ストリーム暗号プロセッサのサイドチャネル解析と設計に関する研究

修 士 論 文

ストリーム暗号プロセッサのサイドチャネル解析と設計に関する研究

情報基礎科学専攻
響 崇史

目次

第 1 章

緒言

近年，ディジタル技術の革新とインターネットの普及により，パーソナルコンピュータ (PC) に限らず，あらゆる機器がネットワークを介して結びつくことで，生活の様々な面で利便性の向上が期待されている．暗号技術は，このようなユビキタス情報社会を支える基盤技術として，経済活動や社会生活に広く浸透している．ネットワークシステムにおける通信内容の保護，利用者の認証，サービスの真正性の保証などの機能は，暗号技術によって実現される．具体的には，電子商取引に用いられる SSL(Secure Socket Layer) や，携帯電話の利用者認証に用いられる SIM カード (Subscriber Identity Module Card)，PC システムの真正性を保証する TPM(Trusted Platform Module) などが挙げられる．これまで，暗号技術はインターネットセキュリティを中心に発展してきたが，近年では，携帯電話やスマートカードを始めとする組込み機器へも応用が進んでいる．組込み機器では限られた電力，リソースでの暗号処理が求められるため，専用のプロセッサ (暗号プロセッサ) の搭載が一般的となっている．近年普及が進んでいる非接触 IC カードや RFID(Radio Frequency IDentification) においては，消費電力およびリソースの制約が特に厳しいため，暗号プロセッサへの需要はさらに高まると考えられる．

現在，広く用いられている暗号方式は，大きく分けて公開鍵暗号方式と共通鍵暗号方式の 2 つに分類される．公開鍵暗号方式は，暗号化と復号で異なる鍵を用いる方式，共通鍵暗号方式は，暗号化と復号に共通の鍵を用いる方式である．公開鍵暗号方式は，鍵の管理や配送，署名や認証に優れた特徴を有するが，複雑な演算処理により処理速度が遅いといった問題がある．一方で，共通鍵暗号方式は，鍵の配送問題があるが，処理が高速であり，データの暗号化/復号に優れた特徴を持つ．現在の社会では，各暗号方式の性質を加味し，用途に応じて使い分けたり，あるいは組み合わせて用いられている．

共通鍵暗号方式はさらに一定の長さのブロック単位ごとに暗号化/ 復号を行うブロック暗号方式とランダムなデータストリームを発生する疑似乱数生成器を使用して暗号化/復号を行うストリーム暗号方式に分けられる．ストリーム暗号方式は，ブロック暗号方式に

比べて，誤り伝搬が小さく同期が取りやすいことなどのメリットから携帯電話や無線通信区間暗号として採用されることが多い．また，ハードウェアに実装する場合，小規模での実装が可能であるため小型機器への搭載が容易であり，処理速度が速いといった特徴を持つ．現在 GSM 携帯電話で使用される A5/1 や無線 LAN(Local Area Network) で用いられる WEP(Wired Equivalent Privacy) のベースとなっている RC4(Rivest's Cipher 4) がストリーム暗号アルゴリズムの一例である．近年では RC4 の危殆化に伴い，新たに多くの暗号アルゴリズムが開発され，その安全性と実装性能に関する研究がさかんに行われている．欧州で eSTREAM プロジェクト (the ECRYPT stream cipher project)[1] が実施され，ISO/IEC18033-4 によるストリーム暗号アルゴリズムの国際標準規格化が進められるなど次世代ストリーム暗号に対する注目は大きい．注目を集める次世代ストリーム暗号の一つとして KCipher-2[2] が挙げられる．この暗号は，現在広く利用されているブロック暗号の AES(Advanced Encryption Standard)[3] に比べて最大 10 倍の速度での処理を達成し，また，第三者機関による評価により安全性が保証されたことで，2012 年に ISO/IEC18033-4 国際標準暗号として認定された．今後，国内外において組込み機器における暗号プロセッサなど幅広い分野での利用が期待されている．

一方で，近年，暗号プロセッサが動作中に放出するサイドチャネル情報 (処理時間，消費電力，漏洩電磁波等) を観測して，内部の秘密情報を奪うサイドチャネル解析の脅威が指摘されている．こうした解析は，暗号アルゴリズムの設計段階では考慮されないことが多いため，理論上安全とされているアルゴリズムであっても脅威となる可能性がある．サイドチャネル解析は，オシロスコープや PC といった比較的安価な設備で行えることや，解析の痕跡が残らないことから，暗号実装に対する従来の攻撃と比べて危険性が高い．例えば，組込み機器の場合，コストのかかる対策を施すことが難しい上に，所有者自身が攻撃者となることもあり得るため，現実的な脅威となっている．特に，その高い解析能力から，暗号処理中のシステムの消費電力変動を利用した電力解析が注目されており，現在までに様々な解析手法が報告されている [4]．また，それに伴って，サイドチャネル解析に耐性を有する暗号プロセッサの開発も重要な課題となっており，効果的な対策の手法が検討されている．

ストリーム暗号プロセッサに対するサイドチャネル解析についても多くの研究が行われている．一例として，eSTREAM プロジェクトにおいて高い評価を得た Grain，Trivium に対して電力解析が可能であることが報告されている [5]．また，国際標準暗号である MUGI のサイドチャネル解析に対する脆弱性 [6] や，近年標準化された Enocoro128-v2 の電力解析に対する脆弱性とその対策法なども近年報告されている [7]，[8]．いずれも第三者機関による評価においてアルゴリズムの安全性が保証された暗号に対する報告であり，次世代ストリーム暗号に対してもサイドチャネル解析が脅威となり得ることを示している．同様に，KCipher-2 に対してもサイドチャネル解析に関して研究報告がなされ

ている [9]．暗号アルゴリズムの強度指標としては，暗号アルゴリズムをもっとも効率のいい方法で解読した場合の解読計算量で評価する．ある暗号が 2^n オーダーの計算量で解読できる場合，その暗号は，n ビットセキュリティの強度を持つという．ストリーム暗号の場合，秘密情報である秘密鍵の鍵長 n により保証される．秘密鍵が 128 ビットである KCipher-2 の計算量的安全性は，鍵の全数探索分の 128 ビットセキュリティとなる．文献 [9] の報告では，電力解析を行うことで，秘密鍵の一部 (32 ビット) の解析を理論的に 2^{11} の計算量でできるものの，残りの 96 ビットの解析には全数探索により 2^{96} の計算量が必要となるため，この解析手法は現実的脅威とはなりえないと指摘している．

これに対し本論文では，KCipher-2 プロセッサに対する高度な電力解析を提案し，現実的な時間で全ての秘密鍵の解析が可能となることを示す．また，サイドチャネル攻撃標準評価ボード (Side-channel Attack Standard Evaluation BOard: SASEBO) [10] 上の FPGA に実装した KCipher-2 プロセッサを用いた実験と ASIC 実装した KCipher-2 プロセッサを用いた実験により提案手法の有効性を例証する．提案手法では，代表的な電力解析である相関電力解析 (Correlation Power Analysis: CPA) [11] を用いる．CPA では，まず，測定した消費電力波形と相関を有する推定電力値を求める．このとき，一般に，値が変化したレジスタのビット数を用いた消費電力モデルにより推定電力値を見積もる．そのため，レジスタの更新前後の値の両方を求めることが攻撃する上で必要となる．KCipher-2 の場合，レジスタの更新後の値しか求められないため，一般的な消費電力モデルを使用することができない．これに対して提案手法では，同一鍵という条件で暗号プロセッサへの入力が選択できる場合に更新前のレジスタの値が定数となることに着目し，出力が 1 となるレジスタのビット数を利用した消費電力モデルを適用する．さらに，得られた相関値の絶対値と極性 (符号) の両方を利用することで，全数探索により解析するビット数を 96 ビットから 32 ビットに削減することが可能となる．32 ビットの全数探索は，現実的な時間で行うことができるため，秘密鍵 128 ビット全てを推定することが可能となる．

本論文では，上記の解析に対して耐性をもつ KCipher-2 プロセッサの構成も合わせて提案する．提案手法では，乱数マスキングに基づく対策を用いる．乱数マスキングとは，演算に用いる値にあらかじめ乱数による変換を施し，観測される消費電力とアルゴリズムによって決まる真の中間値との関係を独立とする手法である．提案する対策手法では，回路中の解析対象となるデータパスにのみマスキングを適用する．FPGA への実装を通して，上記の電力解析に対して高い耐性をもつ KCipher-2 プロセッサが実現できることを示す．

本論文は，以上の内容をとりまとめたものであり，以下に示す 5 章によって構成される．

第 1 章は，本研究の背景と目的および本論文の概説をまとめた緒言である．

第 2 章では，ストリーム暗号とその実装に関する基礎的考察を行う．まず，ストリーム

暗号の構造，疑似乱数生成器とストリーム暗号の関係性，標準化動向について述べる．その後，暗号プロセッサに対するサイドチャネル解析とその対策手法について述べる．

第 3 章では，KCipher-2 プロセッサのサイドチャネル解析について述べる．まず，KCipher-2 アルゴリズムと処理ステップについて述べる．次に, 相関電力解析の KCipher-2 プロセッサへの適用方法と初期鍵の解析手法を提案する．最後に，FPGA と ASIC の両方のプラットフォームに実装した KCipher-2 プロセッサを用いた実験を通して，提案する解析手法の有効性を実証する．

第 4 章では，耐タンパー性を有する KCipher-2 プロセッサについて述べる．ここでは，上記解析に対する耐性を持った KCipher-2 プロセッサを面積重視と速度重視の二通り設計する．まず，設計した面積重視と速度重視のアーキテクチャを示す．次に，乱数マスキングに基づく対策の適用手法について述べる．その後で，提案する対策の有効性を実験により示す．最後に，各アーキテクチャにおいて対策による性能のオーバーヘッドを評価する．

第 5 章は，結言である．

以上，本論文の企図するところを概説した．

第 2 章

ストリーム暗号とその実装に関する基礎的考察

2.1 まえがき

本章では，共通鍵暗号方式に分類されるストリーム暗号とその実装に関する基礎的考察を行う．まず，ストリーム暗号の概要について述べる．次に，暗号プロセッサに対するサイドチャネル解析について，その代表的な手法と対策について述べる．

2.2 ストリーム暗号の概要

本節では，共通鍵暗号方式に分類されるストリーム暗号について，その概要を述べる．以下では，まず，ストリーム暗号の構造について述べた後，ストリーム暗号用途の疑似乱数生成器に求められる性質について述べる．その後，ストリーム暗号の標準化動向について述べる．

2.2.1 ストリーム暗号の構造

暗号は，決められた手続き (アルゴリズム) に従い，暗号鍵を用いて，メッセージ (平文) を第三者が解読不可能なランダム化されたメッセージ (暗号文) に変換するための暗号化処理と，復号鍵を用いて暗号文から元の平文に戻す復号処理から構成される．なお，用語の定義として，復号鍵を使って暗号文から平文に戻すことを「復号」と呼ぶのに対し，復号鍵を使わずに暗号文から平文を求めること，もしくは平文・暗号文から復号鍵を求めることを「暗号解読」あるいは「暗号解析」という．また，解読を試みることを「攻撃」という．

本論文で注目するストリーム暗号は，暗号鍵と復号鍵に同じ鍵を用いる共通鍵暗号方式に分類され，ある乱数系列を鍵 K として扱い，平文 P との排他的論理和を行うことで暗号文 C を生成する ($C = P \oplus K$)．復号はその逆の処理であり，暗号文 C と鍵 K との排

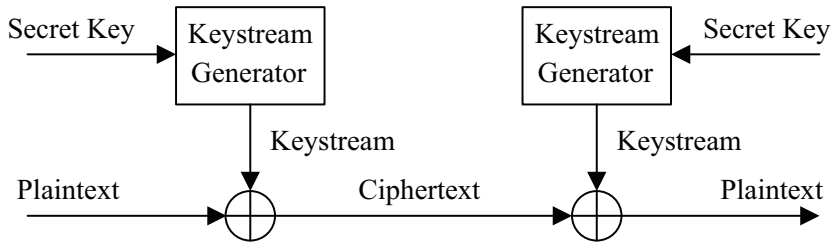


図 2.1　外部同期型ストリーム暗号

他的論理和を行うことで平文 P を得る ($P = C \oplus K$)．平文と鍵長が等しく，かつ鍵として利用する乱数系列が真性乱数系列であれば，鍵 K の全数探索によって暗号文 C から平文 P を復元することは不可能である．これは平文 P と暗号文 C が確立的に独立となるためであり，このような暗号はバーナム (Vernam) 暗号やワンタイムパッド (One-time Pad) と呼ばれる．Vernam 暗号は暗号文単独攻撃に対して完全な安全性を持つ．しかし，通信者間で平文と同じ長さの乱数列を安全に共有しなければならない問題が発生する．そこで，適当な長さの乱数の種を秘密鍵として共有し，確定的アルゴリズムで乱数列を生成する疑似乱数生成器を利用するのが一般的である．したがって，以下では主に疑似乱数生成器を用いたストリーム暗号について述べる．疑似乱数生成器を用いたストリーム暗号の暗号化/復号方式を以下に示す．まず，秘密鍵 K や初期化ベクトル (Initialization Vector: IV) を種として入力し，疑似乱数生成器の内部状態を初期化するアルゴリズムを実行する．その後で，内部状態を更新しながら鍵系列 Z を生成する．生成された鍵系列 Z と平文 P との排他的論理和を行うことで暗号文 C を生成する ($C = P \oplus Z$)．復号はその逆の処理であり，暗号文 C と鍵系列 Z との排他的論理和を行うことで平文 P を得る ($P = C \oplus Z$).

ここで，ストリーム暗号の鍵系列を送受信側で同期させる構造について述べる．一般に外部同期型と自己同期型に分類される．外部同期型は，平文と暗号文の生成とは独立した外部からの信号により鍵生成器の同期が取られる方式である．外部同期型は，送受信者間で完全に同期可能な場合に採用され，再同期が必要な場合のための付加的な仕組みが不可欠である (図 2.1)． これの最大の特長は誤り伝搬がないことにある．常に送受信間で同期しているため，削除，挿入，再送といった能動的な攻撃が行われた場合，即座に同期が取れなくなるので，送受信者がすぐに検知することができる．しかしながら，暗号文に対する改ざんが平文に与える影響を攻撃者が知ることができるので，このような攻撃の対策としてメッセージ認証やデータの完全性を確認できる仕組みも必要となる．一方，自己同期型は，同期のずれを自動的に回復する仕組みを持った方式である (図 2.2)．鍵生成器と外部レジスタとを合わせた構造を持つ．外部レジスタでは暗号文をためておき，レジスタ長

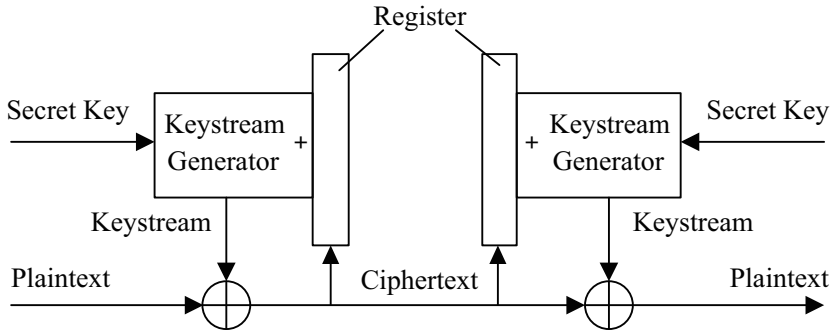


図 2.2　自己同期型ストリーム暗号

だけの暗号文が誤りなく送信できれば，同期が取れなくてもレジスタ内容が一致し同期が回復する．誤りが生じたとしても，最大でもレジスタ長分の伝搬で済む．したがって，削除，挿入といった攻撃が行われた場合，外部同期型と比較して攻撃が行われたのか誤りが生じたのか区別が付きにくい．外部同期型と同様にメッセージ認証やデータの完全性を確認できる仕組みが必要である．

2.2.2　疑似乱数生成器とストリーム暗号

疑似乱数生成器の構成法は様々あるが，ストリーム暗号として用いる場合に，以下の内容が生成される乱数列に求められる．

- 統計的乱数性
- 長周期性
- 予測不可能性

統計的乱数性は，一般的な乱数が重視する乱数性を表す性質であり，長周期性と予測不可能性は暗号用乱数において必要となる安全性に関する性質である．

統計的乱数性とは，出力される乱数系列に 0 と 1 の偏りが生じないこと，連長の出現度数に偏りがないことや顕著なパターンが発見されないことなど統計的な意味で乱数となっているかどうかを示すものである．これがどの程度満たされているかの評価法として代表的なものに，米国標準技術研究所 (National Institute of Standard and Technology: NIST) による評価である NIST FIPS PUB 140-2 ，NIST Special Publication 800-22 や DIEHARD などがある．これらは，いくつかの検定法のセットで構成されており，全ての検定に合格することで乱数性が認められる．

前述したように，暗号用乱数は，乱数性を持つ系列を出力できるだけでなく，安全性に関する性質が求められる．疑似乱数系列の周期性については，総当たり攻撃により内部状

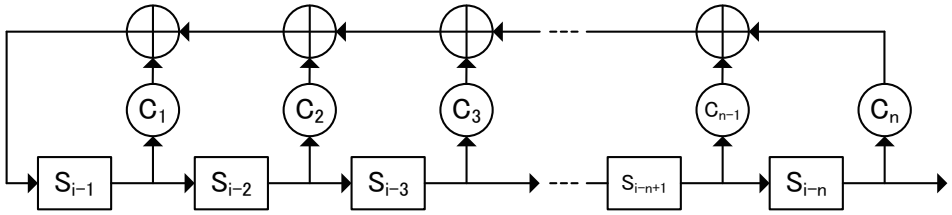


図 2.3　線形フィードバックシフトレジスタ

態を復元されないような長い周期が必要となる．生成する乱数系列には周期はない方が望ましいが，疑似乱数ではある長さの内部状態 (レジスタなど) を持つので，必ず周期をもってしまう．それでも，内部状態の長さが n なら周期は 2^n に近いことが望ましい．予測不可能性については，過去の乱数系列から将来の乱数系列を予測できないことや種の値を逆算できないことを示すが，実際に示すのは容易ではないため，次の性質で代用することも多い．ただし，これらを満たすことは予測不可能性の必要条件に過ぎないことに注意する必要がある．

- 線形複雑度 (linear complexity)
- 免相関性 (correlation immunity)

線形複雑度とは，ある系列が与えられた場合に，それを生成する線形フィードバックシフトレジスタ (Linear Feedback Shift Register: LFSR) が最小何段で構成されるかを示す値である．線形複雑度は大きいことが望ましい．もし，小さい場合，線形複雑度を n とすると，出力系列の連続する $2n$ ビットから全ての出力系列を生成できてしまうためである．LFSR の例を図 2.3 に示す．与えられた系列を生成する LFSR は必ず存在し，線形複雑度は Berlekamp-Massey アルゴリズム [12] などで求めることができる．また，線形複雑度は非線形性と密接に関連しており，非線形性が高いと線形複雑度も大きくなるため，LFSR の出力を非線形に組み合わせる，非線形関数を連結するなどして高度な線形複雑度を実現する場合がある．免相関性は種と出力系列の相関性のことを示す．この免相関性は，そのまま相関攻撃 [13] への耐性へつながる．ストリーム暗号に対する攻撃方法としては様々なものがあるが，もっとも汎用性が高く派生型も多い攻撃方法として相関攻撃が挙げられるため免相関性は重要である．

