博士論文

故障注入を用いた暗号モジュールの動作解析に関する研究

情報基礎科学専攻

遠藤 翔

Analysis of Cryptographic Modules Using Fault Injections

by

Sho ENDO

Graduate School of Information Sciences, Tohoku University, 2015

目次

第 1 章

緒言

現在，暗号技術が情報化社会の基盤技術として用いられている．ネットワークシステムにおける利用者の認証や，サービスの正真性の保証，通信内容の秘匿などの機能は，暗号技術により実現される．具体的な応用例としては，電子マネーやクレジットカード，携帯電話の SIM カード等のスマートカード，電子商取引などに用いられる SSL (Secure Socket Layer) や，PC システムの正真性を保証する TPM (Trusted Platform Module) などが挙げられる．さらに，暗号技術は将来実現されると見込まれている Internet of Things (IoT) 社会において大きな役割を果たすことが予想される．IoT はあらゆる機器がインターネットに接続され相互作用するという概念であり [1]，膨大な数の小型のノードと，データを収集・分析してシステム全体を制御するサーバーからシステムが構成されることを特徴とする．IoT システムでは，ノードとサーバー間の通信を改ざん・盗聴することによりシステム全体への攻撃が可能となることから，システム内の通信に暗号技術を適用しシステムを守る必要がある．

これまで，暗号技術は主に PC やワークステーション等の据え置き型の計算機上に実装されてきたが，近年ではスマートカードや携帯電話を始めとする組込み用途への応用が進んでいる．これらの機器，特に非接触 IC カードや RFID においては，限られた電力，リソースで高速な暗号処理が求められることから専用ハードウェアによる実装が必要不可欠となっている．また，接触型 IC カードや携帯電話などにおいては，専用ハードウェアより安価な暗号実装の形態として，組込みマイコン上のソフトウェアとしての実装も広く用いられている．以上のよ

うな暗号技術のハードウェアもしくはソフトウェアによる実装を本論文では暗号モジュールと呼ぶ.

組込み機器に暗号モジュールを実装した場合，攻撃者自身が暗号モジュールを所持することが想定されることから，暗号アルゴリズムへの攻撃だけでなく，暗号モジュールそのものへの攻撃による秘密情報の奪取の危険性が指摘されている．このような攻撃は，現在スマートカード等で問題となっているのみならず，人に監視されないデバイスが大多数を占める IoT システムを実現する上で大きな問題となることが予想される．暗号アルゴリズムをモジュールとして実装すると，暗号アルゴリズムの設計段階で考慮されていない新たな脆弱性が生じる場合があることから，一般に安全とされているアルゴリズムを用いていても，秘密情報が漏洩する可能性がある．暗号モジュールへの攻撃は物理的なアクセスを伴うことから，物理攻撃と呼ばれている．物理攻撃には様々な手段が存在し，古典的なものでは LSI のパッケージを開封して内部の回路を観察し，リバースエンジニアリングにより各部の機能を明らかにした上で，秘密情報が流れる配線上にプローブを当てて電圧を読み取る攻撃が挙げられる．これに対し，チップを開封すると秘密情報が消去される対策が取られてきたが，近年ではチップの微細化に伴いこのような攻撃を行うこと自体が困難となりつつある [2].

秘密情報への直接的なアクセスが困難になる一方，近年では暗号モジュールを搭載する LSI に存在する非正規の入出力を利用する攻撃の可能性が指摘されている．特に，暗号処理を実行中のモジュールから生じる漏洩情報 (消費電力，電磁波放射，処理時間など) を測定し，それを解析することにより秘密情報を得るサイドチャネル攻撃や，暗号処理の実行中に一時的なビット反転等の故障を生じさせ，暗号文の誤った動作を分析して秘密情報を推定する故障注入攻撃がよく知られている．これらの攻撃は古典的な侵入検知などによる対策が困難であることから，サイドチャネル攻撃に対しては，消費電力と秘密情報の関連性を取り除く対策が取られている．故障注入攻撃に対しては，通常の暗号処理の他に検算を行い，結果の整合性を調べる検算型対策が代表的である.

物理攻撃の分野における研究では，まず現在までに知られていない攻撃の可能性（脆弱性）を指摘（提案）し，その上で脆弱性をカバーする対策が提案されてきた．さらに，提案された対策に対して再度新たな攻撃を提案し，その対策を提案するというように，上記の手順をサイク

ルとして繰り返すことにより暗号モジュールの安全性を高める（脆弱性を減らす）取り組みがなされている．物理攻撃の中で，サイドチャネル攻撃に関しては，攻撃とその対策に関する研究が盛んに行われてきており [3]，プロトコルレベル，アルゴリズムレベル，回路レベルといった様々なレイヤでの対策手法が提案されてきた．これにより，新たに攻撃法が発見された場合でも，既存の対策手法で防ぐことができる場合が多く，安全性は高められてきていると言える．また，物理攻撃に対する暗号モジュールの安全性評価には，上記のような理論的な攻撃の可能性を指摘するのみならず，実際のモジュールに対して攻撃を試行することにより，実験的に評価する技術が不可欠と考えられている．本論文では，この評価を動作解析と呼ぶ．サイドチャネル攻撃の分野では，統一的な基準で安全性評価を行うための実験環境として，サイドチャネル攻撃標準評価ボード (Side-channel Attack Standard Evaluation BOard: SASEBO) [4] が開発された．SASEBO は搭載する暗号モジュールの種類 (ASIC，FPGA，スマートカード上のプログラム) により複数の種類が用意されており，現在，国内外 100 を超える研究機関・企業・大学等で利用され，事実上の世界標準となっている．統一的な基準で安全性評価が可能となったことで，暗号モジュールの試験および認証を行う制度である CMVP(Cryptographic Module Validation Program) [5] 等の認証制度にもサイドチャネル攻撃への耐性評価が加えられ，サイドチャネル攻撃に対する安全性評価と認証の制度が確立されつつある．

一方，故障注入攻撃に関しては，現在でも既存の対策で防ぐことができない攻撃の可能性が指摘されている．代表的なものが故障感度解析と多重故障注入攻撃である．故障感度解析は，暗号ハードウェアの遅延時間が入力に依存して変化することを利用する攻撃で，従来の対策として代表的な検算型対策を無効化できることから，従来の物理攻撃対策で防ぐことは困難であることが知られている [6]．また，多重故障注入攻撃は，組込みマイコンにおいて 1 回の暗号処理の間に複数回故障を注入し，ソフトウェア実装された対策を回避する攻撃である [7]．これらの攻撃では，セットアップ時間違反を用いて故障注入が行われる．セットアップ時間違反は，暗号回路の中の組み合わせ論理回路において演算が完了しないうちに次のクロックサイクルが開始されることにより発生する．セットアップ時間違反を用いた故障注入では，レーザー照射を用いた場合と異なるモデルに基づき故障が注入されるため，上記のように従来の暗号モジュールへの攻撃に対する対策が無効となる場合がある．

さらに安全性評価についても，手法が確立されていないのが現状である．サイドチャネル攻撃とは異なり，故障注入攻撃については依然として統一的な実験環境が存在しないため，各研究者が独自に構築した環境を用いて実験を行っている． このため，特にセットアップ時間違反を用いた故障感度解析などの攻撃で問題となる，秘密情報の取得に要する時間や，暗号化の試行回数といったパラメータの定量的な評価を行い，その結果を比較することはできないのが現状である．

本論文では，上記の課題や要求に基づき，「セットアップ時間違反を用いた故障注入攻撃に対して安全な暗号モジュールの実現」を目的とする．具体的には，(a) 統一的な故障注入攻撃実験システムの開発，(b) 故障感度解析への対策技術の開発，(c) 多重故障注入攻撃の検討と対策技術の開発を行う．

(a) では，サイドチャネル攻撃標準評価ボード SASEBO 上に実装できる故障注入攻撃実験システムを提案する．提案システムは SASEBO と標準的な PC を用いて構築でき，信号発生器などの外部機器が不要である．SASEBO はサイドチャネル攻撃の標準的な実験環境として既に普及していることから，異なる評価者の間で統一的なシステムとして使用できる．故障注入の手段としてはクロックグリッチを用いる．クロックグリッチはセットアップ時間違反を引き起こす故障注入手法の中でも最も高い精度で発生タイミングを制御できることが知られている．そこで，本論文では単一の FPGA 上に実装できるクロックグリッチ生成回路を提案し，これを用いて SASEBO 上に故障注入の機能を実現する．

(b) では，故障感度解析への対策技術を開発する．故障感度解析は，外部から故障感度を観測できないよう，隠蔽することで防止できる．そこで提案対策では，ある一定のタイミングで組み合わせ回路とレジスタを接続することで，外部から観測される故障感度を常に一定とする．本論文では，上記のタイミングを製造後に調整可能な遅延回路により生成する対策回路を提案する．提案回路では遅延時間を製造後に調整することにより，LSI の製造ばらつきを見込んだ動作マージンを削減し，対策に起因する動作速度のオーバーヘッドを削減できる．本論文では，提案対策を施した暗号ハードウェアを ASIC に実装し，(a) で提案した実験システムを用いて故障感度の測定実験を行う．これにより対策の有効性を示す．

(c) では， 組込みマイコン上で動作する暗号ソフトウェアに対して，適応的にタイミングを

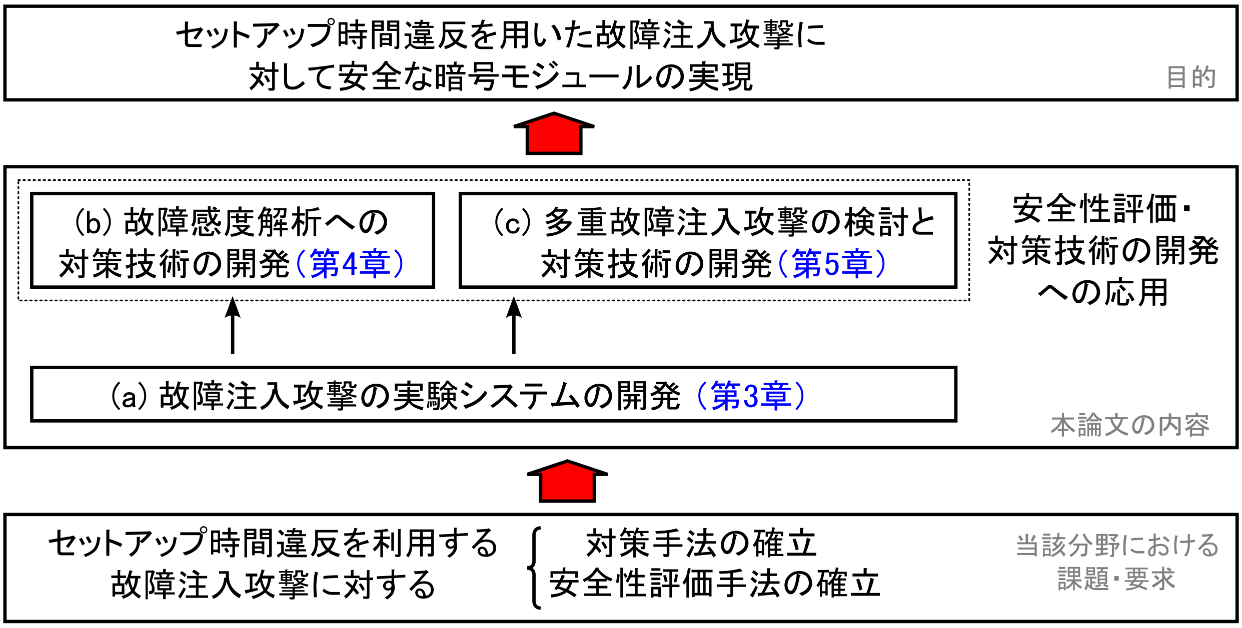


図 1.1　本論文の目的と内容

制御した故障注入により，複数の命令をスキップする多重故障注入攻撃の可能性があることを指摘する．この攻撃では，故障注入のタイミングを探索し，検算型対策を回避する攻撃が可能である．暗号処理の実行中に故障注入を試行し，次の故障注入のタイミングを処理結果に応じて適応的に制御することで，攻撃者が実行中の暗号ソフトウェアの詳細を知らないという条件下においても攻撃が可能となる．本論文では，典型的な故障注入攻撃対策の一つである検算処理が施されている暗号ソフトウェアに対して提案手法が有効となり得ることを実証する．まず，マイコンの出力という情報のみを用いて，差分故障解析に必要なデータ誤りと対策を無効化する故障を同時に発生させる手法について述べる．また，指摘した探索手法の実験を行い，実際の組込みマイコン上で実行できることを示す．その上で，検算型対策を施した AES 暗号ソフトウェアを実行する組込みプロセッサを用いた実験結果を示す．さらに，上記攻撃に対するソフトウェアレベルでの対策法を示す．

図 1.1 に本論文の目的と内容を示す．本論文では，(a) の実験システムを開発し，(b) および (c) は開発したシステムを用いて遂行する．(b) および (c) を通して対策技術を提供するとともに，(a) の実験システムにより安全性評価実験を可能とする．

本論文は，以上の内容を取りまとめたものであり，以下に示す 6 章により構成される．第 1

章は，本研究の背景と目的および本論文の概要をまとめた緒言である．第2章では，暗号技術とその実装に関する基礎的考察を述べる．暗号とその実装技術の概要，代表的な暗号アルゴリズム，暗号モジュールへの物理攻撃とその安全性評価について述べる．また，本論文の主たる内容である故障注入攻撃について，分類と現状の課題について述べる．第3章では，セットアップ時間違反を利用する故障注入攻撃を想定した安全性評価を行うための実験システムを開発する．ここでは，単一の FPGA 上に実装できるクロックグリッチ発生回路を提案し，FPGA を用いたクロックグリッチの生成手法を示す．また，生成したグリッチの時間分解能やジッタの測定を通し，故障注入攻撃に対する安全性評価に十分な性能を有することを示す．第4章では，故障感度解析への対策技術を開発する．まず，暗号回路と組み合わせて実装できる故障感度解析に対する対策回路を提案する．対策回路では，製造後に遅延時間を調整可能な遅延回路の遅延時間により故障感度を隠蔽する．故障感度の漏洩を防ぐには，攻撃者がクロック周波数と電源電圧を変化させた場合でも故障感度が漏洩しないよう遅延時間を適切に設定する必要があることから，与えられた条件の元で統計的に故障感度の漏洩を防ぐ遅延回路の設定方法を示す．また，提案対策を施した AES ハードウェアを ASIC 上に試作し，試作チップを用いた故障感度測定実験により提案対策の有効性を示すとともに，対策回路の回路面積を示し，数 % 程度のオーバーヘッドで実装できることを示す．第5章では，組込みマイコンにおける検算型対策を施した暗号ソフトウェアに対する攻撃の可能性を指摘する．ここでは，暗号化の後に復号を行い，復号結果が入力と等しいことを確認する検算型対策を対象として，適応的な故障注入を行い誤った暗号文を出力させる攻撃アルゴリズムを示す．また，検算型対策を施した AES プログラムを実行する 8 ビットおよび 32 ビットのマイコンに対して上記攻撃の実験を行い，誤った暗号文が取得できることを示す．さらに，上記攻撃への対策として，命令の配置を変更して探索を防ぐ手法を示す．第6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

暗号とその実装に関する基礎的考察

2.1 はじめに

本章では，暗号とその実装に関する基礎的考察を行う．まず，暗号技術全般について概要を述べ，代表的な暗号アルゴリズムを概説する．次に，暗号機能を実装した暗号モジュールとそれに対する物理攻撃について述べる．最後に，本論文で主に取り扱う故障注入攻撃について，概要，分類，対策法の順に述べる．

2.2 暗号技術に関する基礎的考察

本節では，暗号技術の概要について述べる．以下では，まず情報セキュリティ技術における暗号技術の位置付けについて述べた後，特に重要な構成要素である暗号アルゴリズムについて述べる．その後，暗号アルゴリズムの実装について述べる．

2.2.1 暗号技術の概要

情報セキュリティ技術は，さまざまな脅威に対抗して，情報システムを安全に運用するための技術である [8]．広義の情報セキュリティでは，「過失」，「システム障害」，「災害」，「悪意を持った人間による不正行為」を脅威として考えるが，一般に情報セキュリティとよばれるもの（狭義の情報セキュリティ）は，「悪意を持った人間による不正行為」を指す場合が多い．狭義

の情報セキュリティ技術は，これらの脅威に対して，以下に示す 3 つの性質を実現することにより安全性を保証することを目的とする．

- Confidentiality （秘匿性・機密性）
  権限のない第三者から情報の内容を隠すこと
- Integrity （完全性・整合性・一貫性）
  情報が正真であり，改ざんされていないことを保証すること
- Availability （可用性）
  権限があるものはいつでも情報が利用できること

暗号技術は，上記の 3 つの性質のうち，主に秘匿性と完全性を確保するための技術と位置づけられる．秘匿性を守るための「暗号」と，完全性を保証するための「認証・署名」により，さまざまな脅威に対抗する．暗号技術は，(i) 暗号アルゴリズム，(ii) ハッシュ関数，(iii) 擬似乱数生成器の 3 つを基本ブロックとして，上記の性質を保証するような暗号プロトコルを構築する．以下では，それぞれの技術について概説する．

(i) 暗号アルゴリズムは，あるメッセージを第三者が読めない形に変換するためのアルゴリズムであり，暗号化や復号，認証，署名に用いられる．ここで，暗号化 E とは，平文 P を，暗号化鍵 K_E を用いて暗文 C へ変換することである．一方，復号 D は，暗文 C を，復号鍵 K_D を用いて元の平文 P へ変換することを指す．以上の関係は，式 (2.1)，(2.2) でそれぞれ記述される．

$$C = E(P, K_E) \tag{2.1}$$

$$P = D(C, K_D) \tag{2.2}$$

暗号アルゴリズムは，暗文から平文を復元することが，復号鍵 K_D を持たない者にとって非常に困難か不可能となるように設計される．これにより，上記の 3 つの性質を満たすことが可能となる．ここで，暗号化鍵 K_E と復号鍵 K_D が一致するものを共通鍵暗号方式，異なるものを公開鍵暗号方式とよぶ．

(ii) ハッシュ関数とは，任意の長さのメッセージから，一定の長さの値（ハッシュ値）を算出するアルゴリズムである．ハッシュ関数は認証・署名に用いられ，特に (a) 一方向性，(b)

第 2 原像計算困難性，(c) 衝突困難性といった特別な性質を満たすように作られる．(a) 一方向性とは，あるハッシュ値の元となるメッセージを求めることが困難であることである．(b) 第 2 原像困難性とは，あるハッシュ値とその元となるメッセージが与えられたとき，元のメッセージとは異なり，かつハッシュ値が等しいメッセージを求めることが困難であることである．(c) 衝突困難性とは，同じハッシュ値となる 2 つの異なるメッセージを求めることが困難であることである．これらの性質により，あるハッシュ値を作り出せるのは，事実上元のメッセージを知っている場合に限られる．このため，通信の前後でハッシュ値を比較することで，完全性の検証が可能である．通信の前後でメッセージに改変が加えられた場合にはハッシュ値が変化するため，改ざんを検知できる．

(iii) 擬似乱数生成器 (PRNG: Pseudo Random Number Generator) は，統計的に偏りのない系列（乱数）を出力するアルゴリズムである．暗号技術においては，鍵の生成などに用いられている．暗号技術における PRNG が出力する系列には，(a) 無作為性，(b) 予測不可能性，(c) 再現不可能性が求められる．(a) 無作為性とは，統計的に偏りがないという性質であり，(b) 予測不可能性は，過去の系列から次の擬似乱数を予測できないという性質である．

2.2.2　暗号アルゴリズム

暗号アルゴリズムは，暗号技術における最も重要な構成要素である．以下では，暗号方式，暗号アルゴリズムの安全性，暗号アルゴリズムの応用について述べる．現在広く用いられている暗号方式は，大きく分けて共通鍵暗号方式と公開鍵暗号方式の 2 つに分類される．以下では各方式について述べる．

共通鍵暗号方式

共通鍵暗号方式とは，暗号化鍵と復号鍵に同じ鍵を用いる方式である．公開鍵暗号方式に比べ処理が高速であるという利点があり，大量のデータの暗号化や高速な認証に適している．一方，共通鍵暗号を用いて通信するためには，送信者と受信者が事前に鍵を共有しておく必要があり，ネットワーク上で安全に鍵配送を行うことが困難である (鍵配送問題が生じる) という欠点を持つ．共通鍵暗号は，平文と暗号文の関係を第三者には未知の乱数表を介して対応づけることを基本概念とする．しかしながら，一般に巨大で完全な乱数表を実装することは困難で

ある．このため，共通鍵暗号アルゴリズムでは，換字や転置といった簡単な処理を繰り返し適用することで擬似的な乱数表を効率的に実現する．共通鍵暗号アルゴリズムは，一定の長さのブロック単位ごとに換字・転置を行うことで暗号化・復号を行うブロック暗号アルゴリズムと，擬似乱数生成器から出力される鍵系列を利用して暗号化を行うストリーム暗号アルゴリズムに分類される．

公開鍵暗号方式

公開鍵暗号方式とは，暗号化鍵と復号鍵に異なる鍵を用いる方式である．一方の鍵を公開しても暗号の安全性は保たれるため，共通鍵暗号方式で問題となった鍵配送問題を解決できる．このため，不特定多数の暗号通信に適している．また，公開鍵暗号方式は暗号鍵と復号鍵の役割を入れ替えることによりデジタル署名にも利用できる．こうした利点の一方で，暗号化には複雑な演算を繰り返し処理する必要があり，共通鍵暗号方式と比較して，単位データ量あたりの処理に時間がかかるという欠点がある．このため，公開鍵暗号方式はメッセージ自体の暗号化ではなく，共通鍵暗号方式で用いる秘密鍵の配送手段として効率的に用いられる．公開鍵暗号は，公開鍵から秘密鍵を求めることが非常に困難であるという点を基本原理としている．例えば RSA 暗号 [9] を解読することは，大きな数の素因数分解問題または有限体上の離散対数問題に帰着することができる．また，楕円曲線暗号 [10] は楕円曲線上の離散対数問題に帰着できる．

2.2.3　暗号アルゴリズムの具体例

ここでは，共通鍵暗号と公開鍵暗号の代表的なアルゴリズムである AES および RSA 暗号について述べる．

AES

AES は，ISO/IEC 18033 として標準化されており，現在最も広く用いられている共通鍵暗号アルゴリズムである．Substitution-Permutation Network (SPN) 型の 128 ビットブロック暗号アルゴリズムであり [11]，鍵長は 128，192，256 ビットをサポートする．安全性だけではなく，実装の性能も重視したアルゴリズムであり，ソフトウェアとハードウェア共にプラットフォームに応じた多様な設計が可能である．以下では，鍵長 128 ビットにおける AES のア

ルゴリズムについて述べる．

図 2.1 に，AES の暗号化アルゴリズムを示す．AES では，128 ビットの入力データを，1 つの要素が 1 バイトで大きさが 4×4 の行列として解釈する．暗号化・復号の処理は，ラウンド処理と呼ばれる処理を 10 回適用することにより暗号化を行う．ラウンド処理は 4 種類の基本演算である *SubBytes, ShiftRows, MixColumns*, および *AddRoundKey* からなる．ここで，*SubBytes* は 8 ビット入出力の非線形変換テーブル S-box による変換，*ShiftRows* はバイト単位の循環シフト，*MixColumns* は 32 ビット入出力の線形変換，*AddRoundKey* は XOR による ラウンド鍵の加算である．一方，各ラウンドで用いられるラウンド鍵を生成する処理である鍵スケジュールは，XOR，循環シフト，定数 $Rcon_i$，および S-box を用いて行う．

AES の復号は，暗号化と反対の手順を踏むことにより行われる．AddRoundKey は XOR であるため暗号化と共通であり，*SubBytes, ShiftRows, MixColumns* それぞれの逆変換である *invSubBytes, invShiftRows, invMixColumns* が用いられる．

RSA 暗号

RSA 暗号 [9] は Rivest，Shamir，Adleman により考案された公開鍵暗号であり，AES と同様に ISO/IEC 18033 に規定されている．基本的なアルゴリズムは，以下に示すべき乗剰余演算である．M を元のデータ（平文），C を暗号文，E と N を公開鍵，D を秘密鍵とすると，暗号化および復号はそれぞれ式 (2.3)，(2.4) で表される．

$$C = M^E \bmod N \tag{2.3}$$

$$M = C^D \bmod N \tag{2.4}$$

鍵である法 N や秘密鍵 D には，安全性の観点から通常は 1,024 ビット以上の多倍長整数が利用されることが多く，平文 M や暗号文 C も法 N と同一の語長が用いられる．

RSA 暗号のべき乗剰余演算は，指数 E，あるいは D のビットパターンに応じて自剰余算と乗剰余算を繰り返すことによって実現される．べき乗剰余演算のアルゴリズムについては第 4 章で述べる．また，法 N の素数 p，q の値が既知である場合には，中国人剰余定理 (CRT: Chinese Remainder Theorem) を用いた高速演算手法を適用することができる．以下にその計算手順を示す．

