# KCipher-2 への電力解析攻撃対策とその評価

響 崇史† 本間 尚文† 青木 孝文† 仲野 有登‡ 福島 和英‡
清本 晋作‡ 三宅 優‡

†東北大学大学院情報科学研究科
980-8579 仙台市青葉区荒巻字青葉 6–6–05

‡株式会社 KDDI 研究所
356-8502 埼玉県ふじみ野市大原 2–1–15

hibiki@aoki.ecei.tohoku.ac.jp

**あらまし** ストリーム暗号 KCipher-2 への電力解析攻撃対策とその評価について述べる．まず，KCipher-2 への電力解析攻撃について概説する．次に，乱数マスキングに基づく対策手法を提案する．提案手法では，非線形関数部の整数演算と Sub 関数およびレジスタ L1，L2 部分にマスクをかけ，L2 と FSR-B の出力部分でアンマスクを行う．本稿では，90nm-CMOS テクノロジを用いて，対策を施した回路の性能を評価し，整数演算および Sub 関数のアルゴリズムとマスキング方法を適切に設計・適用することで，対策による面積と遅延の増加をそれぞれ 1.5 倍，2.5 倍程度に抑えられることを示す．また，SASEBO-GII を用いた攻撃実験により対策の有効性を示す．

# A countermeasure against power analysis attacks on KCipher-2 and its evaluation

Takafumi HIBIKI† Naofumi HOMMA† Takafumi AOKI†
Yuto NAKANO Kazuhide FUKUSHIMA‡ Shinsaku KIYOMOTO‡ Yutaka MIYAKE

†Graduate School of Information Sciences,
Tohoku University
6–6–05 Aoba, Aramaki, Aoba-ku,
Sendai-shi, Miyagi, 980-8579, JAPAN

‡KDDI R&D Laboratories,Inc.
Ohara 2–1–15,
Fujimino-shi, Saitama, 356-8502 JAPAN

hibiki@aoki.ecei.tohoku.ac.jp

**Abstract** This paper presents a countermeasure against power analysis attacks on a stream cipher KCipher-2 and its evaluation. The proposed concept is to mask intermediate data passing through internal registers L1 and L2 in the non-linear function part, and then unmask the data at the outputs of L2 and FSR-B. We evaluate the performance of the proposed method through ASIC implementations on a 90 nm CMOS technology. Compared with the unprotected design, the performance overheads in circuit area and delay are increased at most 1.5 and 2.5 times, respectively. We also demonstrate the effectiveness of the countermeasure with a SASEBO-GII platform.

## 1 はじめに

近年，暗号アルゴリズムをソフトウェアやハードウェアで実装した暗号モジュールに対する物理攻撃の一つであるサイドチャネル攻撃の危険性が指摘されている．サイドチャネル攻撃は，暗号化/復号処理に伴って生じる副次的な物理量に含まれる情報を利用して暗号解読を行う．

特に，処理中の消費電力の変化を用いる電力解析攻撃は，安価な装置で解析可能，かつ，攻撃の痕跡が残らないという特徴を持つ強力な攻撃であり，Kocher らによる差分電力解析 (Differential Power Analysis: DPA) [1] や相関電力解析 (Correlation Power Analysis: CPA)[2] がよく知られている．

ストリーム暗号に対しては，Fisher らにより，攻撃者が初期ベクトルを観測もしくは選択可能という条件下で，多数の初期ベクトルを用いた電力解析攻撃が可能となることが指摘されている [3]．ストリーム暗号のうち，筆者らは ISO/IEC 国際標準暗号の一つである KCipher-2 [4] に注目し，電力解析攻撃が可能となることを示した [5].

本稿では，上記 [5] に示す電力解析攻撃に対して有効な乱数マスキングに基づく対策方法を示し，その対策を施した KCipher-2 回路の性能を評価する．また，SASEBO-GII を用いた実験を通して，その対策の有効性を示す.