このように，乱数性と安全性の両方を満たすことがストリーム暗号用途に求められており，現在までに多くのものが提案されてきた．ストリーム暗号で使用される疑似乱数生成器の構成法は大きく分けて二つに分けることができる．(i)LFSR などの有限状態機械を利用するもの，(ii) ブロック暗号技術などを応用したものである．(i) は，LFSR から得られるビット列を非線形関数に通すことで鍵系列を生成する手法であり，ハードウェア

表 2.1　ストリーム暗号の標準 (推奨) アルゴリズム

ISO/IEC	SNOW2.0, MUGI, MULTI-S01 Enocoro128 v2, KCipher-2	
CRYPTREC	MUGI, MULTI-S01, 128-bit RC4	
IETF	RC4	
eSTREAM	SW	HC-128, Rabbit, Salsa20/12, SOSEMANUK
	HW	Grain v1, MICKEY v2, Trivium

実装においても小型実装が可能で安価な傾向がある．FCSR(Feedback with Carry Shift Register) など非線形 FSR を利用したものも最近提案されている．(ii) は，ブロック暗号の設計技術を応用したものであり，状態遷移型とも呼ばれる．これには SHA-1 などハッシュ関数を内部関数に利用したものも含まれる．

2.2.3　ストリーム暗号の標準化動向

ここでは，現在のストリーム暗号の標準化動向を述べる．ストリーム暗号の標準技術を定めている主な機関と各機関で標準技術として採用しているものを表 2.1 に示す．ISO/IEC，CRYPTREC(電子政府推奨暗号リスト，2003 年公開)，IETF(Internet Engineering Task Force) は暗号を実装する暗号製品の普及に対して大きな意味を持つ．例えば RC4 は，インターネットの SSL 機能や無線 LAN の暗号化機能の標準を決定している IETF に標準化されているため，結果的には最も普及したストリーム暗号といえる．ISO/IEC 及び CRYPTREC で採用されている MUGI 及び MULTI-S01 はブロック暗号の技術を応用した状態遷移型の構造を持ち，特に MULTI-S01 はメッセージ認証コード機能を有するなど，暗号化機能だけでなく，付加機能をもっており，これまで様々な場面で使用されてきた．また，Enocoro128-v2 や KCipher-2 は LFSR 型の構造を持つ小型高速実装を目指したストリーム暗号であり，Enocoro128-v2 は ISO/IEC29192 国際標準暗号として，KCipher-2 は ISO/IEC18033-4 国際標準暗号として近年認定されたことで，今後，国内外において携帯電話アプリへの実装や RFID など小型デバイスへの利用など幅広い分野での利用が期待されている．一方，eSTREAM はヨーロッパで行われたストリーム暗号評価活動であり，学術成果主体の研究プロジェクトであることから正確には標準化機関ではないが，比較的新しい選定作業を行った (2008 年) ものとして注目されている．この選定の特徴としては，ソフトウェア実装向き (SW) とハードウェア実装向き (HW) と，実装を意識したという点にある．

eSTREAM での選定の特徴にあるように，近年の標準化における評価基準に，実装に関する検討が加えられている．現在，暗号アルゴリズムの実装形態としては，ソフトウェ

ア実装とハードウェア実装に大別できる．ソフトウェア実装とは，暗号アルゴリズムの機能を持つプログラムを作成することで，CPU や DSP(Digital Signal Processor) などの汎用プロセッサ上で暗号機能を実現することを指す．一方，ハードウェア実装は，暗号アルゴリズムを専用回路で実現するものを指す (本論文では暗号プロセッサとしている)．このような実装運用の面での脆弱性により安全に利用できなくなる場合もある．例えば，無線 LAN の暗号化方式の一つである WEP で利用される RC4 は安全性評価が活発に行われ，最近では一般的な PC を利用して約 10 秒で解析できる攻撃結果が得られている．現在，RC4 に限らず多くの暗号に対して，実装に関する研究が活発に行われている．ストリーム暗号も例外ではない．その中で，ディジタルデータのやり取りが複雑化するにつれて，守るべき情報の性質や実装性能の関係から適切な暗号アルゴリズムを選ぶことが重要になりつつある．

2.3 暗号プロセッサに対するサイドチャネル解析に関する基礎的考察

本節では，近年暗号プロセッサに対する脅威として注目されているサイドチャネル解析について述べる．まず，サイドチャネル解析についての概要を述べる．その後に，サイドチャネル解析の中でも強力な電力解析の代表的な手法，対策について述べる．

2.3.1 サイドチャネル解析の概要

近年，暗号プロセッサに対して様々な解析手法が提案されている．Web サーバや銀行の ATM など，暗号プロセッサへの物理的なアクセスを制限できる場合には，サイドチャネル解析は問題とならない．しかし，携帯電話やスマートカードなどの組込み機器は，攻撃者自身により所持されることが多く，暗号プロセッサへの物理的なアクセスが容易であることから，サイドチャネル解析は現実的な脅威であるといえる．

代表的な暗号プロセッサへの攻撃手法は，破壊攻撃と非破壊攻撃の 2 種類に分類される．破壊攻撃は，搭載されている暗号プロセッサに対して狭義タンパー手段 (切る，削る，孔を開ける，溶かす，分解するなど) により実装内部に侵入し，直接内部データの観察や改変を行う攻撃である．しかし，暗号プロセッサに対して直接アクセスする必要があること，LSI の開封には専門知識や高価な設備が必要になること，攻撃の検知が可能であり，対策を比較的容易に施せることから破壊攻撃は実行が非常に困難である．一方，非破壊攻撃は，さらにフォールトベース攻撃 (Fault Based Attack)，サイドチャネル解析 (Side Channel Analysis) に分類される．フォールトベース攻撃は，暗号実装に対して何らかの手段で誤動作を誘発させ，実装の出力結果を観測することで実装内部の秘密情報を読み

取る攻撃である．一方サイドチャネル解析は，暗号プロセッサに正規の入力を与えた際に，暗号プロセッサが動作中に放出するサイドチャネル情報 (処理時間，消費電力，漏洩電磁波等) を観測して，秘密情報を解析する手法である．サイドチャネル解析は，オシロスコープや PC といった比較的安価な設備で実行可能であること，攻撃の痕跡が残らず検知が難しいことから，組込み機器に実装された暗号プロセッサに対する従来の攻撃と比べて危険性が高く，現実的脅威として，非常に注目されている．処理時間の計測に基づくタイミング攻撃 (Timing Attack) や消費電力を用いる電力解析，漏洩電磁波を用いる電磁波解析がサイドチャネル解析の中でも代表的な解析手法として，多くの暗号アルゴリズムへの適用が報告されている．

2.3.2　電力解析

電力解析とは，暗号プロセッサが動作することによって生じる消費電力情報から，内部の演算や中間データ等の解析を行い，秘密情報を解析する手法である．この手法は，暗号プロセッサそのものからだけでなく実装された基板の消費電力波形からも解析可能であることなどから，非常に脅威である手法となっている．以下では，電力解析の中でも代表的な手法である単純電力解析 (Simple Power Analysis: SPA) [14], [15] および相関電力解析 (CPA) について述べる．

単純電力解析 (SPA)

SPA は，測定した一つもしくは複数の消費電力波形から内部情報を解析する手法である．図 2.4 に SPA のイメージを示す．ある暗号処理が秘密鍵に応じて A と B という演算の繰り返しで実行される場合，この A と B の電力波形を見分けることで，その秘密鍵を解析することができる．SPA は，一回の暗号化/復号に多くの計算を必要とする公開鍵暗号 (RSA 暗号 [16] など) が実装された機器に対して有効とされている．特にソフトウェア実装された場合，演算により命令系列が異なっているため，その違いを波形から容易に見分けることができる．複数の波形を用いる SPA では，特徴的な波形が得られる入力を選択し，波形間で同一波形パターンの生じる位置を比較することで秘密情報を解析できる．

相関電力解析 (CPA)

CPA では，暗号処理を実行する際の消費電力を多数測定し，統計処理により秘密鍵を解析する手法である．図 2.5 に CPA の概要を示す．この手法は，ノイズの影響に対しても強いことから，消費電力波形の入力データ依存性が低く，SPA に耐性がある共通鍵暗号 (AES など) が実装された機器に対して有効とされている．以下では，消費電力の測定により得られる情報を利用し，暗号アルゴリズムを解読する手法について述べる．詳細な手法は，使用する電力モデルおよび暗号アルゴリズムに依存するため，ここでは一般的な事項について述べる．

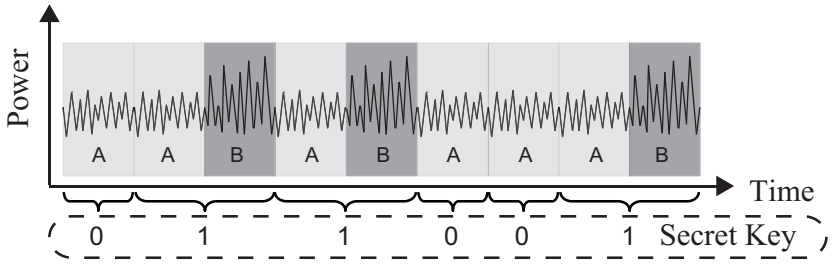


図 2.4　単純電力解析 (SPA)

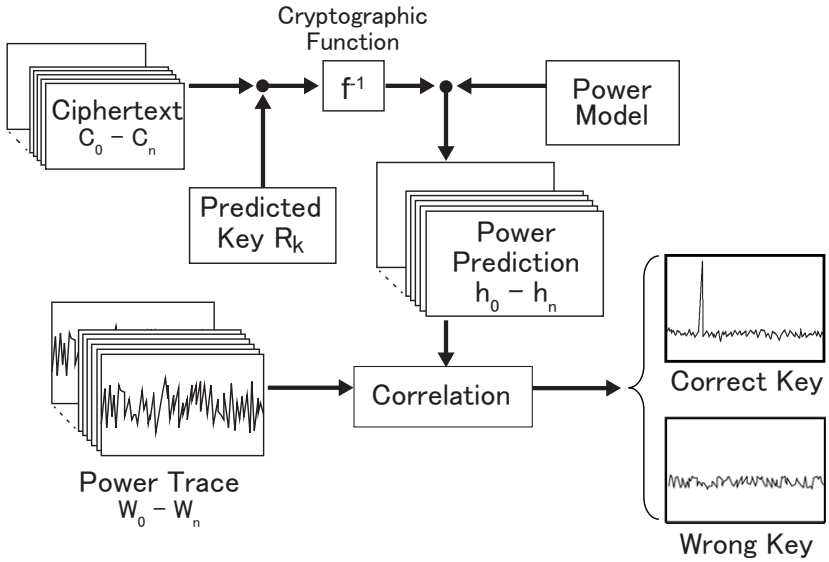


図 2.5　相関電力解析 (CPA)

暗号アルゴリズム E，平文 P_i，暗号文 C_i，鍵 K_E に対し，暗号処理は次式で表現することができる．

$$C_i = E(P_i, K_E) \tag{2.1}$$

攻撃者は，まず，N 個の平文 $P_0 \sim P_{N-1}$ が暗号化される時，攻撃者は対応する暗号文 $C_0 \sim C_{N-1}$ を収集する．これと同時に，処理中の暗号プロセッサの消費電力の計測値 $W_0 \sim W_{N-1}$ を収集する．攻撃者にとって，暗号文 C_i は既知とし，攻撃の目的は鍵 K_E の復元とする．多くの場合，秘密情報である鍵と消費電力は直接の相関を持たないため，直接的に情報を抽出することは困難である．そのため，消費電力の計測値 W_i と相関を有する中間値 I_i をもとに電力を推定する必要がある．この推定電力値を h_i とする．一般的に下記の関係を満たす箇所に注目する．

$$C_i = f(I_i, R_K) \tag{2.2}$$

ここで，f は暗号アルゴリズム E の部分関数で，データ幅が十分に狭く，かつ逆変換 f^{-1} が可能なものである．R_K は暗号アルゴリズム内部で使用される部分鍵である．逆変換ができることから以下の式が成立する．

$$I_i = f^{-1}(C_i, R_K) \tag{2.3}$$

しかし，部分鍵 R_K は未知であり，消費電力の計測値 W_i との相関を調べることは不可能である．そのため，部分鍵 R_K について全数探索を行い，その全てについて中間値 I_i，推定電力値 h_i を求め，消費電力の計測値 W_i との相関を評価する．その結果，最も高い相関を持つ候補を部分鍵の真の値であると見なすことで部分鍵 R_K を求める．解析した部分鍵の情報から秘密鍵 K_E を解読する．多くの場合，鍵スケジュールアルゴリズムを解析することで秘密鍵を解読できる．以上が攻撃の流れである．ここで，関数 f は暗号アルゴリズム E を構成する部分関数と仮定したため，探索空間は鍵長の全探索と比較して大幅に削減され，現実的な時間で実行可能となる．

相関電力解析を行う上で重要となるのは，中間値 I を適切に選ぶこと，推定電力値 h を求めるための中間値 I と消費電力の計測値 W の関係を規定した消費電力モデルを適切に仮定すること，そして，推定電力値 h と消費電力の計測値 W の間に存在する相関の評価を適切に行うことである．現在では，消費電力モデルとして Hamming Weight (HW) モデルや Hamming Distance (HD) モデル [4] が主に用いられる．また，相関値の評価には，式 (2.4) に示すピアソンの相関係数 [11] を用いるのが一般的である．

$$\rho = \frac{\sum_{n=0}^{N-1} \left(h_n - \overline{h}\right) \cdot \left(W_n - \overline{W}\right)}{\sqrt{\sum_{n=0}^{N-1} \left(h_n - \overline{h}\right)^2 \cdot \sum_{n=0}^{N-1} \left(W_n - \overline{W}\right)^2}} \tag{2.4}$$

ここで，$\overline{h}$，$\overline{W}$ はそれぞれ推定電力値の平均値，攻撃対象となる時刻における消費電力の計測値 W の平均値である．

2.3.3　電力解析への対策手法

電力解析への対策には，大きく分けてハイディングとマスキングがある．ハイディングとは，暗号プロセッサから得られる情報と内部処理や中間値との依存関係を隠す対策である．暗号処理中において，入力信号のパターンによらず常にランダムな量もしくは一定量の電力を消費することで対策を実現する．一方，マスキングとは，演算に用いる値にあらかじめ乱数による変換を施し，観測される消費電力とアルゴリズムによって決まる真の中間値 (マスクされていないときの値) との関係を独立とする対策である．消費電力と中間値の関係を推定できたとしても，計算によって得た中間値自体は真の中間値と無関係な

ため解析が不可能となる．これまでに，ハイディングおよびマスキングの両方において，アーキテクチャレベルおよび回路の設計レベルで様々な対策が提案されている．ハイディングの主な手法としては，入力信号に対して相補的なゲート動作をさせることで消費電力を一定にする SABL(Sense Amplifer Based Logic)[17] や二線ロジックの SABL を応用した WDDL(Wave Dynamic Dfferential Logic)[18] などが挙げられる．一方で，マスキングの主な手法としては，MAO(Masked And Operation)[19] や複数の乱数マスクを用いる Threshold Implementation[20]，WDDL に乱数を組み合わせた MDPL(Masked Dual-rail Precharge Logic)[21]，出力許可付きの多数決論理ゲートを用いたトランジスタレベルの対策である RSL(Random Switching Logic)[22] 等が挙げられる．

どの手法も計算時間のみならず，回路面積や消費電力の面でもオーバーヘッドが生じる．未対策の回路に比べて 2 倍以上の演算時間，4 倍以上の回路面積や消費電力を発生させる対策も中には存在する．そのため，小型の組込み機器への暗号プロセッサの搭載が期待される現在において，オーバーヘッドの小さい対策手法が求められている．

2.4　むすび

本章では，ストリーム暗号とその実装に関する基礎的考察を行った．まず，ストリーム暗号の構造を述べた．続いて，ストリーム暗号用途の疑似乱数生成器に求められる乱数性と安全性について述べた．また，ストリーム暗号の標準化動向について述べた．その後，アルゴリズム自体の安全性だけではなく，実装の安全性が重要となることを述べ，サイドチャネル解析について概説した．サイドチャネル解析の中でも，強力な手法として知られる電力解析について，代表的な解析手法と対策手法を述べた．

第 3 章

KCipher-2 プロセッサのサイドチャネル解析

3.1 まえがき

本章では，ストリーム暗号 KCipher-2 をハードウェア実装した暗号プロセッサに対するサイドチャネル解析について述べる．まず，KCipher-2 について述べ，次に，相関電力解析の KCipher-2 への適用方法と初期鍵の解析手法について述べる．その後で，FPGA，ASIC に実装した KCipher-2 プロセッサそれぞれに対して，提案手法による鍵推定実験を行い，提案手法の有効性を示す．以下では，秘密鍵を初期鍵 (Initial Key: IK) と呼ぶ．

3.2 KCipher-2

KCipher-2 は，128 ビットの初期鍵 (IK) と 128 ビットの初期化ベクトル (IV) の 2 つの独立なパラメータを入力として有し，暗号化/復号に使用する 64 ビットの鍵系列 Z を出力するストリーム暗号である．KCipher-2 アルゴリズムは，図 3.1 で示すように，以下の要素で構成されている．

- 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 $R1$，$R2$，$L1$，$L2$ を有する非線形関数部

なお，KCipher-2 内部で使用されるレジスタの語長は全て 32 ビットであり，図中の ZH，ZL は 64 ビットの鍵系列 Z のそれぞれ上位，下位 32 ビットの値である．以下にそれぞれの構成要素，処理ステップについて説明する．