まず，べき乗剰余演算 $C = M^E \bmod N$ $(N = pq)$ の各パラメータから事前に以下の値を計

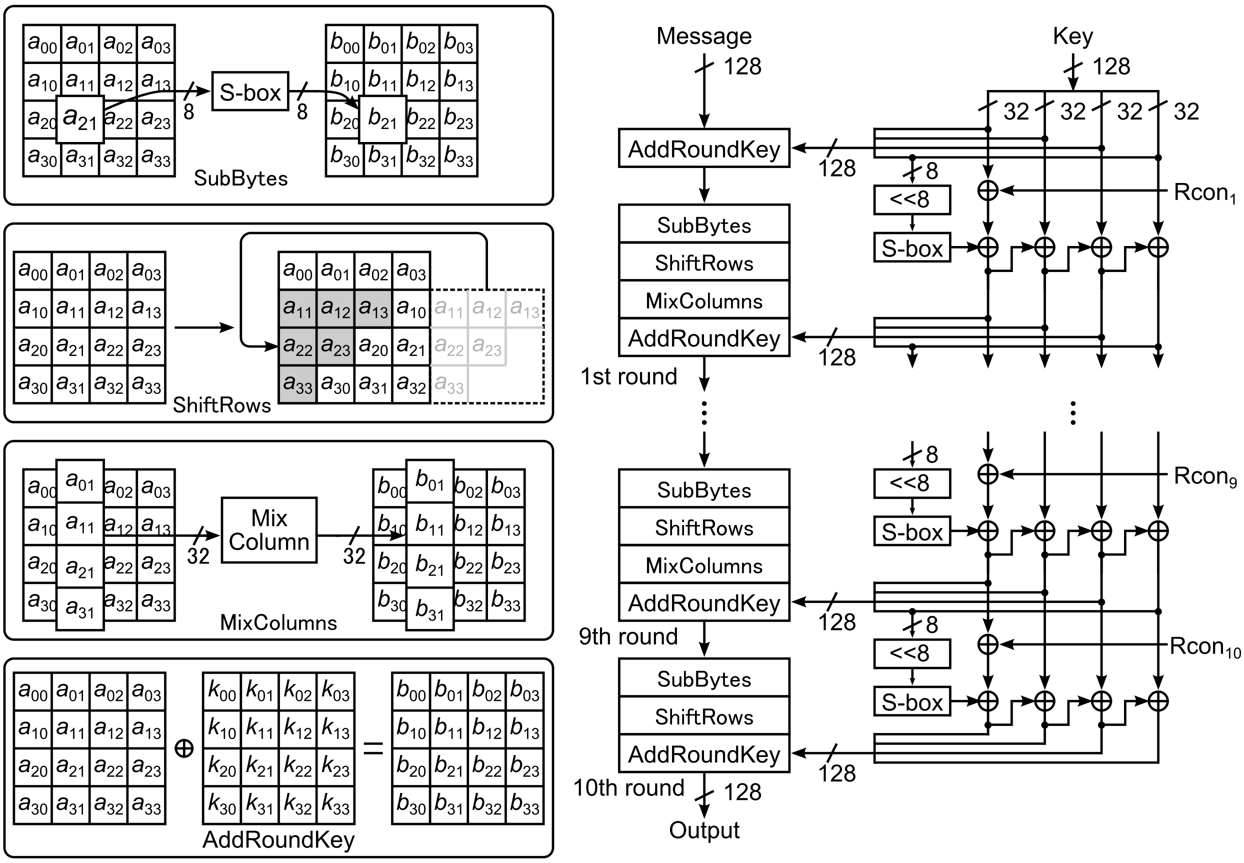


図 2.1　AES の暗号化アルゴリズム

算しておく．

$$p_q^{-1} = p^{-1} \bmod q \tag{2.5}$$

$$E_p = E \bmod (p-1) \tag{2.6}$$

$$E_q = E \bmod (q-1) \tag{2.7}$$

次に，M を p と q で割った余り M_p と M_q を求め，

$$M_p = M \bmod p \tag{2.8}$$

$$M_q = M \bmod q \tag{2.9}$$

これと事前計算した E_p と E_q をそれぞれ指数とするべき乗剰余演算を行い，C_p と C_q を求める．

$$C_p = M_p^{E_p} \bmod p \tag{2.10}$$

$$C_q = M_q^{E_q} \bmod q \tag{2.11}$$

最後にそれらを合わせたものが最終結果 C となる．

$$C = ((C_p - C_q)p_q^{-1} \bmod q)p + C_p \tag{2.12}$$

p と q のビット数は N のおよそ半分なので，べき乗剰余演算の計算量は $(1/2)^3 = 1/8$ となり，それを，C_p と C_q のために 2 回行うので，CRT 演算では演算量を前処理や後処理を除いて，およそ $1/8 \times 2 = 1/4$ に削減できる．

2.2.4　暗号アルゴリズムの安全性

どのような共通鍵暗号アルゴリズムでも，秘密鍵の取りうる値は有限であるため，原理的には秘密鍵を 1 つずつ検査する鍵全数探索攻撃により必ず正しい鍵を見つけることができる．このため，鍵全数探索攻撃に対する耐性が共通鍵暗号アルゴリズムの安全性の上限となる．鍵全数探索攻撃に対する耐性は鍵長によって大きく左右される．現在では，鍵全数探索に十分な耐性を持たせるため，128 ビット以上の鍵長を使うことが一般的である．既知の解読方法に対して，鍵を求めるために必要な計算量が鍵全数探索攻撃の計算量を下回ることがないと証明された場合に，その共通鍵暗号を安全であるとする．言い換えると，鍵全数探索よりも高速な鍵導出アルゴリズムが存在する場合，そのアルゴリズムは解読されたこととなる．

公開鍵暗号アルゴリズムの安全性は，公開鍵から秘密鍵を求めることが，数学的未解決問題と等価であると示すことにより証明される．公開鍵暗号においても，鍵長は安全性を左右するセキュリティパラメータである．鍵長は，数学的未解決問題が定量的に安全であると示される範囲で設定される．

なお，現在の暗号システムで用いられている暗号アルゴリズムは，第三者評価機関によりアルゴリズムの安全性が十分に評価され，推奨暗号として標準化されたものを用いることが一般的である．

2.3　暗号モジュールに関する基礎的考察

本論文では暗号技術を実装したハードウェアやソフトウェアを暗号モジュールと呼ぶ．暗号モジュールの重要な機能の 1 つが鍵の保存・運用である．暗号システムを安全に運用するに

は，秘密鍵を攻撃者に知られることなく保存し，運用する必要がある．通常，秘密鍵は暗号アルゴリズムを実行するチップ上に保存されており，暗号技術の利用者は暗号アルゴリズムを介して秘密鍵を用いることにより，攻撃者に知られることなく利用できる．

2.3.1　暗号アルゴリズムの実装形態

暗号アルゴリズムの実装形態は，ソフトウェア実装とハードウェア実装とに大別できる．ソフトウェア実装は，汎用プロセッサ上で動作するプログラムにより暗号機能を実現する．一方ハードウェア実装は，専用 LSI (Application Specific Integrated Circuit: ASIC) や，Field Programmable Gate Array (FPGA) 上の回路により暗号機能を実現したものを指す．

ソフトウェア実装はプログラムの書き換えにより機能の修正や拡張を容易に行うことができ，CPU やメモリには安価な既製品を用いることができるためコスト面でも有利であるが，既存の命令セットを用いて暗号機能を実現しなければならないため，速度や消費電力はハードウェア実装に比べ不利である．一方ハードウェア実装では，暗号実装に適したアーキテクチャを採用できるため，高速，小型かつ低消費電力で暗号機能を実現できる．特に，携帯電話やスマートカード，RFID などの応用では，小型かつ低消費電力への要求が特に厳しく，ハードウェア実装は必要不可欠である．また，ハードウェア実装は，物理攻撃への対策回路を組み込むことができることから，汎用 CPU 上で動作するソフトウェア実装に比べ高い安全性を実現することも可能である．

2.3.2　暗号モジュールへの物理攻撃

本論文では，攻撃者が暗号モジュールへ物理的にアクセスして秘密情報の抽出を試みることを，物理攻撃とよぶ．暗号アルゴリズムの設計段階では，送信者と受信者が安全でない通信路を介して通信を行うモデルに基づき，通信路の盗聴・改ざんを行う攻撃者を想定する．このため，攻撃者は暗号アルゴリズムの入出力にしかアクセスできないと仮定されている．しかし，物理攻撃では，攻撃者が通信路に流れる情報以外の情報 (暗号処理中の中間値など) へアクセスすることが想定される．物理攻撃では，攻撃者はチップの開封を伴う内部信号の観測や，消

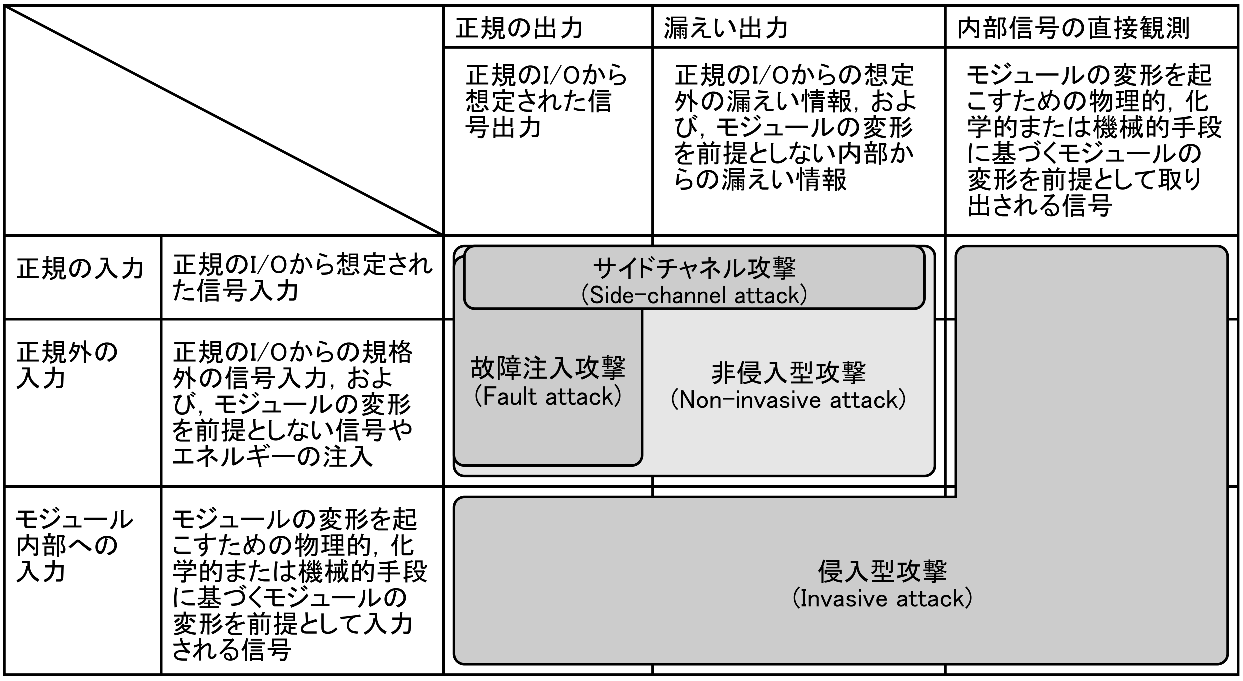


図 2.2　物理攻撃の分類:文献 [12] の表 1.1 および図 1.2 を文献 [8] を参考にして加工したもの

費電流やチップからの漏洩電磁波の測定により，解読に必要な情報を得ることから，上記の仮定が成立せず，安全とされている暗号アルゴリズムでも解読のおそれがある．

Web サーバや銀行の ATM など，暗号モジュールへの物理的なアクセスを制限できる場合には，物理攻撃は大きな問題とならない．しかしながら，近年爆発的に普及している携帯電話やスマートカードなどの組み込み機器は，攻撃者自身により所持されることが多く，暗号モジュールへの物理的なアクセスが容易であることから，物理攻撃は現実的脅威であるといえる．

2.3.3　物理攻撃の分類

図 2.2 に，文献 [12] に示された物理攻撃の分類を示す．図 2.2 は，暗号モジュールへの入力および出力の面から物理攻撃を分類したものである．最も大きな分類として，対象の暗号モジュールの変形 (破壊) を伴うか否かについて侵入型攻撃 (Invasive attack) と非侵入型

(Non-invasive attack) の 2 種類が存在する．侵入型攻撃は，暗号モジュールに対して狭義のタンパー手段 (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，内部信号の観測や改変を行う攻撃である．暗号アルゴリズムを実装した回路に直接アクセスするため非常に高い攻撃能力を有する反面，LSI の開封には専門知識や高価な機器が必要となる．

対策としては，モジュール自体を比較的大型の筐体に収め，侵入監視用のバッテリを設けた上で，筐体を開くと内部の秘密鍵が消去されるスイッチを設けたり，チップの周囲に侵入検知用の配線を配置し，配線が切れると秘密鍵が消去されるといった対策が取られてきた．

一方，非侵入型攻撃は，暗号モジュールの変形を伴うことなく，正規または正規外の入出力を介して行う攻撃である．非侵入型攻撃は，侵入型攻撃と比較して検知が困難であり，攻撃方法によっては安価な機器で攻撃可能なため，現実的な脅威として注目されている．

非侵入型攻撃には，サイドチャネル攻撃 (Side-channel attack)，故障注入攻撃 (Fault attack または Fault injection attack)，およびこれらを組み合わせた攻撃がある．故障注入攻撃にはチップの開封を伴う攻撃も存在するが，通常はチップ本体の改変を伴わないことから，非侵入型に分類している．サイドチャネル攻撃は，暗号モジュールから生じる漏えい情報 (消費電流，電磁波放射，処理時間など) から秘密情報を読み取る攻撃である．故障利用攻撃は，暗号モジュールに対して正規でない入力を印加し，秘密情報の漏洩を誘起する攻撃である．故障注入の手段としては，レーザーや光を照射して記憶素子中のビットを反転させるものや，クロックグリッチや電源電圧の操作によりセットアップ時間違反を引き起こすものがある．これらの攻撃は，レーザー照射を除けばオシロスコープや PC，FPGA ボードといった比較的安価な機器のみで実行可能である．故障利用攻撃は，攻撃の検知に関してはサイドチャネル攻撃と比較すると容易であるが，攻撃方法によっては暗号処理を 4 回行うのみで秘密情報の奪取が可能である [13] など，従来のサイドチャネル攻撃より強力な攻撃となりうる．サイドチャネル攻撃に関しては，従来から盛んに研究されており，安全性評価と認証の制度が確立されつつある．しかし，故障注入攻撃に関してはその対策法が発展途上の段階であり，サイドチャネル攻撃へ対策を施したモジュールであっても攻撃の可能性があるため，近年注目を集めている．

2.3.4　暗号モジュールの安全性評価

暗号モジュールの物理攻撃に対する耐性は，暗号モジュールのアーキテクチャ，暗号モジュールを搭載するチップのプロセスルール等の実装上の特性に大きく依存することから，理論的には攻撃可能とされている攻撃でも実際には攻撃困難である場合やその逆の場合がありうる．従って，暗号モジュールそのものへの攻撃に対する実験的な安全性評価が必要である．物理攻撃に対する暗号モジュールの安全性評価には，実機による実験が必要である．実際に，暗号モジュールの試験および認証を行う制度である CMVP(Cryptographic Module Validation Program) [5] や，IPA が運用する JCMVP(Japan CMVP) [14] においても実験による評価が行われている．攻撃の知識および実験環境をもつテストラボにより，ある制約条件の下で暗号モジュールに対して攻撃が試行される．テストラボはこの試験をもとに報告書を作成し，最終的に認証機関が報告書の内容を元に認定証を発行する．

上記の評価において試行できるのは，当然ながら既知の攻撃に限られるため，可能な限り多くの攻撃を試験することが安全性の向上につながる．このため，できるだけ多くの脆弱性を指摘（攻撃を提案）することが安全性の向上に不可欠と考えられている．また，上記の実験の結果は実験環境に大きく依存することから，統一的な安全性評価のためには，再現性の高い実験を可能とする実験環境が必要となる．

2.4　故障注入攻撃に関する基礎的考察

本節では，本論文で主に取り扱う暗号モジュールへの故障注入攻撃について，分類，具体例，対策法の順に述べる．また，故障注入攻撃の最初の段階である故障注入の手法について述べ，セットアップ時間違反を引き起こす故障注入手法を説明する．

従来の研究では，ある故障が起こったと仮定し，その上で秘密情報の取得の可能性を論じる故障解析 (Fault analysis) と，故障を注入する手法について検討されてきた．実際の攻撃では，まず攻撃者は何らかの方法で故障注入を行い，状況に応じて故障解析の手法を用いることにより秘密情報の取得（攻撃）を実行する．

2.4.1　故障解析の分類

暗号モジュールに対する故障解析 (Fault analysis) に関する研究では，攻撃者が何らかの故障を注入することを仮定し，その仮定の下で秘密情報を抽出する方法について論じられてきた．

暗号モジュールに対する故障解析手法は，最初に公開鍵暗号に対する手法が提案された [13]．この論文では，中国人剰余定理 (Chinese Remainder Theorem) を用いた RSA 暗号において，ある変数に故障を注入することにより因数分解を容易にする攻撃が示された．これは Bellcore attack と呼ばれている．その後共通鍵暗号に対する手法が提案された [15]．これは差分故障解析 (Differential Fault Analysis: DFA) と呼ばれる解析手法であり，正しい暗号文と誤りを含む暗号文の差分を元に秘密鍵の候補を絞り込む攻撃である．現在までに多くの解析手法が提案されている．

故障注入攻撃は，攻撃を行う階層により，(a) プロトコルレベル，(b) アルゴリズムレベル，(c) 回路レベルの攻撃に分けられる．(a) プロトコルレベルの攻撃としては，公開鍵暗号において法 (Modulus) に不正な値を与えて秘密情報を抽出する攻撃 [16] が挙げられる．(b) アルゴリズムレベルの攻撃としては，共通鍵暗号における差分故障解析 [15] や，公開鍵暗号に対する Safe error 攻撃 [17] がよく知られている．(c) 回路レベルの攻撃としては，故障感度解析 [18] がある．

故障注入攻撃では，攻撃者が注入する故障に関して何らかの仮定を設ける．この仮定を故障モデルと呼ぶ．上記の攻撃のうち，(a) や (b) では，暗号アルゴリズム中の変数の一部のビットが反転するという故障モデルが想定されている．一方 (c) では，故障が発生した場所だけでなく，故障の発生タイミング (故障感度) も利用して攻撃する．

一方，故障解析で想定されない故障を利用した故障注入攻撃も存在する．ソフトウェア実装においては，CPU 上で実行される命令そのものをスキップする命令スキップを用いた故障注入による攻撃の可能性も指摘されている [19]．命令スキップが起こった場合，物理攻撃対策が施されているプログラムであっても，対策コードの実行をスキップすることにより秘密情報の取得が可能である場合がある．この攻撃は，暗号アルゴリズム上では対策が困難であることか

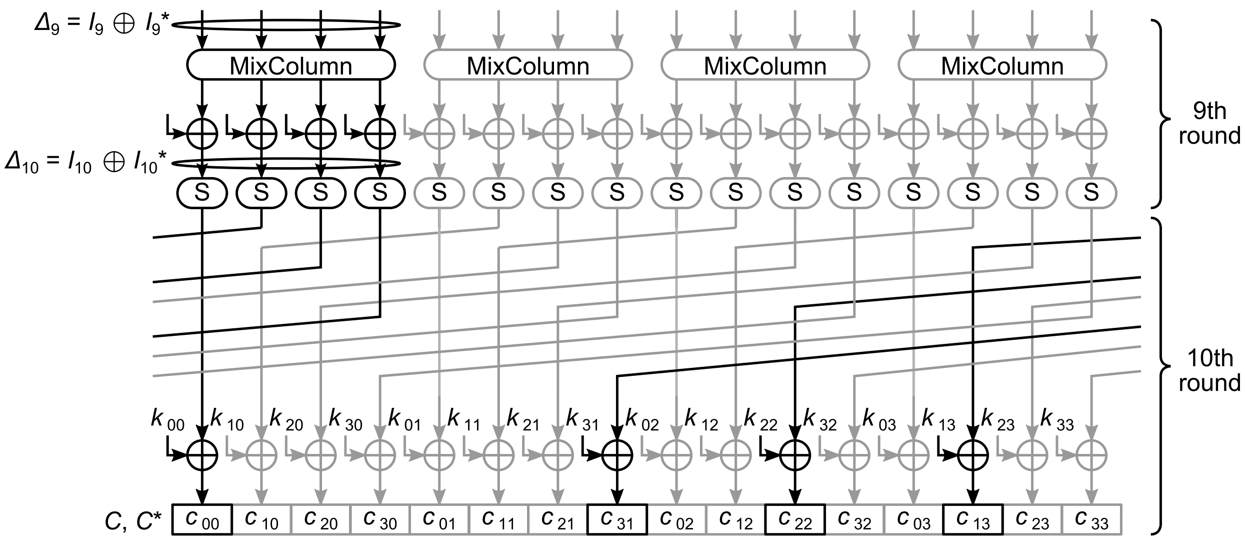


図 2.3　AES の 9 ラウンド目以降のデータの流れ

ら，故障解析の分野では扱われてこなかった．

2.4.2　AES への差分故障解析

ここで，故障解析の具体例として，AES に対する差分故障解析 [20] を示す．差分故障解析 (DFA: Differential Fault Analysis) は，正しい暗号文と誤りを含む暗号文の差分を用いて鍵候補を絞り込む解析手法である．

攻撃者は攻撃対象のデバイスを用いて正しい暗号文 C と誤りを含む暗号文 C^* を計算し，これらを用いて鍵候補を絞り込む．C^* の計算の際，AES の 9 ラウンド目の入力 I_9 のうち 1 バイトに誤りが発生すると仮定する．以降，C と C^* の組を暗号文ペアと呼ぶ．図 2.3 に，AES の 9 ラウンド目以降のデータの流れを示す．ここでは，10 ラウンド目の入力を上流と下流の両方から計算し，両者の取りうる範囲が異なることを利用して鍵候補を絞り込む．

ここでは，図 2.3 において，最も左側の MixColumns に入力される 4 バイトの値のうちいずれかに故障が注入されたと仮定し，その 4 バイトに注目する．ここで，ラウンド入力の i バイト目を $I_n(i)$ とする．誤りがないときおよび誤りを含む 9 ラウンド目の入力をそれぞれ I_9, I_9^* とする．$\oplus$ は排他的論理和である．

$$\varDelta_9 = I_9 \oplus I_9^* \tag{2.13}$$

式 (2.13) において，Δ_9 は故障のあるバイトのみが 0 以外の値を持つことから，4 バイトのうちある 1 バイトのみが 0 以外の値を持ち，残り 3 バイトが 0 であるという制約を持つことが分かる．これより，鍵候補を $2^8 \times 2^2 = 2^{10}$ 通りに絞り込むことができる．

次に，10 ラウンド目の入力における正しい暗号文と誤った暗号文との差分 ΔI_{10} を求める．MixColumns は線形演算であることから式 (2.14) が成り立つ．なお，9 ラウンド目のラウンド鍵 k' は，正しい暗号文と誤った暗号文の差分には影響しないことから式中には含まれない．

$$\begin{aligned}\Delta_{10} &= MixColumns(I_9) \oplus MixColumns(I_9^*) \\ &= MixColumns(I_9 \oplus I_9^*) \\ &= MixColumns(\Delta_9) \end{aligned} \tag{2.14}$$

ここで，Δ_9 は 2^{10} 通りの値に制限されることから，式 (2.14) を用いて取りうる Δ_{10} の値を計算できる．

続く SubBytes の演算は非線形演算であることから計算を続けることができない．そこで，暗号文から 10 ラウンド目の入力を逆算する．ここでは，暗号文 C と推定鍵 k' を用いて逆算した 10 ラウンド目の入力を $J_{10}(C, k')$ と表記する．$J_{10}(C, k')$ は式 (2.15) のように求められる．

$$J_{10}(C, k') = InvSubBytes(InvShiftRows(C \oplus k')) \tag{2.15}$$

これを用いると，10 ラウンド目の入力における差分は式 (2.16) で表される．

$$\Delta J_{10}(k') = J_{10}(C, k') \oplus J_{10}(C*, k') \tag{2.16}$$

差分故障解析では，全ての鍵候補 $(0 \leq k' \leq 2^{32})$ に対し，式 (2.16) の計算を行い，$\Delta J_{10}(k')$ が Δ_{10} の取りうる値に含まれるかどうかを判定する．含まれる場合，k' を鍵候補に追加する．

Δ_{10} は 2^{10} 個であることから，鍵候補を 2^{10} 個に絞り込むことができる．ただし，ここで得られる鍵候補は 2^{10} 通りの部分鍵であり，鍵全体を復元するにはさらなる攻撃が必要である．平文を変更して暗号文ペアを取得し，上記の手順をもう一度繰り返すことにより，鍵候補を 1 つに絞り込むことができる．

この攻撃では，1 つの暗号文ペアに対して $O(2^{32})$ の計算量を要する．

表 2.1　故障注入手法の比較

故障注入手法の分類	故障注入手法	制御性		コスト
		時間的	空間的	
セットアップ時間違反	(i) 電源電圧の変化	○	×	◎
	(ii) クロック周波数の変化	◎	×	◎
	(iii) 温度変化	×	×	○
	(iv) 電磁波照射	×	×	○
電荷の注入	(v) 白色光照射	○	◎	○
	(vi) レーザ照射	○	◎	×

2.4.3　故障注入手法

故障注入攻撃に関する研究としては，故障が注入された後の解析手法だけでなく，故障の注入手法に関する研究も行われている．

故障の種類は，永久故障と過渡故障に大別される．永久故障は回路の破損を伴う故障であり，一度注入すると元の状態に復元できないため，複数回暗号処理を要する故障利用攻撃には適用できない．また，電源投入時の自己診断などを用いることにより比較的容易に検知できることから，永久故障による故障利用攻撃の脅威は小さいと考えられる．一方，過渡故障は種々の外乱により生じる一時的な値の反転を指す．過渡故障は繰り返し注入することができるため，永久故障と比較して多様な攻撃が考えられる．安全性評価の観点でも，永久故障は過渡故障により模擬できる場合が多い．以上の理由により，本論文では過渡故障を発生させる故障注入手法を扱う．