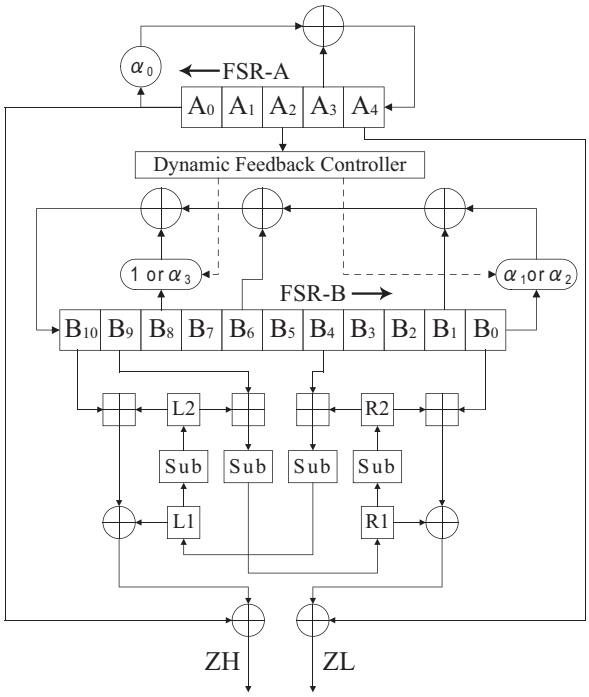



図 1: ストリーム暗号 KCipher-2 の概要

## 2 KCipher-2

KCipher-2 は，128 ビットの初期鍵 ($IK$) と，128 ビットの初期ベクトル ($IV$) の 2 つの独立なパラメータを入力とするストリーム暗号である．KCipher-2 は，図 1 で示すように，以下の要素で構成される.

- 線形フィードバックシフトレジスタ (FSR-A) および動的フィードバックシフトレジスタ (FSR-B)
- 4 つの内部レジスタ $R1$，$R2$，$L1$，$L2$ を有する Finite State Machine (FSM)

また，KCipher-2 内部で使用されるレジスタの語長は全て 32 ビットである.

### 2.1 構成要素の概要

FSR-A，FSR-B は，それぞれ 5 個のレジスタと 11 個のレジスタで構成される．FSR-A，FSR-B の更新は，以下の式によって行われる.

$$A_i^{(t+1)} = \begin{cases} A_{i+1}^{(t)} & (i \neq 4) \\ \alpha_0 A_0^{(t)} \oplus A_3^{(t)} & (i = 4) \end{cases} \quad (1)$$

$$B_j^{(t+1)} = \begin{cases} B_{j+1}^{(t)} & (j \neq 10) \\ \alpha_{(cl1^{(t)})} B_0^{(t)} \oplus B_1^{(t)} \oplus B_6^{(t)} \oplus \alpha_3^{cl2^{(t)}} B_8^{(t)} & (j = 10) \end{cases} \quad (2)$$

ここで，$\alpha_{(cl1^{(t)})}$ は，

$$\alpha_{(cl1^{(t)})} = \alpha_1^{cl1^{(t)}} + \alpha_2^{1-cl1^{(t)}} - 1 \quad (3)$$

である．また，時刻 $t$ の FSR-A，FSR-B の各レジスタの出力を $A_i^{(t)}$，$B_j^{(t)}$ とし，$i$，$j$ をレジスタの番号とする．$i$，$j$ の範囲は $0 \leq i \leq 4$，$0 \leq j \leq 10$ である．各レジスタの $n$ 番目のビットの値は $A_2^{(t)}[n] \in \{0,1\}$ のように表す．$n = 31$ のとき，レジスタの最上位ビットの値を表す．式 (2) の $cl1$，$cl2$ は，クロック制御ビットであり，FSR-A における $A_2$ の値を用いてそれぞれ $cl1^{(t)} = A_2^{(t)}[30] \in \{0,1\}$，$cl2^{(t)} = A_2^{(t)}[31] \in \{0,1\}$ と与えられる.

一方 FSM は，整数加算器と置換関数 Sub を含み，FSR-A の 2 つのレジスタ，FSR-B の 4

つのレジスタおよび 4 つの内部レジスタの値を入力とし，1 サイクルごとに 64 ビットの鍵系列を出力する．置換関数 Sub は，入力された 32 ビットの値をバイト単位に分割し，8 ビット入出力の換字処理を各バイトに施す Substitution 処理と，その出力に対し，32 ビット入出力の Permutation 処理から構成される．これらは，それぞれ共通鍵暗号 AES [6] の S-box 処理，MixColumns 処理と同一の関数である．

### 2.2 初期化処理

KCipher-2 の処理ステップは，内部状態を初期化する初期化処理と内部状態を更新しながら鍵系列を生成する鍵系列出力処理に分けられる．

初期化処理は，初期鍵 $IK$ と初期ベクトル $IV$ の読み込み処理と，内部状態の初期化処理で構成される．以下では，128 ビットの初期鍵 $IK$ を上位から 32 ビットずつに分割し，$IK = (IK_3, IK_2, IK_1, IK_0)$ と表す．また，128 ビットの初期ベクトル $IV$ も同様に 32 ビットずつに分割し，$IV = (IV_3, IV_2, IV_1, IV_0)$ と表す．

