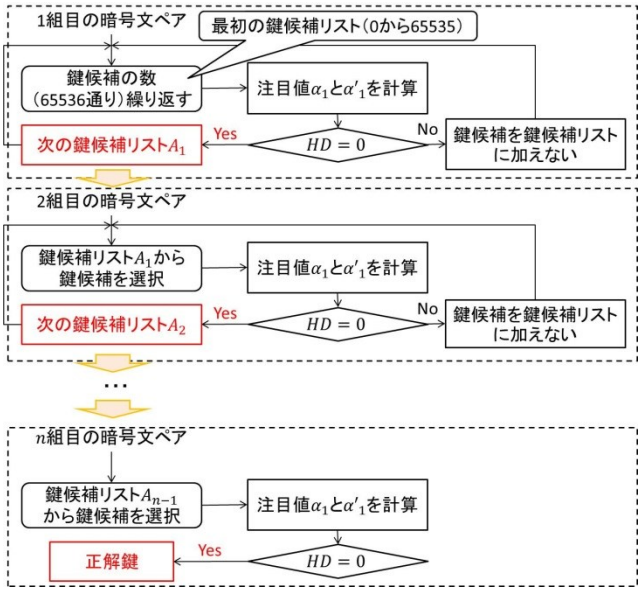

図 8 枝刈りアルゴリズムの処理
Figure 8 Processing of pruning algorithm

4. 評価実験

4.1 シミュレーション実験

提案手法の有効性を検証するために，シミュレーションによる評価実験を行う．ここでは，3.1 節のフォールトモデルに対して，ゾーン 2 からゾーン 4 の部分に任意のフォ

ールトを混入する．具体的には，乱数によって決定した位置のビット値を反転させる．

評価実験はフォールトの混入個数は対象とする暗号中間値の 20%のビット数とし，乱数で正しい暗号文とフォールト入り暗号文のペアを 20 組生成して行った. 提案手法を用いて，秘密鍵を解析した結果を図 9 に示す．図 9 の縦軸（正解部分鍵数）は秘密鍵 128bit のうち，正解した部分鍵の数（部分鍵 16bit で 1 つ）を示したものである．また，横軸は解析に使用した暗号文ペア数である．

図 9 より，暗号文ペア数が 13 組のとき，全ての秘密鍵の解析に成功していることが分かる．したがって，提案手法の有効であると考えられる．

次に，枝刈りアルゴリズムの有効性を検証するための実験を行った．ここでは，枝刈りアルゴリズムを使用しない場合，すなわち，統計処理を用いた攻撃モデルのみで解析を行った場合と，枝刈りアルゴリズムを使用した場合の比較を行う．具体的には，暗号文ペアを 20 組用意し，1 つの部分鍵 16bit の予測における処理時間と計算量の比較を行った．この結果を表 2 に示す．

表 2 より，枝刈りアルゴリズムを使用した場合，処理時間は 28[sec]となっている．一方で，枝刈りアルゴリズムを使用した場合，処理時間は 471[sec]である．したがって，枝刈りアルゴリズムを使用した場合は，枝刈りアルゴリズムを使用しない場合と比較して，1/20倍ほど処理時間が早くなっていることが分かる．また，枝刈りアルゴリズムを使用しない場合は，65536 通りの鍵候補を暗号文ペア 20 組分の計算を行う必要がある．一方で，枝刈りアルゴリズムを使用した場合，その計算量は表 2 に示すように，69904 回となっており，大幅に計算量を削減することが出来る．

以上より，枝刈りアルゴリズムを導入することで，処理時間，計算量を大幅に削減することができ，枝刈りアルゴリズムが有効であることを確認した．

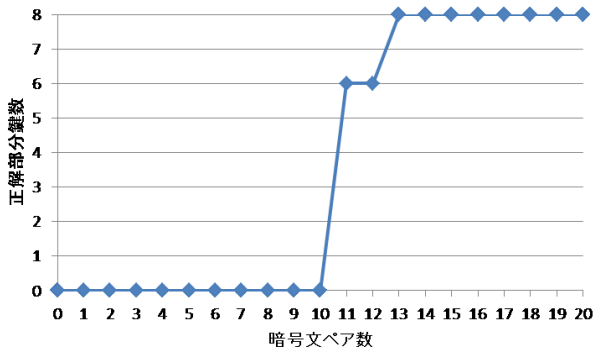

図 9 実験結果

Figure 9 Simulation result

表 2 処理時間の比較

Table II Comparison of processing time

	処理時間	計算量
枝刈りアルゴリズムなし	471[sec]	65536×20
枝刈りアルゴリズムあり	28[sec]	69904

5. まとめ

本研究では，軽量暗号 PRINCE に対する新たなフォールト解析手法を提案した．提案手法では，統計処理を用いた解析手法を階層的に適用することで, 秘密鍵の解析を行う．また，試行する鍵候補の数を減らす枝刈りアルゴリズムを導入することで，計算量を大幅に削減した．そして，シミュレーション実験を通して, 提案手法の有効性を実証した．

今後は，提案手法に対する対策の検討などを行っていく予定である．

謝辞 本研究は科学技術振興機構（JST）の戦略的創造研究推進事業（CREST）における「ディペンダブル VLSI システムの基盤技術」の研究の一環として行われた．

参考文献

1) Borghoff, J. et al.: PRINCE - A Low-latency Block Cipher for Pervasive Computing Applications, ASIACRYPT 2012, LNCS 7658, pp.208-225 (2012).
2) Boneh, D. et al.: On the Importance of Checking Cryptographic Protocols for Faults, EUROCRYPT 1997, LNCS 1233, pp.37-51 (1997).
3) Biham, E. and Shamir, A.: Differential Fault Analysis of Secret Key Cryptosystems, Proc. of CRYPTO'97, pp.513-525 (1997).
4) NIST, "Advanced Encryption Standard (AES)", FIPS PUB 197, http://csrc.nist.gov/publications/fips/index.html/ http://office.microsoft.com/ja-jp/word-help/CH010097020.aspx
5) Piret, G. and Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structure, with Application to the AES and Khazad, Proc. of CHES 2003, pp.77-88 (2003).
6) Chen, C.-N. and Yen, S.-M.: Differential fault analysis on AES key schedule and some countermeasures, Proc. 8th Australasian Conf. Information Security and Privacy (ACISP 2003), vol.2727, pp.118-129 (2003).
7) Giraud, C.: DFA on AES, Proc. 4th Int. Conf. Advanced Encryption Standard-AES (AES 2004), vol.3373, pp.27-41 (2005).
8) Bogdanav, A. et al.: PRESENT: An Ultra-Lightweight Block Cipher, Proc. of CHES 2007, LNCS 4727, pp.450-466, Springer-Verlag (2007).
9) Shibutani, K. et al.: Piccolo: An Ultra-Lightweight Blockcipher, Proc. of CHES 2011, LNCS 6917, pp.342-357, Springer-Verlag (2011).
10) Wang, G. and Wang, S.: Differential Fault Analysis on PRESENT Key Schedule, Conf. on Computational Intelligence and Security (CIS 2010), pp.362-366 (2010).
11) Jeong, K.: Differential Fault Analysis on Block Cipher Piccolo, Cryptography ePrint Archive, Report 2012/399 (2012). http://eprint.iacr.org/
12) Song, L. and Hu, L.: Differential Fault Attack on the PRINCE Block Cipher, Cryptography ePrint Archive, Report 2013/043, LNCS 8162, pp.43-54, Springer-Verlag (2013). http://eprint.iacr.org/