代表的な故障注入手法を表 2.1 に示す．過渡故障を発生させる故障注入方法には，(i) 電源電圧の変化 [21]，(ii) クロック周波数の変化 [22]，(iii) 温度変化 [23]，(iv) 電磁波照射 [24]，(v) 光照射 [25]，(vi) レーザ照射 [23] が挙げられる．

この中で，電磁波照射，電源電圧，クロック周波数および温度変化では，フリップフロップ

がセットアップ時間違反を起こすことにより誤りが注入される．一方，光やレーザ光を照射する場合，記憶素子 (メモリやフリップフロップ) の値が反転することにより誤りが注入される．

(v) および (vi) は LSI のパッケージを開封する必要がある侵入型の攻撃である．(vi) のレーザやイオンビームの照射では，半導体チップ上の特定の回路に誤りを注入することが可能なため，故障注入への対策回路を避けて故障を注入できる場合があるが，照射装置が高価である．(v) は蛍光灯や LED などの比較的安価な装置で実行可能であるが，空間的分解能は (vi) ほど高くない．

一方， (i) – (iv) は，LSI のパッケージを開封せずに行うことができる．(iii) 温度変化や (iv) 電磁波照射は，攻撃対象の機器に一切手を加えることなく攻撃可能である．特に (iv) 電磁波照射は，遠隔からの故障注入が可能であるという特長を持つことから強力な攻撃となりうる．これらの故障注入手法では，攻撃に有用な故障を発生させることは可能であるものの，再現性の高い故障注入は困難であるため，安全性の定量的評価に用いることは難しい．

(i) ， (ii) は正規の入力ポート (電源ピン，クロック入力) へ非正規の入力を与えることにより故障を注入する．これらの手法はモジュールが搭載されている基板上の配線を変更する必要があるが，特殊な装置を用いないため安価に攻撃可能であり，故障の注入タイミングの制御も容易である．特に，クロック周波数を変化させる攻撃は，数百ピコ秒のオーダで誤りの注入タイミングを制御できる．

故障解析の手法によっては，故障モデルに制限があるものもある．2.4.1 節で述べた Bellcore attack や差分故障解析，Safe error 攻撃などは，暗号アルゴリズム中の変数に故障を注入できればよいので，どの故障注入手法を用いても実行できる．一方，故障感度解析は，故障注入を用いてクリティカルパスの遅延時間を測定する攻撃であることから，セットアップ時間違反により故障注入を行う必要がある．

2.4.4 故障注入攻撃への対策

フォールトトレランス分野との相違点

回路に生じる過渡的な故障への対策は，古くからフォールトトレランスの分野で盛んに行われてきた．フォールトトレランスの分野では，チップのパッケージに含まれる放射性物質

や，宇宙から飛来する放射線により記憶素子の値が反転する故障に対する対策が取られてきた[26]．しかし，故障注入攻撃への対策を従来のフォールトトレランスの枠組みで対処することを考えると，以下で述べる問題点がある．

まず，フォールトトレランスの手法により故障注入への対策を施したとしても，攻撃を防げない場合がある．例として，近年提案された故障感度解析は，誤った出力を必要としないことから，故障検知では防止できないことが知られている．このような攻撃には，従来のフォールトトレランスの枠組みを超えた対策が必要である．

また，従来のフォールトトレランスの枠組みで故障注入攻撃対策を行うことを考えると，コストが膨大となる問題がある．自然発生する故障の場合，多数の故障が同時に発生する可能性は低いことから，想定する故障の同時発生数を制限することにより，故障への耐性を持たせるためのコストを削減する．しかし，故障注入攻撃では，悪意を持った攻撃者が故障を注入することから，任意の数の故障を同時に注入できる．このような故障を従来のフォールトトレランスの枠組みで対処することを考えると，コストはしばしば膨大となり，実用的ではないことが多い．故障注入攻撃対策は，スマートカード等のリソースが少ないデバイスに搭載されるため，対策実現のコストは低く抑えなければならない．

また，ソフトウェア的な対策が困難であるという問題もある．組込みマイコン上の暗号ソフトウェアにより暗号処理を行うシステムが広く用いられているが，物理攻撃への対策を施した暗号処理専用の組込みマイコンはコストが高いため，ハードウェア的な対策を持たない安価なデバイスで暗号処理を行うことが求められている．このようなデバイスでは，ハードウェアを変更せずにソフトウェアのみに故障注入攻撃対策を施す必要がある．ソフトウェア的な故障検知手法としては，プログラムフローを監視する手法がある．これは，プログラムの節点に特徴的なデータを埋め込み，これを加算していくことでプログラムの異常な動作を検知するものである．この手法でも，加算した結果に基づき正常か異常かを判断する判定処理を攻撃して，故障検知を回避できる可能性がある．

故障注入攻撃対策

前節の問題点を解決し低コストな対策を実現するため，故障注入攻撃に特化した対策が研究されている．故障注入攻撃への対策法も，攻撃法と同じく (a) プロトコルレベル，(b) アルゴ

リズムレベル， (c) 回路レベルの対策に分けられる．

(a) プロトコルレベルの対策は，鍵を暗号アルゴリズムに入力する前に乱数と混合し，秘密鍵が暗号アルゴリズムによりそのまま処理されないようにする対策 [27] が挙げられる．この対策では，暗号化の処理 1 回ごとに鍵が変更されるため，暗号化を複数回試行する必要がある攻撃を阻止できる．(b) アルゴリズムレベルの対策としては，検算により誤りを検知し，出力を停止する手法 [28] や，秘密情報と出力結果の誤りのパターンを無関係にする対策 [29] などが挙げられる．(c) 回路レベルでの対策としては，クロック周波数または電源電圧の異常を検知する回路 [30] が挙げられる．この中で，入力が正規のものかどうか調べる対策は，入力を操作する攻撃にのみ有効である．検算型の対策と，アルゴリズムの特性を利用するチェックは，入力を操作する攻撃には無効であるが，暗号処理の途中に注入された故障を検出できる．共通鍵暗号では通常入力ではなく暗号処理の途中に故障を注入することから，共通鍵暗号における対策としては一般的である．しかし，近年，検算型の対策を無効化できる故障感度解析の可能性が指摘されている．故障感度解析では，暗号モジュールの出力から故障感度を測定するが，誤りが生じたかどうかという情報のみで測定できるため誤った暗号文そのものは不要である．このため検算型対策を無効化できる．

ソフトウェア実装における対策としては，上記の検算型対策の他に，命令を複製して故障が注入されることを防ぐ対策 [19] がある．この対策では，2.4.1 節で述べた命令スキップ攻撃を想定しており，ある 1 個の命令がスキップされても，複製された命令が実行されることにより故障が注入されることを防ぐ．しかし，ソフトウェアによる対策では，1 回の故障注入のみ考慮するものが多かった．複数回の故障注入を想定した場合，暗号処理と対策コードの両方に故障注入して，暗号処理に誤りを生じさせた上で対策そのものをスキップし，誤った暗号文を得る攻撃が考えられることから，プログラムフローへの攻撃を防ぐ一般的な手法は提案されていない．

2.4.5 セットアップ時間違反による故障注入

2.4.3 節で述べた故障注入手法の中で，特に近年問題となっているのがセットアップ時間違反を用いた故障注入手法である．これを用いると，ハードウェア実装では故障感度解析，組込

みマイコンでは多重故障注入攻撃がそれぞれ可能となる．

セットアップ時間違反 (Setup time violation) は，同期式論理回路において，組み合わせ回路の遅延時間がクロック信号の間隔に対して長いことを意味する．セットアップ時間は，記憶素子が入力を保持するために必要な，入力を保持すべき最小の時間である [31]．図 2.4(a) に同期式論理回路の概要を示す．ここで，Clock はクロック信号，F1, F2 は D フリップフロップ，Q1 は F1 の出力，D2 は F2 の入力である．また，各時間の定義は，t_{cq} がクロック信号の立ち上がりから Q にデータが出力されるまでの時間，t_c が組み合わせ回路の遅延時間，t_{setup} がフリップフロップ F2 のセットアップ時間である．

同期式論理回路では，タイミング制約として，回路中のある記憶素子から次の記憶素子にデータが伝搬する時間よりクロック信号の周期 T が大きくなければならない．図 2.4(b) に，式 (2.17) に示すタイミング制約を満たす場合の各時間の関係を示す．

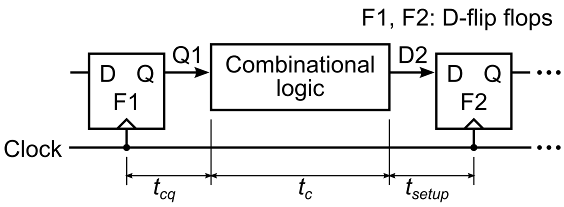
$$T \geq t_{cq} + t_c + t_{setup} \tag{2.17}$$

一方, タイミング制約を満たさない場合には, 以下の 2 つの場合が考えられる．まず, 図 2.4(c) のように，十分なセットアップ時間が確保されない場合である．次に，図 2.4(d) のように，$t_{cq} + t_c$ が T より大きい場合である．この場合にはセットアップ時間は定義できないが，セットアップ時間違反と同様に誤った値が F2 に格納される．

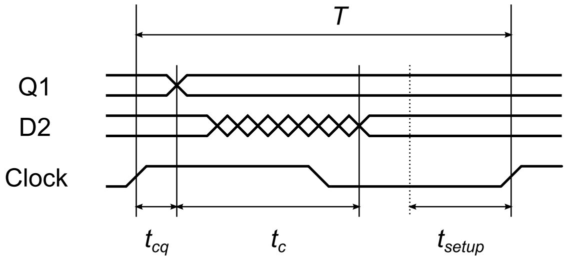
上記の場合のうち，図 2.4(c) のみをセットアップ時間違反と呼ぶ場合と，図 2.4(c) および図 2.4(d) の両者をセットアップ時間違反と呼ぶ場合 [30] があるが，どちらの場合でも故障が発生することから，本論文では両者をセットアップ時間違反と呼ぶ．

2.5　結び

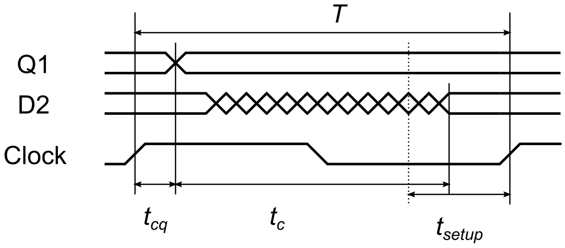
本章では，暗号とその実装に関する基礎的考察を行った．まず，暗号技術全般について概要を述べ，代表的な暗号アルゴリズムである AES と RSA 暗号を概説した．次に，暗号機能を実装した暗号モジュールとそれに対する物理攻撃について説明し，アルゴリズム自体の安全性だけではなく実装の安全性が重要となることを述べた．その後，本論文で主に取り扱う故障注入攻撃について，概要，分類，対策法の順に述べた．特にセットアップ時間違反を用いた故障



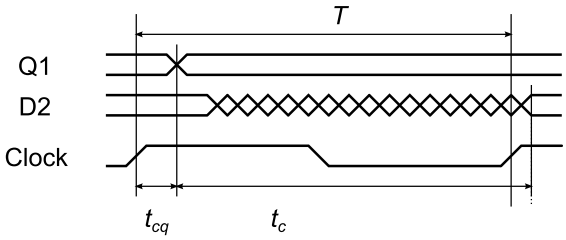
(a) 同期式論理回路



(b) タイミング制約を満たす場合



(c) タイミング制約を満たさない場合



(d) セットアップ時間が定義できない場合

図 2.4　同期式論理回路のタイミング図

注入が問題となっていることから，セットアップ時間違反による故障注入の原理を述べた.

第 3 章

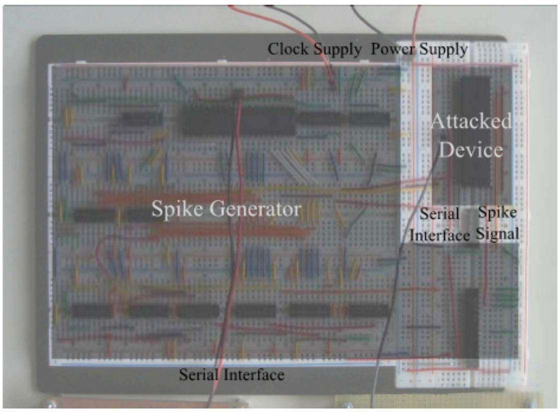
故障注入攻撃実験システムの構築

3.1 はじめに

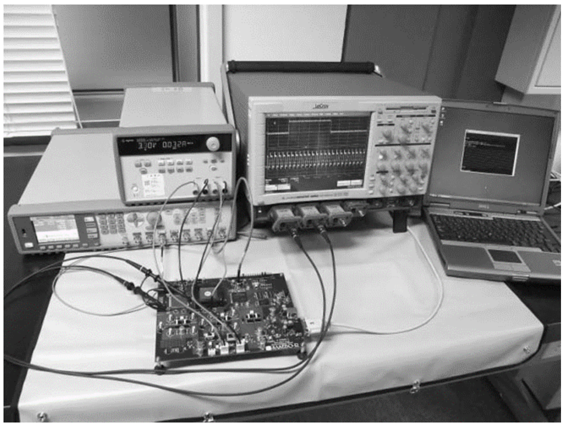
本章では，セットアップ時間違反を利用する故障注入攻撃を想定した安全性評価を行うための実験システムを開発する．2.4.5 節で述べたように，セットアップ時間違反を用いる故障注入の中で最も高い制御性を有するクロックグリッチにより故障注入を行う．本章では，まず従来のクロックグリッチ生成手法の問題点を述べ，それを解決する新たな生成手法を提案する．また，提案システムの評価として，SASEBO–G および SASEBO-W 上にクロックグリッチ発生回路を実装し，生成したグリッチの時間分解能や時間的なゆらぎ (ジッタ) の測定を通し，セットアップ時間違反を用いた故障注入攻撃に対する安全性評価を行うために十分な分解能と再現性を有することを示す.

3.2 関連研究

ここでは，従来の故障注入実験環境とその問題点を述べ，その後サイドチャネル攻撃の分野における統一的な実験環境を整備する試みと，故障注入攻撃の評価への応用について述べる.



(a) 文献 [21] における実験環境 (文献 [21] より引用)



(b) 文献 [22] における実験環境 (文献 [22] より引用)

図 3.1　従来の実験環境

3.2.1　従来の故障注入実験環境

従来，故障注入攻撃の実験では，独自に故障注入のためのシステムを構築した上で実験が行われてきた．初期の実験では，正確な故障注入が不要な攻撃について実験が行われることが多かった．文献 [21] では，RSA 暗号を実行する組込みマイコンにおいて，ある 1 回の自乗剰余算に故障を注入する攻撃が行われている．RSA 暗号における自乗剰余算は数 ms のオーダーの時間を要することから，この演算に故障を注入するには数 ms 程度のタイミング制御で十分であった．実験環境としては図 3.1(a) に示す電源電圧を一時的に低下させる装置が用いられている．一時的な回路の実験で用いられるブレッドボード上に実験システムが構築されていることから再現は難しい．一方，近年では高精度なタイミング制御を要する故障注入攻撃の実験が行われている．文献 [22] で用いられた実験装置を図 3.1(b) に示す．信号発生器を用いて生成したクロック信号を合成することでクロックグリッチを生成し，故障を発生させている．この論文では，128 ビット AES において，クロックグリッチの幅（最初の立ち上がりエッジから 2 番目の立ち上がりエッジまでの時間）を変化させた時の，誤りが発生するビットの位置と数を調べている．この論文では，AES ハードウェアで実行される特定のラウンドへ故障を注入する必要があることから，故障を注入するタイミングをクロックサイクル単位で制御する必要があった．また，クロックグリッチの幅により誤りが生じるビット数が変化することから，

グリッチの幅を数百 ps 単位で制御が必要であった．また，正確なタイミングの故障注入を要する攻撃手法である故障感度解析 [18] においても，[22] と同じ構成の実験環境が用いられている．しかしながら，故障感度解析では，クロックグリッチのジッタがノイズとして作用することにより，ジッタのある実験環境ではより多くの平文を必要とすることが知られている [32]．従って，セットアップ時間違反を用いた攻撃の安全性評価を行うには，統一的な実験環境を実現する必要がある．セットアップ時間違反により注入される故障の強度 (通常の動作条件から離れる度合い) は，信号発生器の種類や，信号発生器から故障注入回路までの配線の長さ等により左右されることから，再現実験を行うにはこれらの条件を同じにする必要がある．別々のハードウェアを用いた実験環境で上記の条件を揃えるのは困難である．

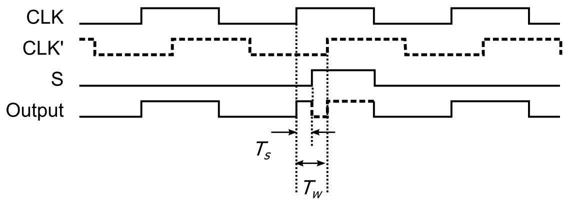
また，故障注入攻撃の実験に特化した市販の実験環境として，Riscure 社の VC Glitcher[33] がある．VC Glitcher は差分故障解析の実験に用いることを想定したプログラマブルなクロックグリッチ発生器であり，500M サンプル毎秒の分解能で波形を生成できる．この数値はグリッチ幅にすると 2 ns の分解能となる．しかしながら，故障感度解析では数十 ps のオーダーの分解能が必要であることから，故障感度解析には利用できないという問題がある．

3.2.2 サイドチャネル攻撃の分野における実験環境標準化の試み

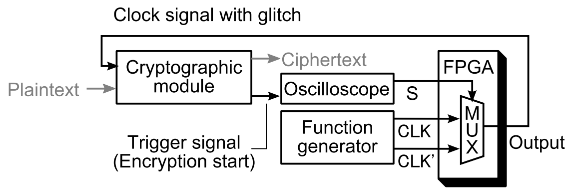
サイドチャネル攻撃の分野では，サイドチャネル攻撃標準評価ボード (Side-channel Attack Standard Evaluation Board: SASEBO) と呼ばれる実験ボードが広く用いられている [4]．現在 6 種類のボードが開発されており，表 3.1 に示すように，対象とする暗号モジュールに合わせてボードを選択することで，あらゆる実装を想定した実験が可能である．SASEBO はサイドチャネル情報である暗号モジュールの消費電流や暗号モジュールから発生する磁界を高精度に測定するため，ノイズを低減する設計となっている．暗号処理用と制御用の 2 つの LSI を備えており，制御用 LSI は主に PC との通信に用いられ，暗号処理用 LSI は暗号処理のみを行う．これにより，暗号処理と無関係な雑音成分を排除し，暗号処理に起因する電流のみを測定することが可能である．制御用 LSI には，全ての SASEBO において FPGA が用いられており，この FPGA に回路を追加することにより，機能を追加できる．

表 3.1 SASEBO ボードの仕様

名称	暗号 LSI の種類
SASEBO–G	FPGA (Xilinx XC2VP7)
SASEBO–R, RII	実験者が用意する ASIC (着脱可能)
SASEBO–B	FPGA (Altera EP2S30)
SASEBO–GII	FPGA (Xilinx XC5VLX30)
SASEBO–W	実験者が用意する IC カード (着脱可能)



(a) タイミング図



(b) ブロック図

図 3.2 文献 [22] におけるグリッチ生成手法

3.2.3 従来のクロックグリッチ生成手法

文献 [22] で用いられている従来のクロックグリッチ生成手法を図 3.2 に示す．図 3.2(a) に示すように，位相の異なる 2 つのクロック信号 CLK および CLK’ を，グリッチを発生させるタイミング S を用いて切り替えることにより，グリッチを含んだクロック信号を生成する．これを実現する回路として，図 3.2(b) が用いられている．信号生成器およびオシロスコープの 2 つの外部機器を用いて，信号 CLK, CLK’ および S を生成する．まず，暗号モジュールはオシロスコープへ暗号化開始の信号を出力する．この信号をトリガとして，オシロスコープのタイムアウトトリガ機能により，トリガ入力から一定時間経過した後に立ち上がる信号である信号 S を FPGA へ出力する．出力は信号 S が 0 のとき CLK，S が 1 のとき CLK’ となる．これにより，切り替えを行ったタイミングで信号が立ち下がり，クロックグリッチが生成される．

従来，クロックグリッチ生成回路を実現するには，外部トリガ出力機能を持つオシロスコー

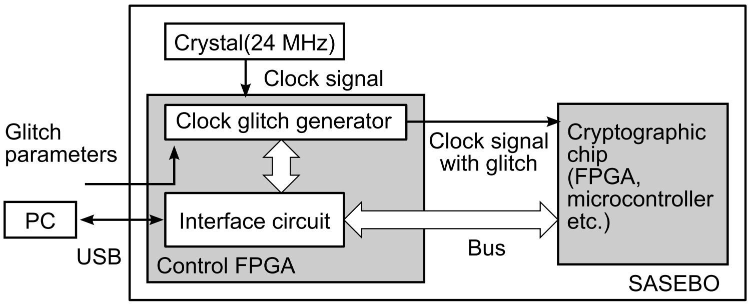


図 3.3　提案システムの概要

プと，2 系統以上の出力系統を持つ信号発生器が必要であった．これを複数回の故障注入へ拡張することを考えると，信号発生器の出力が $2n$ 系統必要であることから，多重故障注入攻撃はあまり現実的でないと見做されてきた．そこで本論文では，FPGA 上に実現できるクロックグリッチ発生回路を提案する．信号発生器の機能を持つ回路ブロックは現在市販されている FPGA において一般的なものとなっていることから，これを用いることで信号発生器を不要とできる．これにより，既に普及しているサイドチャネル攻撃標準評価ボード SASEBO 上に実現でき，多重故障注入の実験も容易に実行可能となる．

3.3　クロックグリッチ生成回路の実装

3.3.1　提案するシステムの概要

提案システムのブロック図を図 3.3 に示す．ハードウェアとしては SASEBO と PC から構成され，SASEBO 上の制御 FPGA にクロックグリッチ発生回路が搭載される．クロック信号は SASEBO 上に搭載されている 24MHz の水晶発振器で生成され，制御 FPGA に供給する．クロックグリッチ発生回路はこのクロック信号を元にグリッチを含むクロック信号を生成する．本論文では，SASEBO–G と SASEBO–W を対象としてクロックグリッチ生成回路を実装する．

生成するクロックグリッチの模式図を図 3.4 に示す．図 3.4 において，T_s は最初の立ち上がりエッジと立ち下がりエッジの間の幅，T_w は最初と 2 番目の立ち上がりエッジの幅であり，

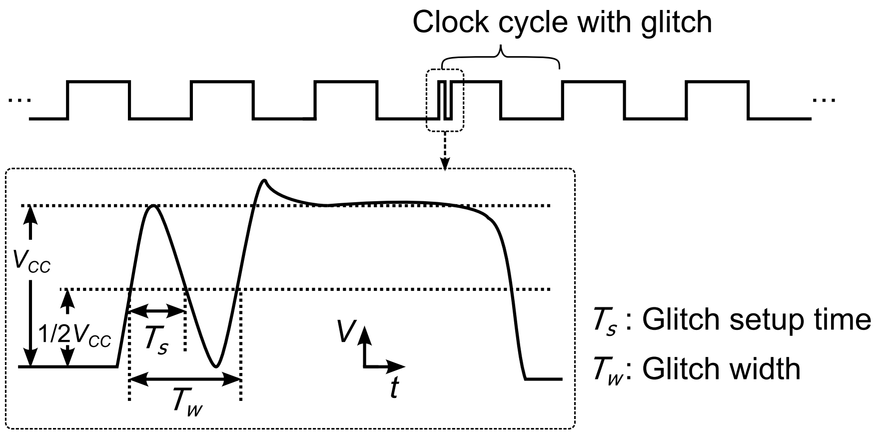


図 3.4　グリッチのパラメータ

本論文ではそれぞれグリッチ整定時間，グリッチ幅とよぶ．

ここで，本論文で実装するオンチップ故障注入回路の要件を定める．本論文では，現在までに提案されている故障利用攻撃の中で最も高い分解能を必要とする故障感度解析に対応するための条件を考える．故障感度解析の実験では，グリッチ幅の分解能 $\varDelta$ が 0.035 ns 以下 [18]，グリッチ幅の標準偏差 σ が 0.07 ns 以下である必要がある [32].