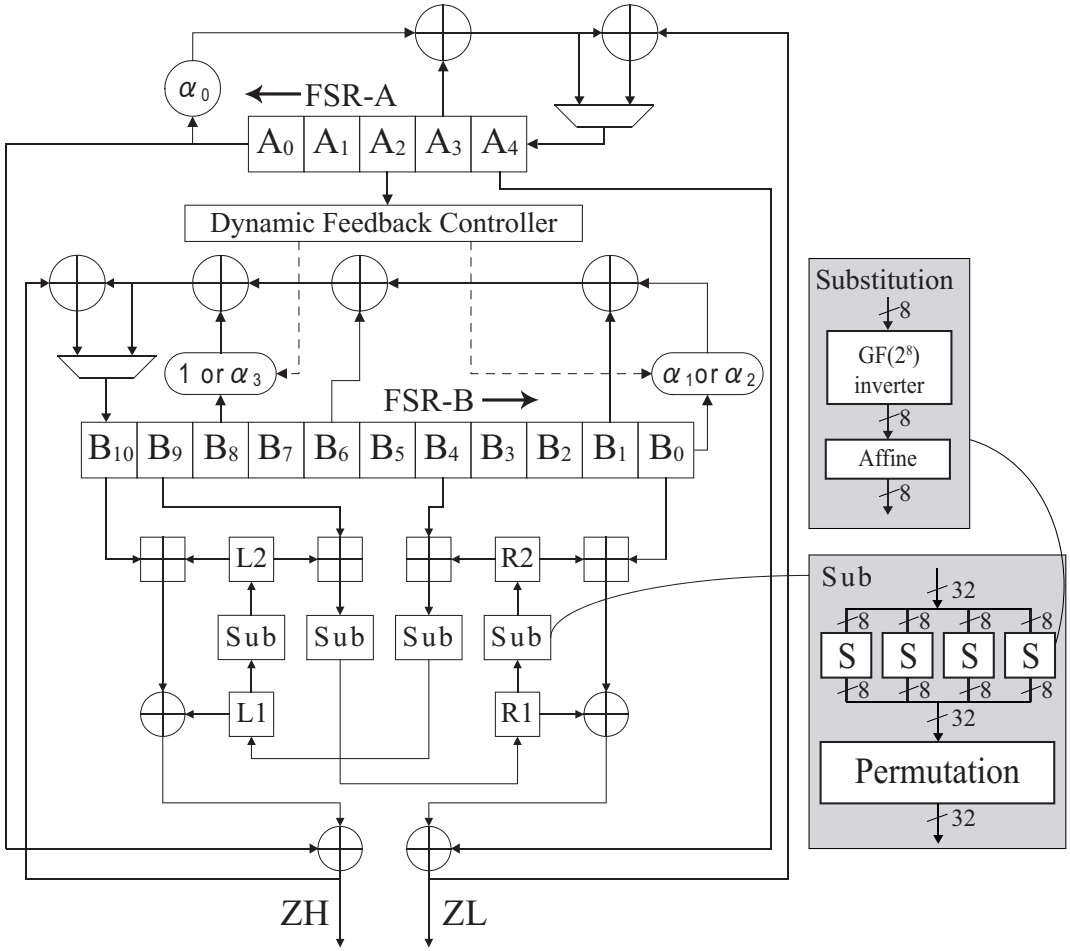


図 3.1　ストリーム暗号 KCipher-2

3.2.1　FSR-A と FSR-B

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．ここで，β を原始多項式 $x^8 + x^7 + x^6 + x + 1 \in GF(2)[x]$ の根とする．また，1 バイトの値 y を $(y_7, y_6, \ldots, y_1, y_0)_2$ と表す．ただし，y_7 が最上位ビット，y_0 が最下位ビットである．この時，y を $GF(2^8)$ 上の元とすると，$y = y_7\beta^7 + y_6\beta^6 + \ldots + y_1\beta^1 + y_0$ のように表すことができる．同様に，γ，δ，ζ をそれぞれ式 (3.1)，式 (3.2)，式 (3.3) の原始多項式の根とする．

$$x^8 + x^5 + x^3 + x^2 + 1 \in GF(2)[x] \tag{3.1}$$
$$x^8 + x^6 + x^3 + x^2 + 1 \in GF(2)[x] \tag{3.2}$$
$$x^8 + x^6 + x^5 + x^2 + 1 \in GF(2)[x] \tag{3.3}$$

また，α_0 を以下の 4 次既約多項式の根とする．

$$x^4 + \beta^{24}x^3 + \beta^3x^2 + \beta^{12}x + \beta^{71} \in GF(2)[x] \tag{3.4}$$

さらに，32 ビットの値 Y を (Y_3,Y_2,Y_1,Y_0) で表すこととする．ただし，Y_i は 1 バイトの値であり，Y_3 が最上位バイトである．この時，Y は $Y = Y_3\alpha_0^3 + Y_2\alpha_0^2 + Y_1\alpha_0 + Y_0$ で表される．同様に，α_1，α_2，α_3 をそれぞれ以下の 4 次既約多項式の根とする．

$$x^4 + \gamma^{230}x^3 + \gamma^{156}x^2 + \gamma^{93}x + \gamma^{29} \in GF(2^8)[x] \tag{3.5}$$

$$x^4 + \delta^{34}x^3 + \delta^{16}x^2 + \delta^{199}x + \delta^{248} \in GF(2^8)[x] \tag{3.6}$$

$$x^4 + \zeta^{157}x^3 + \zeta^{253}x^2 + \zeta^{56}x + \zeta^{16} \in GF(2^8)[x] \tag{3.7}$$

上記の定義を用いて，FSR-A のフィードバック関数と FSR-B のフィードバック関数は，それぞれ以下で表される．

$$A_i^{(t+1)} = \begin{cases} A_{i+1}^{(t)} & (i \neq 4) \\ \alpha_0 A_0^{(t)} \oplus A_3^{(t)} & (i = 4) \end{cases} \tag{3.8}$$

$$B_j^{(t+1)} = \begin{cases} B_{j+1}^{(t)} & (j \neq 10) \\ \alpha_{(cl1^{(t)})} B_0^{(t)} \oplus B_1^{(t)} \oplus B_6^{(t)} \oplus \alpha_3^{cl2^{(t)}} B_8^{(t)} & (j = 10) \end{cases} \tag{3.9}$$

ここで，$\alpha_{(cl1^{(t)})}$ は，

$$\alpha_{(cl1^{(t)})} = \alpha_1^{cl1^{(t)}} \oplus \alpha_2^{1-cl1^{(t)}} \oplus 1 \tag{3.10}$$

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 (3.9) の $cl1^{(t)}$，$cl2^{(t)}$ 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．

3.2.2　非線形関数部

非線形関数部は，4 つの 32 ビット入出力の整数加算器 (図 3.1 中の ⊞) と 4 つの置換関数 Sub，4 つの内部レジスタ $R1$, $R2$, $L1$, $L2$ から構成される．FSR-A の 2 つのレジスタの値 A_0, A_4，FSR-B の 4 つのレジスタの値 B_0, B_4, B_9, B_{10} 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，Substitution と Permutaiton から構成され，32 ビット単位で置換を施す．以下でこの関数について述べる．

Substituition(以下，S-box とよぶ) 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関数である．この関数は，ガロア体 $GF(2^8)$ 上の逆元演算と $GF(2)$ 上のアフィン変換によって実現される．ここで，$GF(2^8)$ を構成する既約多項式は $\beta^8+\beta^4+\beta^3+\beta+1$ である．また，アフィン変換では以下に示す行列演算を行う．

$$\boldsymbol{a'} = \boldsymbol{A_f a} \tag{3.11}$$

ここで，$\boldsymbol{a'}$，$\boldsymbol{a}$ は $GF(2^8)$ 上の元をベクトル表現したものであり，$a = a_7\beta^7 + a_6\beta^6 + \cdots + a_1\beta^1 + a_0\beta^0$，$a' = a'_7\beta^7 + a'_6\beta^6 + \cdots + a'_1\beta^1 + a'_0\beta^0$ と表されるとき，

$$\boldsymbol{a} = [a_0, a_1, a_2, a_3, a_4, a_5, a_6, a_7]^T \tag{3.12}$$

$$\boldsymbol{a'} = [a'_0, a'_1, a'_2, a'_3, a'_4, a'_5, a'_6, a'_7]^T \tag{3.13}$$

である．また，アフィン行列 $\boldsymbol{A_f}$ は，以下のように定義される．

$$\boldsymbol{A_f} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \tag{3.14}$$

ここで，a_i，$a'_i \in GF(2)$ $(0 \le i < 8)$ はそれぞれアフィン変換の入力，出力を表す．

Permutation は，4 バイトの値を用いてビット演算を行い，別の 4 バイトの値に変換する．Permutation で行われる演算はガロア体 $GF(2^8)$ の行列演算である．Permutation の入力を上位から 1 バイト毎に a_3，a_2，a_1，a_0，出力を a'_3，a'_2，a'_1，a'_0 とすると，Permutation は以下の行列演算により実現される．

$$\begin{bmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{bmatrix} = \begin{bmatrix} v_0 & v_1 & v_2 & v_3 \\ v_3 & v_0 & v_1 & v_2 \\ v_2 & v_3 & v_0 & v_1 \\ v_1 & v_2 & v_3 & v_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{3.15}$$

ここで，$v_i (0 \le i < 4)$ は $GF(2^8)$ の定数であり，$v_0 = (02)_{16}$，$v_1 = (03)_{16}$，$v_2 = (01)_{16}$，$v_3 = (01)_{16}$ である．

3.2.3　処理ステップ

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

初期化処理は，初期鍵 IK の内部鍵 K への拡張，内部鍵 K と初期ベクトル IV の読み込み，内部状態の初期化の順に処理が行われる．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．

まず，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．

$$K_k = \begin{cases} IK_{3-k} & (0 \le k \le 3) \\ K_{k-4} \oplus f(K_{k-1}) & (k = 4n) \\ K_{k-4} \oplus K_{k-1} & (k \neq 4n) \end{cases} \tag{3.16}$$

ここで，$f(K_{k-1})$ は，次式のような処理を行う．

$$f(K_{k-1}) = \mathrm{Sub}(\mathrm{RotWord}(K_{k-1})) \oplus \mathrm{rc} \tag{3.17}$$

ただし，RotWord はバイト単位の循環シフトを表す．すなわち，$K_i = (K_{i,3}, K_{i,2}, K_{i,1}, K_{i,0})$（$K_{i,j}$ は 1 バイトのデータ）に対して，$\mathrm{RotWord}(K_i) = (K_{i,2}, K_{i,1}, K_{i,0}, K_{i,3})$ である．また，Sub は関数 Sub の処理を表す．rc は定数であり，$k = 4$ の時，$rc = (01000000)_{16}$，$k = 8$ の時，$rc = (02000000)_{16}$ となる．

次に，内部鍵 K と初期化ベクトル IV を読み込み，KCipher-2 の初期状態を以下のように決定する．

$$\begin{aligned} &A_i = K_{4-i}\ (i = 0, \ldots, 4), B_0 = K_{10}, B_1 = K_{11}, B_2 = IV_3, B_3 = IV_2, \\ &B_4 = K_8, B_5 = K_9, B_6 = IV_1, B_7 = IV_0, B_8 = K_7, B_9 = K_5, B_{10} = K_6, \\ &R1 = R2 = L1 = L2 = 0 \end{aligned}$$

データの読み込み後，KCipher-2 を 24 クロック $(t = 0, 1, \ldots, 23)$ 動作させ，内部状態の撹拌を行う．

$$R1^{(t+1)} = \mathrm{Sub}(L2^{(t)} \boxplus B_9^{(t)}) \tag{3.18}$$

$$R2^{(t+1)} = \mathrm{Sub}(R1^{(t)}) \tag{3.19}$$

$$L1^{(t+1)} = \mathrm{Sub}(R2^{(t)} \boxplus B_4^{(t)}) \tag{3.20}$$

$$L2^{(t+1)} = \mathrm{Sub}(L1^{(t)}) \tag{3.21}$$

$$A_i^{(t+1)} = \begin{cases} A_{i+1}^{(t)} & (i \neq 4) \\ \alpha_0 A_0^{(t)} \oplus A_3^{(t)} \oplus ZL^{(t)} & (i = 4) \end{cases} \tag{3.22}$$

$$B_j^{(t+1)} = \begin{cases} B_{j+1}^{(t)} & (j \neq 10) \\ \alpha_{(cl1^{(t)})} B_0^{(t)} \oplus B_1^{(t)} \oplus B_6^{(t)} \oplus \alpha_3^{cl2^{(t)}} B_8^{(t)} \oplus ZH^{(t)} & (j = 10) \end{cases} \tag{3.23}$$

ここで，$\alpha_{(cl1^{(t)})}$ は式 (3.10) と等しい．また，$R1^{(t)}$, $R2^{(t)}$, $L1^{(t)}$, $L2^{(t)}$ は時刻 t の非線形関数部の内部レジスタの値である．初期化処理の間は，式 (3.8), (3.9) とは異なり，鍵系列 Z は出力されず，フィードバック関数に加えられる．

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 $Z^{(t)} = (ZH^{(t)}, ZL^{(t)})$ 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．

$$ZH^{(t)} = B_{10}^{(t)} \boxplus L2^{(t)} \oplus L1^{(t)} \oplus A_0^{(t)} \tag{3.24}$$

$$ZL^{(t)} = B_0^{(t)} \boxplus R2^{(t)} \oplus R1^{(t)} \oplus A_4^{(t)} \tag{3.25}$$

鍵系列出力後，内部状態の更新を行い，時刻 $t+1$ の内部レジスタの値を得る．内部状態の更新は，式 (3.8), (3.9), (3.18)〜(3.21) によって行われる．

3.3　KCipher-2 に対する相関電力解析 (CPA)

ストリーム暗号に対するサイドチャネル解析の場合，攻撃者が初期化ベクトルを観測もしくは選択可能という条件下で，多数の初期化ベクトルを用いるという想定がされる場合が多い．そこで，提案する解析手法では，同一鍵で初期化ベクトル 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．本手法では，初期鍵を直接復元するのではなく，複数の 32 ビットの内部鍵を CPA により解析し，その情報をもとに初期鍵を復元していく．また，解析に利用する鍵候補を削減するために，8 ビットごとの CPA による内部鍵の解析を行う．上位バイトから解析を開始し，得られた鍵情報を以降の解析に使用する．以下に解析の詳細を述べる．

はじめに，CPA に使用する推定電力値を決定するために必要となる，消費電力波形と相関を有する中間値を求める．中間値には，入力情報が完全に攪拌される前の初期化処理開始直後の内部レジスタの値を利用する．具体的には，初期化処理開始後の 3 クロック目と 4 クロック目における非線形関数内の内部レジスタ $L1$ の値 $(L1^{(3)}, L1^{(4)})$ に注目する．その値は以下の式で与えられる．

$$L1^{(3)} = \mathrm{Sub}(IV_1 \boxplus \mathrm{Sub}(\mathrm{Sub}(K_5))) \tag{3.26}$$

$$L1^{(4)} = \mathrm{Sub}(IV_0 \boxplus \mathrm{Sub}(\mathrm{Sub}(K_6 \boxplus \mathrm{Sub}(0)))) \tag{3.27}$$

式 (3.26) と式 (3.27) において，内部鍵 K_5 と K_6 で決定される $\mathrm{Sub}(\mathrm{Sub}(K_5))$ と $\mathrm{Sub}(\mathrm{Sub}(K_6 + \mathrm{Sub}(0)))$ を推定対象とし，それぞれ CK_5，CK_6 とおく．ここで，注目するレジスタ $L1^{(t)}$ の値を求めるためには，推定する鍵の値と IV_1，IV_0 との整数加算中に生じる桁上がりの影響とそれに続く Sub 関数内の Permutation に注意する必要がある．鍵の最下位バイトを推定する際には，上位バイトの情報を利用できるため，IV と予

測鍵により中間値を正しく推定できるが，鍵の上位・中間バイトを推定する際には，推定するバイトとそれ以前の推定により得た情報しか使用できず，整数加算の桁上がりの影響と Permutation の出力を完全に再現することが困難となる．これを解決するために，推定バイトより下位のビットが 0 となるような IV_1，IV_0 を選択する．これにより，桁上げの伝搬を無視できるだけでなく，Permutation の演算の一部を定数として扱えるため，中間値を近似的に求めることが可能となる．

CK_5 の上位から 2 バイト目の推定に使用する中間値の算出を例に挙げる．IV_1 と CK_5 を上位から 1 バイトずつ分割し，$IV_1 = (iv_{1,3}, iv_{1,2}, iv_{1,1}, iv_{1,0})$，$CK_5 = (ck_{5,3}, ck_{5,2}, ck_{5,1}, ck_{5,0})$ と表す．$L1^{(3)}$ も同様に 1 バイトずつに分割し，$L1^{(3)} = (l_3^{(3)}, l_2^{(3)}, l_1^{(3)}, l_0^{(3)})$ と表す．中間値 $l_2^{(3)}$ は式 (3.14) より以下のように表せる．

$$\begin{aligned} l_2^{(3)} = {} & S(iv_{1,3} \boxplus ck_{5,3} \boxplus \mathrm{carry}) \otimes (03)_{16} \oplus S(iv_{1,2} \boxplus ck_{5,2}) \otimes (02)_{16} \\ & \oplus S(iv_{1,1} \boxplus ck_{5,1}) \oplus S(iv_{1,0} \boxplus ck_{5,0}) \end{aligned} \tag{3.28}$$

ここで，S は関数 S-box の出力を表し，$\otimes$ は $GF(2^8)$ 上の乗算を表す．また，carry は $(iv_{1,2} \boxplus ck_{5,2})$ の桁上げを表す．$ck_{5,2}$ の推定の場合には，$ck_{5,3}$ は既知であり，$iv_{1,1} = iv_{1,0} = 0$ となる IV_1 を選択するため，式 (3.28) の第 3 項と第 4 項は定数項となる．定数項は統計処理を行う上で無視できるため，解析に使用する中間値 $l_2^{(3)}$ は以下で近似できる．

$$l_2^{(3)} \simeq S(iv_{1,3} \boxplus ck_{5,3} \boxplus \mathrm{carry}) \otimes (03)_{16} \oplus S(iv_{1,2} \boxplus ck_{5,2}) \otimes (02)_{16} \tag{3.29}$$

同様にして，他の鍵バイトの解析に使用する中間値 $L1^{(t)} = (l_3^{(t)}, l_2^{(t)}, l_1^{(t)}, l_0^{(t)})$ の値は，S-box の出力 $Sout = (s_3^{(t)}, s_2^{(t)}, s_1^{(t)}, s_0^{(t)})$ を用いて以下のように近似できる．

$$l_3^{(t)} \simeq s_3^{(t)} \otimes (02)_{16} \tag{3.30}$$

$$l_2^{(t)} \simeq s_3^{(t)} \otimes (03)_{16} \oplus s_2^{(t)} \otimes (02)_{16} \tag{3.31}$$

$$l_1^{(t)} \simeq s_3^{(t)} \oplus s_2^{(t)} \otimes (03)_{16} \oplus s_1^{(t)} \otimes (02)_{16} \tag{3.32}$$

$$l_0^{(t)} = s_3^{(t)} \oplus s_2^{(t)} \oplus s_1^{(t)} \otimes (03)_{16} \oplus s_0^{(t)} \otimes (02)_{16} \tag{3.33}$$

次に，使用する消費電力モデルを設定する．中間値として注目するレジスタ値の直前のクロックにおける値は，それぞれ以下の式で与えられる．

$$L1^{(2)} = \mathrm{Sub}(K_9 \boxplus \mathrm{Sub}(0)) \tag{3.34}$$

$$L1^{(3)} = \mathrm{Sub}(IV_1 \boxplus CK_5))) \tag{3.35}$$

式 (3.34)，(3.35) 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，攻撃者が求めることはできない．そのため，CPA で通常用いられる Hamming Distance(HD) モデ