読み込み処理の前に，初期鍵 $IK$ を以下の方法で 12 個の 32 ビット内部鍵 $K_k (k = 0, 1, ..., 11)$ に拡張する．

$$K_k = \begin{cases} IK_{3-k} & (0 \le k \le 3) \\ K_{k-4} \oplus Sub(k_{k-1}) \oplus rc & (k = 4n) \\ K_{k-4} \oplus K_{k-1} & (k \neq 4n) \end{cases} \quad (4)$$

$$k_{k-1} = (K_{k-1} \ll 8) \oplus (K_{k-1} \gg 24) \quad (5)$$

ここで，$Sub$ は Sub 関数の処理を表す．また，$rc$ は定数であり，$k = 4$ の時，$rc = (01000000)_{16}$，$k = 8$ の時，$rc = (02000000)_{16}$ となる．

内部鍵の生成後，内部鍵と初期ベクトルにより KCipher-2 の内部状態を初期化する．その後，以下に示すような内部状態の撹拌を 24 クロック $(t = 0, 1, \ldots, 23)$ 行う．

$$R1^{(t+1)} = Sub(L2^{(t)} + B_9^{(t)}) \quad (6)$$

$$R2^{(t+1)} = Sub(R1^{(t)}) \quad (7)$$

$$L1^{(t+1)} = Sub(R2^{(t)} + B_4^{(t)}) \quad (8)$$

$$L2^{(t+1)} = Sub(L1^{(t)}) \quad (9)$$

$$A_i^{(t+1)} = \begin{cases} A_{i+1}^{(t)} & (i \neq 4) \\ \alpha_0 A_0^{(t)} \oplus A_3^{(t)} \oplus ZL^{(t)} & (i = 4) \end{cases} \quad (10)$$

$$B_j^{(t+1)} = \begin{cases} B_{j+1}^{(t)} & (j \neq 10) \\ \alpha_{(cl1^{(t)})} B_0^{(t)} \oplus B_1^{(t)} \oplus B_6^{(t)} \oplus \alpha_3^{cl2^{(t)}} B_8^{(t)} \oplus ZH^{(t)} & (j = 10) \end{cases} \quad (11)$$

ここで，$\alpha_{(cl1^{(t)})}$ は式 (3) と等しい．また，$ZH^{(t)}$ と $ZL^{(t)}$ は，時刻 $t$ に出力する 64 ビットの鍵系列のそれぞれ上位，下位 32 ビットの値である．同様に，$R1^{(t)}$, $R2^{(t)}$, $L1^{(t)}$, $L2^{(t)}$ は時刻 $t$ の FSM の内部レジスタの値である．初期化処理中は，式 (1), (2) とは異なり，鍵系列がフィードバック関数に加えられる．

### 2.3 鍵系列出力処理

初期化処理が終わると，鍵系列出力処理に移行する．時刻 $t$ に出力される 64 ビットの鍵系列 $Z^{(t)}$ は，内部レジスタの値を用いて以下の式により出力される．

$$ZH^{(t)} = B_{10}^{(t)} + L2^{(t)} \oplus L1^{(t)} \oplus A_0^{(t)} \quad (12)$$

$$ZL^{(t)} = B_0^{(t)} + R2^{(t)} \oplus R1^{(t)} \oplus A_4^{(t)} \quad (13)$$

鍵系列出力処理中の内部状態の更新は，式 (1), (2), (6)〜(9) によって行われる．

## 3 KCipher-2 への電力解析攻撃

KCipher-2 に対する 電力解析攻撃では，同一鍵で初期ベクトル $IV$ が選択可能という条件下で初期鍵 $IK$ を復元することを目的とする．攻撃対象は 32 ビットの内部鍵であり，相関電力解析 (CPA) により 8 ビットごとに鍵の推定を行う．下位ビットから鍵推定を行い，推定した鍵情報を順次上位ビットの鍵推定に使用する．

はじめに，CPA に使用する予測電力値の決定方法について述べる．予測電力値には初期化開始直後の内部レジスタの値を利用する．まず，開始後の 3 クロック目と 4 クロック目の FSM

の内部レジスタ $L1$ の値 $L1^{(3)}, L1^{(4)}$ に注目すると，その値は以下の式で与えられる．

$$L1^{(3)} = Sub(IV_1 + Sub(Sub(K_5))) \quad (14)$$

$$L1^{(4)} = Sub(IV_0 + Sub(Sub(K_6 + Sub(0)))) \quad (15)$$

ここで，内部鍵 $K_5$ と $K_6$ で決定される $Sub(Sub(K_5))$ と $Sub(Sub(K_6 + Sub(0)))$ を推定対象とし，それぞれ $CK_5$，$CK_6$ とおく．注目するレジスタ $L1^{(t)}$ の値を求めるためには，それぞれ $IV_1$，$IV_0$ との整数加算中の桁上がりの影響とそれに続く Permutation 処理を考慮する必要がある．鍵の中間・下位ビットを推定する際には，任意の $IV$ では Permutation 処理を完全に再現することが困難となる．そこで，Permutation 処理の演算の一部を定数として扱えるような範囲で $IV$ を選択する．具体的には，予測する位置より上位のビットが 0 となるものを $IV_1$，$IV_0$ として選択する．これにより，Permutation 処理を近似できるだけでなく，桁上げの伝搬も無視できるため，中間値を求めることが可能となる．鍵の最上位ビットを推定する際には，下位ビットの情報を利用できるため，任意の $IV$ で中間値を正しく推測できる．