3.3.2　グリッチ生成回路の実装

単一の FPGA でクロックグリッチを生成する基本的な回路を図 3.2(b) に示す．3.2.3 節で述べた従来の実験環境においては，信号発生器とオシロスコープを用いてクロック信号のタイミングを制御していたが，この機能を FPGA 上の DLL (Delay Locked Loop) により実現した回路である．DLL に相当する機能は現在の FPGA では広く搭載されており (Xilinx 社では DCM，Altera 社では PLL が利用できる)，どのベンダーの FPGA においても本回路の実装は容易である．図 3.5 において，DLL–A は信号発生器，DLL–B およびカウンタはオシロスコープに対応する．DLL はクロック信号に遅延（位相シフト）を与えることができるモジュールであり，遅延時間は DLL の入力 θ_w および θ_s で変化させることができる．本回路では，カウンタにより一定時間を作り出し，DLL–B により生成された遅延のあるクロック信号との論理積を取ることにより，信号 S を生成する．

しかしながら，図 3.5 の回路には，DLL の仕様により出力出力周波数の範囲が制限されると

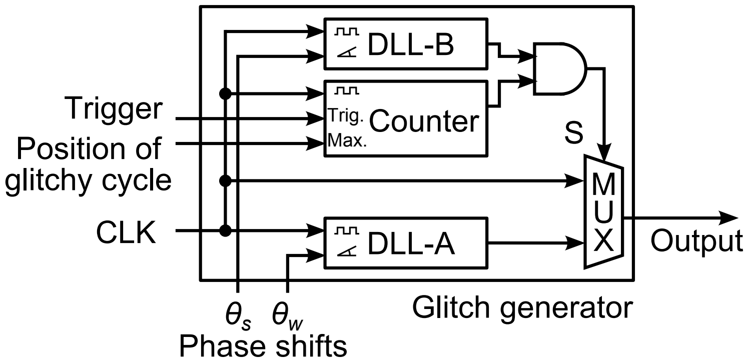


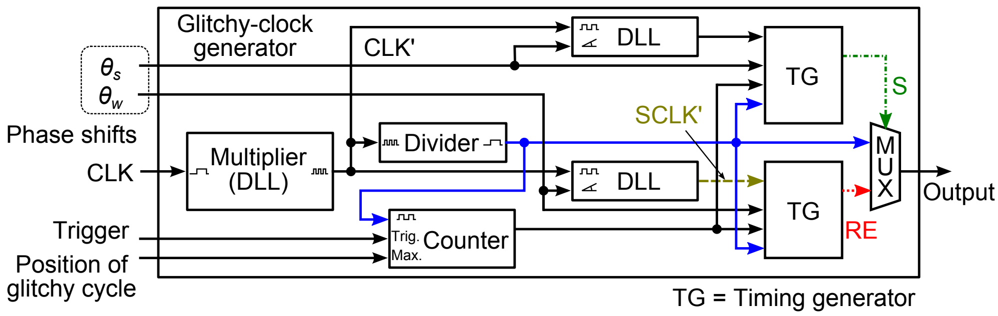
図 3.5　基本的なグリッチ生成回路

いう問題がある．本回路では DLL により生成されたクロック波形の一部を直接出力信号へ反映するため，出力信号の最低周波数は DLL の最低周波数に制限される．また，SASEBO–G に搭載されている FPGA (Virtex-II Pro) では，DLL における位相シフトの分解能が出力周波数により変化するため，出力周波数によっては 3.3.1 節で示した要件である分解能 0.035 ns 以下を達成することができないという問題もある．

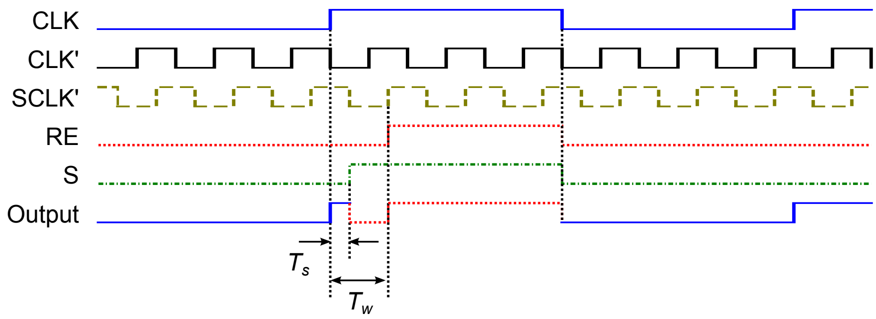
そこで，本論文では，分解能を確保しつつ，任意の出力周波数を設定可能なグリッチ生成手法を提案する．提案回路では，DLL の出力信号をそのまま使用するのではなく，クロックグリッチのタイミングの制御のみに用いることにより，任意の周波数のクロック信号にグリッチを重畳させることを可能とする．

図 3.6(a) に提案回路のブロック図を示す．本回路では DLL の分解能を確保するため，入力クロック信号を逓倍したものを DLL の入力とする．DLL の出力は タイミング生成器 (Timing generator: TG) へ入力され，TG の出力が信号 RE および 切り替え信号 S となる．図 3.6(b) に提案手法のタイミング図を示す．入力クロック信号 CLK を逓倍したクロック信号を CLK’ とする．まず，DLL を用いて CLK’ の位相をシフトする．このクロックの立ち上がりエッジのタイミングを用いて，出力信号における 2 番目の立ち上がりエッジに相当する信号 RE を生成する．出力信号は，通常のクロック信号と RE を切り替えることにより生成する．

図 3.7 にタイミング生成器内部のタイミング図を示す．TG はクロック信号，グリッチ生成のトリガ信号，DLL の出力，カウンタの最大値を入力として取り，DLL の出力信号と同じタ



(a) ブロック図



(b) タイミング図

図 3.6　提案するクロックグリッチ生成手法

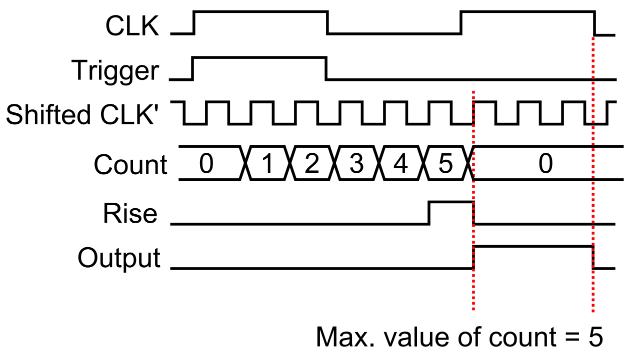


図 3.7　タイミング生成器内部のタイミングチャート

イミングで立ち上がる信号を生成する．TG は DLL のクロック信号に同期するカウンタを持ち，カウンタが最大値に達すると出力信号が立ち上がる．カウンタの最大値はパラメータ θ_s の上位ビットにより決定される．これにより，カウンタの最大値によりタイミングの粗い調整，DLL における位相シフト量により細かい調整が可能となる．

クロックグリッチ生成回路において，ジッタが発生するのは主に DLL 内部である．従って

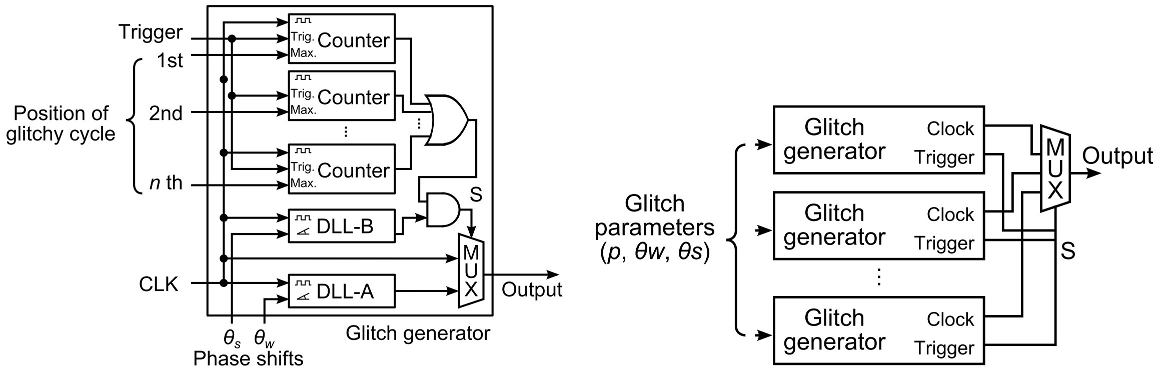
信号が通過する DLL の個数が少ないほどジッタも少なくなる．提案回路では，一度逓倍したクロックを再び分周し，出力信号のベースとなる信号を得る．これは DLL が直列に接続されることを避け，グリッチ整定時間 T_w のジッタの増加を抑えるためである．ここで，入力したクロックをそのまま用いた場合を考えると，T_w のジッタは DLL2 個分となる．一方，提案回路の構成では，逓倍されたクロックが位相の基準となるため，T_w のジッタは DLL 1 個分である．

逓倍比の決定方法を以下に示す．まず，DLL の入力周波数 f_{in} を決定する．SASEBO-G に搭載されている FPGA (Virtex-II Pro) では，DLL の位相シフトの分解能は出力クロック周期の 1/256 または 遅延線の 1 段の遅延 (約 30 ps) のうち大きい方となる．従って，分解能を最大化するには，入力周期の 1/256 が 30 ps 以下となるよう，$f_{in} > 130$ MHz とすればよい．本論文では，入力クロック周波数が 24 MHz のため，逓倍比を 6 とし，$f_{in} = 144$MHz とした．

3.3.3　複数回の故障注入への対応

3.3.2 節で述べたクロックグリッチ生成回路では，一度クロックグリッチを生成してからもう一度生成するには回路のリセットを要するため，短時間のうちに連続して複数回クロックグリッチを生成することはできない．図 3.8 に，多重故障注入に対応したクロックグリッチ発生回路を示す．全てのクロックグリッチを同じパラメータ（整定時間と幅）で生成する場合，図 3.8(a) に示すブロック図のように，クロックグリッチ生成回路に含まれるカウンタを故障の回数分用意するのみで生成できる．いずれかのカウンタの出力が 1 になると，クロックグリッチが生成される．

さらに，一連のクロックグリッチを異なるパラメータで生成する場合，図 3.5 に示した回路を複数個配置し，それらを切り替えることにより複数回のクロックグリッチを発生させることができる．図 3.8(b) に上記機能を実現する回路のブロック図を示す．DLL によりクロック信号の位相を変化させるには，データシートで定められた時間（数千クロックサイクル程度）を要することから，短時間の間に異なるパラメータを持ったクロックグリッチを生成する場合，回数だけ DLL を搭載する必要がある．この場合，クロックグリッチを連続で



(a) 同じパラメータのクロックグリッチを複数回生成する場合

(b) 異なるパラメータのクロックグリッチを生成する場合

図 3.8　多重故障注入に対応したクロックグリッチ発生回路

表 3.2　実験条件

オシロスコープ	Tektronix DPO7254 (4GHz, 40GSa/s)	
クロック周波数	24 MHz	
	SASEBO-G	SASEBO-W
FPGA	Xilinx XC2VP30	Xilinx XC6SLX150
FPGA の I/O 電源電圧	3.3V	2.5V
FPGA のコア電源電圧	1.6V	1.2V

発生させることができる回数は FPGA に搭載された DLL の個数により決まる．DLL の個数は，SASEBO-G に搭載されている FPGA(Xilinx XC2VP30) では 8 個，SASEBO-W の FPGA(Xilinx XC6SLX150) では 18 個であり，それぞれ 4 回，9 回までの故障注入に対応する．

3.4　クロックグリッチ発生回路の評価

提案したクロックグリッチ発生回路を SASEBO 上に実装し，生成した波形の分解能および再現性を評価する．表 3.2 に実験条件を示す．

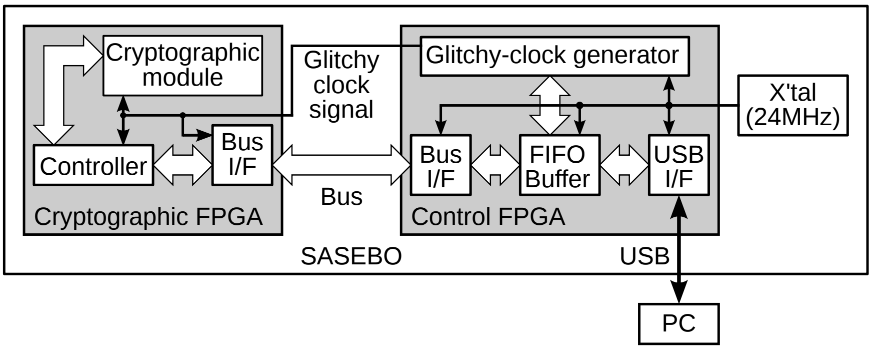


図 3.9　SASEBO–G 上に実装したオンチップ故障注入回路

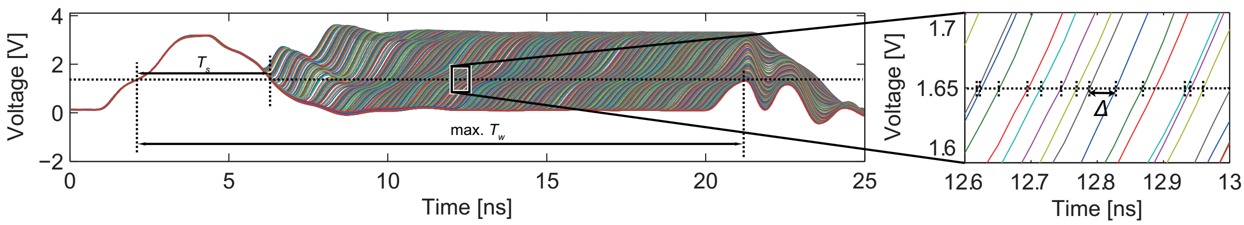


図 3.10　異なるグリッチ幅の設定値におけるオンチップ故障注入回路の出力波形

図 3.9 は SASEBO–G に実装されたオンチップ故障注入回路のブロック図である．提案するオンチップ故障注入回路は SASEBO 上の制御用 FPGA (XC2VP30) へ実装される．故障注入回路の出力はターゲットとなる暗号用 FPGA (XC2VP7) に供給される．位相シフトのパラメータ θ_s, θ_w および グリッチの生成タイミングを決定するカウンタの値は通信インターフェイス (USB インターフェイス，FIFO) を通して PC から制御される．なお，SASEBO-W では暗号モジュール（スマートカード）は搭載せずに実験を行った．

3.4.1　グリッチ幅の分解能

図 3.10 に，T_s を 4.12 ns に固定し，T_w を変化させたときの提案回路の出力波形を示す．提案回路により生成された信号にはランダムな時間的誤差 (ジッタ) が含まれるため，波形を 10 回取得し，その電圧の平均値を図 3.10 に表示している．オシロスコープを用いて，T_w を 4.9–17.9 ns まで変化させたときのグリッチの波形を観測した．図 3.10 は異なるグリッチ整定時間の波形を重畳させて表示している．図 3.10 中の拡大図より，T_w は平均して 0.026 ns

の分解能で変化させることができることが分かる．この値は $\Delta < 0.035$ ns の要件を満たしており，提案回路により要件を満たす回路が実装できることを示した．なお，この値は FPGA に搭載されている遅延素子の遅延により決まる値である．SASEBO–W についても同様に実験を行い，Δ は平均して 0.0232 ns となり，要件を満たしていることを確認した．

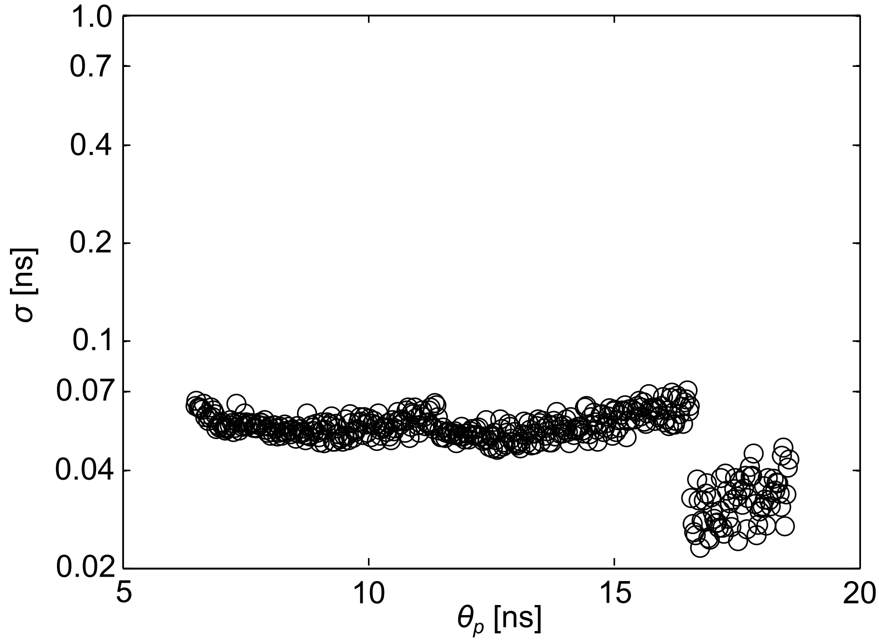
3.4.2　グリッチ幅の再現性の評価

提案回路により生成されるクロック信号の再現性を評価する．ここでは，再現性の指標として，グリッチ幅設定値 θ_w を固定した上でグリッチ幅 T_w を複数回測定したときの標準偏差 σ を用いる．ここではグリッチ整定時間を $\theta_s = 4.12$ ns に固定し，各グリッチ幅の設定値において T_w を 10000 回測定した．図 3.11 に，θ_w を変化させたときの T_w の標準偏差 σ の変化を示す．SASEBO-G および SASEBO-W の両方の実装において，σ は概ね 0.050 ns 以下に収まっている．この値は 3.4.2 節で示した要件である 0.070 ns 以下を満たしており，安全性評価実験に十分な再現性を有していると言える．

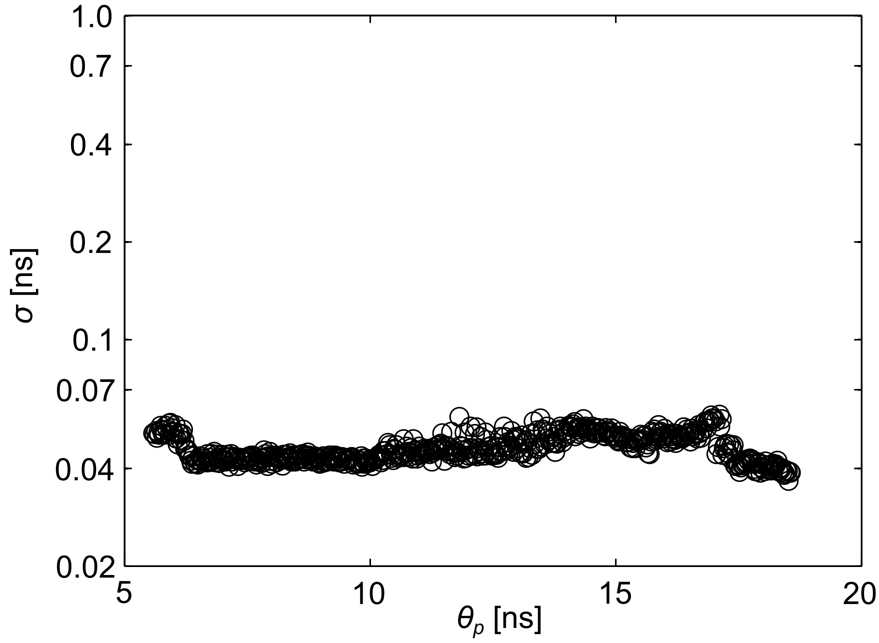
3.4.3　提案回路による故障注入実験

提案回路を用いて，FPGA に実装された 32 ビット加算回路への故障注入実験を行い，提案回路により実際にセットアップ時間違反による故障注入が可能であることを示す．図 3.12 に加算回路のブロック図を示す．ここで，加算器の入力はそれぞれ $a = (a_{31}a_{30}...a_1a_0)_2$，$b = (b_{31}b_{30}...b_1b_0)_2$ とし，加算器間のキャリーは $c = (c_{30}c_{29}...c_1c_0)_2$，D フリップフロップの出力は $d = (d_{31}d_{30}...d_1d_0)_2$ とする．加算器の出力に D フリップフロップが接続されており，D フリップフロップはクロック信号の立ち上がりエッジで加算器の出力を取り込む．加算器にはキャリーの伝搬遅延が存在するため，加算器での演算が終了する以前にクロック信号の立ち上がりがあった場合，D フリップフロップには誤った演算結果が格納される．

実験方法を以下に示す．まず, 加算器へ $a = (FFFFFFFF)_{16}, b = (00000000)_{16}$ を入力する．この操作により, 前ビットのキャリーが 0 となる．次に, 加算器へ $a = (FFFFFFFF)_{16}$，$b = (00000001)_{16}$ を入力する．この条件ではすべての桁でキャリーが 1 となり，加算器の出



(a) SASEBO-G



(b) SASEBO-W

図 3.11　$\theta_s = 4.12$ ns における標準偏差 σ の変化

力結果は下位ビットから順に確定していくため，上位のビットほど結果が確定する時刻が遅くなると予想される．従って，初期状態としてクロック周期を本回路の最低周期より長く設定し，徐々に短くした場合，上位ビットから順に誤りが開始し，その後誤りの開始位置が下位

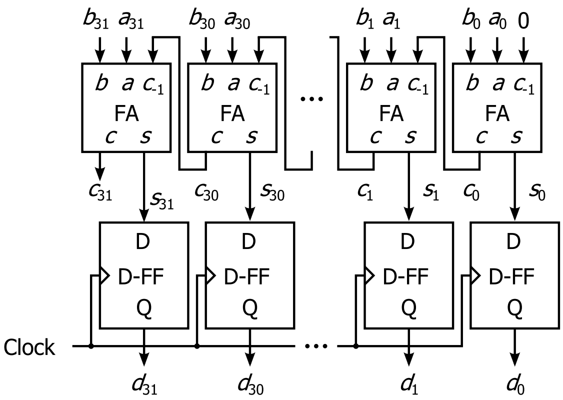


図 3.12　故障注入実験に用いる加算回路

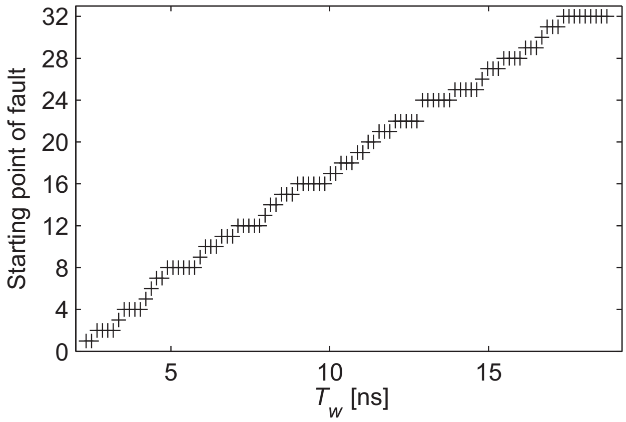


図 3.13　グリッチ幅 T_w と誤り開始点の関係

ビットへと移動すると予想される．

実験方法を以下に示す．提案回路および加算回路をそれぞれ SASEBO–G の制御用 FPGA および暗号用 FPGA に実装し，D フリップフロップのクロック信号として提案回路の出力信号を与えた．グリッチ整定時間は $T_s = 2.5$ ns に固定し，グリッチ周期 T_w を 2.3–18.7 ns の間で変化させながら，フリップフロップの出力を観測した．この結果，予想通り，あるクロック周期から誤りが開始され，それ以降クロック周期を短くすると，誤りが開始されるビットが上位ビットから下位ビットへ移動するという結果が得られた．ここでは，出力結果 d において誤りが開始されるビットを誤り開始点と呼ぶ．

図 3.13 に，グリッチ幅 T_w と誤り開始点の関係を示す．誤りは上位ビットから始まり，T_w が小さくなるに従い下位ビットへ開始点が移動する．これは加算器の入力に基づく予想と一致しており，故障注入が正しく行われていることを示している．以上の結果より，提案回路により FPGA に実装した順序回路に誤りを注入できることを示した．

3.5　結び

本章では，サイドチャネル攻撃標準評価ボード SASEBO 上に搭載できる故障注入攻撃実験システムを構築した．提案システムは単一の FPGA 上に実装可能なクロックグリッチ発生回

路に基づく．本章ではまず，クロック周期の操作による故障注入の原理および従来のグリッチ生成手法を述べた．次に，オンチップ故障注入回路の実装における従来手法の問題点を述べ，従来手法の問題点を解決する新たなグリッチ生成手法を提案し，提案手法を実現する回路構成を示した．さらに，提案手法の評価として，SASEBO-G および SASEBO-W 上に提案回路を実装して生成したクロックグリッチの特性を測定し，本回路が安全性評価実験の要件を満たすことを示した．

第 4 章

故障感度解析への対策技術の開発

4.1　はじめに

本章では，故障感度解析への対策技術を開発する．従来の物理攻撃対策のほとんどは故障感度解析へ脆弱であることが示されており [6]，現在のところ故障感度解析への対策法は知られていない．従来の故障注入攻撃対策は，故障を検知して出力を停止するものや，故障注入を行っても攻撃者に有用な情報が得られないようにアルゴリズムを変更するものが代表的であるが，これらの対策は，故障が発生したかどうかという情報は隠蔽できないため，故障感度解析に対して脆弱である．故障感度解析への対策は，基本的なアイディアは文献 [34] で述べられているものの，対策ハードウェアとして実現されていないのが現状である．そこで本章では，可変遅延素子により生成したタイミングに基づき故障感度を隠蔽する対策を提案する．また，提案対策を施した AES ハードウェアを ASIC 上に試作し，故障感度 (遅延時間) の測定実験を通して提案対策の有効性を示す.