表 3.1　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

ルの使用は難しい．これは，値が変化したレジスタのビット数を用いた消費電力モデルであるためである．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，出力が 1 となるレジスタのビット数を利用した消費電力モデルである Hamming Weight(HW) を適用するため，1 ビットの消費電力モデルを考える．表 3.1 に 1 ビット HD モデルと 1 ビット 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 の推定では，消費電力モデルに 1 ビット HW モデルを使用する．第 j バイト $(0 \leq j \leq 3)$ の第 i ビット $(0 \leq i \leq 7)$ の CK_5 の推定電力値 $h_{j,i}^{(3)}$ $(\in \{0,1\})$ は次式で与えられる．

$$h_{j,i}^{(3)} = l_{j,i}^{(3)} \tag{3.36}$$

一方，CK_6 の推定では，CK_5 を推定できたならば，式 (3.35) 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 の推定電力値 $h_{j,i}^{(4)}$ $(\in \{0,1\})$ は次式で与えられる．

$$h_{j,i}^{(4)} = l_{j,i}^{(4)} \oplus L1_{i+8j}^{(3)} \tag{3.37}$$

$h_{j,i}^{(3)}$，$h_{j,i}^{(4)}$ と取得した消費電力波形との相関を式 (2.4) で表されるピアソンの相関係数 ρ 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．

$$\rho_j^{(3)} = \sqrt{\frac{1}{8}\sum_{i=0}^{7}\left(\rho_{j,i}^{(3)}\right)^2} \tag{3.38}$$

$$\rho_j^{(4)} = \sqrt{\frac{1}{8}\sum_{i=0}^{7}\left(\rho_{j,i}^{(4)}\right)^2} \tag{3.39}$$

式 (3.38)，(3.39) 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 (3.16) から K_2 を計算できる．以上までが相関値の絶対値を利用した解析手法であり，これにより得られる内部鍵は K_2，K_5，K_6 である．

次に相関値の極性を利用した K_9 の推定について述べる．CK_5 が既知である場合，CK_5 の推定時に得られた相関値の極性から，直前の定数値 $L1^{(2)}$ を推定することが可能となる．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 3.1 より，2 つの消費電力モデルの間には以下の関係が成り立つ．

$$HD(v_{n-1}, v_n) = HW(v_n) \qquad (v_{n-1} = 0 \text{ のとき}) \tag{3.40}$$
$$HD(v_{n-1}, v_n) = 1 - HW(v_n) \qquad (v_{n-1} = 1 \text{ のとき}) \tag{3.41}$$

$v_{n-1} = 0$ の時，2 つの消費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 つの消費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，CK_5 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．すべての $\rho^{(3)}_{j,i}$ から $L1^{(2)}$ を推定できれば，式 (3.34) により K_9 が計算できる．なお，この消費電力モデルの関係から推定できるものは直前の値だけではないことに注意されたい．上記の関係は v_{n+1} が定数の場合も成立する．そのため，式 (3.27) より $L1^{(4)}$ の値も IV_0 を固定することで定数となることから，$L1^{(2)}$ の推定同様，$L1^{(4)}$ の値が推定でき，その値から K_6 の値を直接推定することが可能である．内部鍵 K_9 が求められた場合，式 (3.16) より K_8，K_{10} が求められる．

その後で，K_3 を 2^{32} の全数探索により推定することで，式 (3.16) よりすべての内部鍵 K，初期鍵 IK が計算できる．

3.4　実験

本節では，FPGA と ASIC の 2 種類のプラットフォームにおいて，提案する解析手法の実証実験を行う．まず，SASEBO-GII 上の FPGA を用いた実験について述べ，その後に ASIC 実装に対する実験結果について述べる．なお，それぞれのプラットフォームにおいて，面積重視と速度重視の 2 種類の KCipher-2 プロセッサに対して提案手法の実証実験を行う．実装の詳細については，次章で説明する．

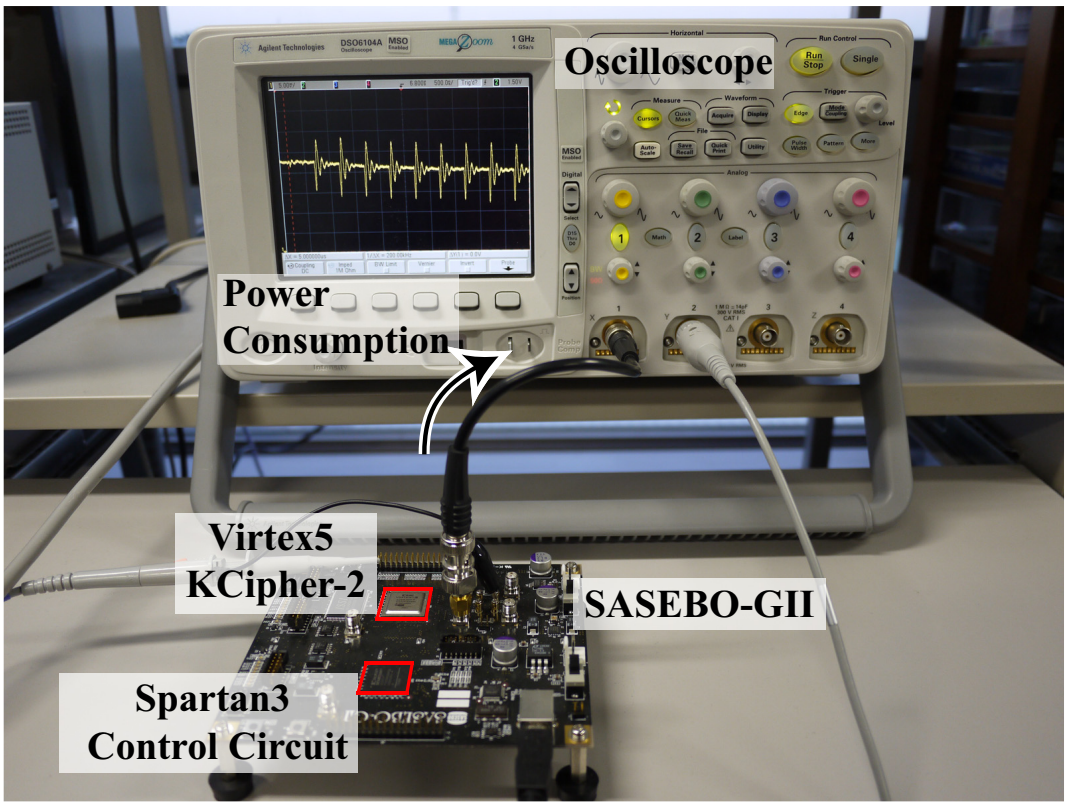


図 3.2　実験環境 (FPGA)

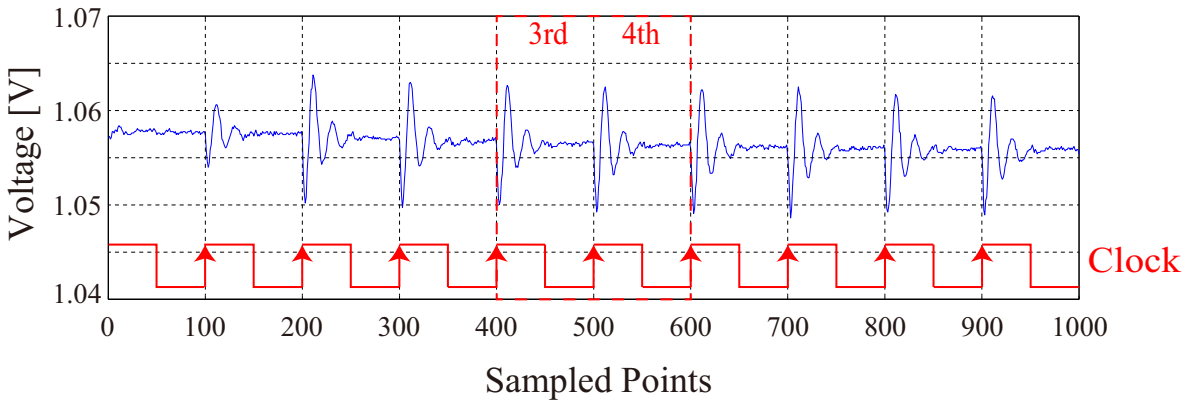


図 3.3　FPGA から測定した消費電力波形の例

3.4.1　FPGA 実装に対する実験

FPGA 実装を用いた実験環境を図 3.2 に示す．図に示す SASEBO-GII には，FPGA が 2 つ搭載されており，一方の FPGA1(Xilinx Virtex5) に KCipher-2 プロセッサを実

装し，もう一方の FPGA2(Xilinx Spartan3) に制御回路を実装した．Agilent 社のディジタルオシロスコープ (MSO6104A) を用いて，SASEBO-GII の電源ラインに設置された抵抗 (1 [Ω]) の消費電力 (電圧降下) を観測した．図 3.2 に示すように計測用 SMA ソケットにプローブを接続することで計測を行った．PC から FPGA2 経由で FPGA1 に IK と IV を入力するたびに鍵生成を行い，その時の消費電力を計測した．なお，本実験では KCipher-2 プロセッサの動作周波数を 2[MHz]，計測時のサンプリングレートを 200 [MSample/s] に設定した．消費電力の計測結果の例を図 3.3 に示す．図 3.3 中の青線は計測した消費電力の波形であり，KCipher-2 の初期化処理に対応する数のスパイクが確認できた．図には，初期化処理の開始直後の波形を示してあり，注目する 3 クロック目と 4 クロック目の処理にあたる部分を赤枠で囲んでいる．縦軸は電圧値，横軸は標本点を表す．以下に，相関値の絶対値を用いた CK_5，CK_6 の推定実験の結果を示し，その後，相関値の極性を用いた K_9 の推定実験の結果を示す．

まず，CK_5 の推定実験の結果を示す．前述した通り，推定する鍵候補を削減するために 1 バイト単位で上位から鍵の推定を行った．消費電力の波形はそれぞれの推定バイトで 100,000 枚使用した．3 クロック目 (400 ∼ 500 点間) における消費電力の計測値と推定鍵を用いて求めた推定電力値との相関結果の内，絶対値が最大となる値をその推定鍵における相関値とする．実験結果を図 3.4，図 3.5 に示す．これらの横軸は推定鍵の候補を表しており，縦軸は相関値を表す．この図で相関値が最大となる鍵候補が実験により得た CK_5 となる．全ての推定位置で相関値が最大となる鍵候補を調べた結果，推定鍵 $(\mathrm{dcc6a716})_{16}$ が得られた．この推定鍵は正しい CK_5 の値と一致した．また，鍵の推定が可能となる最小の波形数を調べる指標である Measurement To Disclosure (MTD) の解析結果を図 3.6，図 3.7 に示す．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，縦軸は対応する相関値であり，全ての鍵候補における相関値を重ねてプロットしてある．この図において，正しい鍵候補とその他の鍵候補とを区別できる交差点が MTD となる．図 3.6 より，面積重視の実装では，いずれの推定位置でも波形数 10,000 枚で正しい鍵の相関値が他と区別できることがわかった．また，下位の推定バイトでは 10,000 枚より少ない波形数で推定が可能になった．最上位バイトの推定 (図 3.6(a)) において正しい鍵以外の相関値が相対的に高い原因としては，IV の選択が 256 通りに限られることが考えられる．一方で，速度重視の実装では，いずれも推定位置でも波形数 7,000 枚程度で鍵の推定が可能であることが図 3.7 から確認できた．

次に，CK_6 の推定実験の結果について述べる．CK_5 の推定と同様に，バイト単位の推定を行った結果，相関値が最大となる鍵候補から得られた推定鍵と真値が一致することを確認した．また，各推定位置での MTD の解析結果を示した図 3.8 と図 3.9 から，いずれのアーキテクチャにおいても全ての鍵推定が成功することを実証できた．また，各推定位置において 10,000 枚程度の消費電力波形により鍵推定が可能であることがわかった．

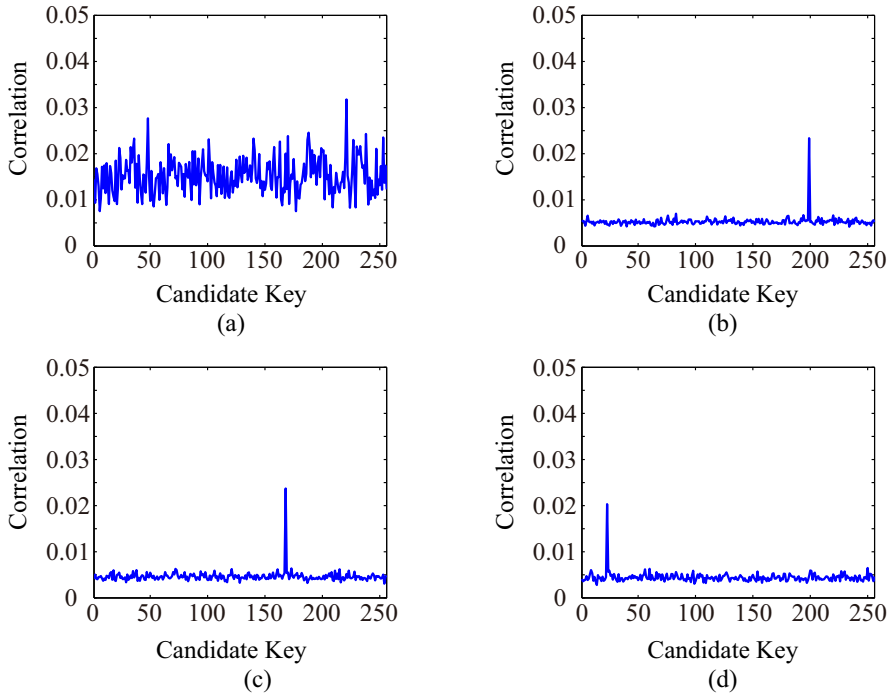


図 3.4　面積重視の FPGA 実装に対する CK_5 推定時の各鍵候補の相関値: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

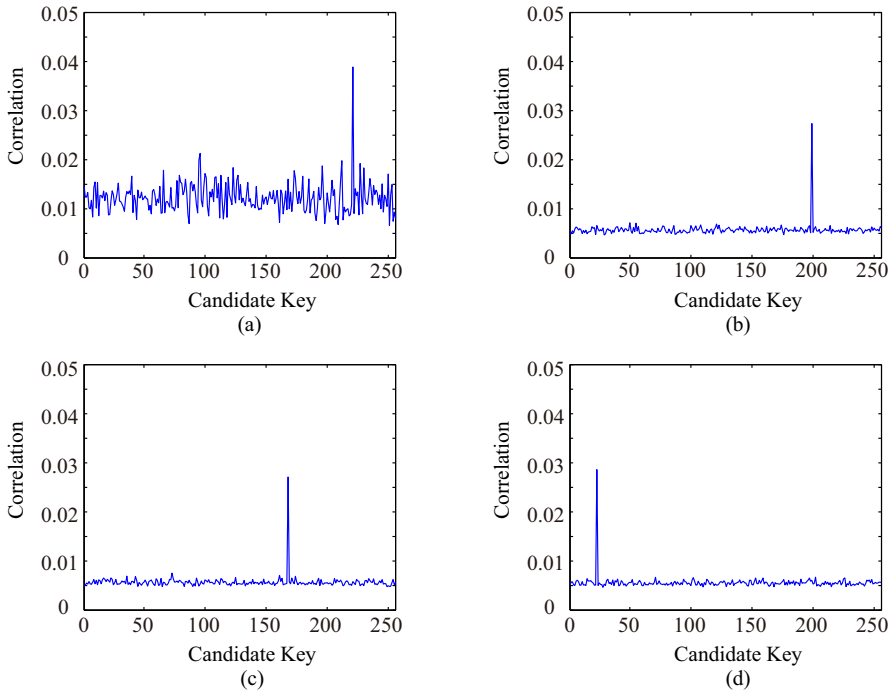


図 3.5　速度重視の FPGA 実装に対する CK_5 推定時の各鍵候補の相関値: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

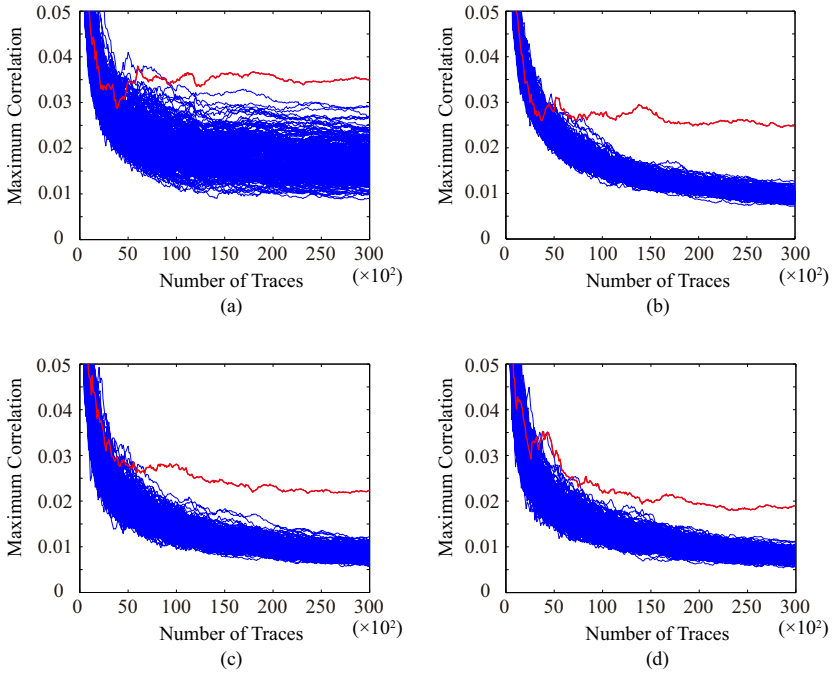


図 3.6　面積重視の FPGA 実装に対する CK_5 推定時の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

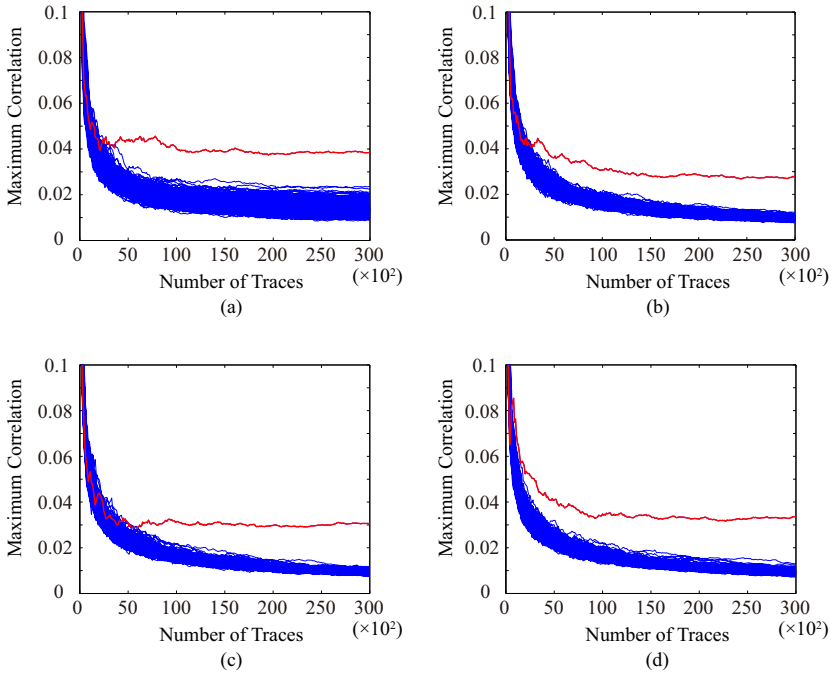


図 3.7　速度重視の FPGA 実装に対する CK_5 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