結果として，注目するレジスタ $L1^{(t)}$ の値は，既知情報から求められる Substitution 処理の出力 $Sout = (s_3, s_2, s_1, s_0)$ を用いて，式 (16) ～式 (19) で表される．

$$l_3 = s_3 \otimes (02)_{16} \oplus s_2 \oplus s_1 \oplus s_0 \otimes (03)_{16} \quad (16)$$

$$l_2 = s_2 \otimes (02)_{16} \oplus s_1 \oplus s_0 \quad (17)$$

$$l_1 = s_1 \otimes (02)_{16} \oplus s_0 \quad (18)$$

$$l_0 = s_0 \otimes (02)_{16} \quad (19)$$

ここで，$\otimes$ は $GF(2^8)$ 上の乗算を表す．また，$l_3$, $l_2$, $l_1$, $l_0$ は，32 ビットの $L1^{(t)}$ の値のうち，それぞれ 31~24 ビット，23~16 ビット，15~8 ビット，7~0 ビットの 8 ビットの値を表す．

次に，使用する消費電力モデルを設定する．予測するレジスタの直前のクロック時の値は，それぞれ以下の式で与えられる．

$$L1^{(2)} = Sub(K_9 + Sub(0)) \quad (20)$$

$$L1^{(3)} = Sub(IV_1 + CK_5))) \quad (21)$$

表 1: 1 ビットの消費電力モデルの関係

| $v_{n-1}$ | $v_n$ | HD($v_{n-1}$,$v_n$) | HW($v_n$) |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

$v_n$: 状態 $n$ におけるレジスタ値

式 (20)，(21) のように直前の値は初期鍵に依存した未知の値であるため，CPA で通常用いられる Hamming Distance(HD) モデルの使用は難しい．しかし，同一鍵という条件を想定した場合，$L1^{(2)}$ の値は一定値となる．そこで，1 ビットの消費電力モデルを考える．表 1 に 1 ビット HD モデルと 1 ビット Hamming Weight(HW) モデルの関係を示す．今回のように直前の値が定数の場合，極性は逆になる可能性があるが，HD モデルと HW モデルを等価とみなせることがわかる．この特性を利用して，$CK_5$ の予測では，消費電力モデルに 1 ビットの HW モデルを使用する．このとき，第 $j$ バイト $(0 \leq j \leq 3)$ の第 $i$ ビット $(0 \leq i \leq 7)$ の $CK_5$ の予測電力値 $h_{j,i}^{(3)}$ $(\in \{0, 1\})$ は次式で与えられる．

$$h_{j,i}^{(3)} = l_{j,i} \quad (22)$$

一方，$CK_6$ の予測では，$K_5$ を推定できたならば，式 (21) より直前のレジスタの値 $L1^{(3)}$ が求められるため，HD モデルを使用できる．このとき，$CK_6$ の予測電力値 $h_{j,i}^{(4)}$ $(\in \{0, 1\})$ は次式で与えられる．

$$h_{j,i}^{(4)} = l_{j,i} \oplus L1_{i+8j}^{(3)} \quad (23)$$

$h_{j,i}^{(3)}$，$h_{j,i}^{(4)}$ と取得した消費電力波形との相関をピアソンの相関係数 $\rho$ により求める．その相関係数をそれぞれ $\rho_{j,i}^{(3)}$，$\rho_{j,i}^{(4)}$ とし，ある 8 ビットの推定に使用する相関値は以下の式により与える．

$$\rho_j^{(3)} = \sqrt{\frac{1}{8}\sum_{i=0}^{7}\left(\rho_{j,i}^{(3)}\right)^2} \quad (24)$$

$$\rho_j^{(4)} = \sqrt{\frac{1}{8}\sum_{i=0}^{7}\left(\rho_{j,i}^{(4)}\right)^2} \quad (25)$$

式 (24)，(25) の値が最大になる鍵候補を正しい $CK_5$，$CK_6$ と推定する．これらから $K_5$，$K_6$ が計算でき，式 (4) から $K_2$ を計算できる．