4.2　関連研究

4.2.1　故障感度解析

故障感度解析 (Fault Sensitivity Analysis: FSA) は，サイドチャネル情報の 1 つである故障感度を用いた攻撃である．故障感度とは，故障注入に必要な最小の故障注入の強度を指す．

故障強度は，セットアップ時間違反を用いた故障注入の手段 (クロック周波数の上昇，電源電圧の低下) の程度の大きさを表す．一般に，チップの設計者が指定する動作条件から離れるほど故障が発生する可能性が高いことから，本論文では動作条件から離れる度合いが大きいほど故障強度は大きいとする．クロック周波数であれば高いほど，電源電圧であれば低いほど故障感度は大きい．故障感度解析は，故障感度と入力（平文）の関係から秘密鍵を復元する攻撃であり，従来の故障注入攻撃で用いられてきた誤りを含む暗号文を必要としないため，従来の対策を無効化できる可能性がある．例えば，故障感度は，故障が発生したかどうか（正常動作かエラーか）という情報のみで測定できるため，故障を検知して出力を停止する対策を無効化できる．

故障感度はクロックグリッチによる一時的なクロック周波数の上昇を用いて測定できる [18]．故障感度を測定するには，平文を固定し，クロック周波数を上昇させながら複数回暗号化する．故障感度は最初に誤りが生じたときのクロック周波数で与えられる．攻撃の例として，AES の S-box に対する攻撃が挙げられる．文献 [18] では，PPRM 形式に基づく S-box において，故障感度が入力のハミング重みモデルで表せることを利用した攻撃が行われている．この論文で行われた実験では，50 個の平文から秘密鍵の抽出に成功している．

4.2.2 Collision FSA

故障感度解析は，Correlation Collision Attack [35] と呼ばれる電力解析の手法と組み合わせることにより，Collision FSA [6] として拡張されている．この攻撃は入力と故障感度の関係（故障感度モデル）が明らかではない場合でも適用できる上に，攻撃対象のデバイスからモデルを構築するプロファイリングも必要としない．AES への攻撃では，複数の S-box 間で故障感度と入力の関係が等しいことを利用する．衝突 (Collision) とは，2 つの S-box の入力が等しい (衝突している) ことを意味する．

ここでは，128 ビット AES を例に攻撃の原理を説明する．i をバイト番号 $(0 \leq i \leq 15)$，10 ラウンド目におけるラウンド鍵の i バイト目を K_i^{10}，暗号文の i バイト目を C_i とすると，i バ

イト目の入力 I_i は

$$I_i = InvSbox(C_{SR(i)} \oplus K^{10}_{SR(i)}) \tag{4.1}$$

と表せる．ここで，ShiftRows によりシフトされたバイト番号 $SR(i)$ は式 (4.2) で表せる．

$$SR(i) = \{i - 4(i \bmod 4)\} \bmod 16. \tag{4.2}$$

S-box の故障感度は入力により決まるため，i バイト目の S-box の入力を I_i とすると，i バイト目の故障感度は $F(I_i)$ と表せる．従って，1 バイト目と 2 バイト目の故障感度 F_1, F_2 は以下のように表せる．

$$F_1 = F(InvSbox(C_{SR(1)} \oplus K^{10}_{SR(1)})) \tag{4.3}$$
$$F_2 = F(InvSbox(C_{SR(2)} \oplus K^{10}_{SR(2)})) \tag{4.4}$$

Collision FSA では，平文をランダムに生成して暗号化を行い，全ての暗号文 ($0 \leq C_{SR(1)} \leq 255, 0 \leq C_{SR(2)} \leq 255$) が得られるまで暗号化を繰り返し，上記の暗号文に対する F_1, F_2 の表を作成する．そこで，任意の暗号文 $0 \leq C \leq 255$ に対する F_1, F_2 の値を考えると，それぞれ式 (4.5)，(4.6) となる．

$$F_1 = F(InvSbox(C \oplus K^{10}_{SR(1)})) \tag{4.5}$$
$$F_2 = F(InvSbox(C \oplus K^{10}_{SR(2)})) \tag{4.6}$$

ここで，2 つの部分鍵の差分を $\Delta^{10}_K = K^{10}_{SR(1)} \oplus K^{10}_{SR(2)}$ とすると，F_2 は以下のように表せる．

$$\begin{aligned} F_2 &= F(InvSbox(C \oplus K^{10}_{SR(2)})) \\ &= F(InvSbox(C \oplus K^{10}_{SR(1)} \oplus \Delta^{10}_K)). \end{aligned} \tag{4.7}$$

$F(InvSbox(C \oplus K^{10}_{SR(1)} \oplus \Delta^{10}_K))$ は，F_1 を秘密鍵 Δ^{10}_K の下で並び替えた値である．そこで，予測鍵 $0 \leq \Delta'^{10}_K \leq 255$ の下で $F'_1 = F(InvSbox(C \oplus K^{10}_{SR(1)} \oplus \Delta'^{10}_K))$ を計算し，$F_2 = F'_1$ となったとき，予測鍵が正しいことが分かる．

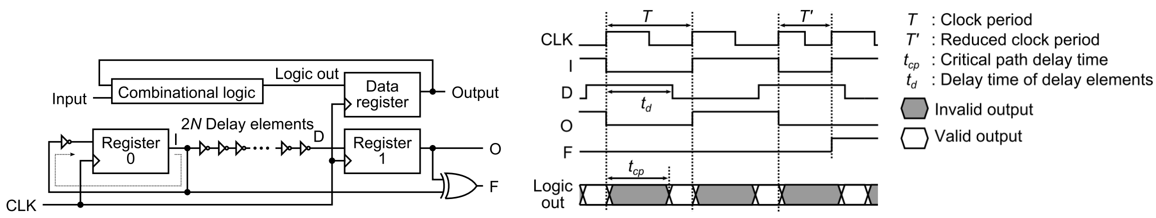
4.2.3 従来の対策に対する故障感度解析

従来のサイドチャネル攻撃および故障注入攻撃に対する対策のほとんどは，FSA に対して脆弱であることが示されている．例えば，DFA に有効な対策として，AES における Simultaneous error detection [36] がある．この手法は，AES の 1 ラウンドを 2 サイクルに分けて実行する．ShiftRows および SubBytes は最初のサイクル，MixColumns および AddRoundKey は 2 番目のラウンドでそれぞれ実行される．しかし，この対策は原理的に故障感度の漏洩を完全に防ぐことはできず，実際にこの手法を用いた AES ハードウェアが FSA により破られている [37]．

ある種のランダムマスキングも FSA により破られる可能性がある．マスクの値が一様乱数の場合，誤った暗号文 C' の出現頻度は一様ではなく，入力に依存する．従って，Collision FSA が適用可能であり，実際に Masked AND Operation (MAO) により対策が施された AES ハードウェアが FSA により破られている [6]．サイドチャネル攻撃と DFA の両方に有効な対策である WDDL も，FSA に対して脆弱である可能性がある [34]．WDDL に対する故障感度解析では，正しい暗号文において値 1 を持つビットから故障感度データを取得する．以上のことから，従来のサイドチャネルおよび故障注入攻撃対策では故障感度を隠蔽またはマスクすることが困難である．

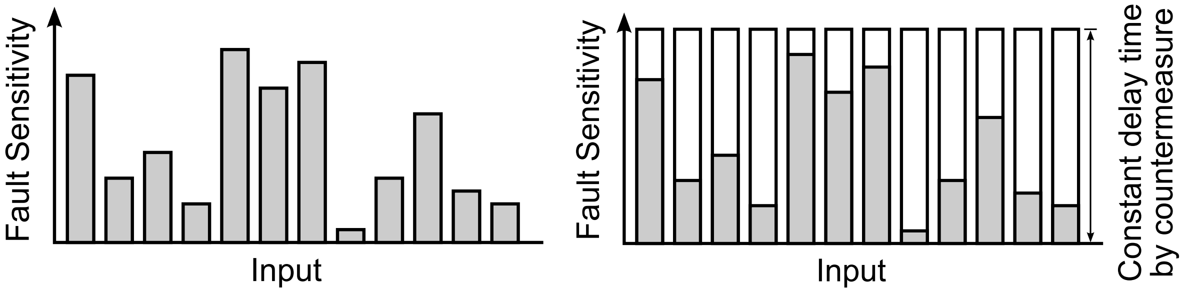
4.2.4 故障感度解析対策の検討

故障感度解析では，セットアップ時間違反を引き起こすことが必要であるため，セットアップ時間違反への対策が有効であると考えられる．文献 [30] では，暗号回路のクリティカルパスより長い遅延回路を用いたセットアップ時間違反検出回路が提案されている．遅延回路は直列接続された遅延素子を用いて構成される．図 4.1 に，文献 [30] のアイディアにより差分故障解析への対策を実装した例を示す．図 4.1(a) に対策回路のブロック図を示す．遅延素子により，クロック信号の立ち上がりから一定時間後に立ち上がる信号を生成し，これを用いて故障注入の有無を判断する．図 4.1(b) にタイミング図を示す．T, T', t_{cp}, t_d はそれぞれクロック周期，



(a) ブロック図　　　　(b) タイミング図

図 4.1　文献 [30] に基づくセットアップ時間違反対策



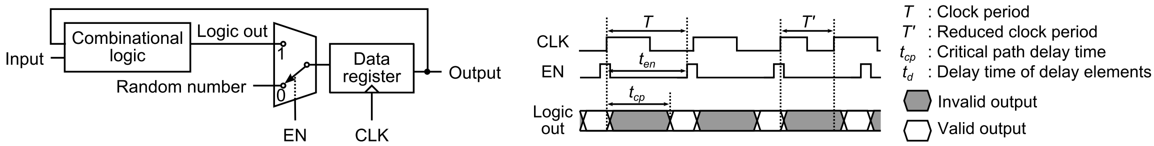
(a) 対策なし　　　　(b) 対策あり

図 4.2　文献 [34] の対策における故障感度の概念図

一時的に短縮されたクロック周期，組み合わせ回路中のクリティカルパスの遅延時間，遅延回路の遅延時間を表す．レジスタ 0 の出力 (I) はクロック信号の立ち上がりで反転し，クロック周期が遅延回路の遅延時間を下回ると，故障検知を表す信号 F が 1 になる．この対策では，信号 F が 1 の時は暗号化結果（誤った暗号文）を出力しないことにより，差分故障解析を防ぐ．

現在までに故障感度解析への対策が実装された例はないが，文献 [34] において故障感度の隠蔽による対策の可能性があることが示唆されている．この論文では，イネーブル信号と呼ばれる信号により組み合わせ回路（暗号回路）の故障感度を隠蔽する対策が提案されている．図 4.2 にこの対策を施したときの故障感度の概念図を示す．対策なしでは図 4.2(a) のように故障感度が入力に依存して変化するのに対し，この対策では図 4.2(b) のように入力にかかわらず常に一定の故障感度が観測されるようにする．ここで，攻撃者から観測される故障感度は対策回路により生成される．

上記の故障感度を持つ回路を実現した例が文献 [34] に示されている．図 4.3(a) にブロック図を示す．組み合わせ回路とレジスタの間にマルチプレクサを挿入した回路構成を持つ．ここ



(a) ブロック図　　　　(b) タイミング図

図 4.3　文献 [34] の対策回路

で EN はイネーブル信号である．マルチプレクサはイネーブル信号により制御され，イネーブル信号が 1 のときのみ出力結果をレジスタへ格納する．イネーブル信号が 0 の場合には乱数が格納されるため，出力は誤りとなり，攻撃者はこの時の故障強度を故障感度として得る．組み合わせ回路の状態が整定するタイミングはイネーブル信号が立ち上がるタイミングより早いため，組み合わせ回路の整定時間 (故障感度) は攻撃者から観測できない．図 4.3(b) に理想的な各信号のタイミングを示す．組み合わせ回路が整定するまでは，EN 信号は 0 であり，整定後に 1 になり，次のクロックエッジで 0 に戻る．EN 信号が 0 のときにはレジスタには乱数が格納される．

なお，イネーブル信号が 0 の場合にクロック信号が立ち上がると乱数が入力される理由を以下に示す．仮に定数が入力される場合，特定のラウンドの出力を定数にする故障を注入できることになる．この場合，例えば AES の 9 ラウンド目の出力を定数にして，10 ラウンド目を実行することにより，攻撃者が知っている状態から 1 ラウンドだけ暗号処理を実行できる．1 ラウンドの暗号化では攪拌が十分でないことから，様々な攻撃が可能となるおそれがある．以上の理由により，上記対策では乱数を用いる．

上記の対策の有効性を保証するためには，イネーブル信号が組み合わせ回路の整定よりも常に後に立ち上がらなければならない．そのため，本対策においてイネーブル信号のタイミングは非常に重要であるが，その具体的な生成方法は文献 [34] には示されていない．イネーブル信号は攻撃者がタイミングを変更できないよう，回路内部で生成しなければならない．

サイドチャネル攻撃の分野では，回路内部で生成した遅延時間に基づく対策の例として，Random Switching Logic [38] がある．この対策では，直列に接続された遅延素子によりイネーブル信号を生成する．Random Switching Logic では，遅延素子の個数は製造前のシミュ

レーションに基づき決定される．一方，故障感度解析への対策を考える際には，攻撃者がクロック周波数だけでなく電源電圧を変化させることも想定しなければならないことから，最適な遅延素子の個数を製造前に決定することは困難である．特に，高度に微細化されたプロセスにおいて，製造後のチップにおける遅延時間を正確に予測することはより困難となる．この問題を解決するため，本章で提案する対策では，製造後に遅延時間を調整可能な可変遅延回路によりイネーブル信号のタイミングを決定する．

4.3 可変遅延回路を用いた故障感度の隠蔽

4.3.1 提案対策の回路構成

提案対策では，4.2.4 節で述べた Li らの対策のコンセプトを回路として実現するため，可変遅延回路を用いてイネーブル信号を生成する．図 4.4 に提案対策を施した暗号ハードウェアの構成を示す．暗号コア，可変遅延回路 (Configurable Delay Block: CDB) を含むイネーブル信号発生器，乱数生成器，ワンタイムメモリ (OTM) からなる．本対策は (i) 製造後の調整を行うための設定フェーズ，(ii) 暗号処理を行う動作フェーズの 2 つのフェーズを持つ．OTM は可変遅延回路の設定値を保持し，設定フェーズの時に可変遅延回路へ値を書き出す．CDB は遅延素子を直列に接続した回路で構成され，製造後に調整可能とする．製造前に，調整可能な遅延時間の範囲および 1 段の分解能をあらかじめ決定する．これらのパラメータは，暗号回路の正確な遅延時間が不明でも，おおよその値から決定できるため，静的タイミング解析 (STA) により得られたおよその遅延時間と，チップの製造プロセスに依存する製造ばらつきの大きさの見積もりに基づき決定する．最終的な遅延回路の設定値は，遅延回路の特性の測定結果および決定される．製造後に，それぞれのパラメータの設定値における実際の遅延時間を測定し，OTM に設定値を書き込んだ後，回路は動作フェーズに移行する．一度動作フェーズに移行すると，以降の OTM への値の書き込みは不可能となることから，攻撃者は設定値を変更することはできない．

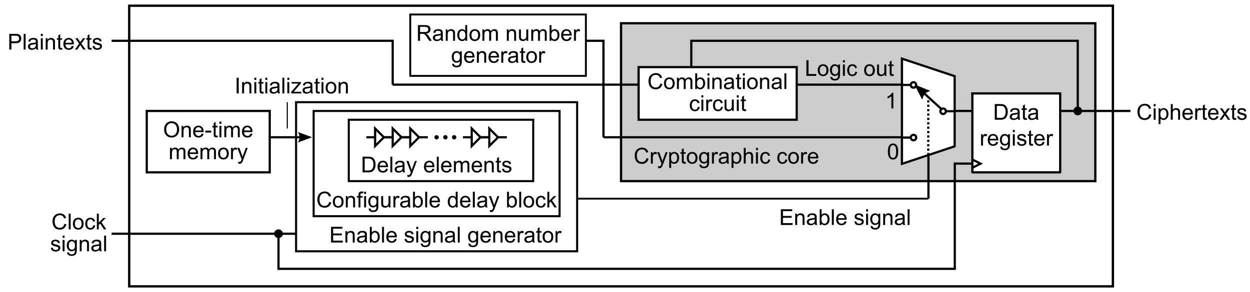


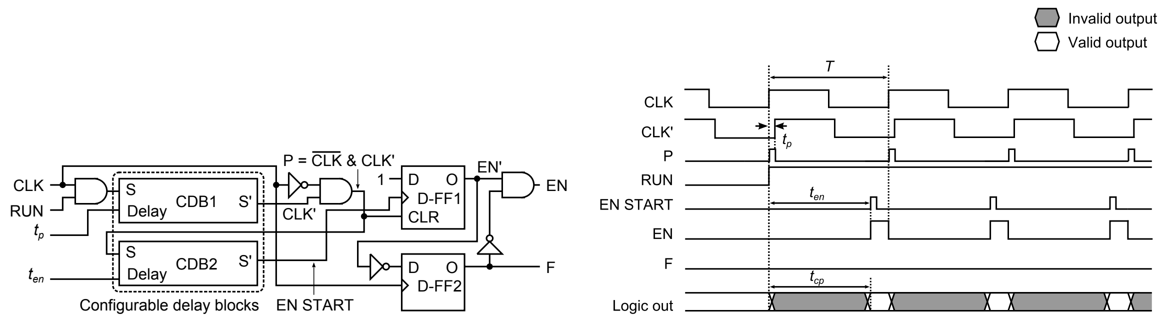
図 4.4　提案対策のブロック図

4.3.2　イネーブル信号の生成方法

図 4.5 に，イネーブル信号の生成方法を示す． 信号 P の幅を t_p ，クロック信号 CLK と EN 信号のそれぞれの立ち上がりエッジ間の時間を t_{en} とする．最初に，CDB1 により CLK の位相をわずかに遅らせた信号 CLK' を生成し，CLK および CLK' から信号 P を生成する．次に，信号 P を CDB2 に通して信号 EN START を生成する． 信号 EN START は，暗号処理が実行中であることを示す信号 RUN が 1 である間だけ生成される．最後に，EN 信号は EN START と信号 P のそれぞれの立ち上がりエッジの間だけ 1 となる信号として生成される．CLK が立ち上がったときに EN 信号が 0 であった場合，攻撃検知信号 F が 1 になり，暗号コアは動作を停止する．EN 信号は信号 F が 0 のときのみ生成されるため，2 サイクル連続して故障が注入された場合でも対策は機能する．EN 信号はクリア付き D フリップフロップにより生成されることから，EN START の立ち上がり時に信号 P が 1 であった場合，EN 信号の立ち上がりが妨げられる．従って，$t_p < t_{en}$ と設定しなければならない． また，t_{en} は組み合わせ回路において起こりうる最大の遅延時間 t_{cp} より大きくなければならない．適切な t_{en} の値を求める方法は 4.3.4 節で述べる．

4.3.3　提案手法による攻撃検知

ここでは，タイミング違反による故障注入のモデルとして，クロック周期が短くなる場合（オーバークロック）と，電源電圧の低下を考える．また，前節で述べた条件 $t_{cp} < t_{en}$ が動作



(a) ブロック図　　　　(b) タイミング図

図 4.5　イネーブル信号生成器

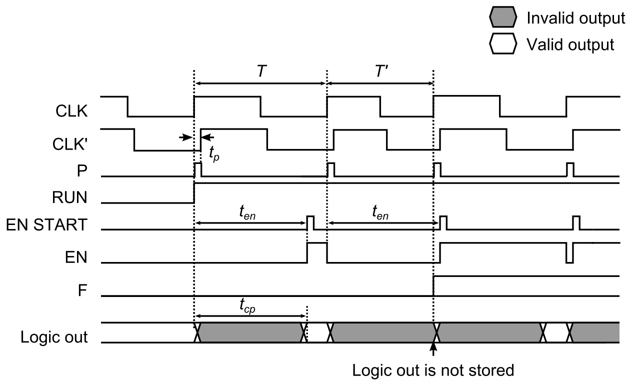
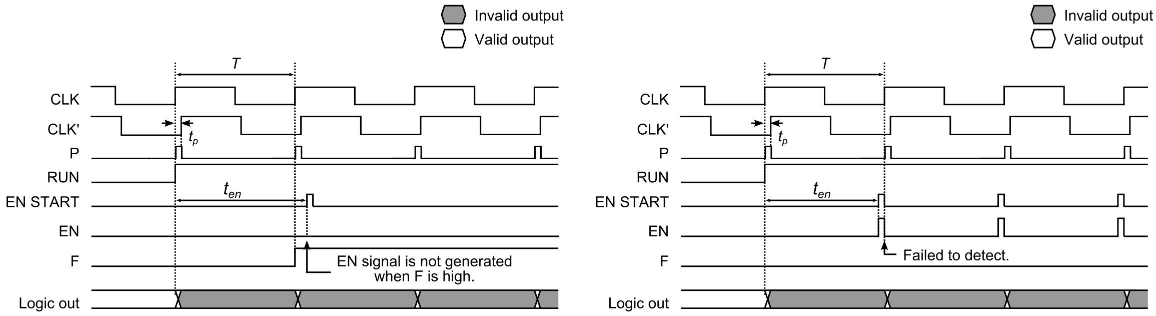


図 4.6　クロック周波数の変化の検出

電圧において満たされていると仮定する． 図 4.6 に，クロック周期が一時的に短くなった場合の各信号のタイミング図を示す．条件 $t_{cp} < t_{en}$ が常に満たされるため，クロック信号の 2 番目の立ち上がりで組み合わせ回路の出力は得られず，対策は正常に動作する．

一方，電源電圧が低下した場合，条件は常に満たされるとは限らない．図 4.7 に，電源電圧が低下した場合の各信号のタイミング図を示す． クロック周波数は一定であるが，電源電圧の低下により t_{cp} および t_{en} の両者が増加する．このとき，$t_{cp} > t_{en}$ となる場合，図 4.7(b) に示すように，整定していない組み合わせ回路の出力が漏洩してしまう．このため，暗号コアが動作する（攻撃に有用な情報が得られる）電源電圧において，常に $t_{cp} \leq t_{en}$ が満たされるよう CDB を設定しなければならない．

図 4.6 および図 4.7 は暗号処理の実行中に故障が注入されている場合を示しているが，本対策では信号 RUN が 1 になる前から故障が継続して注入されている場合でも検知できる．ま



(a) $t_{cp} \leq t_{en}$ の場合　　(b) $t_{cp} > t_{en}$ の場合

図 4.7　電源電圧変化の検出

た，クロック周波数と電源電圧が同時に変化した場合にも対応できる．

4.3.4　V–T 曲線に基づく可変遅延回路の設定

ここでは，電源電圧の変化にかかわらず $t_{en} > t_{cp}$ となるように遅延回路を設定する方法を述べる．図 4.8 に，温度変化がない場合に理想的な設定を行った場合の V–T 曲線を示す．横軸が電源電圧，縦軸が時間（クロック周期および遅延時間）であり，V_{min} および V_{max} はそれぞれ暗号コアが動作する最低電圧，最高電圧である．標準的な CMOS (Complementary Metal Oxide Semiconductor) 回路では，遅延時間は電源電圧にほぼ反比例し，動作する最低電圧付近で急激に増加する．暗号ハードウェアは t_{en} より上の領域では正常動作し，t_{cp} より下の領域では正常に動作しない．t_{en} と t_{cp} の間の領域は動作マージンとして確保されている領域であり，暗号コアは正常に動作するが，提案対策では乱数を出力する．ここで，故障感度解析を防ぐためには，V_{min} から V_{max} までの電圧範囲を考えなければならない．

提案対策では，最大動作周波数が t_{cp} ではなく t_{en} により制限されることから，回路を最大動作周波数で動作させることを想定すると，$t_{en} - t_{cp}$ が提案対策のオーバーヘッドとなる．従って，最適な t_{en} は，$t_{en} > t_{cp}$ を満たす最小の t_{en} である．温度変化についても考慮し，より安全な t_{en} を見つけるには，図 4.9 に示すように，電源電圧，クロック周期，温度変化を考慮した関係を用いて，t_{en} を表す平面が t_{cp} を常に上回るように設定する．

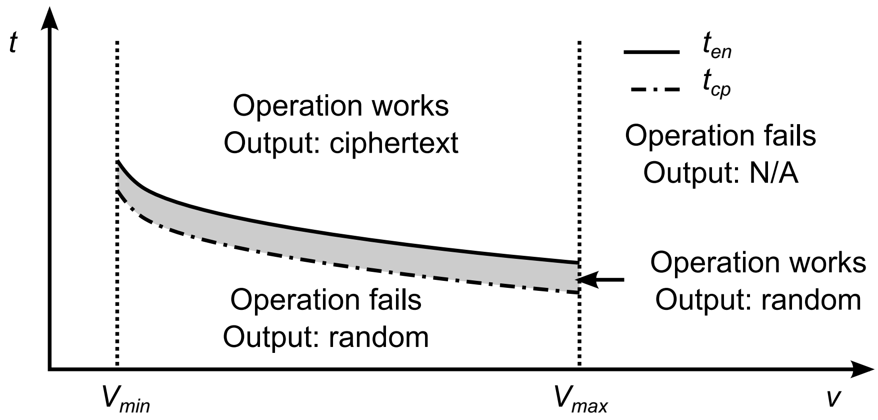


図 4.8　t_{en} と t_{cp} の間の関係の概念図

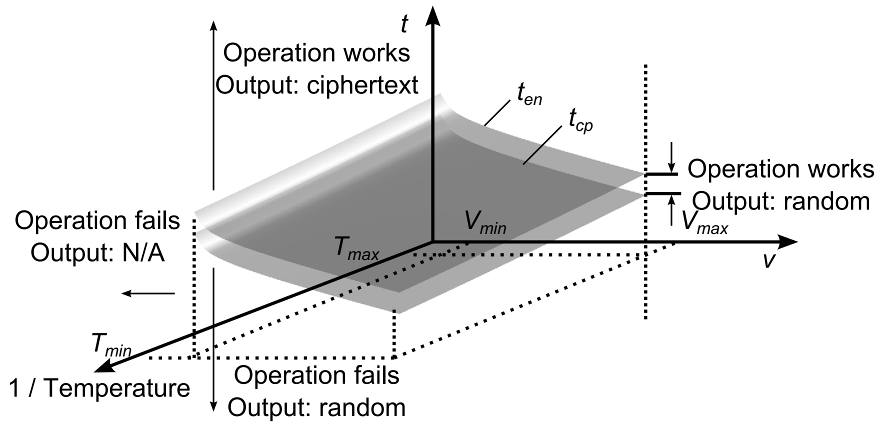


図 4.9　温度変化がある場合の t_{en} と t_{cp}

4.3.5　許容できる最小の遅延時間の見積もり

上記の手法はクリティカルパスの遅延 t_{cp} に基づき遅延回路の遅延時間を設定するが，暗号回路の t_{cp} を知ることは一般に困難である．クリティカルパスは製造時に変化する場合があるため，真のクリティカルパスの遅延を求めるには入力を総当りしなければならない．しかし，暗号回路の入力は一般にビット数が多く総当りが困難である．例えば，128 ビットの AES では，平文 128 ビット，秘密鍵 128 ビットの合計 256 ビットの入力を持つため，総当りは困難である．