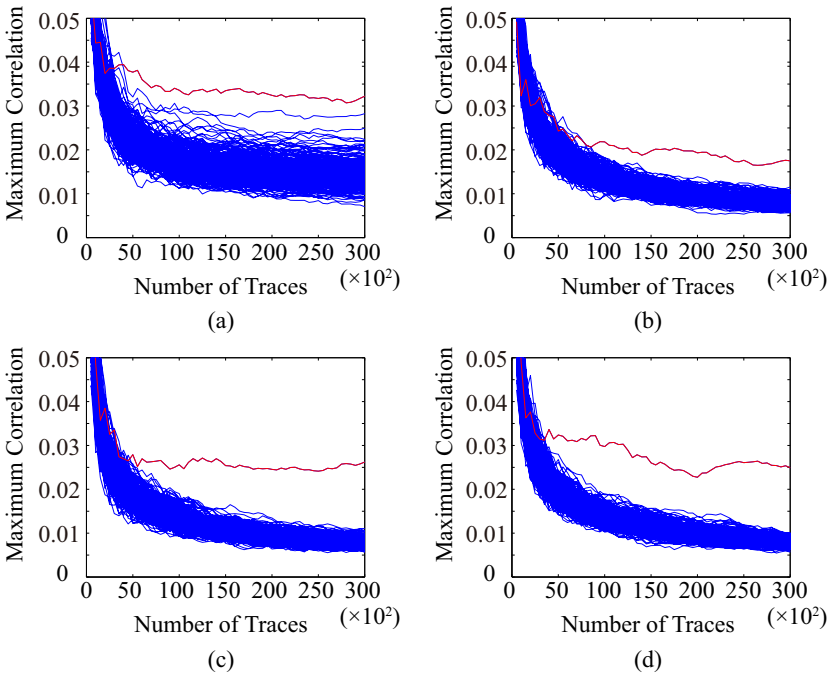


図 3.8　面積重視の FPGA 実装に対する CK_6 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

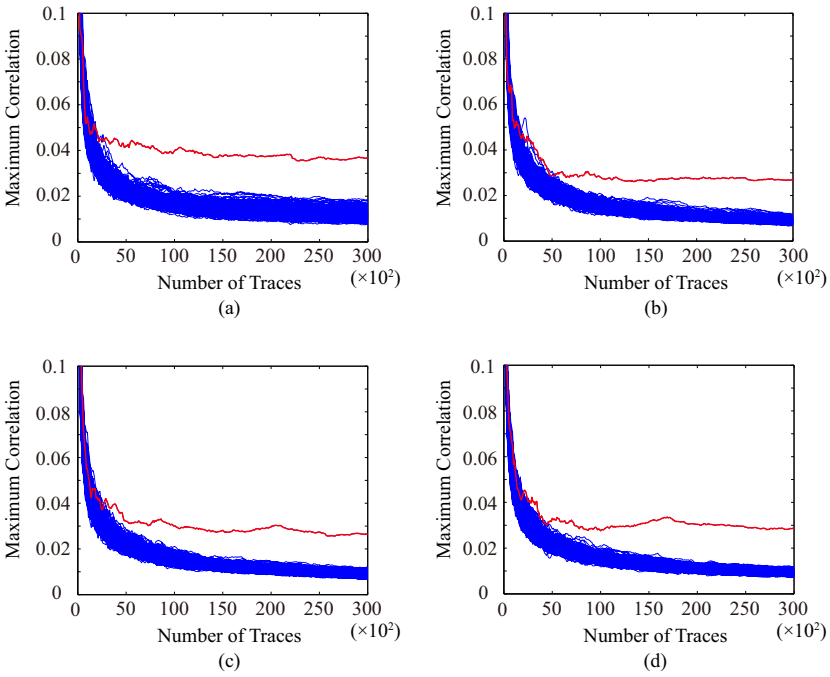
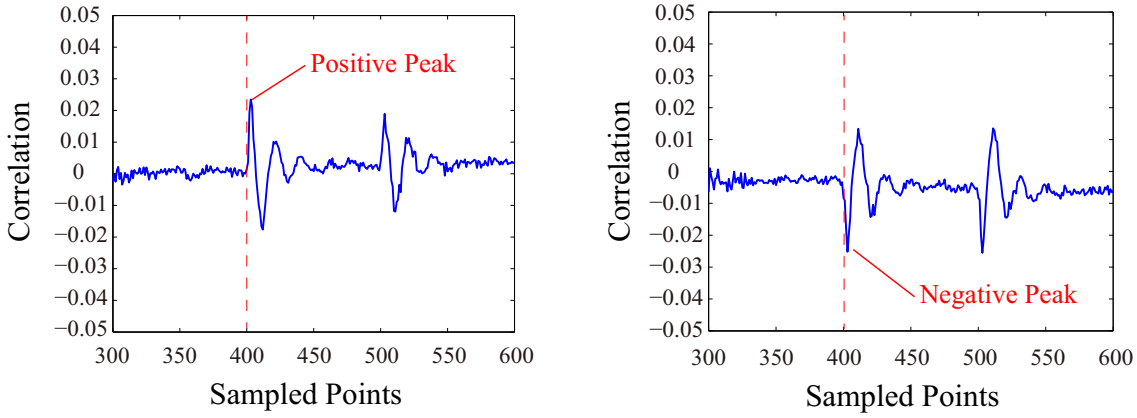


図 3.9　速度重視の FPGA 実装に対する CK_6 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット



(a)　(b)

図 3.10　相関値の極性の例: (a) 正の相関 (b) 負の相関

K_9 の推定の実験結果について述べる．この実験は CK_5 の値が全て既知であることを前提に行った．各推定位置において，CK_5 の真値を用いて推定電力値を計算し，消費電力の計測値との相関を求めた．なお，CK_5 の値が既知のため，IV_1 の選択をこの実験において行う必要はないことに注意されたい．また，$L1^{(2)}$ の値だけではなく，$L1^{(4)}$ の値も推定できることを示すために IV_0 を 0 としている．図 3.10 に極性が正の場合と負の場合の相関値をそれぞれ示す．ここで，横軸は標本点を表し，縦軸は相関係数の値を表す．図 3.3 中の 3 クロック目と 4 クロック目に相当する位置に存在するピークの極性を調べることで定数値を推定する．図 3.10(a) は 3 クロック目における相関値の極性が正の場合であり，この時，攻撃者は定数値を 0 と推定できる．これに対して図 3.10(b) は，極性が負の場合であり，攻撃者は定数値を 1 と推定できる．図 3.11 に，面積重視の実装に対する解析により得られた 1 バイト分の相関値を示す．図 3.11(a) は最上位ビットを推定電力値に使用したものであり，(h) は最下位ビットを推定電力値に使用したものである．この図において，3 クロック目における相関値の極性から 2 クロック目の値を $(00101010)_2$ と推定でき，これは真値と一致した．また，4 クロック目における相関値の極性からその時の値を $(01000011)_2$ と推定でき，これも真値と一致した．以上のように，全てのビットの相関値の極性から定数値を推定することができ，実際に推定した値と真値が一致することを確認した．速度重視の実装に対する解析結果を図 3.12 に示す．この図から面積重視の実装に対する解析結果と同様の結果が得られていることが確認できた．

以上の実験により，アーキテクチャの違いで解析に必要な波形数に差があるが，提案手法により K_5，K_6，K_9 を解析できることを実証した．この後で K_3 の全数探索を行うことで初期鍵の解析が完了する．したがって，2^{32} の計算量で初期鍵の解析が可能であることが示された．

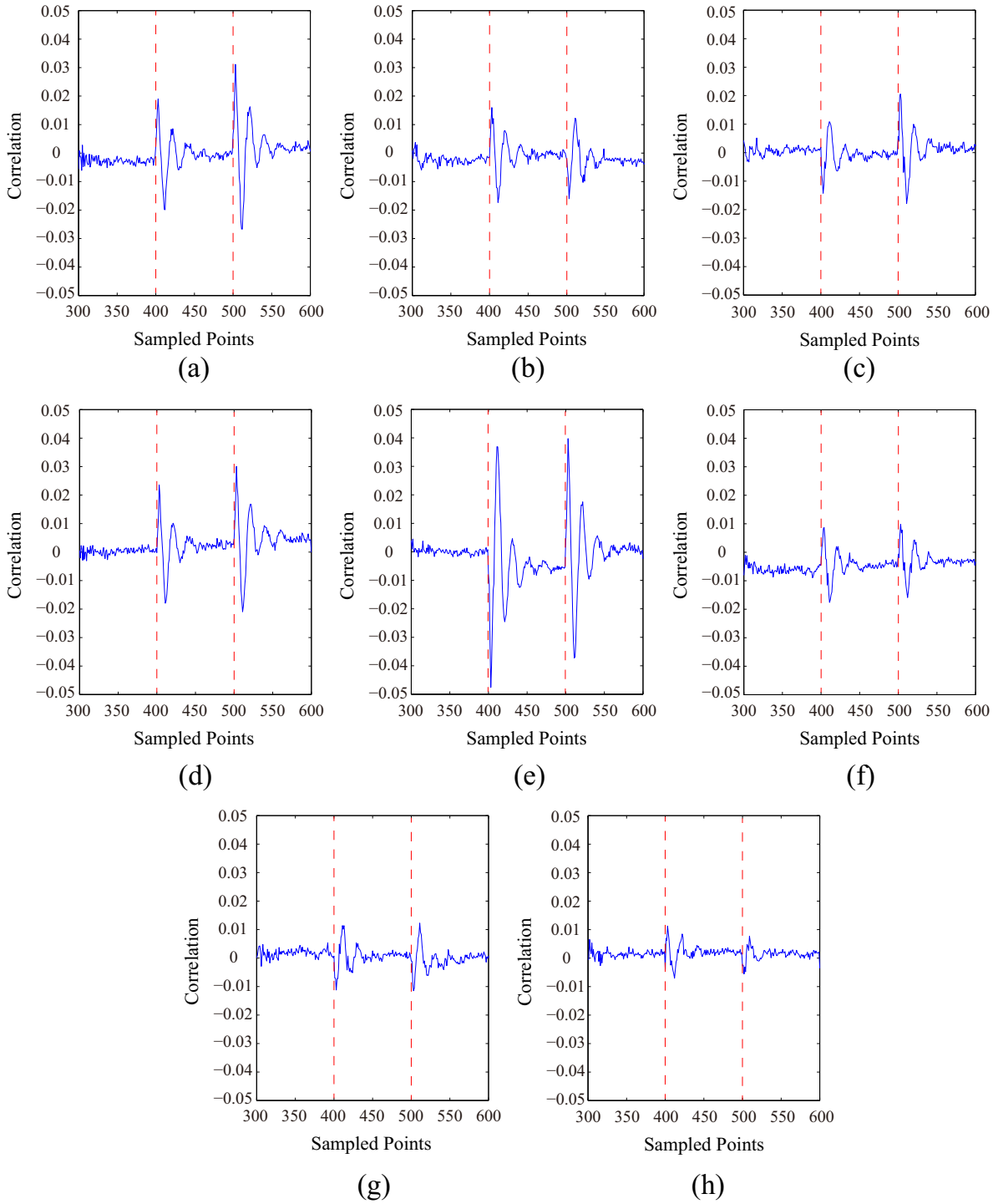


図 3.11　1 バイト分の相関値の極性 (面積重視): (a) 最上位ビット，(b) 7 ビット目，(c) 6 ビット目，(d) 5 ビット目，(e) 4 ビット目，(f) 3 ビット目，(g) 2 ビット目，(f) 最下位ビット

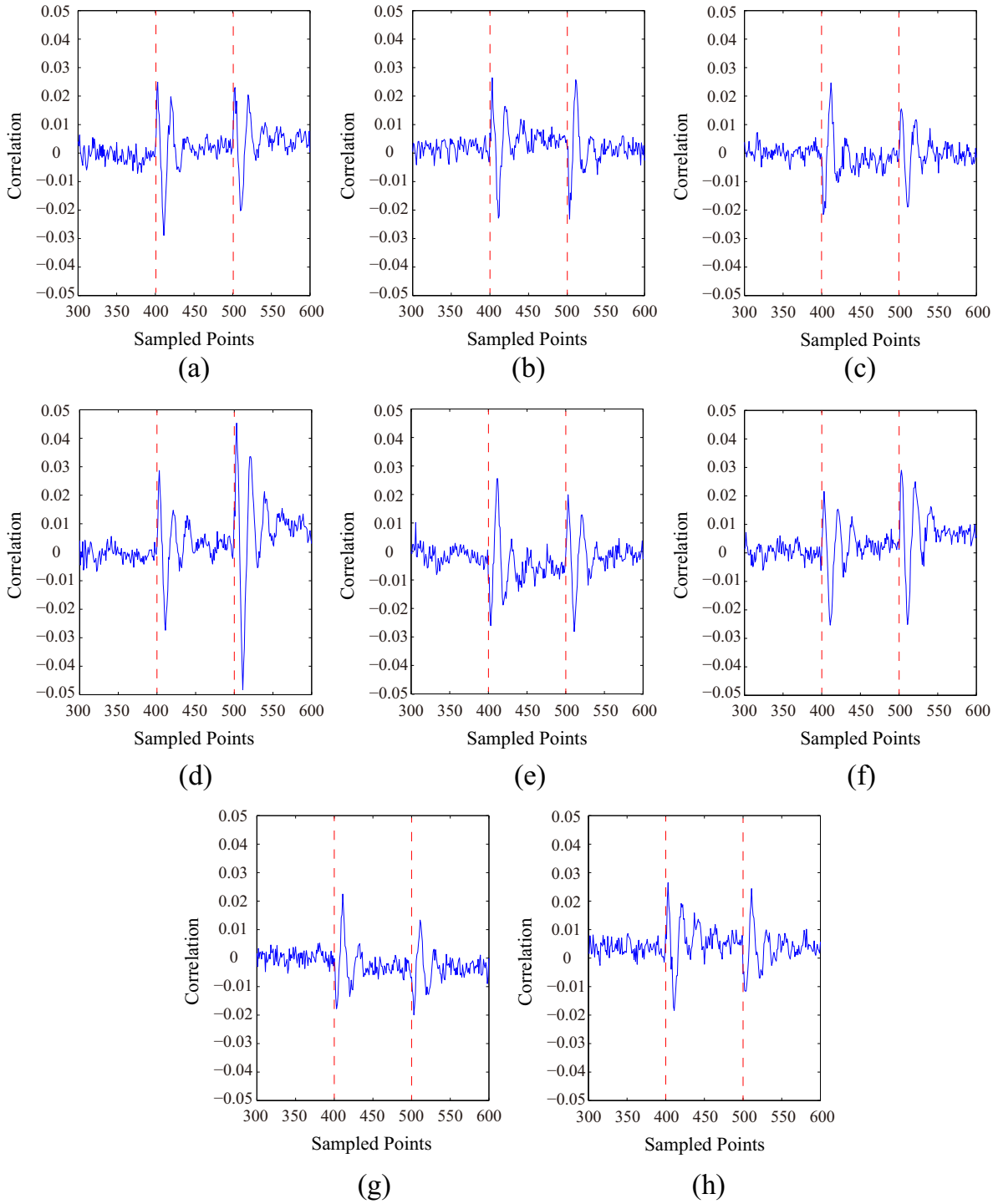


図 3.12　1 バイト分の相関値の極性 (速度重視): (a) 最上位ビット，(b) 7 ビット目，(c) 6 ビット目，(d) 5 ビット目，(e) 4 ビット目，(f) 3 ビット目，(g) 2 ビット目，(f) 最下位ビット

3.4.2　ASIC 実装に対する実験

ASIC 実装を用いた実験環境を図 3.13 に示す．SASEBO-W 上に試作プロセッサを搭載したドータボードを接続し，SASEBO-W 上の FPGA には制御回路を実装した．消費電力波形は，FPGA の場合と同様に，SASEBO の電源ラインに設置された抵抗間の電圧降下として計測した．図 3.13 に示すように計測用 SMA ソケットにプローブを接続することで計測を行った．なお，本実験では KCipher-2 プロセッサの動作周波数を 2 [MHz]，計測時のサンプリングレートを 200 [MSample/s] に設定した．消費電力の計測結果の例を図 3.14 に示す．図 3.14 中の青線は計測した消費電力の波形であり，FPGA の場合と同様に，KCipher-2 の初期化処理に対応する数のスパイクが確認できた．図には，初期化処理の開始直後の波形を示してあり，注目する 3 クロック目と 4 クロック目の処理にあたる部分を赤枠で囲んでいる．縦軸は電圧値，横軸は標本点を表す．

ASIC 実装に対する解析結果として，CK_5 の推定実験の結果のみを示す．解析には消費電力波形を 100,000 枚使用した．図 3.15 に面積重視の実装に対する解析結果を示す．最上位バイト (図 3.15(a)) 以外の推定位置で鍵の解析可能であることがわかる．鍵の解析に必要な波形数はいずれの位置でも 40,000 枚程度であり，この数は FPGA に比べて多かった．最上位バイトの鍵情報は，100,000 波形を用いても解析できなかったが，この情報を既知として実施した次の推定 (図 3.15(b)) が成功していることから，高々 16 ビットの鍵候補による CPA で推定できると考えられる．また，図 3.16 に速度重視の実装に対する解析結果を示す．図 3.16 に示すように，全ての推定位置において 10,000 波形以下での解析が可能であった．ただし，中間値が正確に求められる下位バイトの推定では，FPGA 実装の場合と比べて多くの波形が解析に必要であった．

以上の実験により，解析に必要な波形数は FPGA と ASIC の両者で異なるが，いずれのプラットフォームにおいても提案手法による鍵の解析が可能であることを実証した．

3.5　むすび

本章では，ストリーム暗号 KCipher-2 プロセッサに対するサイドチャネル解析について述べた．まず，KCipher-2 について，そのアルゴリズムと処理ステップを示した．次に，サイドチャネル解析の中でも強力な電力解析の代表的手法である相関電力解析を，KCipher-2 に適用する方法と秘密情報の解析手法について述べた．これにより，従来の解析に必要な計算量 2^{96} を 2^{32} に削減できることを示した．その後，SASEBO-GII の FPGA 上に実装した KCipher-2 プロセッサを用いた実験と ASIC 実装した KCipher-2 プロセッサを用いた実験を通して，提案する解析手法が有効であることを実証した．