さらに，$CK_5$ の推定時に得られた相関値の情報と使用した電力モデルの関係から定数値 $L1^{(2)}$ が推定できる．すでに直前のレジスタの値が定数の場合，1 ビット HW モデルと 1 ビット HD モデルが等価となることを述べた．表 1 より，2 つの電力モデルの間には以下の関係が成り立つ．

$$HD(v_{n-1}, v_n) = HW(v_n) \quad (v_{n-1} = 0 \text{ のとき}) \quad (26)$$

$$HD(v_{n-1}, v_n) = 1 - HW(v_n) \quad (v_{n-1} = 1 \text{ のとき}) \quad (27)$$

$v_{n-1} = 0$ の時，2 つの電力モデルは正の相関関係にあり，$v_{n-1} = 1$ の時は負の相関関係にある．すなわち，2 つの電力モデルが正の相関を持つとき直前の定数値 ($v_{n-1}$) は 0，負の相関を持つときは 1 と推定できる．ここで，$L1^{(2)}_{j,i}$ の推定に利用する相関値 $\rho^{(3)}_{j,i}$ に注目する．正解の $CK_5$ での相関値 $\rho^{(3)}_{j,i}$ は，処理開始から 3 クロック目でピークを有する．そのピークが正ならば直前の定数値 $L1^{(2)}_{j,i}$ の値は 0，負ならば 1 と推定できる．ただし，相関の正負と推定する定数値の関係は，電力の測定点が GND の場合に成り立つものであり，測定点を VDD にすると，その相関関係は逆転することに注意されたい．すべての $\rho^{(3)}_{j,i}$ から $L1^{(2)}$ を推定できれば，式 (20) により $K_9$ が計算できる．その後で $K_7$ を $2^{32}$ の全探索により推定すると，式 (4) よりすべての内部鍵 $K$，初期鍵 $IK$ が計算できる．

# 4 電力解析攻撃への対策

本節では，上記の CPA への対策として乱数マスキングによる対策について述べる．

## 4.1 提案手法

上記の CPA は，初期化処理のタイミングで FSM 内部のレジスタ $L1$ の消費電力に対して行われる．$L1$ の入力側には FSR-B，内部レジスタ $R2$ の値を入力とする整数加算器と Sub 関数があり，その出力側には Sub 関数と内部レジスタ $L2$ がある．それらの動作中に消費される電力は上記 CPA の対象となり得るため，乱数マスキングによる対策では，それらを含む $L1$ のデータパス全体に乱数マスクを施す必要がある．そこで，整数加算器の入力からマスクし，内部レジスタ $L2$ の出力でアンマスクすることを考える．

乱数マスキングでは，アンマスク用の値をあらかじめ，もしくは並行して計算しておくことが求められる．演算器への入力を $x$，マスク値を $m_x$，演算を $f$ としたとき，以下の式が成立する場合は，演算器を二重化し，片方でマスクの値を入力として演算することでアンマスクの値を計算する．

$$f(x \oplus m_x) = f(x) \oplus f(m_x) \quad (28)$$

一方で，上記の式が成立しない場合もある．$L1$ のデータパスでは整数加算器と Sub 関数内の Substitution 処理がこれにあたる．これらのマスキングの実現方法を以下に述べる．

## 4.2 整数加算器のマスキング

整数加算器へのマスキングを実現するために，Golic によるマスクを考慮した AND 演算 (Masked AND) [7] を応用する．Masked AND 演算は，入力 $A = a \oplus m_a$ と $B = b \oplus m_b$ に対して，$C = (a \wedge b) \oplus m_c$ という値を出力する．ここで，$\wedge$ は AND 演算を表す．このとき，出力をマスクする $m_c$ は，$m_a$ か $m_b$ のどちらか一方の値で実現できる．この Masked AND 演算を整数加算器中の AND 演算と置き換えることでアンマスクが容易な整数加算器を設計する．ある桁の入力値を $x_i$，$y_i$，マスク値を $m_{x,i}$，$m_{y,i}$，出力値を $z_i$ とする整数加算器を考える．$X_i = x_i \oplus m_{x,i}$，$Y_i = y_i \oplus m_{y,i}$ を入力したとき，出力 $Z_i$ と下位ビットからの桁上げ $C_i$ は以下の式で表される．

$$Z_i = X_i \oplus Y_i \oplus C_i \quad (29)$$

$$C_i = X_{i-1} \wedge Y_{i-1} \oplus C_{i-1} \wedge (X_{i-1} \oplus Y_{i-1}) \quad (30)$$

式 (29) より，出力 $Z_i$ のアンマスクを考える場合，$C_i$ をアンマスクするための値が必要となる．そのためには，式 (30) の項にある $C_{i-1}$ のマスク値を考慮しなければならないが，AND 演算を Masked AND 演算に変換することで，以下のように $C_{i-1}$ のマスク値に依存しない形で表すことができる．

$$\begin{aligned} C_i &= (x_{i-1} \wedge y_{i-1}) \oplus m_{x,i-1} \oplus (c_{i-1} \wedge \\ &\quad (x_{i-1} \oplus y_{i-1})) \oplus (m_{x,i-1} \oplus m_{y,i-1}) \\ &= c_i \oplus m_{y,i-1} \end{aligned} \tag{31}$$