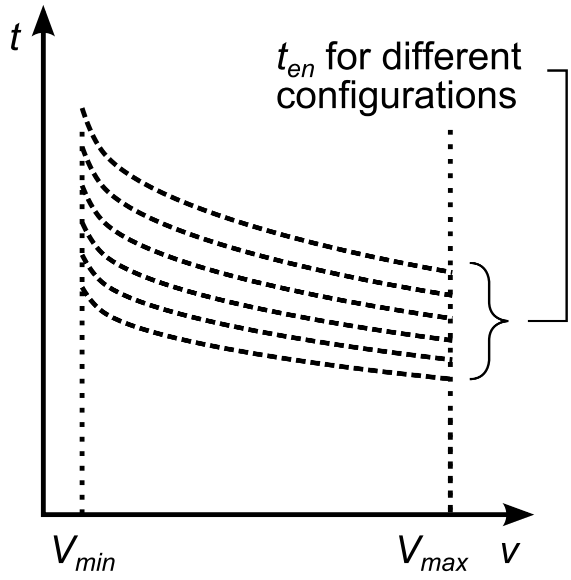
この問題を解決するため，ある程度の故障感度の漏洩を許容し，与えられた条件下で攻撃者

が故障感度解析に必要な故障感度のデータ数を取得できないように t_{en} を設定する統計的な手法を提案する．N_a を故障感度解析が成功するために必要な故障感度のデータ数とする．N_a は，製造したチップへ故障感度解析を行い求める． なお，実際のチップ上で故障感度解析を行わなくとも，1 個の故障感度により解析が成功する場合 ($N_a = 1$) が設計者にとっての最悪値であることは明らかである．N_l を実際の測定で取得した故障感度の数とすると，$N_l < N_a$ のとき故障感度解析による鍵復元は失敗する．N_o を，攻撃者が故障感度解析へ用いることができる測定回数とする．これは攻撃者が許容できる，攻撃に要する時間によって決まる．

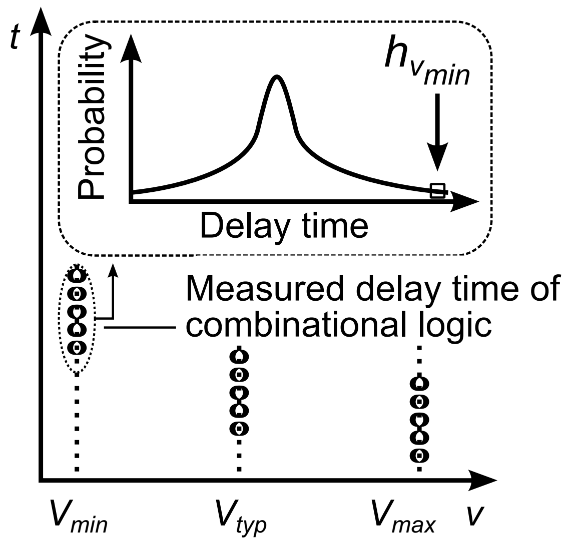
図 4.10 に，t_{en} を設定するための見積もり手法を示す．V_{typ} は暗号回路の標準動作電圧であり，V_{min} および V_{max} はそれぞれ暗号コアが動作する最低電圧，最高電圧である．最初に，図 4.10(a) のように，電源電圧と t_{en} を変化させながら t_{en} を測定する．この時 t_{en} は設定値を変更可能となっており，ここで測定した遅延時間を元に最終的な設定値を決定する．次に，ある温度下で 3 つの電源電圧 $V_{min}, V_{typ}, V_{max}$ において測定した遅延時間を元に確率密度関数を算出する．

図 4.10(b) に，測定した遅延時間と見積った確率密度関数を示す．本対策では，暗号回路の遅延時間が遅延回路の遅延時間を上回るとき，故障感度が漏洩するとする．f を遅延時間の確率密度関数とすると，故障感度の漏洩確率，すなわちある平文に対する故障感度 t がある遅延時間 h を上回る確率は $g(h) = f(t \geq h)$ と表せる．攻撃者は平文と故障感度の関係を知らないことから，ランダムに平文を変えて故障感度を複数回測定し，故障感度を漏洩させることを試みる．ここでは，1 つの平文に対して故障強度を変えながら出力の正誤を判定（故障感度を測定）することを 1 回の試行と呼ぶ．対策の遅延時間を h に設定し，攻撃者が故障感度を N_o 回取得することを試行した場合，得られる故障感度値の個数の期待値 E は $E = N_o g(h)$ と表せる．$\lceil E \rceil < N_a$ のとき，対策は有効に機能すると期待できる．ここで，対策が有効に機能する最小の遅延時間を h_v とすると，$\lceil N_o g(h_v) \rceil = N_a$ となる．

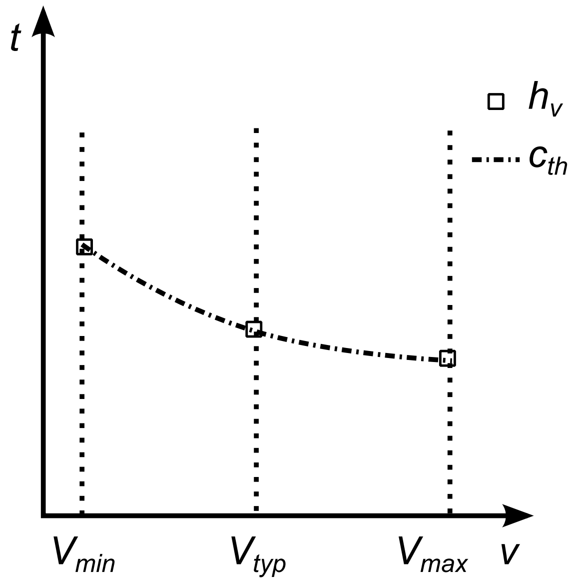
従って，式 $g(h_v) = N_a/N_o$ を h_v について解くことにより，設定値が求められる．電源電圧は $V_{min}, V_{typ}, V_{max}$ とする．すると，図 4.10(c) のように，h_v を通る線 c_{th} を求めることができる．最後に，図 4.10(d) のように，最適な t_{en} は，図 4.10(a) の中で c_{th} を常に上回る線として求められる．結果として，漏洩する故障感度の個数は N_a を下回ることが期待できる．



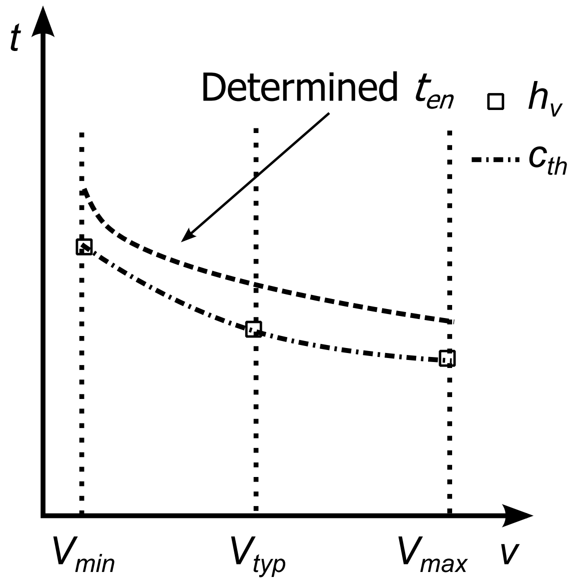
(a) 異なる設定値に対する t_{en} の測定



(b) V_{min}，V_{typ}，V_{max} の確率密度関数の見積もり



(c) c_{th} の計算



(d) t_{en} の計算

図 4.10　統計的手法による t_{en} の決定

上記のアプローチでは，与えられたパラメータ N_a, N_o の下で統計的な安全性を実現する．より安全なアプローチとしては，静的タイミング解析 (Static Timing Analysis: STA) により算出された回路の遅延時間に基づき t_{en} の設定値を設定する方法である．最初に，図 4.10(a) – 4.10(c) の手順を実行する．その後，図 4.11 に示すように，静的タイミング解析によって算出された回路の遅延時間 t_{STA} を常に上回る t_{en} を選択する．

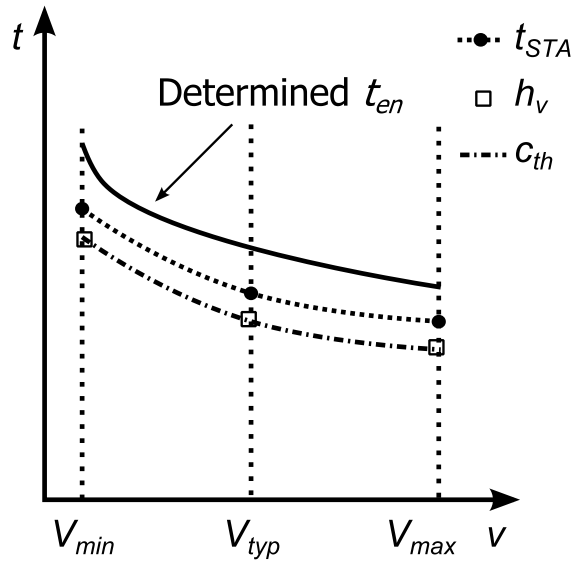


図 4.11　t_{STA} に基づく t_{en} の設定

上記の議論は温度が一定であると仮定しているが，温度変化を考慮するには，最高，最低，標準の 3 つの温度で最適な t_{en} を算出し，最大のものを選択する．

4.4　提案対策の評価実験

4.4.1　提案対策の ASIC 実装

提案手法の評価実験を行うため，提案対策を施した AES ハードウェアを ASIC 上に実装した．AES ハードウェアは典型的なループアーキテクチャに基づくもので，合成体 $GF(((2^2)^2)^2)$ を用いた S-box [39] を搭載したものとする．図 4.12 に，実装した n 段の可変遅延回路を示す．i 段目はマルチプレクサと，直列接続された 2^{i-1} 個の遅延素子から成る．t_m と t_e をそれぞれ 1 個のマルチプレクサと遅延素子の遅延時間とすると，n 段の遅延素子の遅延時間は,

$$t_d = nt_m + dt_e \tag{4.8}$$

となる．ただし，d は遅延時間の設定値で，0 から $2^n - 1$ の範囲で表される．d が最大値の時，信号は全ての遅延素子を通る．従って，遅延時間は間隔 t_e で $nt_m \leq t_d \leq (2^n - 1)t_e$ の範囲

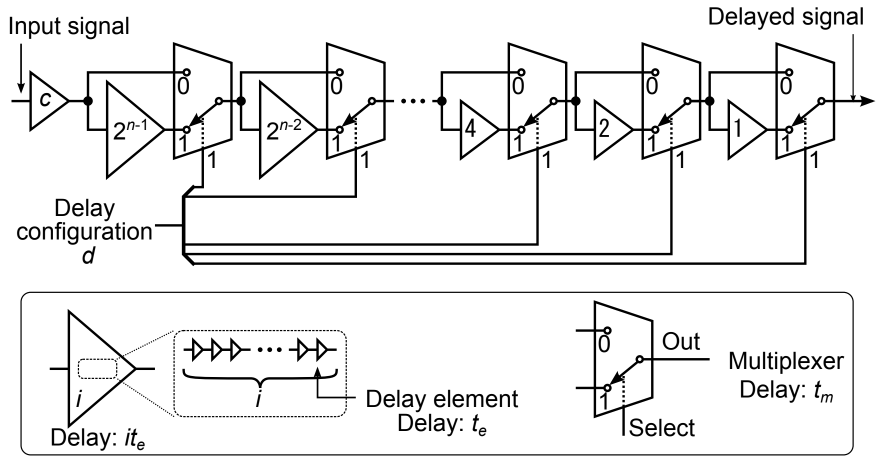


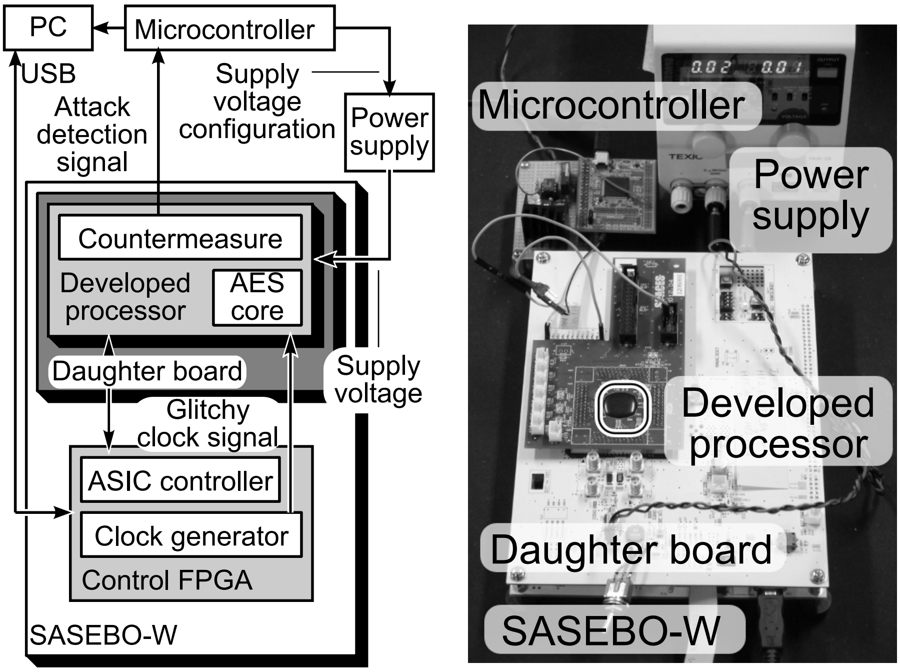
図 4.12　可変遅延回路

で変化させることができる．遅延回路の段数 n は，静的タイミング解析に基づく暗号回路の遅延時間を元に決定する．

4.4.2　遅延時間の測定実験

上記の試作回路を用いた実験により，提案対策の有効性を示す．図 4.13 に実験環境を示す．3 章で構築した故障注入攻撃実験システムを用いて，故障感度解析と同様の手法で遅延時間を測定した．試作した ASIC はドーターボード上に実装され，さらにそれをサイドチャネル攻撃標準評価ボード SASEBO-G 上に接続した．本実験では，可変遅延回路は 4 段，温度は 25°C で，試作した 2 つのチップ A, B を対象として実験を行った．ここでは，可変遅延回路の遅延時間 t_{en} および AES の最初のラウンドにおける AES 回路の遅延時間を測定し，4.3.5 節で述べた遅延時間の設定方法により可変遅延回路の遅延時間を設定した．

図 4.14 に，チップ A での遅延時間の測定結果を示す．本図では，実線は異なる設定値における可変遅延回路の遅延時間 (t_{en}) を表し，太い実線は静的タイミング解析に基づき設定された t_{en}，白い円は V_{min}(0.87V)，V_{typ}(1.08V)，V_{max}(1.21V) で測定された AES 回路の遅延時間を表す．白い四角は h_v の値，一点鎖線は c_{th} の軌跡，黒い円は t_{STA} の値，点線は t_{STA} の線を表す．本実験では，1.08V(最低電圧)，1.20V(標準電圧)，1.32V(最高電圧) で静的タイミング解析を行い，t_{STA} を求めた．これらの電圧値はチップ製造者の標準セルライブラリによ



(a) ブロック図 (b) 外観

図 4.13 実験環境

り規定されている．電源電圧 v は 0.05 V の間隔で 0.87V (チップが動作した最低電圧) から 1.21V(チップが動作した最高電圧) の間で変化させた．電圧下降に従い，AES 回路の遅延時間と t_{en} の両者が増加した．4.3.4 節と 4.3.5 節の手法に基づき，図 4.14 において t_{en} の値を決定した．

ここで，h_v の具体的な計算方法を示す．ここでは，許容できる故障感度の漏洩回数を $N_a = 1$，攻撃者による故障感度の測定回数を $N_o = 10^8$ 回とする．故障感度解析においては，攻撃対象のデバイスによらず，1 回の故障感度測定で秘密鍵を推定できることが評価者にとっての最悪値であることに注意されたい．なお，10^8 回の測定には，1 秒に 1 回の測定と仮定すると 3.17 年 (1147 日) かかる．まず，暗号回路の遅延時間の分布を求める．図 4.15 に，電源電圧 $V_{min}(= 0.87V)$ で 10000 の平文を用いて測定した AES の遅延時間の分布を示す．遅延時間は正規分布に従い，平均は $\mu = 2.904$，標準偏差は $\sigma = 0.2407$ であることが分かる．

ある故障感度 t が遅延時間 h を超える確率 $g(h)$ は，誤差関数を erf とすると

$$g(h) = 1 - \frac{1}{2}\left\{1 + \mathrm{erf}\left(\frac{h-\mu}{\sqrt{2\sigma^2}}\right)\right\} = \frac{1}{2}\left\{1 - \mathrm{erf}\left(\frac{h-\mu}{\sqrt{2\sigma^2}}\right)\right\} \tag{4.9}$$

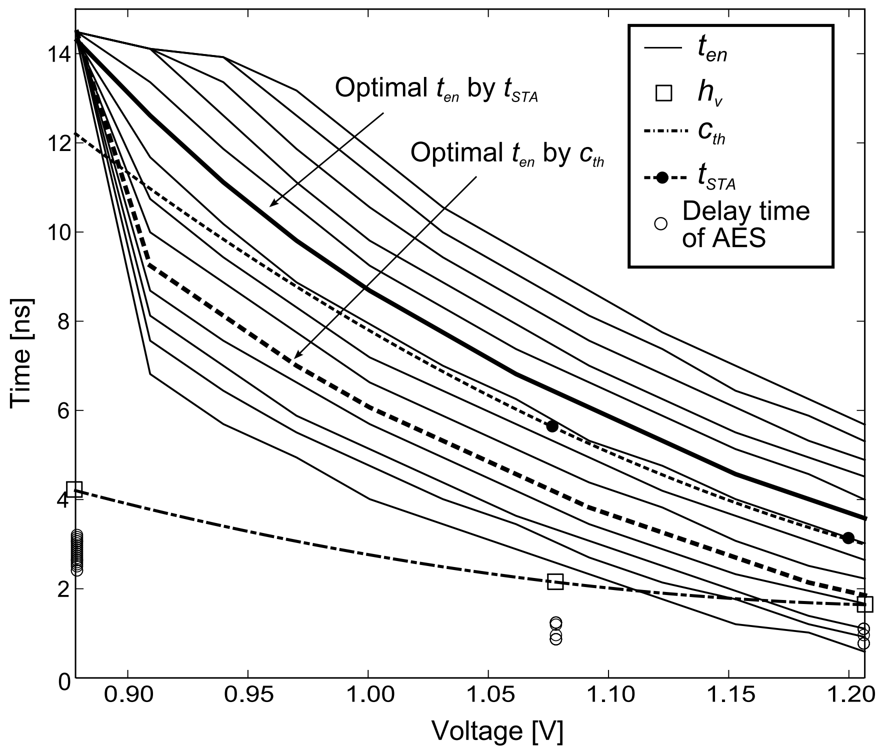


図 4.14 チップ A における遅延時間の測定結果

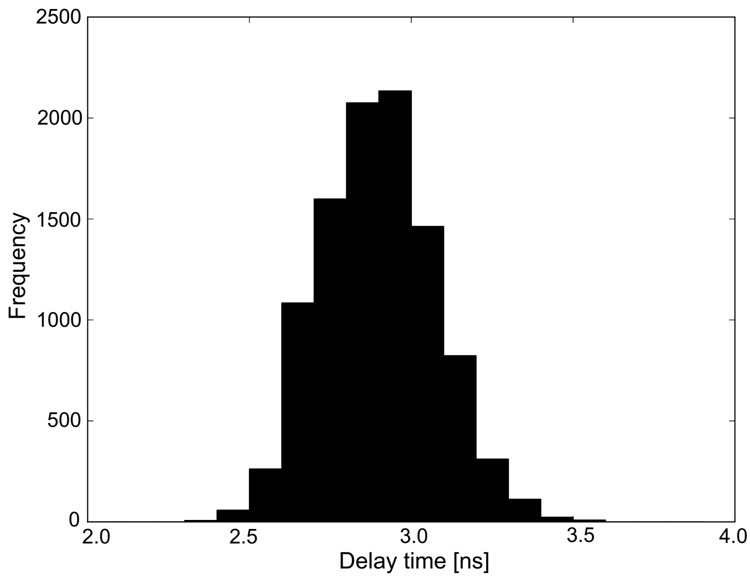


図 4.15 電源電圧 0.87 V における AES 回路の遅延時間の分布

で表される．上記の μ, σ の値および $f(h_{0.87})$ を式 (4.9) に代入すると，$h_{0.87} = 4.254$ns が得られる．電源電圧 $V_{typ} = 1.08$ V および $V_{max} = 1.21$ V における h_v の値は，それぞれ

$h_{1.08} = 2.041$ ns および $h_{1.21} = 1.605$ns となった．従って，図 4.14 のように h_v，c_{th}，最適な t_{en} を描くことができる．

図 4.16 に，チップ B における遅延時間の測定結果および最適な t_{en} の値を示す．チップ B の AES 回路は電源電圧が 0.87 V から 1.27 V の範囲で正常に動作した．チップ B での結果を図 4.14 の結果と比較すると，AES 回路の遅延時間および動作範囲は製造ばらつきによりチップ A の分布とは異なることが分かる．このように，提案対策では，チップの製造ばらつきの影響を受けずに最適な設定値を選択できる．

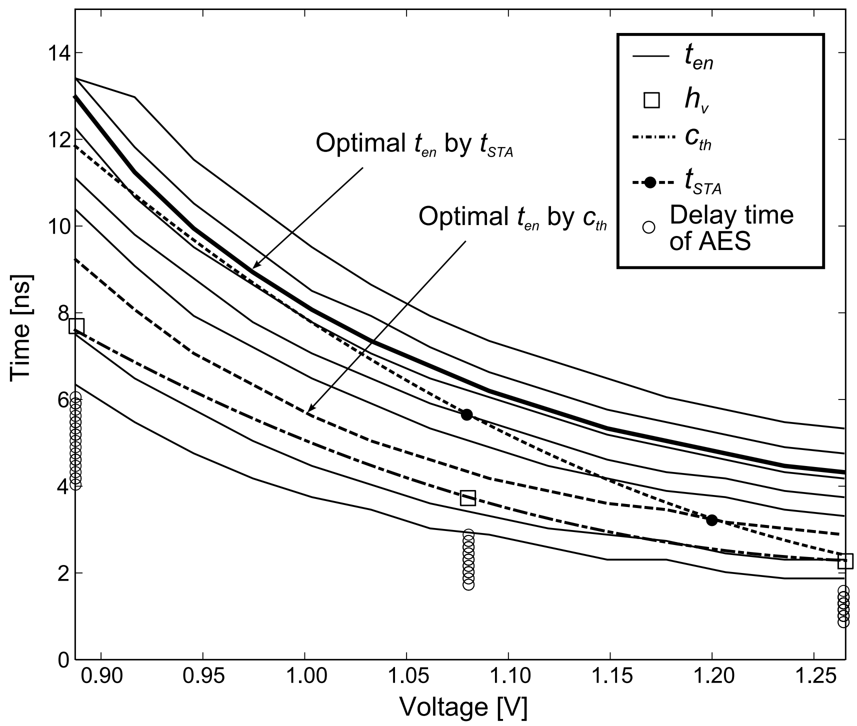
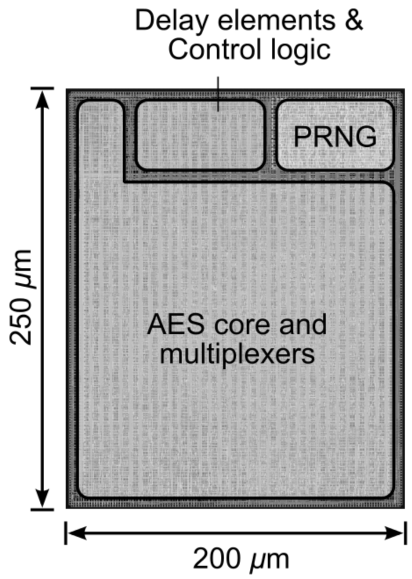


図 4.16　チップ B におけるプロファイリング結果

4.4.3　提案対策の実装コスト

図 4.17 に，製造したチップのレイアウト図と各構成要素の実装コストを示す．図 4.17 中の表は，Synopsys 社の設計ツール Design Compiler D-2010.03 で見積もったゲート換算の回路面積であり，AES core は対策なしの AES 回路を表す．提案対策を施した AES モジュールは組み合わせ回路とレジスタの間に設けられるマルチプレクサを含む．



Components	Gate Equivalence (%)
AES core	28702 (100)
Multiplexer	1882 (6.56)
CDB	1068 (3.72)
EN generator	35 (0.12)
PRNG	1624 (5.66)
Total	33311 (116.06)
Total w/o PRNG	31687 (110.40)