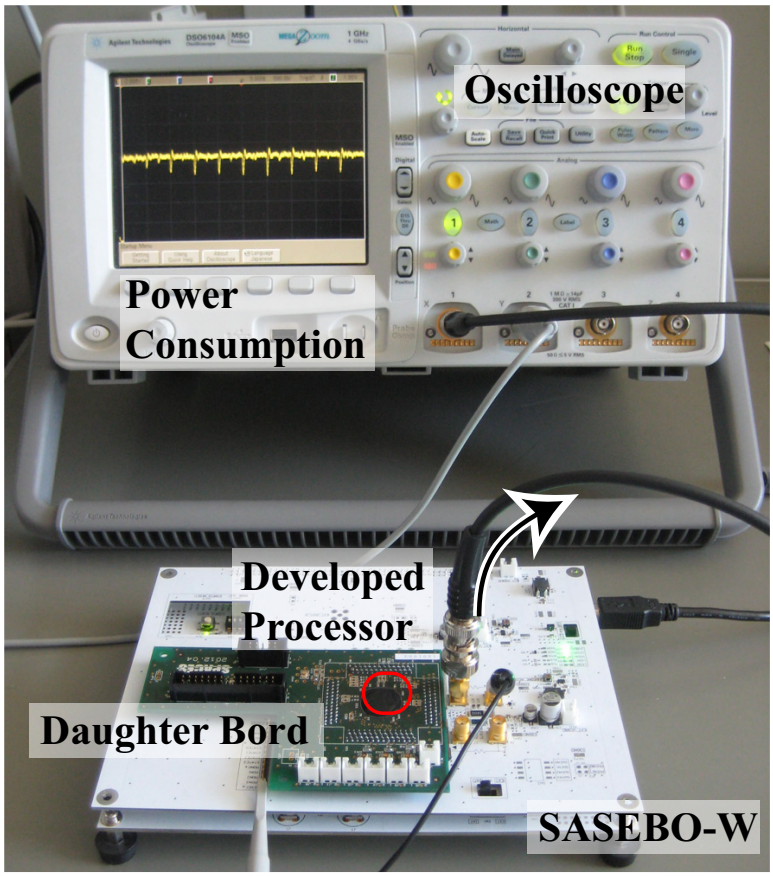


図 3.13　実験環境 (ASIC)

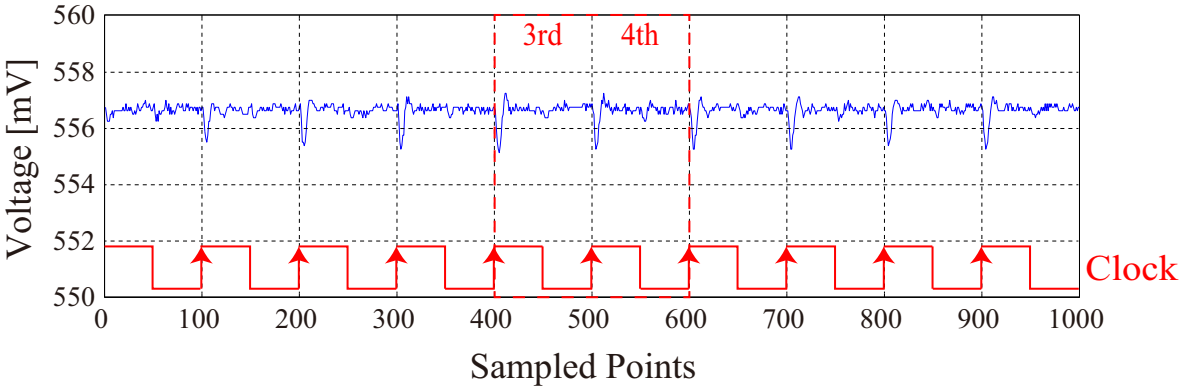


図 3.14　ASIC から測定した消費電力波形の例

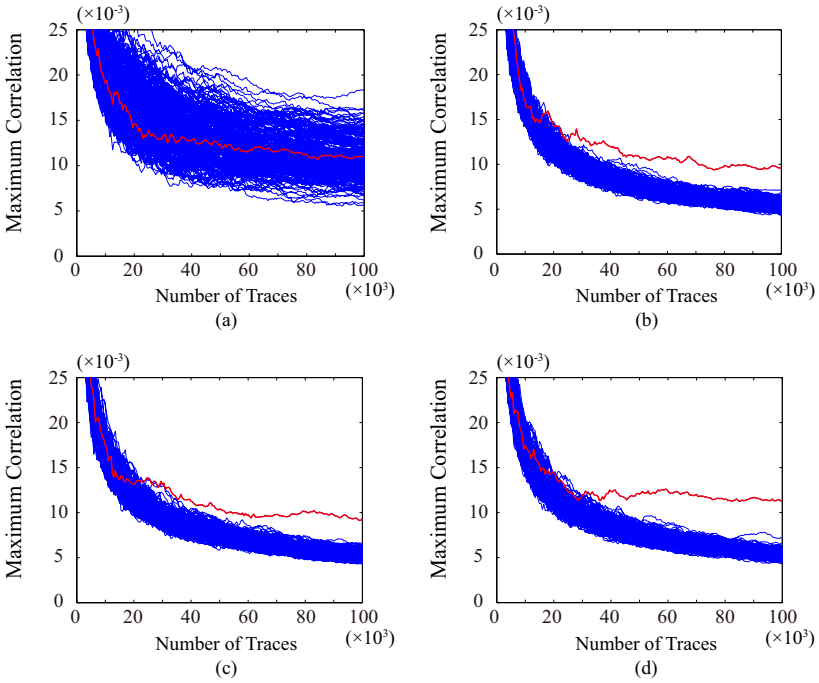


図 3.15　面積重視の ASIC 実装に対する CK_5 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

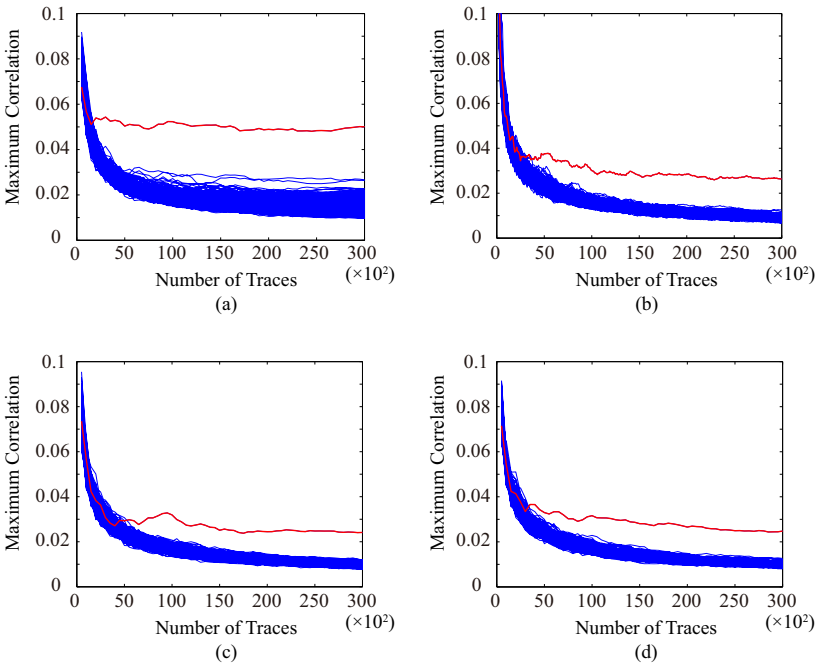


図 3.16　速度重視の ASIC 実装に対する CK_5 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

第 4 章

耐タンパー性を有する KCipher-2 プロセッサ

4.1 まえがき

本章では，耐タンパー性を有する KCipher-2 プロセッサの設計を行う．第 3.3 節で述べた解析に対する耐性を持った KCipher-2 プロセッサを面積重視と速度重視の 2 通り設計する．まず，設計した面積重視と速度重視のアーキテクチャについて説明する．次に，乱数マスキングによる対策の適用手法について述べる．その後で，FPGA 上に実装したプロセッサを用いた実験を通して提案する対策の有効性を実証する．最後に性能に対するオーバーヘッドの評価について述べる．

4.2 設計仕様

ここでは，KCipher-2 の設計仕様について述べる．面積重視と速度重視の 2 通りのアーキテクチャで設計するにあたり，KCipher-2 の非線形関数部を構成する整数加算器の算術アルゴリズムと S-box の実装方式について述べる．これら演算器は，KCipher-2 プロセッサのクリティカルパスに含まれており，これらの伝搬遅延が全体の処理速度に与える影響が大きいためである．速度重視の設計では，これら演算器のアーキテクチャの中でも最も伝搬遅延の小さいものを使用する．また，回路面積重視の設計では，これら演算器のアーキテクチャの中でも回路面積が最小となるものを使用する．したがって，整数加算器の算術アルゴリズムと S-box の実装方式として，面積重視の設計では Ripple Carry Adder(RCA) と合成体による実装 (Composite field arithmetic: Comp) を採用し，速度重視の設計では Kogge-Stone Adder(KSA)[23] と組み合わせ回路によるルックアップテーブルの実装 (TBL) を採用する．

これら 2 通りの設計に対して，第 3.3 節で述べた解析手法への対策を施す．以下に各演算器のアーキテクチャについて詳細を述べる．

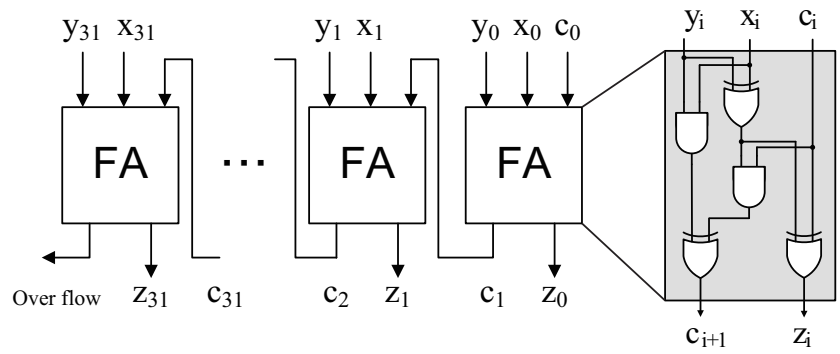


図 4.1　RCA のブロック図

Ripple Carry Adder (RCA)

図 4.1 に 32 ビットの RCA のブロック図を示す．ここで，加算器の入力値をそれぞれ $x = (x_{31}x_{30}...x_1x_0)_2$，$y = (y_{31}y_{30}...y_1y_0)_2$ とし，加算器間の桁上げは $c = (c_{30}c_{29}...c_1c_0)_2$，出力を $z = (z_{31}z_{30}...z_1z_0)_2$ とする．図からわかるように，この加算アルゴリズムは，以下の式で表される 1 ビットの加算器である全加算器 (Full Adder: FA) を必要なビット数だけ直列に接続しており，桁上げを順繰りにすべての FA を通して上位に伝搬していく．そのため，ゲート遅延の点ではビット数分だけ遅延が蓄積されるため，伝搬遅延が大きくなる．その反面，回路面積が最小になる．

$$z_i = x_i \oplus y_i \oplus c_i \tag{4.1}$$

$$c_{i+1} = x_i \wedge y_i \oplus c_i \wedge (x_i \oplus y_i) \tag{4.2}$$

ここで，$\wedge$ は AND 演算を表す．

Kogge-Stone Adder (KSA)

KSA は，桁上げを並列に計算する Prefix Adder の 1 つである．Prefix Adder では，下位ビットからの桁上げとは無関係に求められる Generate (G) と Propagate (P) という信号を用いて桁上げを並列に計算する加算アルゴリズムである．ここで，加算器の入力値と桁上げ，出力は上記 RCA の説明のときと同様のものを想定する．このとき，第 i 桁において，各信号は $g_i = x_i \wedge y_i$，$p_i = x_i \oplus y_i$ で表される．これは，g_i が 1 のときこの桁で桁上げが発生し，p_i が 1 のとき下位からの桁上げが上位に伝搬することを示す．これら信号を用いて式 (4.2) を以下のように表せる．

$$c_{i+1} = g_i \oplus c_i \wedge p_i \tag{4.3}$$

P 信号と G 信号の計算は各桁で独立であり，全桁で並列に行える．また，c_i が求まれば，z_i の計算も式 (4.2) により各桁で独立であり，全桁で並列に求められる．

第 i 桁から第 $i+1$ 桁の 2 桁からなるブロックの桁上げの信号を $(g_{i+1:i}, p_{i+1:i})$ とする

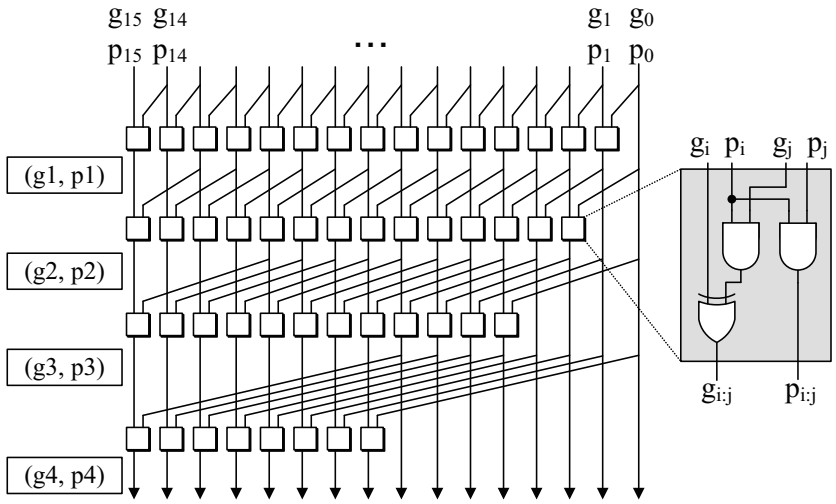


図 4.2　16 ビット KSA における桁上げ計算のブロック図

と，これらは

$$g_{i+1:i} = g_{i+1} \oplus (p_{i+1} \wedge g_i) \tag{4.4}$$
$$p_{i+1:i} = p_{i+1} \wedge p_i \tag{4.5}$$

という計算により求まる．この計算を

$$(g_{i+1:i}, p_{i+1:i}) = (g_{i+1}, p_{i+1}) \circ (g_i, p_i) \tag{4.6}$$

と定義する．$\circ$ を Prefix 演算とする．この演算では結合律が成立するが，交換律は成立しない．以降，第 j 桁から第 i 桁 $(i \geq j)$ までからなるブロックの桁上げの信号を $(g_{i:j}, p_{i:j})$ とする．$(g_{i:j}, p_{i:j})$ は，

$$(g_{i:j}, p_{i:j}) = (g_i \oplus (p_i \wedge g_{i-1:j}),\ p_i \wedge p_{i-1:j}) \tag{4.7}$$
$$= (g_i, p_i) \circ \cdots \circ (g_{j+1}, p_{j+1}) \circ (g_j, p_j) \tag{4.8}$$

により計算できる．したがって $(g_{i-1:j}, p_{i-1:j})$ が求まれば

$$c_i = g_{i-1:j} \oplus c_j \wedge p_{i-1:j} \tag{4.9}$$

により桁上げ c_i が計算できる．$\circ$ は結合律が成立するため，再帰的な計算を並列化可能である．演算を並列化するときの組み合わせや順序により，様々な構成が考えられる．

図 4.2 に 16 ビットの KSA における桁上げ計算のブロック図を示す．図のように KSA では，1 段目では，隣接する右側のビットから，2 段目では 2 ビット，3 段目では 4 ビット，4 段目では 8 ビット右側から入力を取る Prefix 演算を行っており，バイナリツリーで桁上げを生成している．最終段の Prefix 演算の出力を $(g4, p4)$ とした時，加算器の出力

z は以下の式で表せる．

$$\begin{aligned} z &= x \oplus y \oplus c \\ &= x \oplus y \oplus (g4 \ll 1) \oplus c_0 \wedge (p4 \ll 1) \\ &= x \oplus y \oplus (g4 \ll 1) \end{aligned} \tag{4.10}$$

ここで，$\ll 1$ は 1 ビット左シフトを表す．また，c_0 は 0 ビット目の加算における下位からの桁上げを表し，常に 0 を想定する．式 (4.10) より最終段の出力の G 信号のみで桁上げを表せることがわかる．KSA のアルゴリズムは，桁上げ伝搬段数が最小となるように桁上げ演算を行うアルゴリズムであり，他の Prefix Adder と比べて最も遅延が小さい反面，回路面積が大きい．

ルックアップテーブルによる実装 (TBL)

S-box は，入出力 8 ビットのルックアップテーブル (TBL) として実装可能である．入力 8 ビットに対する 256 通りの出力を定義通りに記述し，論理合成ツールを用いて組合せ回路を生成する．TBL 実装では，回路量と配線が多くなるため，回路面積の面では不利だが，遅延は小さい．

合成体による実装 (Comp)

S-box は数学的な構造を有しており，ガロア体 $GF(2^8)$ 上の逆元演算とアフィン変換を組み合わせで定義される．そのため，S-box を演算器として実装することが可能である．演算器による実装手法は，文献 [24]，[25]，[26] 等で多く研究されており詳細に分析されている．

本実装で利用する合成体による小型実装を図 4.3 に示す．この実装では，要素数が同じガロア体が同型であることを利用し，ガロア体 $GF(2^8)$ 上の逆元演算を，以下で定義される既約多項式を使って得られる合成体 $GF(((2^2)^2)^2)$ で行うものである [24]．これは，ガロア体 $GF(2)$ の 2 次拡大を繰り返したものであり，$GF(2^8)$ と同型な体上の逆元演算回路としては，現在知られている限り最小のものである．

$$\begin{array}{cc} GF(2^2) & : x^2 + x + 1 \\ GF((2^2)^2) & : x^2 + x + \phi \\ GF(((2^2)^2)^2) & : x^2 + x + \lambda \end{array} \tag{4.11}$$

ここで，$\phi = (10)_2$，$\lambda = (1100)_2$ である．

図 4.3 に示すように，逆元の計算は，もとの体 $GF(2^8)$ 上の元を同型写像 δ で合成体 $GF(((2^2)^2)^2)$ へマップし，$GF(((2^2)^2)^2)$ 上で乗法逆元を計算したのち，同型写像 δ^{-1} で $GF(2^8)$ へマップする，という 3 段階で行う．入力 $P \in GF(((2^2)^2)^2)$ の逆元計算は，部分体 $GF((2^2)^2)$ の演算を用いて次の式で行える．

$$P^{-1} = (P \cdot P^{16})^{-1} \cdot P^{16} \tag{4.12}$$

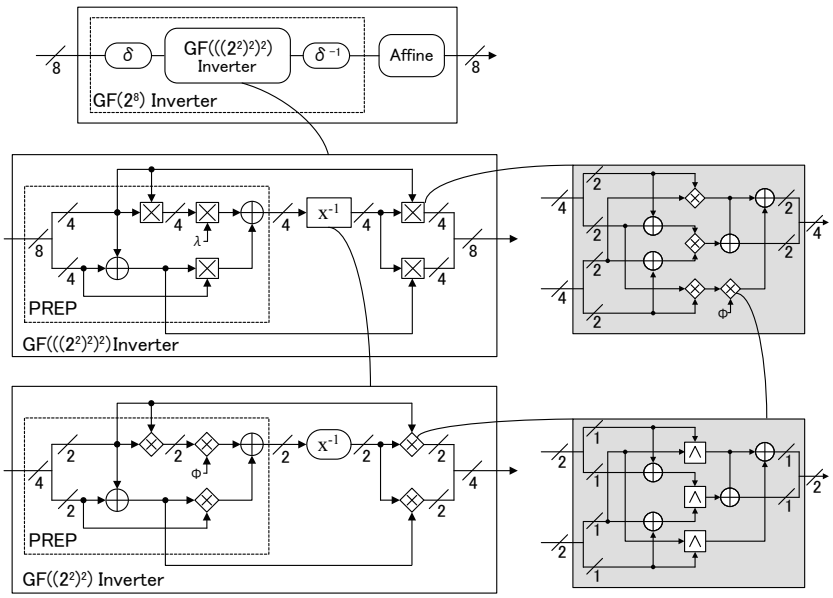


図 4.3　S-box の合成体による実装

ここで右辺の乗算や逆元演算は，$GF((2^2)^2)$ の演算であり，それらはさらに $GF(2^2)$ の演算を組み合わせて行える．逆元演算は上式に従って構成され，階層的な構造を持つ．また，図 4.3 の前処理部 (PREP) が $(P \cdot P^{16})$ の演算に相当する．一般に回路遅延が増加する一方で，回路面積を削減可能であり，小型実装に適するといえる．