式 (29)，(31) より，マスクされた値を入力した整数加算器の出力 $Z_i$ は以下の式で表せる．

$$\begin{aligned} Z_i &= x_i \oplus y_i \oplus c_i \oplus m_{x,i} \oplus m_{y,i} \oplus m_{c,i} \\ &= z_i \oplus m_{x,i} \oplus m_{y,i} \oplus m_{y,i-1} \end{aligned} \tag{32}$$

これを n ビットに拡張させた場合，アンマスクの値 ($m_z$) は，$m_x \oplus m_y \oplus (m_y \ll 1)$ となる．

上記は，Ripple Carry Adder (RCA) のアルゴリズムへの適用方法である．この加算アルゴリズムは，回路面積が小さくなる一方で，桁上げ伝搬により，演算遅延が大きくなる．KCipher-2 のハードウェア実装において，整数加算器はクリティカルパスに含まれるため，この伝搬遅延が全体の処理速度に与える影響は大きい．一方，遅延の小さい高速な加算アルゴリズムとしては Prefix Adder が知られている．そこで，高速な実装方法として，上記の Masked AND 演算を Prefix Adder にも応用する．Prefix 加算アルゴリズムでは，下位ビットからの桁上げとは無関係に求められる Propagate (P) と Generate (G) という信号を用いた以下の Prefix 演算で桁上げを並列に計算する．

$$p_k = p_i \wedge p_j \tag{33}$$

$$g_k = g_i \oplus (g_j \wedge p_i) \tag{34}$$

ここで，P 信号は $p_i = x_i \oplus y_i$ で与えられ，G 信号は $g_i = x_i \wedge y_i$ で与えられる．Prefix Adder のマスキングは，先に述べた RCA のマスキングの時と同様に，式 (33)，(34) の AND 演算を Golic による Masked AND 演算に置き換えることで実現可能である．一方，Prefix Adder の出力のアンマスクは，式 (32) とは異なり，Prefix 加算アルゴリズムごとに求める必要がある．本稿で用いた Kogge-Stone の加算アルゴリズム (KSA) [8] では，アンマスク値 ($m_z$) は以下のように求められる．

$$m_z = mG4 \oplus mG4 \ll 16 \tag{35}$$

$$mG1 = m_y \oplus m_y \ll 1 \tag{36}$$

$$mG2 = mG1 \oplus mG1 \ll 2 \tag{37}$$

$$mG3 = mG2 \oplus mG2 \ll 4 \tag{38}$$

$$mG4 = mG3 \oplus mG3 \ll 8 \tag{39}$$

### 4.3 Substitution 処理のマスキング

Substitution 処理では，ガロア体上の逆元演算に対するマスキングを実現する必要がある．逆元演算に対するマスキングとしては，加法的なマスキング手法と乗法的なマスキング手法が考えられる．加法マスキングでは，合成体を使用した逆元演算回路の最も小さい部分体上の演算において，加法マスクの値が出力の真値と分離可能であるという性質を利用する [9]．一方，乗法マスキングでは，乗法マスクが逆元演算後に出力とマスク値に分離可能であるという性質を利用する [10]．ここで，乗法的なマスク・アンマスクをかける前後で入出力に依存した消費電力が生じないように加法マスクも使用することに注意されたい．それぞれのマスキング手法の詳細については，[9], [10] を参照されたい．

### 4.4 アンマスキング

図 2 に上記のマスク処理を施した KCipher-2 回路を示す．ここでは，マスクされた中間値の流れを赤線で，マスク・アンマスク用の値の流れを青線で表す．マスク処理は $B_4$ の出力および $L1$ 側のパスへの $R2$ の出力で施し，アンマスク処理は $L2$ および $B_{10}$ の出力で施す．ここで，$L1$ のアンマスク処理を，$L1$ の出力ではなく，その出力先である FSR-B のレジスタ $B_{10}$ の出力で行うことに注意されたい．これは $L1$ の出力を用いた XOR 演算により，真値に依存した消費電力が生じるのを防ぐためである．整