図 4.17 実装した AES 回路のレイアウト図と実装コスト

遅延回路，マルチプレクサ，擬似乱数生成器のオーバーヘッドはそれぞれ AES コアの 3.72%，6.56%，5.66% であった．従って対策回路全体の面積オーバーヘッドは 16.0% となる．この対策は，検算を用いた故障検知など，従来の対策とともに実装できる．

4.5 結び

本章では，製造後に調整可能な遅延回路を用いた故障感度解析への対策を提案した．まず，遅延回路を用いた故障感度解析対策のアーキテクチャを示した．その後，遅延回路の設定方法を示した．攻撃を防ぐための遅延回路の設定値を統計的に設定する手法を示した．また，提案対策を施した AES ハードウェアを ASIC 上に実装し，遅延回路および AES 回路の遅延時間を測定し，最適な設定値を決定する実験を行い，提案対策の有効性を示した．また，対策回路の面積についても評価を行い，面積オーバーヘッドは約 16% であった．

第 5 章

多重故障注入攻撃の検討とその対策技術の開発

5.1　はじめに

本章では，組込みマイコンにソフトウェア実装された暗号モジュールに対する多重故障注入攻撃とその対策を検討する．まず，新たな多重故障注入攻撃を提案する．提案攻撃では，検算型対策を施した暗号ソフトウェアに対して，適応的に故障注入を行うことにより，適切な故障注入のタイミングを探索し，攻撃者がプログラムを知らない状態でも攻撃が可能となる．本章では上記攻撃の実験を行い，提案攻撃により誤りを含む暗号文を取得できることを示す．また，提案攻撃へのソフトウェアレベルでの対策を提案する．提案攻撃では分岐命令のスキップを利用するが，提案対策では，分岐命令をスキップした場合でも暗号文を出力しないようプログラムを変更する．

5.2　関連研究

多重故障注入攻撃は，最初に故障解析手法の分野で検討されている．ここでは，暗号アルゴリズム中の変数に複数の故障が注入される場合を想定しており，文献 [40] では，複数の故障が AES の差分故障解析へ及ぼす影響を明らかにしている．また，文献 [41] では，AES の実行中

に，中間値に起こりうる全ての故障をモデリングしている．このように，注入される故障を一般化しようとする試みがなされているが，暗号アルゴリズムの解析に主眼を置いていることから，プログラムの流れ (プログラムフロー) が変化する故障モデルは考慮していない．

プログラムフローへの故障注入のモデルとして，命令スキップがある．これは実行中の命令を阻止するという故障モデルであり，従来から，組込みマイコン上に実装された暗号モジュールに対して故障注入を行い，実行中の命令がスキップされたという報告がある [21]–[19]．命令スキップは，8 ビットマイコンにおける実験では電源電圧の低下 [21] や電磁パルス [42] により起こることが確認されており，32 ビットのマイコンでもレーザー照射 [7] と電磁パルス [43] により報告されている．命令スキップを用いた攻撃では，誤りを引き起こし，得られた出力から差分故障解析等の故障解析手法を用いて秘密鍵を復元する．また，命令スキップをモデルとして想定した場合，対策コードの実行そのものをスキップすることが考えられるため，対策コードをスキップして誤った暗号文を得る攻撃 [7] も報告されている．

一方，命令スキップへの対策も提案されている．組込みマイコンにおける暗号実装では，対策ハードウェアの追加が難しいことから，ソフトウェアの変更による対策手法が提案されている．最も典型的な対策が検算型の対策である．この対策は，同じ演算を複数回行うもの [19] と，逆演算やチェックサムのような暗号化の演算とは異なる演算を行うもの [36],[44] とに大別できる．どちらの対策においても，最終的には故障検知のために複数の演算結果を比較し，結果が誤りの場合暗号文を出力しない．多重故障注入を想定した場合，暗号化と比較処理の両方に故障を注入できる．ソフトウェアにおける条件分岐では，分岐の条件を判定する比較命令の後に，分岐命令を配置することが多い．このため，分岐命令をスキップすることにより，これらの対策を無効化できるおそれがある．

また，冗長性の粒度も複数の場合が考えられる．典型的なものが，暗号処理全体に冗長性を持たせるもので，暗号化を完了した後に検算のための処理を行う対策 [23] である．一方，各命令ごとに冗長性を持たせる対策として，命令の複製を行う対策 [19] もある．この対策は任意の 1 命令をスキップした場合でも複製された命令が実行されることから故障注入を阻止できるが，複製された命令を全てスキップした場合無効化できる．

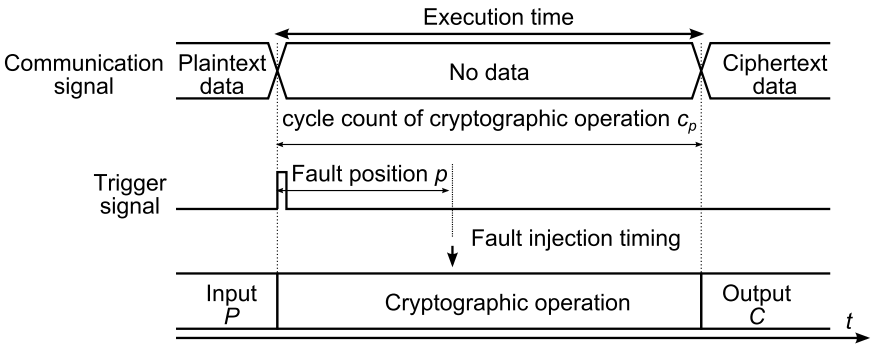


図 5.1　暗号処理のタイミング図

5.3　適応的な故障注入に基づく多重故障注入攻撃

本節では，文献 [23] に示されているような典型的な検算型対策に対して，攻撃者がプログラムを知らない状態（ブラックボックス）での多重故障注入攻撃手法を提案する．提案攻撃では，故障注入のタイミングを 1 サイクルずつ変化させ，暗号モジュールの出力から適切なタイミングかどうか判断することにより，探索を行う．

5.3.1　攻撃の前提

提案攻撃では，攻撃者が以下の事実を知っていることを仮定する．

(a) ソフトウェアで暗号アルゴリズムが実装されていること
(b) 検算型の対策が実装されていること

検算型の対策では，暗号処理に故障が注入されるとエラー信号が出力されることから，攻撃者はある故障注入を行うことにより検算型の対策が実装されていることを知ることができる．また，攻撃者は IC カードリーダー等の機器と暗号モジュール間の通信信号から暗号処理が行われているタイミングを知ることができると仮定する．図 5.1 のように，通信の信号は，暗号処理の前後に発生することから，攻撃者は暗号処理の開始タイミング (=暗号処理前の通信の終了タイミング) および暗号処理のサイクル数 c_p を知ることができる．

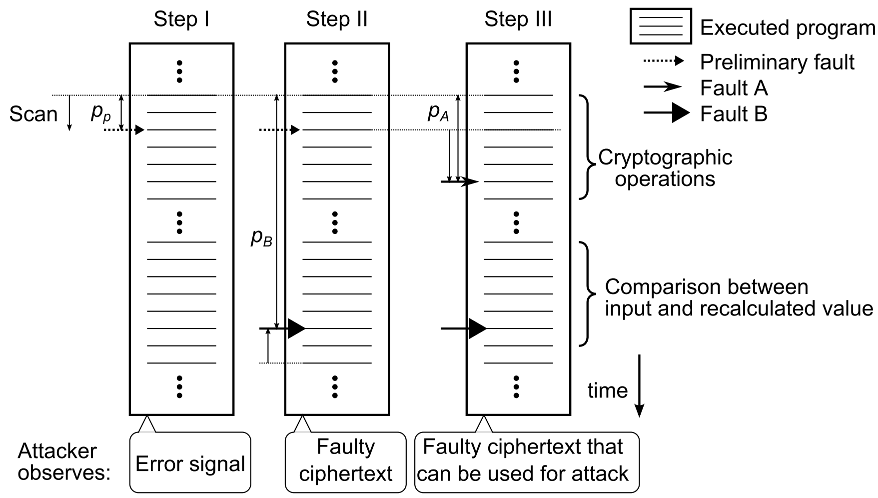


図 5.2　提案攻撃の概念図

図 5.1 において，トリガ信号は通信信号のみを用いて生成できる．以降，トリガ信号の立ち上がりからクロックグリッチを含むサイクルまでの間隔 p を故障位置と呼ぶ．

5.3.2　提案攻撃のアルゴリズム

図 5.2 に提案攻撃の概念図を示す．故障 A および故障 B を，それぞれプログラム (命令列) 中の暗号処理および対策へ注入される故障とする．まず，攻撃の開始前に故障注入なしで暗号化を行い，正しい暗号文を取得しておく．提案攻撃は 3 ステップから成り，ステップ I では対策を動作させるための故障である予備故障のタイミングを探索する．予備故障の位置 p_p を 0（暗号処理の開始タイミング）から 1 ずつ増加させながら探索する．予備故障が適切な位置に注入された時，暗号モジュールからエラー信号が出力されるため，エラー信号の出力をもってステップ I は終了となる．

ステップ II では，ステップ I で得たタイミングで予備故障を注入しつつ，故障 B の適切な注入タイミングを探索する．故障 B の位置は暗号処理の終了タイミングから探索を開始し，1 サイクルずつ減少させる．この段階では，故障 B が適切なタイミングで注入されると，エラー信号ではなく誤った暗号文が出力されることから，誤った暗号文を得た時に終了となる．

この段階で誤った暗号文は既に得られているが，予備故障の注入位置は必ずしも攻撃に適した位置であるとは限らない．例えば，AES に対する差分故障解析では，故障が 8 ラウンド目ま

Algorithm 1 暗号化の後に復号を行う検算型対策 (EncWithRecal)

Input: Plaintext P

Output: Ciphertext C or error signal

$C \leftarrow$ Encryption(P)

$P_2 \leftarrow$ Decryption(C)

if $P = P_2$ **then**

　　return C

end if

return error signal

たは 9 ラウンド目の入力に注入されていることが前提となる．そこで，ステップ III では，予備故障を取り除き，攻撃に適した暗号文を得るための故障である故障 A を注入する．故障 A の注入は予備故障の注入タイミング p_p から開始し，1 ずつ増加させながら探索する．ステップ III は攻撃に適した暗号文が得られた時終了となる．

上記の攻撃のアルゴリズムを示す．攻撃対象とする暗号ソフトウェアは，アルゴリズム 1 に示すように，暗号化の後に復号を行い，復号結果を入力と比較する．

ステップ I, II, III に対応するアルゴリズムはそれぞれアルゴリズム 2，3，4 である．まず，アルゴリズム 2 を用いて予備故障の位置を求める．ここで，5.3.1 節で述べたように，c_p は暗号処理に要するクロックサイクル数である．予備故障の位置 p_p および予備故障のグリッチ幅 w_p はそれぞれ $0 \leq p_p \leq c_p$ および $w_{min} \leq w_p \leq w_{max}$ の範囲で変化させる．w_{max} および w_{min} はそれぞれ最大と最小のグリッチ幅である．なお，w_p は 1 ステップごとにクロックグリッチ発生器の時間分解能 r_w だけ減少させる．

次に，故障 B の位置 p_B とグリッチ幅 w_B を探索するため，アルゴリズム 3 を実行する．予備故障を位置 p_p に注入しつつ，p_B はプログラムの最後から探索する．また，w_B は予備故障と同様に $w_{min} \leq w_B \leq w_{max}$ の範囲で変化させる．アルゴリズム 3 は誤った暗号文が得られた時終了となる．

最後にアルゴリズム 4 を実行し，差分故障解析 (DFA) に利用できる暗号文を取得する．本

Algorithm 2 対策を動作させるための故障注入

Input: Random plaintext P, cycle count of cryptographic operation c_p, maximum and minimum glitch with w_{max}, w_{min}

Output: Correct ciphertext C, position and glitch width of preliminary fault p_p, w_p

$C \leftarrow$ EncWithRecal(P) # Correct ciphertext
for $p_p = 0$ to c_p **do**
　SetPreliminaryFaultPosition(p_p)
　for $w_p = w_{max}$ downto w_{min} **do**
　　SetPreliminaryFaultGlitchWidth(w_p)
　　$C_f \leftarrow$ EncWithRecal(P)
　　if Error signal is observed **then**
　　　return p_p, w_p
　　end if
　end for
end for

章では AES 暗号を例として，9 ラウンド目に故障を注入された結果として得られた暗号文を探索する．故障 B をステップ II で得たタイミングで注入しつつ，故障 A の位置 p_A を $p_p \leq p_A$，グリッチ幅 w_A を $w_{min} \leq w_p \leq w_{max}$ の範囲で変化させて探索する．アルゴリズム 4 は，一部に誤りを含む暗号文が得られた時終了となる．8 ビットマイコンでは，4 バイトの誤りを含む暗号文を探索する．

図 5.3 に，128 ビット AES において，9 ラウンド目の入力に 1 バイトの故障が注入された場合の故障の伝搬を示す．この故障は Piret らによる DFA[20] を用いた解析に利用できる．従って，8 ビットマイコンでは，アルゴリズム 4 は 4 バイトの誤りを含む暗号文が得られた時点で終了となる．

一方 32 ビットマイコンでは，最初に 12 バイトより少ない誤りを含む暗号文を探索し，適切な p_A を探索する．AES に対する DFA では，16 バイト全てに誤りを含む暗号文を用いたと

Algorithm 3 対策を回避するための故障注入

Input: Random plaintext P, correct ciphertext C, cycle count of cryptographic operation c_p, position and glitch width of preliminary fault p_p, w_p, maximum and minimum glitch width w_{max}, w_{min}

Output: Position and glitch width of Fault B p_B, w_B

SetPreliminaryFaultPosition(p_p)
SetPreliminaryFaultGlitchWidth(w_p)
for $p_B = c_p$ to p_p **do**
　　SetFaultBPosition(p_B)
　　for $w_B = w_{max}$ downto w_{min} **do**
　　　　SetFaultBGlitchWidth(w_B)
　　　　$C_f \leftarrow$ EncWithRecal(P)
　　　　if $C_f \neq C$ and Error signal is not observed **then**
　　　　　　return p_B, w_B
　　　　end if
　　end for
end for

き最も少ない暗号文で攻撃できる [45] ことから，ここからさらに 16 バイトの故障が得られるパラメータを探索する．故障位置 p_A において，16 バイトの故障が得られるようグリッチ幅 w_A を探索する．

5.3.3　攻撃に要するコスト

提案攻撃に要するコストは，DFA に利用できる暗号文を取得するまでに要する故障注入の試行回数に依存する．c_1，c_2，c_3 をそれぞれステップ I, II, III における故障注入の試行回数とする．また，p_p，p_A，p_B をそれぞれ予備故障，故障 A，故障 B の位置とし，p_p，p_A，p_B をそれぞれ予備故障，故障 A，故障 B のグリッチ幅とする．w_{max}, w_{min}, w_i をそれぞれ最大

Algorithm 4 一部に誤りを含む暗号文の取得

Input: Random plaintext P, correct ciphertext C, cycle count of cryptographic operation c_p, position and glitch width of preliminary fault p_p, Position and glitch width of Fault B p_B, w_B

Output: Position and glitch width of Fault A p_A, w_A, faulty ciphertext C_f

```
SetFaultBPosition(p_B)
SetFaultBGlitchWidth(w_B)
for p_A = p_p to c_p do # Loop for glitch position
    SetFaultAPosition(p_A)
    for w_A = w_max downto w_min do # Loop for glitch width
        SetFaultAGlitchWdith(w_A)
        C_f ← EncWithRecal(P)
        d ← C_f ⊕ C
        n =CountNonZeroByte(d)
        if n ≤ 12 then
            return p_A, w_A, C_f
        end if
    end for
end for
```

のグリッチ幅と最小のグリッチ幅，最大と最小の差 ($w_i = w_{max} - w_{min}$) とする．w のループの試行回数は w/r_w (r_w はグリッチ生成回路の分解能) で表される．総試行回数 c は以下の式で表される．

$$c_1 = p_p(w_i/r_w + 1) + (w_{max} - w_p)/r_w + 1, \tag{5.1}$$

$$c_2 = (c_p - p_B)(w_i/r_w + 1) + (w_{max} - w_B)/r_w + 1, \tag{5.2}$$

$$c_3 = (p_A - p_p)(w_i/r_w + 1) + (w_{max} - w_A)/r_w + 1, \tag{5.3}$$

$$c = c_1 + c_2 + c_3. \tag{5.4}$$

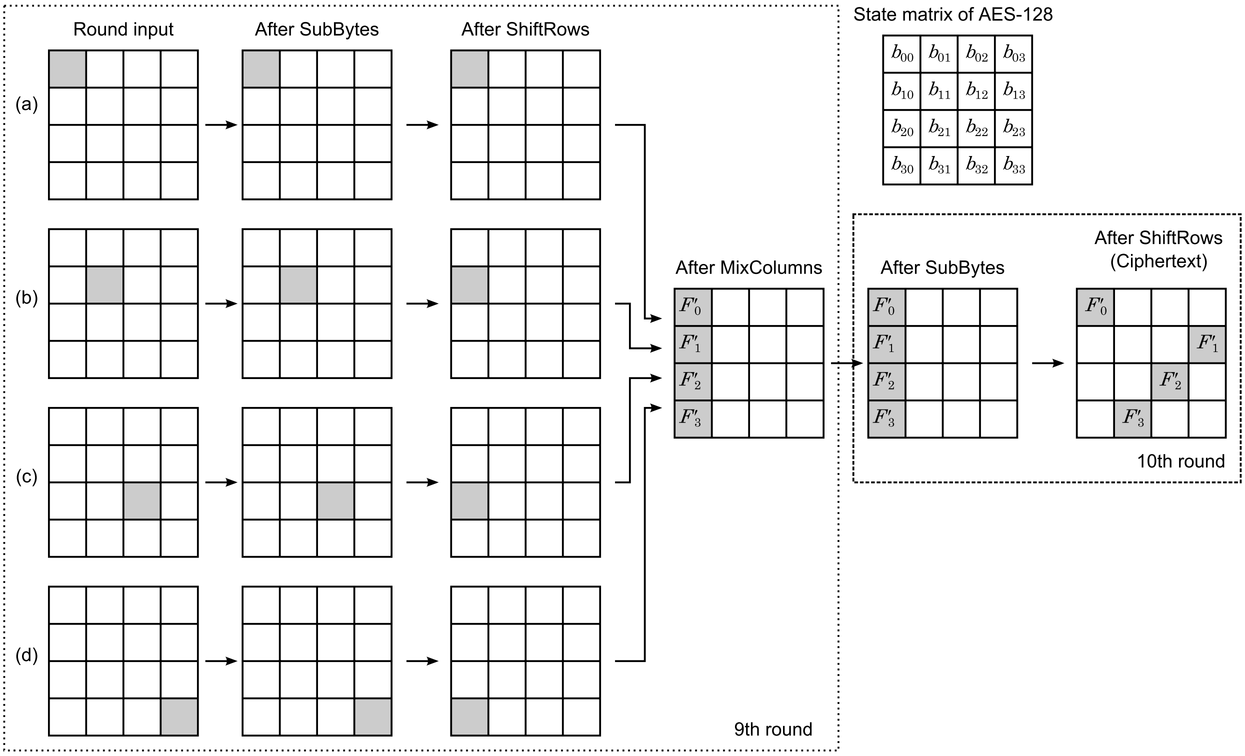


図 5.3　128 ビット AES における故障の伝搬

この場合，最も探索回数が大きくなる条件である最悪の条件は，式 (5.5) および (5.6) により表される．また，最悪の条件における試行回数は式 (5.7)–(5.10) で与えられる．

$$w_p = w_A = w_B = w_{min} = 0, \tag{5.5}$$

$$p_B = p_A + 1. \tag{5.6}$$

$$\begin{aligned}
c_1 &= p_p(w_{max}/r_w + 1) + w_{max}/r_w + 1, && (5.7)\\
c_2 &= (c_p - p_A - 1)(w_{max}/r_w + 1) \\
&\quad + w_{max}/r_w + 1 \\
&= (c_p - p_A)(w_{max}/r_w + 1) && (5.8)\\
c_3 &= (p_A - p_p)(w_{max}/r_w + 1) + w_{max}/r_w + 1, && (5.9)\\
c &= c_1 + c_2 + c_3 \\
&= (c_p + 2)(w_{max}/r_w + 1). && (5.10)
\end{aligned}$$

8 ビットマイコンでは，5.6.2 節に示すように，グリッチ幅 w の探索をせずに故障注入のターゲットとなる命令をスキップできることから，グリッチ幅 w を探索するためのループを

省略できる．この条件は，式 (5.7)–(5.10) で $w_{max}/r_w = 0$ となっている場合と等価であり，式 (5.10) は $c = c_p + 2$ となる．現在用いられている多くの共通鍵暗号アルゴリズムは，高々数十万サイクルの実行サイクル数 (=c_p) のソフトウェアで実装できることから，上記の数は 128 ビットの鍵を総当りで求める場合と比較して十分小さいといえる．

32 ビットのマイコンでは，ステップ III は 2 つの段階に分けられ，それぞれステップ III(a) および III(b) と呼ぶ．c'_3 と c''_3 はそれぞれステップ III(a) および III(b) における試行回数とする．また，I_s はグリッチ幅 w の設定値における刻み数であり，$w = r_w \cdot I_s$ とする．ステップ III(a) では，一部が誤った暗号文が得られる時の故障 A の位置 p_A を知るため，グリッチ幅の刻み幅を粗くした探索を行う．グリッチ幅は w_A は刻み幅 $r_w \cdot I_s$ で探索される．次に，ステップ III(b) では，位置を上記で求めた p_A に固定し，幅 w_A を細かく探索する．

$$\begin{aligned} c'_3 &= (p_A - p_p)\{w_i/(r_w \cdot I_s) + 1\} \\ &\quad + (w_i - w_A)/(r_w \cdot I_s) + 1, && (5.11) \\ c''_3 &= (w_i - w_A)/r_w + 1, && (5.12) \\ c &= c_1 + c_2 + c'_3 + c''_3 && (5.13) \end{aligned}$$

式 (5.13) を，探索回数が最も多くなる条件である式 (5.5) および (5.6) の下で計算した場合，式 (5.14) となる．

$$\begin{aligned} c &= (c_p + p_p - p_A + 1)(w_{max}/r_w + 1) \\ &\quad + (p_A - p_p)\{w_{max}/(r_w \cdot I_s)\} + 1 && (5.14) \end{aligned}$$

ここで，p_A および p_p の最悪値 (探索回数が最も多くなる場合) は，それぞれ $p_A = c_p - 1$，$p_p = c_p - 1$ となる．従って，試行回数の最大値は式 (5.15) で表される．

$$c = (c_p + 1)(w_{max}/r_w + 1) \tag{5.15}$$

提案攻撃において，故障注入のタイミングは，平文と秘密鍵からなる入力データに依存しないため，攻撃者は一度各故障の適切な注入タイミングを同定すると，以降は平文を変えるのみ

表 5.1 実験条件

Cryptographic algorithm	128-bit AES with recalculation-based countermeasure (Written from scratch)
8-bit microcontroller	Atmel ATmega163
32-bit microcontroller	Atmel AT91SAM7S256 (with ARM7TDMI core)
Compiler	gcc 4.3.3 (Optimized by the option -Os)
FPGA	Xilinx XC6SLX150
Plaintext	$(00112233445566778899aabbccddeeff)_{16}$
Secret key	$(000102030405060708090a0b0c0d0e0f)_{16}$

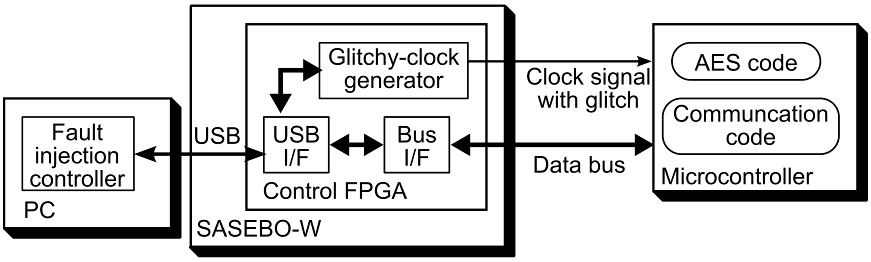
で複数の誤りを含む暗号文を入手できる．また，故障注入の総試行回数も入力に依存しない．なお，提案攻撃はリバースエンジニアリングのコストを必要としない．

5.4 実験

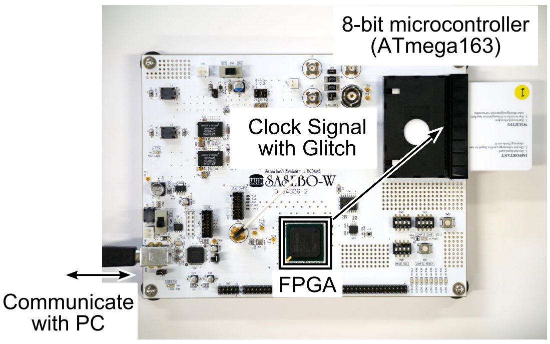
5.4.1 実験の概要

本節では，検算型対策を施した AES 暗号ソフトウェアに対して，提案攻撃の実験を行う．図 5.4(a) に実験環境を示す．第 3 章で提案したシステム上に組込みマイコンを接続し，マイコンへクロックグリッチを供給する．図 5.4(b) および図 5.4(c) にそれぞれ 8 ビットおよび 32 ビットの組込みマイコンを用いた実験環境の写真を示す．両者ともクロック信号はサイドチャネル攻撃標準評価ボード SASEBO 上の FPGA で生成され，FPGA からマイコンのクロック入力端子へ信号が供給される．