4.3　乱数マスキングに基づく対策手法

第 3.3 節で述べた CPA は，初期化処理のタイミングにおける非線形関数部の内部レジスタ $L1$ を対象としている．そこで，乱数マスキングに基づく対策を $L1$ を含むデータパスにのみ適用する．$L1$ 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提案する CPA の対象となり得るため，本手法では，整数加算器の入力値 B_4，$R2$ 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．なお，マスク・アンマスク処理は排他的論理和 (XOR) により行う．

図 4.4 に乱数マスキングによる対策を施した KCipher-2 プロセッサを示す．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

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．

$$m_{L1} = \mathrm{Perm}(A_f(m_{add})) \tag{4.13}$$
$$m_{L2} = \mathrm{Perm}(A_f(m_{L1})) \tag{4.14}$$

ここで，Perm は Permutation，A_f はアフィン変換を表す．

図 4.4 に示すように，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，図中のアフィン変換，Permutation のように演算器を二重化し，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．

$$f(x \oplus m_x) = f(x) \oplus f(m_x) \tag{4.15}$$

一方で，上記の式が成立しない場合は，マスク値あるいはマスクされた値を用いた演算により出力の真値を分離できるように演算器を設計する必要がある．整数加算器と Sub 関数内部の S-box 中の逆元演算回路がこれにあたる．これらのマスキングの実現方法を以下に述べる．

4.3.1　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) [27] 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，以下の値を出力する．

$$C = A \wedge' B = (a \wedge b) \oplus m_c \tag{4.16}$$

ここで，$\wedge'$ を Masked AND 演算とする．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．整数加算器中の AND 演算を Masked AND 演算に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．本論文では，面積重視の整数加算器には RCA，速度重視の整数加算器には KSA をそれぞれ使用するため，両方の加算アルゴリズムに対して Golic による Masked AND 演算を適用する．

まず，32 ビットの RCA に適用する方法について述べる．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．$X_i = x_i \oplus m_{x,i}$，$Y_i = y_i \oplus m_{y,i}$ 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．

$$Z_i = X_i \oplus Y_i \oplus C_i \tag{4.17}$$
$$C_i = X_{i-1} \wedge Y_{i-1} \oplus C_{i-1} \wedge (X_{i-1} \oplus Y_{i-1}) \tag{4.18}$$

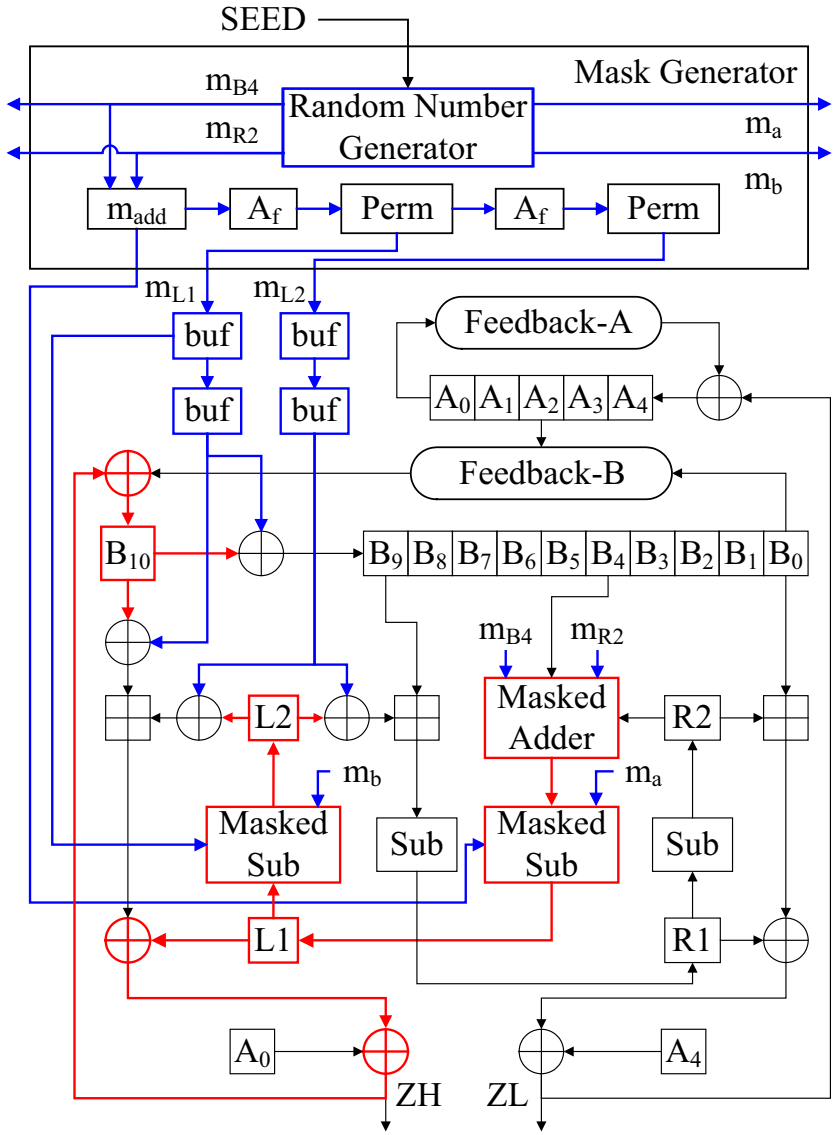


図 4.4　マスク処理を施した KCipher-2 回路

式 (4.17) 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 (4.18) 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．

$$\begin{aligned} C_i &= (x_{i-1} \wedge y_{i-1}) \oplus m_{x,i-1} \oplus (c_{i-1} \wedge (x_{i-1} \oplus y_{i-1})) \oplus (m_{x,i-1} \oplus m_{y,i-1}) \\ &= c_i \oplus m_{y,i-1}, \end{aligned} \tag{4.19}$$

式 (4.17)，(4.19) 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．

$$\begin{aligned} Z_i &= x_i \oplus y_i \oplus c_i \oplus m_{x,i} \oplus m_{y,i} \oplus m_{c,i} \\ &= z_i \oplus m_{x,i} \oplus m_{y,i} \oplus m_{y,i-1}. \end{aligned} \tag{4.20}$$

これを n ビットに拡張させた場合，アンマスクの値 m_z は以下の式で表せる．

$$m_z = m_x \oplus m_y \oplus (m_y \ll 1) \tag{4.21}$$

次に，32 ビットの KSA に適用する方法について述べる．演算器自体は，RCA へのマスキングと同様に AND 演算を，Masked AND 演算に置き換えることで実現可能である．ただし，出力をアンマスクする値が RCA の場合と異なる．以降では，32 ビットの KSA 内の AND 演算を Masked AND 演算に置き換えた演算器における出力のアンマスク値を求める方法を述べる．

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 KSA 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，G，P 信号 (G_i, P_i) はそれぞれ以下の式で与えられる．

$$G_i = X_i \wedge' Y_i = (x_i \wedge y_i) \oplus m_{y,i} \tag{4.22}$$

$$P_i = X_i \oplus Y_i = x_i \oplus y_i \oplus m_{x,i} \oplus m_{y,i} \tag{4.23}$$

また, 32 ビット KSA における各段の Prefix 演算出力を ($G1$, $P1$)，($G2$, $P2$)，($G3$, $P3$)，($G4$, $P4$)，($G5$, $P5$) とし，そのアンマスクの値を ($mG1$, $mP1$)，($mG2$, $mP2$)，($mG3$, $mP3$)，($mG4$, $mP4$)，($mG5$, $mP5$) とする．1 段目の Prefix 演算の出力 ($G1_i$, $P1_i$) は，隣接する右側のビットとの演算であり，以下の式で与えられる．

$$G1_i = G_i \oplus (G_{i-1} \wedge' P_i) = g_i \oplus (g_{i-1} \wedge p_i) \oplus (m_{y,i} \oplus m_{y,i-1}) \tag{4.24}$$

$$P1_i = P_i \wedge' P_{i-1} = (p_i \wedge p_{i-1}) \oplus m_{x,i} \oplus m_{y,i} \tag{4.25}$$

上式のように Masked AND 演算のアンマスク値を選択することで，G 信号のアンマスク値は G 信号のマスク値のみで求められる．P 信号についても同様である．これを n ビットに拡張させた場合，($mG1$, $mP1$) は

$$mG1 = m_y \oplus m_y \ll 1 \tag{4.26}$$

$$mP1 = m_x \oplus m_y \tag{4.27}$$

となる．同様に 2 段目から 5 段目までのアンマスクの値を求める．

$$mG2 = mG1 \oplus mG1 \ll 2 \tag{4.28}$$

$$mG3 = mG2 \oplus mG2 \ll 4 \tag{4.29}$$

$$mG4 = mG3 \oplus mG3 \ll 8 \tag{4.30}$$

$$mG5 = mG4 \oplus mG4 \ll 16 \tag{4.31}$$

$$mP5 = mP4 = mP3 = mP2 = mP1 \tag{4.32}$$

P 信号のアンマスク値は，式 (4.25) のように Masked AND 演算において P_i のアンマスク値を選択すると，常に $m_x \oplus m_y$ となる．

式 (4.10)，式 (4.26) ∼ (4.31) からマスクされた値を入力したときの出力 Z，そのアンマスク値 (m_z) について以下が成立する．

$$Z = X \oplus Y \oplus (G5 \lll 1) \tag{4.33}$$
$$= z \oplus m_x \oplus m_y \oplus (mG5 \lll 1) \tag{4.34}$$
$$= z \oplus m_z \tag{4.35}$$

4.3.2　逆元演算回路のマスキング

ガロア体 $GF(2^8)$ 上の逆元演算回路に対するマスキングには，合成体の実装に対する加法的なマスキング手法と乗法的なマスクを利用して逆元演算のマスキングを実現する手法があり，文献 [28]，[29]，[30] 等で多く研究されており詳細に分析されている．本論文では，面積重視の逆元演算には合成体 $GF(((2^2)^2)^2)$ の実装に対する加法的なマスキング手法 [29]，速度重視の逆元演算には乗法的なマスキング [30] を使用する．

合成体 $GF(((2^2)^2)^2)$ の実装に対する加法的なマスキング手法では，部分体 $GF(2^2)$ 上の演算において以下の式のように，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

$$(x \oplus m'')^{-1} = (x \oplus m'')^2 = x^2 \oplus m''^2 = x^{-1} \oplus m''^{-1} \tag{4.36}$$

図 4.5 にマスク処理を施した $GF(((2^2)^2)^2)$ 上の逆元演算回路を示す．この回路は，8 ビットのデータ a，8 ビットのマスク値 m，4 ビットのマスク値 y とした時，$A = a \oplus m$，m，y の 3 つの入力に対して，$a^{-1} \oplus m$ を出力する．マスクなしの逆元演算回路 (図 4.3) の前処理部 (PREP) を 2 つ使用し，さらに，部分体の逆元演算回路の前後に補正項計算を追加する．この回路はマスクなしの場合と同じく再帰的な回路構造を取る．そのため，マスク処理を施した部分体 $GF((2^2)^2)$ 上の逆元演算回路も図 4.5 と同様の構造を取る．

一方，乗法的なマスキングでは，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．したがって，ガロア体 $GF(2^8)$ 上の逆元をテーブルで実装することが可能となるため加法的なマスキング手法に比べて速度の面で有利である．ただし，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．逆元演算への入力を $A = a \oplus m$，XOR によるマスクを m，乗法マスクを y としたとき，マスク処理を施した逆元演算は図 4.6 の流れで行われる．

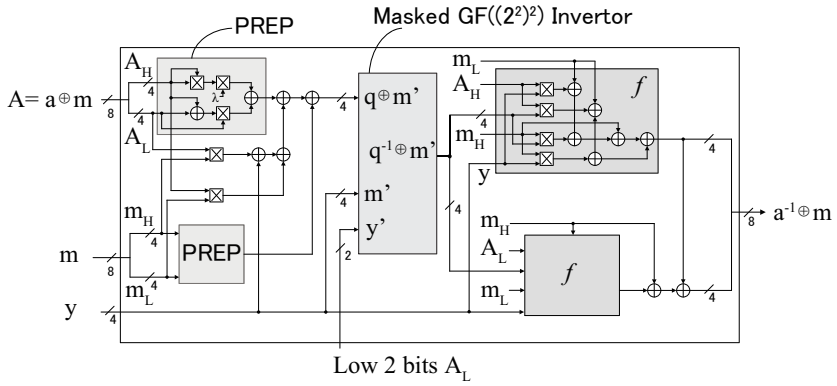


図 4.5　マスク処理を施した $GF(((2^2)^2)^2)$ 上の逆元演算回路

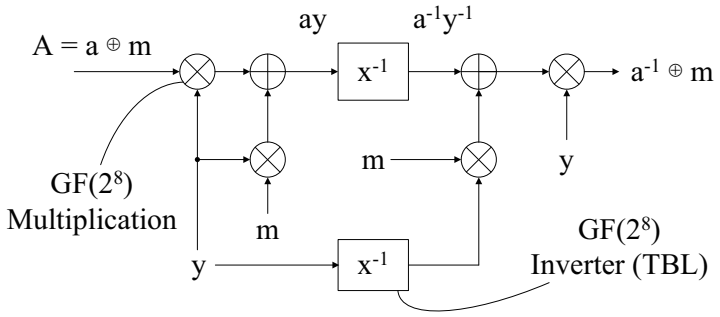


図 4.6　乗法的なマスキング手法による逆元演算回路

4.4　実験

本節では，乱数マスキングに基づく対策を施した KCipher-2 プロセッサに対して CPA 実験による耐性と対策による回路性能に対するオーバーヘッドを評価する．まず，設計した回路面積重視と速度重視のアーキテクチャの両方について対策を施し，第 3.3 節で提案した CPA に対して耐性を有することを実験により示す．その後で，対策による回路面積と遅延のオーバーヘッドを FPGA と ASIC の両方をターゲットデバイスとして評価する．最後に，設計した 4 つの KCipher-2 プロセッサを ASIC 実装し，実装コストの評価を行う．

4.4.1　耐性評価

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．実験環境，実験条件については，第 3.4 節と同様のものを使用する．面積重視と速度重視の KCipher-2 プ

ロセッサ両方を FPGA に実装し，CPA の耐性を実験により確認する．攻撃の対象は内部鍵 K_5 とする．図 4.7 に面積重視の対策回路に対する CPA 実験の結果を，図 4.8 に速度重視の対策回路に対する CPA 実験の結果を示す．100,000 枚の消費電力波形を用いても，第 3.3 節で提案した攻撃を防ぐことが確認できる．攻撃の性質上，最も鍵の推定が容易となる最下位バイトの推定であっても鍵の推定が困難となっていることが結果からわかる．以上より，提案する対策手法の有効性を確認することができた．

4.4.2　性能評価

設計した KCipher-2 プロセッサについて，FPGA(Virtex) と ASIC の 2 種類のターゲットデバイスに対して，対策を施した時のオーバーヘッドについての評価結果を示す．以下の 4 つのプロセッサの論理合成結果によりこれを評価する．

- 未対策，面積重視 (RCA, Comp)
- 未対策，速度重視 (KSA, TBL)
- 対策，面積重視 (RCA, Comp)
- 対策，速度重視 (KSA, TBL)

FPGA をターゲットデバイスにした場合の評価には，論理合成ツールに Xilinx 社の ISE13.1 を使用した．論理合成結果を表 4.1 に示す．FPGA での回路面積の評価には，Xilinx 社の FPGA において論理ブロックを構成する要素である論理セル (スライス) の使用数を用いている．遅延の評価には，クリティカルパスにおける動作遅延を用いる．表 4.1 からは，面積重視の設計の場合，対策を施すことで面積がおよそ 1.5 倍に増加し，速度重視の設計の場合，遅延がおよそ 1.9 倍に増加することがわかった．

ASIC をターゲットとした評価では，論理合成ツールには Synopsys 社の Design Compiler を使用し，セルライブラリには TSMC 65nm LP Standard Cell Libraries を使用した．表 4.2 に論理合成結果を示す．ASIC での回路面積の評価には，配線の面積を無視したセルの面積を用いている．遅延の評価には，FPGA 同様クリティカルパスにおける動作遅延を用いる．表 4.2 から，面積重視の設計の場合，対策を施すことで面積がおよそ 1.6 倍に増加することがわかった．また，速度重視の設計の場合，対策を施すことで遅延は 2.6 倍に増加することがわかった．以上の結果から，FPGA や ASIC といったプラットフォームの違いでオーバーヘッドに差が見られるが，面積重視の設計では，1.6 倍の面積増加で対策が実装可能であり，速度重視の設計では，2.6 倍の遅延増加で対策が実装可能であることがわかった．

最後に，試作した KCipher-2 プロセッサの実装コストの評価について述べる．TSMC 65nm LP Standard Cell Libraries をも用いて合成し，遅延の目標値は 24 [MHz] とし