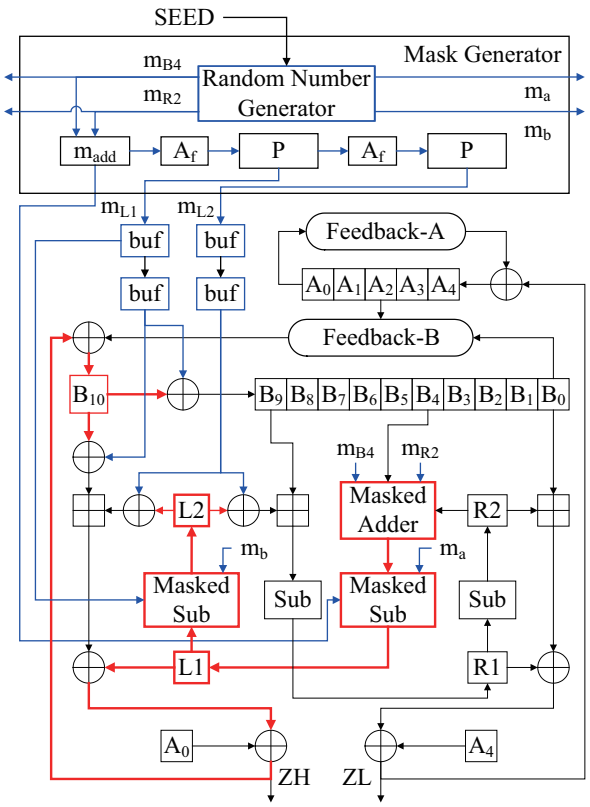



図 2: マスク処理を施した KCipher-2 回路

数加算器の入力の真値は $B_4$，$R2$ であり，その加法マスクの値を $m_{B4}$，$m_{R2}$ とし，出力のマスク値を $m_{add}$ とする．ここで，図中の $m_a$ と $m_b$ は，Sub 関数内部で使用するマスク値を示す．$L1$ および $L2$ から得られた値をアンマスクするには，それぞれ以下の値を用いる．

$$m_{L1} = P(A_f(m_{add})) \quad (40)$$

$$m_{L2} = P(A_f(m_{L1})) \quad (41)$$

ここで，$P$ は Permutation 処理，$A_f$ はアフィン変換を表す．$L2$ の出力は整数加算器に入力する前に $m_{L2}$ を用いてアンマスクされる．一方，$L1$ の値は FSR-B のレジスタである $B_{10}$ の出力で，$m_{L1}$ を用いてアンマスクされる．

# 5 実験

本節では，上記の対策を施した KCipher-2 回路の性能を評価する．また，その対策の有効性を FPGA 実装への CPA 実験により示す．

## 5.1 性能評価

KCipher-2 回路では，未対策回路，対策回路ともに，非線形関数部内の $L1$，$L2$ を含むパスがクリティカルパスとなるため，そのパスに含まれる演算器を変えた場合の性能の違いを評価する．整数加算器には RCA と KSA を，Substitution 処理には，合成体を用いた実装 (Comp) とテーブル実装 (Table) を評価に利用する．それぞれの論理合成結果を表 2 に示す．論理合成ツールには Synopsys 社の Design Compiler を，スタンダードセルライブラリには ST Microsystems 90nm-CMOS ライブラリを用いた．表 2 から，未対策回路において，RCA と Comp の組み合わせが面積最小になり，KSA と Table の組み合わせが遅延最小となることがわかる．対策回路においても同様の結果が確認できる．また，同一の演算器構成で，未対策回路と対策回路を比較したところ，面積と遅延の増加はそれぞれ 1.5 倍，2.5 倍程度となることが確認できる．

## 5.2 CPA 耐性評価

次に，上記対策回路の FPGA 実装に対する CPA 攻撃実験の結果を示す．本実験には，サイドチャネル標準評価 FPGA ボード SASEBO-GII (Side-channel Attack Standard Evaluation Board)[11] を使用した．SASEBO-GII には FPGA が 2 つ搭載されており，一方の FPGA1(Xilinx Virtex5) に KCipher-2(RCA+Comp) を実装し，もう一方の FPGA2(Xilinx Spartan3) に実験用の制御回路を実装した．詳細な実験条件は表 3 の通りである．攻撃の対象は，内部鍵 $K_5$（32 ビット）とし，第 3 節で述べた通り，鍵候補を削減するため鍵の予測は 8 ビット単位で下位ビットから順次行った．上位ビットの推定の際には，下位ビットの鍵を既知として利用した．CPA の実験結果を図 3 に示す．これは，攻撃の性質上最も推定が容易となる 31 ～ 24 ビットにおける正しい鍵候補と誤った鍵候補の相関値の推移である．(a) には未対策回路の結果を，(b) には対策回路の結果を示す．ここで，赤線は正しい鍵候補，青線は誤った鍵候補による相関値を表す．横軸は解析に用いた波形数，縦軸は対