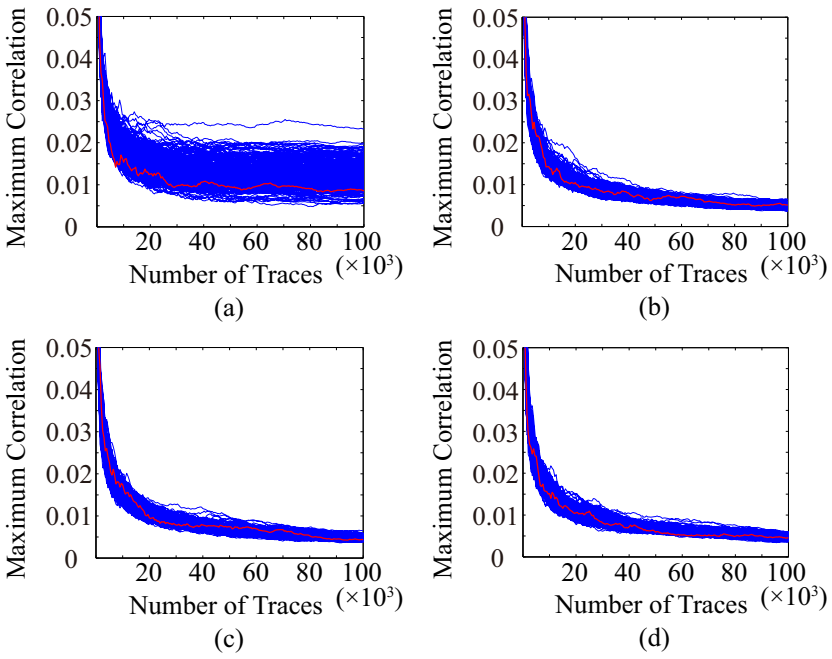


図 4.7　面積重視の対策設計に対する CK_5 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

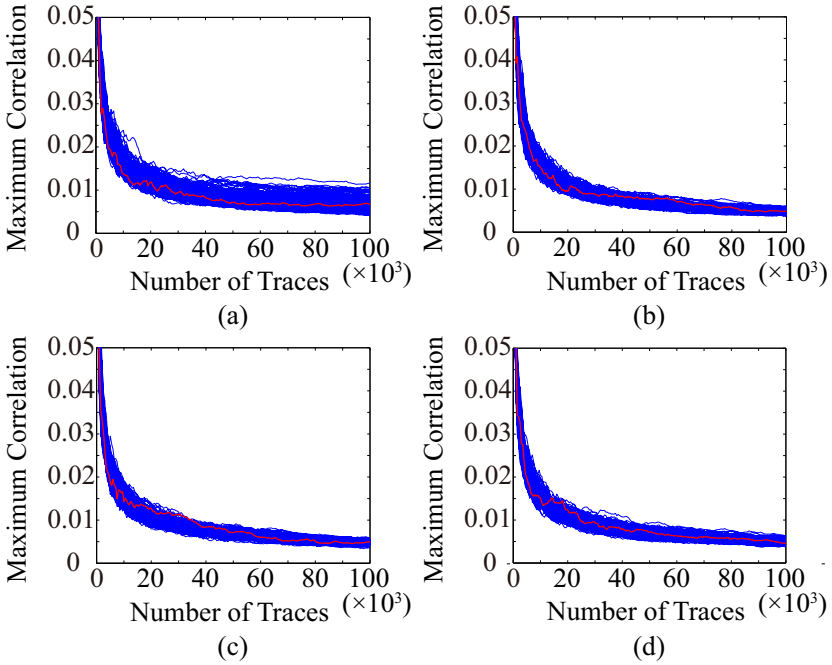


図 4.8　速度重視の対策設計に対する CK_5 推定時の各鍵候補の MTD の解析結果: (a) 31∼24 ビット (b) 23∼16 ビット (c) 15∼8 ビット (d) 7∼0 ビット

表 4.1　論理合成結果: FPGA(Virtex5)

	Without Countermeasure		With Countermeasure	
Architecture	Area	Speed	Area	Speed
Area [Slice]	2721	5232	4001	8817
Delay [ns]	14.77	7.56	23.56	13.97

表 4.2　論理合成結果: ASIC

	Without Countermeasure		With Countermeasure	
Architecture	Area	Speed	Area	Speed
Area [μm^2]	30131	56611	47930	77621
Delay [ns]	6.50	2.27	13.44	5.99

表 4.3　試作した KCipher-2 プロセッサの実装コスト

	Without Countermeasure		With Countermeasure	
Architecture	Area	Speed	Area	Speed
Area [μm^2]	49818.24	66254.76	80145.36	106666.92

た．与えた遅延制約が十分に遅いため，回路面積についてのみ評価を行った．表 4.3 に結果を示す．対策による回路面積のオーバーヘッドは，面積重視と速度重視の両方でおよそ 1.6 倍であることがわかった．ただし，速度を重視して遅延制約をきつくした場合，回路面積はさらに増加すると考えられる．

図 4.9，4.10 に試作したプロセッサのレイアウト図を示す．各図において (a) に未対策版のレイアウト図を，(b) に対策版のレイアウト図を示している．図中の C_L_Area(C_L_Speed) は，内部レジスタ $L1$，$L2$ を含む非線形関数部のモジュールであり，C_R_Area(C_R_Speed) は，内部レジスタ $R1$，$R2$ を含む非線形関数部のモジュール，FSR は，FSR-A と FSR-B のモジュールである．また，対策版における C_L_Area_mask(C_L_Speed_mask) は，対策したデータパスを表すモジュールであり，MG_Area(MG_Speed) は，乱数マスクの生成，アンマスクの生成を行うモジュールである．本対策手法は，対策を $L1$，$L2$ を含むモジュールのみに施すものであり，その様子がレイアウト図からも確認できる．対策したモジュールの面積は回路全体の半分を占めているが，回路全体に施す必要のある他の対策手法に比べて，相対的にオーバーヘッドは小さいと言える．

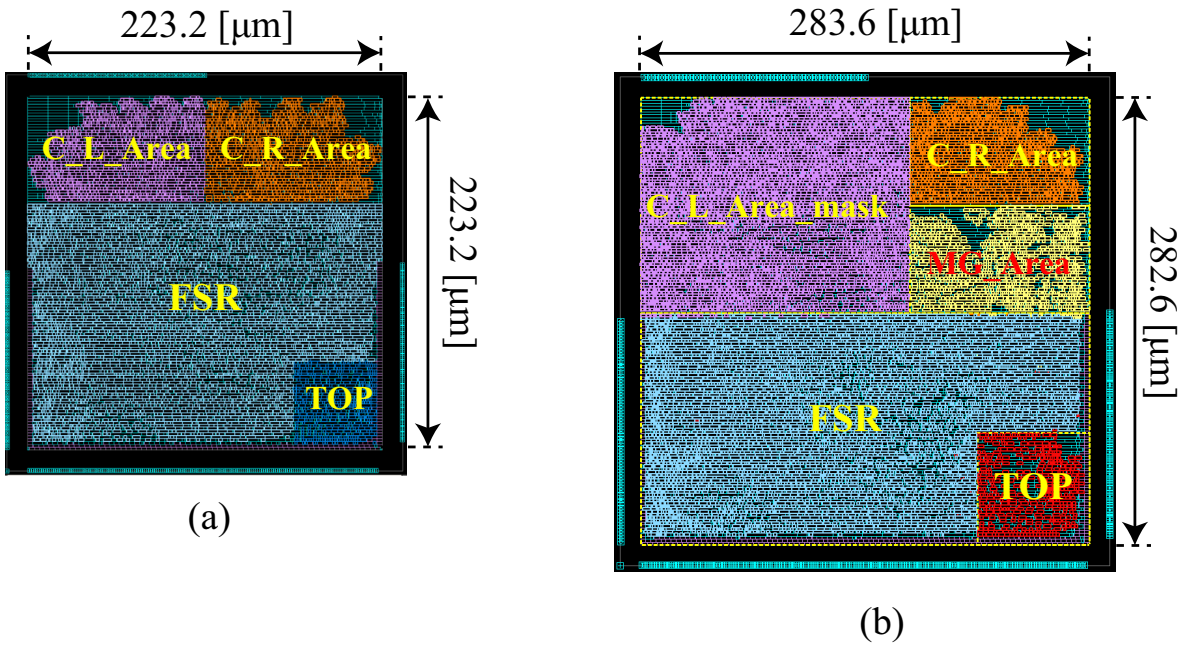


図 4.9　面積重視の KCipher-2 プロセッサ: (a) 未対策版 (b) 対策版

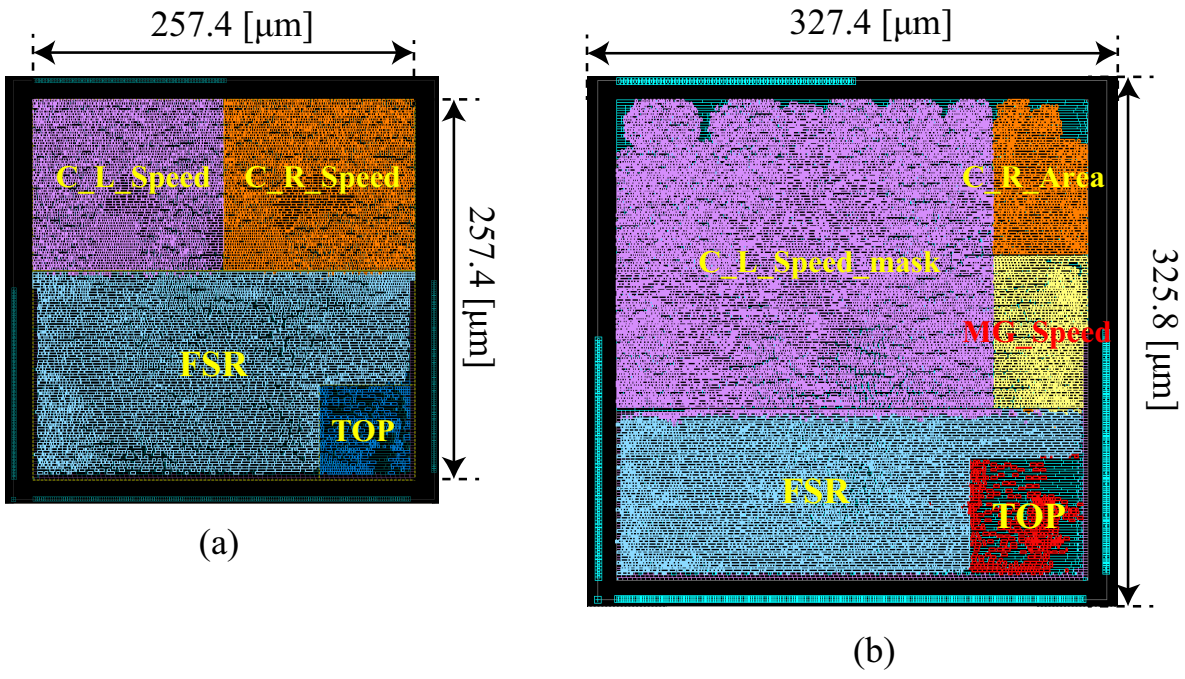


図 4.10　速度重視の KCipher-2 プロセッサ: (a) 未対策版 (b) 対策版

4.5　むすび

本章では，耐タンパー性を有する KCipher-2 プロセッサの設計について述べた．まず，設計した面積重視と速度重視のアーキテクチャについて述べた．次に，各アーキテクチャに対して乱数マスキングに基づく対策を適用する方法について述べた．その後，SASEBO-GII の FPGA 上に実装したプロセッサを用いた実験を通して，提案する対策の有効性を示した．また，FPGA と ASIC を想定した論理合成により提案手法のオーバーヘッドを評価した結果，面積重視の設計では 1.6 倍の面積増加で，遅延重視の設計では 2.6 倍の遅延増加で対策を実装できることを示した．さらに，設計した 4 つの KCipher-2 プロセッサを ASIC 実装し，実装コストを評価した．その結果，対策の実装にかかる回路面積のオーバーヘッドは 1.6 倍程度であることを確認した．

第 5 章

結言

以上，第 2 章から第 4 章まで，ストリーム暗号 KCipher-2 プロセッサに対するサイドチャネル解析の脅威の把握とその対策回路の設計について述べた．

第 2 章では，ストリーム暗号とその実装に関する基礎的考察を行った．まず，共通鍵暗号方式に分類されるストリーム暗号の構造，ストリーム暗号用途の疑似乱数生成器に求められる性質について述べた上で，ストリーム暗号の標準化動向について述べた．また，暗号のアルゴリズム自体の安全性だけではなく実装の安全性が重要となることを述べ，サイドチャネル解析について概説した．サイドチャネル解析の中でも強力な手法として知られる電力解析について，代表的な解析手法と対策手法について述べた．

第 3 章では，KCipher-2 プロセッサのサイドチャネル解析について述べた．まず，KCipher-2 アルゴリズムと処理ステップについて述べた．次に, 相関電力解析の KCipher-2 への適用方法と初期鍵の解析手法を提案した．提案手法では，暗号プロセッサへの入力を適切に選択し，測定した消費電力と求めた推定電力値との相関値の絶対値と極性の両方を秘密鍵の推定に利用する．これにより最大でも 2^{32} の探索により秘密鍵の推定が可能となることを示した．最後に，FPGA と ASIC の両方のプラットフォーム上に実装した KCipher-2 プロセッサを用いた実験を通して，提案する解析手法が有効であることを示した．

第 4 章では，耐タンパー性を有する KCipher-2 プロセッサについて述べた．まず，KCipher-2 プロセッサの設計仕様として面積重視と速度重視の 2 通りのアーキテクチャについて述べた．次に，乱数マスキングに基づく対策の適用手法について述べた．その後で，提案する対策を施したプロセッサに対して第 3 章で提案した解析実験を行い，対策の有効性を示した．また，各アーキテクチャの対策回路は，面積重視の場合は 1.6 倍，速度重視の場合は 2.6 倍のオーバーヘッドで実装できることを論理合成結果により示した．さらに，設計した 4 つのプロセッサを ASIC 実装し，実装コストを評価した．その結果，対策の実装にかかる回路面積のオーバーヘッドが 1.6 倍程度であることを確認した．

今後の展望としては, KCipher-2 プロセッサへの他の解析手法, 例えば, フォールトベース攻撃などの脅威の把握が挙げられる．また，提案した耐タンパー性を有する KCipher-2 プロセッサの高性能化も検討している．

参考文献

[1] "The eSTREAM Project." http://www.ecrypt.eu.org/stream/.

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

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

[4] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the secrets of smart cards*. Springer, 2007.

[5] W. Fischer, B. Gammel, O. Kniffler, and J. Velten, "Differential power analysis of stream ciphers," *CT-RSA, Lecture Notes in Computer Science*, Vol. 4377, pp. 257–270, 2007.

[6] J. Takahashi, T. Fukunaga, and K. Sakiyama, "Differential fault analysis on stream cipher mugi," *Proc. IEICE Trans. Fundamentals Electron. Commun. Comput. Sci*, Vol. E95-A, No. 1, pp. 242–251, 2012.

[7] 三上修吾 渡辺大, "ストリーム暗号 Enocoro-128 v2 に対する相関電力解析," 電子情報通信学会研究報告, Vol. 111, No. 337, pp. 1–6, 2011.

[8] 三上修吾, 吉田博隆, 渡辺大, 崎山一男, "Threshold implementation を利用したストリーム暗号 Enocoro-128 v2 の 相関電力解析対策," 暗号と情報セキュリティシンポジウム, No. 2C2-1, 2012.

[9] M. Henricksen, W. Yap, C. Yian, S. Kiyomoto, and T. Tanaka, "Side-channel analysis of the K2 stream cipher," *Information Security and Privacy, Lecture Notes in Computer Science*, Vol. 6168, pp. 53–73, 2010.

[10] "Side-channel Attack Standard Evaluation BOard." http://www.morita-tech.co.jp/SASEBO/en/board/sasebo-g2.html.

[11] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," *CHES2004, Lecture Notes in Computer Science*, Vol. 3156, pp. 16–29, 2004.

[12] E.R. Berlekamp, *Algebraic coding theory*. McGraw-Hill, 1968.

[13] N.T. Courtois and W. Meier, "Algebraic attacks on stream ciphers with linear feedback," *Proc. EUROCRYPT, Lecture Notes in Computer Science*, Vol. 2656, pp. 345–359, 2003.

[14] P.C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," *Proc. CRYPTO '99, Lecture Notes in Computer Science*, Vol. 1666, pp. 388–397, 1999.

[15] P.C. Kocher, R. Lee, G. McGraw, and A. Raghunathan, "Security as a new dimension in embedded system design," *Proc. DAC*, pp. 753–760, 2007.

[16] R.L. Rivest, A. Shamir, and L. Adleman, "A method for obtainning digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, No. 2, pp. 120–126, 1978.

[17] K. Tiri, M. Akmal, and I. Verbauwhede, "A dynamic and differential CMOS logic with signal independent power consumption to withstand differential power analysis on smart cards," *Proc. ESSCIRC*, pp. 403–406, 2002.

[18] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," *Proc. DATE*, pp. 246–251, 2004.

[19] E. Trichina, "Combinational logic design for aes subbyte transformation on masked data," *Cryptology ePrint Archive*, Vol. 2003/236, 2003.

[20] S. Nikova, C. Rechberger, and V. Rijmen, "Threshold implementations against side-channel attacks and glitches," *Proc. ICICS, Lecture Notes in Computer Science*, Vol. 4307, pp. 529–545, 2006.

[21] T. Pop and S. Mangard, "Masked dual-rail pre-charge logic: DPA-resistance without routing constrain," *CHES2005, Lecture Notes in Computer Science*, Vol. 3659, pp. 172–186, 2005.

[22] D. Suzuki, M. Saeki, and T. Ichikawa, "Random switching logic: A new countermeasure against DPA and second-order DPA at the logic level," *Proc. IEICE Tras. Fundamental Electron. Commun. Comput. Sci*, Vol. E90-A, No. 1, pp. 160–168, 2007.

[23] P.M. Kogge and H.S. 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.

[24] A. Satoh and S. Morioka, "Hardware-focused performance comparison for the standard block ciphers AES, Camellia, and Triple-DES," *Information Security, Lecture Notes in Computer Science*, Vol. 2851, pp. 252–266, 2003.

[25] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," *CHES2002, Lecture Notes in Computer Science*, Vol. 2523, pp. 172–

186, 2003.

[26] A. Rudra, P.K. Dubey, C.S. Julta, V. Kumar, J.R. Rao, and P. Rohatgi, "Efficient rijndael encryption implementation with composite field arithmetic," *CHES2001, Lecture Notes in Computer Science*, Vol. 2162, pp. 171–184, 2001.

[27] J.D. Golic, "Techniques for random masking in hardware," *IEEE Trans. Circuits and Systems*, Vol. 54, No. 2, pp. 291–300, 2007.

[28] E. Oswald, S. Mangard, N Pramstaller, and V Rijmen, "A side-channel analysis resistant description of the AES S-Box," *FSE2005, Lecture Notes in Computer Science*, Vol. 3557, pp. 413–423, 2005.

[29] 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.

[30] M.L. Akkar and C. Giraud, "An implementation of DES and AES, secure against some attacks," *CHES2001, Lecture Notes in Computer Science*, Vol. 2162, pp. 309–318, 2001.

謝辞

本論文は，著者が東北大学 大学院情報科学研究科 情報基礎科学専攻 計算機構論分野 (青木 (孝)・本間 (尚) 研究室) において行った研究を取りまとめたものです.

本研究を推し進めるにあたり，恩師青木孝文教授には，筆者が学部 4 年次の分野配属以来，終始熱心な御指導と御鞭撻を頂きました．先生の研究・教育に対する真摯な御姿勢から多くを学んだことを銘記し，ここに改めて感謝の意を表します.

本論文をまとめるにあたり，本論文の審査員として御専門の立場から有意義な御意見・御批判を頂いた亀山充隆教授ならびに静谷啓樹教授に感謝いたします.

本間尚文准教授には，研究に対する懇切なる御指導と終始変わらぬ励ましを頂くとともに，本論文の執筆においても様々な御助言を賜りました．ここに改めて感謝の意を表します.

株式会社 KDDI 研究所の清本晋作氏，福島和英氏，および，仲野有登氏には，在学中を通して共同研究をさせていただき，有益な御助言と御支援を頂きました．ここに深く感謝いたします.

菅原健氏 (現 三菱電機株式会社)，齋藤和也氏 (現 富士通株式会社)，遠藤翔氏 (青木 (孝)・本間 (尚) 研究室) には，本研究に関する様々なご意見やご協力を頂きました．ここに改めて感謝の意を表します.

最後に，日頃の研究室生活において様々な面で御協力頂いた伊藤康一助教をはじめとする研究室諸氏に心より御礼申し上げます.

2013 年 2 月 8 日