表 2: 論理合成結果

| | Adder | Sub | Delay[ns] | Area[kG] |
|---|---|---|---|---|
| W/o counter-measure | RCA | Comp | 5.43 | 16.33 |
| | RCA | Table | 4.46 | 24.11 |
| | KSA | Comp | 3.25 | 24.26 |
| | KSA | Table | 2.26 | 28.88 |
| With counter-measure | RCA | Comp | 12.41 | 24.43 |
| | RCA | Table | 11.53 | 33.79 |
| | KSA | Comp | 6.44 | 27.74 |
| | KSA | Table | 5.63 | 37.84 |

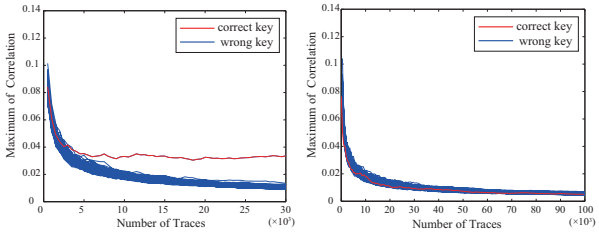
表 3: 実験条件

| Experimental FPGA board (SASEBO-GII) | |
|---|---|
| FPGA1 | Xilinx Virtex5 |
| Clock frequency | 2 MHz |
| Measurement point | Resistor (1 Ω) attached to VDD line |
| **Experimental setting** | |
| Digital oscilloscope | Agilent MSO6104A |
| Probe | Coaxial cable (50 Ω) |
| Sampling frequency | 200 M samples/sec |

応する相関値である．これにより，正しい鍵候補とその他の候補と区別するのに必要な波形数を評価した．使用した消費電力波形数は，未対策回路では 3 万波形，対策回路では，10 万波形を用いた．その結果，未対策回路では，5 千から 6 千波形で攻撃が成功するのに対し，対策回路では 10 万波形を用いても攻撃を防ぐことが確認できた．また，表 2 に示す全ての対策回路においても同様に対策の有効性を確認した．

## 6 まとめ

本稿では，KCipher-2 への電力解析攻撃対策を提案し，その評価を行った．90nm-CMOS スタンダードセルライブラリを用いた結果，整数演算および Sub 関数のアルゴリズムとマスキング方法を適切に設計・適用することで，対策による面積増加と遅延増加をそれぞれ 1.5 倍，2.5 倍程度に抑えられることを示した．また，KCipher-2 回路を SASEBO-GII 上の FPGA に実装し，電力解析攻撃実験を行った結果，対策を施すことで鍵の推定が 10 万波形を用いても




(a) 未対策回路　　(b) 対策回路

図 3: 相関値の推移

困難となることを示した．今後の課題としては，他のサイドチャネル攻撃（故障利用攻撃等）に対する脆弱性とその対策に関する検討などが挙げられる．

## 参考文献


[1] P. Kocher, J. Jaffe, and B. Jun, “Differential power analysis,” Proc. CRYPTO ’99, Lecture Notes in Computer Science, vol.1666, pp.388–397, 1999.

[2] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” Proc. CHES, Lecture Notes in Computer Science, vol.3156, pp.16–29, 2004.

[3] W. Fischer, B. Gammel, O. Kniffler, and J. Velten, “Differential power analysis of stream ciphers,” Proc. CT-RSA, Lecture Notes in Computer Science, vol.4377, pp.257–270, 2007.

[4] S. Kiyomoto, T. Tanaka, and K. Sakurai, “K2: A stream cipher algorithm using dynamic feedback control,” Proc. SECRYPT, pp.204–213, 2007.

[5] 響崇史, 齋藤和也, 本間尚文, 青木孝文, 仲野有登, 福島和英, 清本晋作, 三宅優, “KCipher-2 に対する相関電力解析とその対策,” 暗号と情報セキュリティシンポジウム, no.3C1-2, 2012.

[6] J. Daemen and V. Rijmen, “The Design of Rijndael: AES–the advanced encryption standard”, Springer, 2002.

[7] J. Golic, “Techniques for random masking in hardware,” IEEE Trans. Circuits and Systems, vol.54, no.2, pp.291–300, 2007.

[8] P. Kogge and H. Stone, “A parallel algorithm for the efficient solution of a general class of recurrence equations,” IEEE Trans. Computers, vol.C-22, no.8, pp.786–793, 1973.

[9] 森岡澄夫, 秋下徹, “合成体を用いた AES S-Box 回路に対する DPA 攻撃,” コンピュータセキュリティシンポジウム, pp.679–684, 2004.

[10] M. Akkar and C. Giraud, “An implementation of DES and AES, secure against some attacks,” Proc. CHES, Lecture Notes in Computer Science, vol.2162, pp.309–318, 2001.

[11] “Side-channel Attack Standard Evaluation BOard(SASEBO-GII).” `http://www.aoki.ecei.tohoku.ac.jp/crypto/web/sasebo.html`.