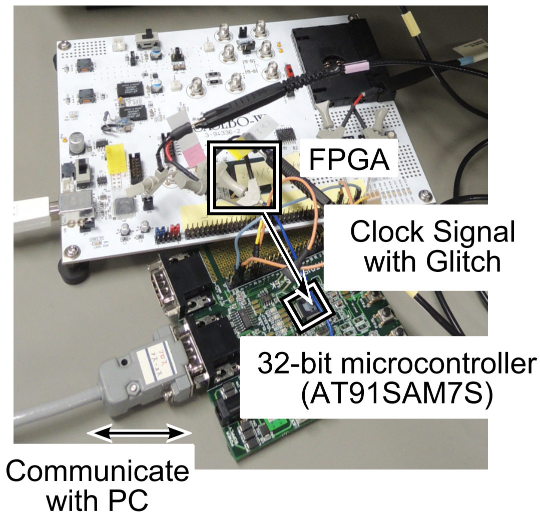
表 5.1 に実験条件を示す．どちらのマイコンでも，アルゴリズム 1 に示す 128 ビットの AES プログラムを実行する．このプログラムは暗号演算用ライブラリを用いずに C 言語で書かれており，gcc 4.3.3 でコンパイルしている．本実験では，アルゴリズム 2,3,4 を順に実行する．アルゴリズム 4 は，5.3.2 節で述べたように，8 ビットマイコンでは 4 バイト，32 ビットマイコンでは 16 バイトの誤りを含む暗号文を得た時点でそれぞれ終了となる．



(a) ブロック図



(b) 8 ビットマイコンを用いた実験環境



(c) 32 ビットマイコンを用いた実験環境

図 5.4　実験環境

5.4.2　予備実験

8 ビットマイコンの実験では，攻撃のターゲットとなる命令をスキップできるグリッチ幅 w をあらかじめ測定した．攻撃のターゲットとなる命令は，分岐 (BRNE)，排他的論理和 (XOR)，ロード (LD)，ストア (ST) である．これらの命令へ故障注入を行った結果，中央値として $w = 54$ ns を選択した．32 ビットのマイコンでは，10.2 ns から 5.64 ns までの範囲で w を走査した．

Algorithm 5 検算型対策を施した AES (AESEncWithRecal)

Input: Plaintext P

Output: Ciphertext C

$C \leftarrow$ AESEncryption(P)

$P_2 \leftarrow$ AESDecryption(C)

r1 $\leftarrow 1$

if $P = P_2$ **then**

  r1 $\leftarrow 0$

end if

return C, r1

5.4.3 8 ビットマイコンにおける実験結果

アルゴリズム 5 に，本実験で用いた関数 AESEncWithRecal を示す．これは，暗号化および検算としての復号を行い，その結果を比較する．この関数はアルゴリズム 5.3.2 の 1 行目から 3 行目に対応するが，P と P_2 の比較結果を返す点が異なる．AESEncWithRecal では，暗号化結果と検算結果が等しい時 0 をレジスタ r1 に保存し，そうでない場合 1 を保存する．コード 1 は本実験で用いたプログラムのメイン関数である．ここで，関数 SendData および SendError は，それぞれ暗号化結果およびエラー信号を出力する関数である．コード 1 において，暗号化が終了すると，3 行目のテスト (TST) 命令により，レジスタ r1 の値を元に分岐の成否を確定する．テスト命令はレジスタ r1 の値が 1 の場合，分岐条件を表すゼロフラグ (Z) を 0 にクリアし，r1 $= 0$ の場合 Z $= 1$ とする．その後，4 行目の分岐命令 (BRNE) が実行され，Z $= 0$（故障が検知された場合）のときにはラベル LBL_FAIL へのジャンプが行われた後に “call SendError” が実行され，エラー信号が出力される．Z $= 1$ の場合にはジャンプは行われず，“call SendData” が実行され，暗号文が出力される．しかしながら，4 行目の分岐命令の実行中に故障が注入されると，分岐命令がスキップされ，レジスタ r1 の値にかかわらず 5 行目の “call SendData” が実行される．

表 5.2 8 ビットマイコンにおける実験結果

Step	Starting position	Final position	Number of trials
I	$p_p = 0$	$p_p = 3$	4
II	$p_B = 30530$	$p_B = 30527$	4
III	$p_A = 3$	$p_A = 9996$	9994
Total			10002

コード 1 本実験に用いたメイン関数

```
1: LBL_MAIN:
2: CALL AESEncWithRecal
3: TST r1     ;Z ← 1 when r1=0
4: BRNE LBL_FAIL  ;Jumps when Z=0
5: CALL SendData
6: JMP LBL_FINISH
7: LBL_FAIL:
8: CALL SendError
9: LBL_FINISH:
```

表 5.2 に，8 ビットマイコンにおいて提案攻撃を完了するのに要した故障注入の試行回数を示す．最初に，暗号処理に要するサイクル数 c_p を測定し，$c_p = 30530$ との数値を得た．本実験では，合計で 10002 回の故障注入により，差分故障解析に適した 4 バイトの誤りを含む暗号文を得た．

本実験において，実際に故障が注入された処理を以下に示す．故障 A は 8 ラウンド目の AddRoundKey に注入された．コード 2(a) に本実験で用いた AddRoundKey のコードを示す．変数 k および s はそれぞれラウンド鍵と中間変数を表す．コード 2(b) は，コード 2(a) をコンパイルして得られたアセンブリコードである．このコードの中で，故障 A は 5 行目のロード命令 (LD) に注入された．また，故障 B は，コード 1 の 4 行目にある分岐命令 (BRNE) に注入された．

コード 2 AddRoundKey

(a) C 言語のソースコード

```
add_round_key(void)
{
 u8_t i;
 for (i=0; i<16;i++){
   s[i] ^= k[i];
 }
}
```

(b) gcc 4.3.3 でコンパイルされたアセンブリコード

```
1: LDS r26, 0x0000
2: LDS r27, 0x0000
3: LDI r30, 0x00
4: LDI r31, 0x00
5: LD r24, X
6: LD r25, Z+
7: EOR r24, r25
8: ST X+, r24
9: LDI r24, 0x00
10: CPI r30, 0x00
11: CPC r31, r24
12: BRNE .+0
13: RET
```

5.4.4 32 ビットマイコンにおける実験結果

32 ビットのマイコンでは，4 バイトを 1 命令で処理することから，誤りを含む暗号文のバイト数はグリッチ幅に依存する．この事実を示すため，まず，8 ラウンド目の AddRoundKey に故障を注入する実験を行った．図 5.5 に，異なるグリッチ幅における誤りのバイト数を示す．ここでは，コード 2(b) の 7 行目にある排他的論理和命令 (EOR) に故障を注入した．グリッチ幅を w に固定し 10 回の測定を行い，その平均値をグリッチ幅 w における誤りバイト数とした．図 5.5 では，グリッチ幅の減少に伴い誤りバイト数が増加している．5.6 ns より小さなグリッチ幅の場合には，マイコンの入力回路にグリッチとして認識されなかったため故障は注入されなかった．この結果より，差分故障解析に利用できる暗号文の取得には，グリッチ幅の走査が必要であることが分かる．

表 5.3 に，32 ビットの ARM マイコンにおける実験結果を示す．本実験では，暗号処理に要するサイクル数 c_p は 32667 であった．表 5.3 では，カッコ内の数値は，グリッチ発生回路の刻み数を表す．合計で 446191 回の故障注入により差分故障解析に利用できる暗号文を取得できた．探索の前に数回故障注入を行った結果，グリッチ幅 $w > 10.2$ns および $w < 5.64$ns のとき故障は注入されないことを観測した．そこで，以降の探索では，グリッチ幅の最小値

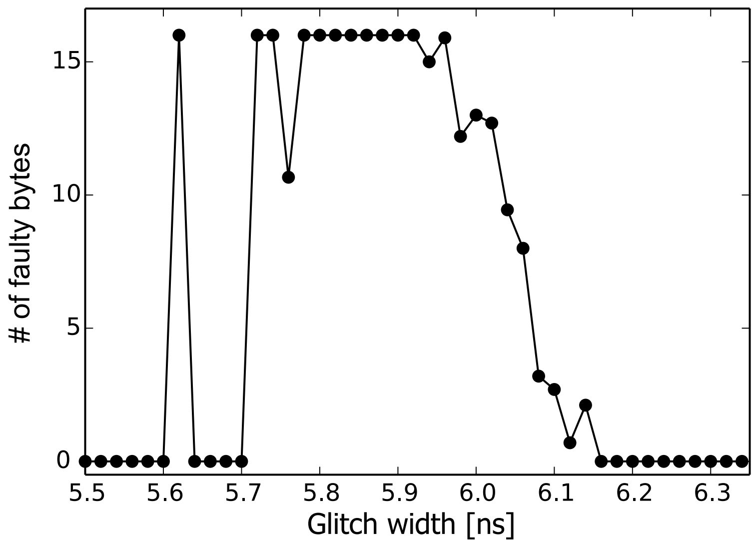


図 5.5 注入された故障のバイト数

を 5.64ns とした．ここで，各ステップにおける探索回数の詳細を示す．まず，予備故障の位置 p_p を探索した．この探索では p_p だけでなく，予備故障のグリッチ幅 w_p も同時に変化させた．w_p の探索の範囲は，10.2ns(刻み数 400) から 5.64ns(刻み数 159) とし，p_p は 0 から 1 ずつ増加させた．ステップ I の探索は，$p_p = 106$ および $w_p = 8.11$ns で完了し，探索完了までに要したサイクル数は，(400-159)+(400-290)+1=25657 回であった．ステップ II の探索も同様に行った．合計で 2378 $(= (32667 - 32658) \times (400 - 159) + (400 - 192) + 1)$ 回の故障注入で完了した．5.3.3 節で述べたように，ステップ III ではステップ I および II に比べ非常に多くの試行を要することから，粗密戦略により試行回数を削減する．ステップ III(a) では，グリッチ幅 w_A を 10.18ns(刻み数 400) から 5.66ns(刻み数 160) まで，分解能 0.188ns(刻み数 10) で変化させ，一部が誤りである暗号文を探索した．このステップは 417940 回 $(=(17448 - 107) \times (400 - 160)/10 + (400 - 190)/10)$ の試行で完了した．ステップ III(a) が完了した時の故障 A の位置 $p_A = 17448$ において故障を注入すると，AES の 9 ラウンド目の入力に故障を注入できることが分かる．ステップ III(b) では，故障 A のグリッチ幅 w_A を変化させて，全バイトが誤りの暗号文を得る．このステップは 216 回 (=400-185+1) 回の故障注入で完了した．

表 5.3 32 ビットマイコンにおける実験結果

Step	Parameters	Starting position	Final position	Number of trials
I	p_p	0	106	25657
	w_p	10.18ns (400)	8.11ns (290)	
II	p_B	32667	32658	2378
	w_B	16.20ns (400)	9.49ns (192)	
III(a)	p_A	107	17448	417940
	w_A	10.18ns (400)	6.23ns (190)	
III(b)	p_A	17448	17448	216
	w_A	10.18ns (400)	6.14ns (185)	
Total				446191

5.4.5 提案攻撃の応用

提案攻撃はアルゴリズム 1 に示した検算型対策に加え，様々な対策手法を無効化できる可能性がある．最初に，同じ演算を 2 回行う検算型対策への応用を考える．この場合は，攻撃アルゴリズム 2 から 4 を変更せずに適用できるだけでなく，ステップ II(アルゴリズム 3) を変更し，2 回の演算に同じ故障を生じさせて故障検知を回避することもできる．アルゴリズム 2 により予備故障の位置 p_p を発見した後に，アルゴリズム 3 において，故障 B の位置 p_B を p_p を起点として 1 ずつ増加させて探索する．ここでは予備故障と故障 B が同じ演算を誤らせた場合に暗号文を取得できる．さらに，ステップ III で差分故障解析に利用できる暗号文を得るには，予備故障の位置 p_p と故障 B の位置 p_B の時刻の間隔を一定に保ちつつ，故障注入の位置を探索する．

提案攻撃は命令を複製する対策も無効化できる可能性がある．コード 3 に命令を複製する対策の例を示す．コード 3 は，コード 2 のプログラムに対策を施した例である．コード 3 の対策は，4 行目の分岐命令 (BRNE) をスキップしても，5 行目の分岐命令が実行されることか

ら，1 回の故障注入による攻撃を防ぐ [19]．しかしこの対策は 4 行目と 5 行目の命令を両方ともスキップすると回避できる．また，この回避を行うための故障注入タイミングを探索することもできる．この場合は，2 回故障注入を行い，最初の故障の位置 p および最初の故障の位置から 2 番目の故障の位置までの間隔 i を同時に探索すれば良い．上記攻撃に要する故障注入回数は，n を暗号処理全体のコードの実行にかかるサイクル数とすると，$O(n^2)$ となる．

コード 3 命令複製の対策例

```
1: LBL_MAIN:
2: CALL AESEncWithRecal
3: TST r1
4: BRNE LBL_FAIL
5: BRNE LBL_FAIL
6: CALL SendData
7: JMP LBL_FINISH
8: LBL_FAIL:
9: CALL SendError
10: LBL_FINISH:
```

暗号処理の前後にランダムな時間の遅延を挿入する対策も提案攻撃に対して脆弱である．この対策では，乱数を生成し，ループカウンタに乱数を格納した上でループを実行することから，終了条件を満たさない場合にループの先頭へ戻る分岐命令をスキップすると，ループカウンタの値にかかわらず直ちにループを終了させることができる．攻撃者は，通信の信号などを通して暗号処理の時間を観測することにより，ループがスキップされていることを観測できる．ランダムな遅延が 1 回だけ挿入されている場合，ループをスキップすると暗号処理の時間が一定となる．一方，ランダムな遅延を暗号処理の前後に挿入し，全体の処理時間が一定となる実装では，一方のループをスキップすると暗号処理の時間が毎回ランダムに変化するようになる．上記対策を無効化した後は，差分故障解析に利用できる故障の注入タイミングを探索できる．

5.5　提案攻撃への対策手法

本節では，提案攻撃へのソフトウェアによる対策を述べる．対策の前提として，攻撃者は提案対策の存在を知らないものとする．暗号文に故障が生じると，提案対策も従来の検算型対策

と同様にエラー信号を出力することから，攻撃者は提案対策と従来の対策を判別できない．最初に，分岐命令をスキップする攻撃を考える．コード 4 に，分岐命令のスキップに耐性を持つコードを示す．コード 4 の 8 行目にある分岐命令をスキップすると，プログラムカウンタは 10 行目へ移動し，エラー信号出力ルーチンが呼び出されることから，誤った暗号文は得られない．このように，命令が正常動作しない場合にエラーとなるプログラム構成法は”Default fail” と呼ばれる [46]．さらに，コード 4 では出力ルーチンの呼び出しが暗号化の前にあることから，暗号化後に出力ルーチンへ到達するには，分岐命令で分岐されなければならない．これにより，暗号化と出力ルーチンの間の全ての命令をスキップする攻撃を防ぐ．

コード 4　Default fail による対策

```
1: JMP LBL_MAIN
2: LBL_SUCCESS:
3: CALL SendData
4: JMP LBL_FINISH
5: LBL_MAIN:
6: CALL AESEncWithRecal
7: TST r1
8: BREQ LBL_SUCCESS
9: LBL_FAIL:
10: CALL SendError
11: LBL_FINISH:
```

コード 5 提案対策

```
1: JMP LBL_MAIN
2: LBL_SUCCESS:
3: CALL SendData
4: JMP LBL_FINISH
5: LBL_TEST:
6: TST r1
7: BREQ LBL_SUCCESS ;jumps when Z=1
8: JMP LBL_FAIL
9: LBL_MAIN:
10: CALL AESEncWithRecal
11: CLZ            ;Z ← 0
12: BRNE LBL_TEST ;Go to LBL_TEST when Z=0
13: LBL_FAIL:
14: CALL SendError
15: LBL_FINISH:
```

しかしながら，コード 4 にも，故障注入に対する脆弱性がある．7 行目にあるテスト命令 (TST) をスキップすると，ゼロフラグ (Z) が更新されないことから，レジスタ r1 の値にかかわらず分岐命令によりラベル LBL_SUCCESS へジャンプする可能性がある．ゼロフラグの値は最後にゼロフラグを更新した命令に依存し，その値が 1 の時攻撃に成功する．この攻撃を防ぐには，ゼロフラグ初期化するルーチンを加えれば良い．コード 5 に，コード 4 に対策を施したコードを示す．暗号化の後，11 行目でゼロフラグを初期化する．初期化の値は，ゼロフラグが更新されない場合に出力ルーチンへ進まないような値とする．コード 5 では，ゼロフラグが 1 の時にジャンプする命令である BRNE が用いられているため，ゼロフラグを 0 に

初期化する．初期化の後，ゼロフラグが正しく初期化されている場合にのみテストルーチンへ移動する．コード 5 では，正しく初期化されている時 (ここでは Z=0 の時) テストルーチン LBL_TEST へ移動する．これによりテストルーチンへ進んだ後には Z=0 が保証される．このプログラムは，6 行目のテスト命令や 11 行目の初期化命令をスキップした場合には出力ルーチンへジャンプしないため，上記の攻撃を防止できる．

この対策は，フラグの操作と分岐命令によりプログラムフローを制御する多くのアーキテクチャで実装できる． さらに，アーキテクチャが特定の命令をサポートしている場合にはより効率的な対策ができる．例えば，ARM アーキテクチャのうち一部の命令セットでは，分岐条件の判定とジャンプを 1 命令で行う CBZ 命令がサポートされており，コード 4 においてテスト命令および分岐命令を CBZ 命令へ置き換えるのみで対策が完了する．

5.6　結び

本章では，組込みマイコンにおいて適応的な故障注入に基づく多重故障注入攻撃とその対策について述べた．提案攻撃は攻撃者がプログラムの詳細を知らないブラックボックスの条件で実行できる．提案攻撃は従来のレーザーによる攻撃と異なり，クロックグリッチを用いることから非侵襲で行うことができる．本章では，8 ビットおよび 32 ビットのマイコンに実装された典型的な検算型対策を施した AES ソフトウェアに対して，提案攻撃の実験を行った．実験結果より，攻撃者がプログラムの詳細を知らない場合でも，多重故障注入攻撃により誤った暗号文を得ることができることを示した．加えて，上記攻撃に対する対策を提案した．提案対策は，分岐命令や分岐判定命令をスキップして出力ルーチンの呼び出しへ到達する攻撃を阻止できる．

第 6 章

結言

以上，第 2 章から第 5 章まで，セットアップ時間違反を用いた故障注入攻撃に対して安全な暗号モジュールの実現を目的として，(a) 統一的な故障注入攻撃実験システムの開発，(b) 故障感度解析への対策技術の開発，(c) 多重故障注入攻撃の検討と対策技術の開発について述べてきた．

第 2 章では，暗号とその実装に関する基礎的考察を行った．まず，暗号とその実装技術に関する概要を述べ，暗号アルゴリズムに関する基礎的考察を行った．共通鍵暗号および公開鍵暗号の概念を述べ，それぞれの代表的な暗号アルゴリズムである AES 暗号および RSA 暗号の概要を述べた．また，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，物理攻撃について概説した．物理攻撃の中でも，近年特に注目を集めている故障利用攻撃について，その分類と故障注入手法，対策を述べた．

第 3 章では，セットアップ時間違反を利用する故障注入攻撃を想定した安全性評価を行うための実験システムを開発した．ここではセットアップ時間違反を用いる故障注入の中で最も高い制御性を有するクロックグリッチにより故障注入を行うシステムを開発した．従来のクロックグリッチ生成手法の問題点を述べ，それを解決する新たな生成手法を提案した．また，提案システムの評価として，SASEBO–G および SASEBO-W 上にクロックグリッチ発生回路を実装し，生成したグリッチの時間分解能や時間的なゆらぎ (ジッタ) の測定を通し，セットアップ時間違反を用いた故障注入攻撃に対する安全性評価を行うために十分な分解能と再現性を有

することを示した．

第 4 章では，故障感度解析への対策技術を開発した．まず，故障感度解析に対する対策回路を提案した．対策回路では，製造後に遅延時間を調整可能な遅延回路の遅延時間により故障感度を隠蔽する．故障感度の漏洩を防ぐには，攻撃者がクロック周波数と電源電圧を変化させた場合でも故障感度が漏洩しないよう遅延時間を適切に設定する必要があることから，与えられた条件の元で統計的に故障感度の漏洩を防ぐ遅延回路の設定方法を示した．また，提案対策を施した AES ハードウェアを ASIC 上に試作し，試作チップを用いた故障感度測定実験により提案対策の有効性を示すとともに，対策回路の回路面積を示し，十数 % 程度のオーバーヘッドで実装できることを示した．

第 5 章では，組込みマイコンにおける検算型対策を施した暗号ソフトウェアに対する攻撃の可能性を指摘した．ここでは，暗号化の後に復号を行い，復号結果が入力と等しいことを確認する検算型対策を対象として，適応的な故障注入を行い誤った暗号文を出力させる攻撃アルゴリズムを示した．また，検算型対策を施した AES プログラムを実行する 8 ビットおよび 32 ビットのマイコンに対して上記攻撃の実験を行い，誤った暗号文が取得できることを示した．さらに，上記攻撃への対策として，命令の配置を変更して探索を防ぐ手法を示した．

今後の展望としては，第 3 章で提案した故障注入回路を用いて，故障注入攻撃に対する定量的な安全性評価手法を確立することが挙げられる．提案回路を用いて構築した実験環境では再現性の高い実験が可能なため，秘密情報の推定に必要な暗号処理の試行回数や，制約条件の有無などにより安全性の定量化が可能となる．また，本論文で提案した対策のさらなる安全性評価を行うことも課題である．第 4 章で提案した故障感度解析対策では，暗号回路の遅延時間の分布に基づき許容できる遅延時間の最小値を求めることから，AES 以外の暗号アルゴリズムに対して提案対策を適用する場合，遅延時間の分布を調べる必要がある．また，第 5 章で示した多重故障注入攻撃対策については，対策を自動的に施すコンパイラの実装も挙げられる．

参考文献

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito, "The internet of things: A survey," *Computer Networks*, Vol. 54, No. 15, No. 15, pp. 2787 – 2805, 2010.

[2] Ross Anderson, Mike Bond, Jolyon Clulow, and Sergei Skorobogatov, "Cryptographic processors – a survey," No. UCAM-CL-TR-641, Aug. 2005.

[3] Stefan Mangard, Elisabeth Oswald, and Thomas Popp, *Power Analysis Attacks – Revealing the secrets of smart cards*, p. 1. Springer Science&Business Media, LLC, 2007.

[4] Research Center for Information Security (RCIS) of National Institute of Advanced Industrial Science and Technology, "SASEBO project overview." `http://satoh.cs.uec.ac.jp/SASEBO/en/board/index.html`.

[5] National Institute of Standards and Technology and Communications Security Establishment Canada, "Cryptographic module validation program, http://csrc.nist.gov/groups/stm/cmvp/index.html," 2009.

[6] A. Moradi, O. Mischke, and C. Paar, "One attack to rule them all: Collision timing attack versus 42 AES ASIC cores," *IEEE Transactions on Computers*, Vol. 62, No. 9, pp. 1786–1798, Sept. 2013.

[7] E. Trichina and R. Korkikyan, "Multi fault laser attacks on protected CRT-RSA," in *Proceedings of the 7th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Santa Barbara, CA, USA), pp. 75–86, Aug. 2010.

[8] NTT 情報流通プラットフォーム研究所, NTT R&D 情報セキュリティシリーズ最新暗号

技術. 株式会社アスキー, 2006.

[9] 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, Feb. 1978.

[10] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comp.*, Vol. 48, No. 177, pp. 203–209, Jan. 1987.

[11] NIST FIPS PUB. 197, "Advanced encryption standard (AES)." `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`.

[12] INSTAC, "平成 14 年度 耐タンパー性調査研究委員会報告書." `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/documents/INSTAC_rep.pdf`, 2003.

[13] Dan Boneh, Richard DeMillo, and Richard Lipton, "On the importance of checking cryptographic protocols for faults," *Advances in Cryptology - EUROCRYPT '97*, Vol. 1233, pp. 37–51, 1997.

[14] 独立行政法人情報処理推進機構, "暗号モジュール試験および認証制度 (JCMVP), http://www.ipa.go.jp/security/jcmvp/index.html," 2009.

[15] Eli Biham and Adi Shamir, "Differential fault analysis of secret key cryptosystems," in *Advances in Cryptology, CRYPTO '97*, vol. 1294 of *Lecture Notes in Computer Science*, pp. 513–525, 1997.

[16] M. Kara-Ivaniov, E. Iceland, and A. Kipnis, "Attacks on authentication and signature schemes involving corruption of public key (modulus)," in *Proceedings of the 5th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Washington, DC, USA), pp. 108–115, Aug 2008.

[17] S. M. Yen and M. Joye, "Checking before output may not be enough against fault-based cryptanalysis," *IEEE Transactions on Computer*, Vol. 49, No. 9, pp. 967–970, Sept. 2000.

[18] Y. Li, K. Ohta, and K. Sakiyama, "New fault-based side-channel attack using fault sensitivity," *IEEE Trans. Information Forensics and Security*, Vol. 7, No. 1, pp. 88–

97, Feb. 2012.

[19] Alessandro Barenghi, Luca Breveglieri, Israel Koren, Gerardo Pelosi, and Francesco Regazzoni, “Countermeasures against fault attacks on software implemented AES: effectiveness and cost,” in *Proceedings of the 5th Workshop on Embedded Systems Security (WESS)*, (New York, NY, USA), pp. 7:1–7:10, Oct. 2010.

[20] Gilles Piret and Jean-Jacques Quisquater, “A differential fault attack technique against SPN structures, with application to the AES and Khazad,” in *Cryptographic Hardware and Embedded Systems - CHES 2003*, vol. 2779 of *Lecture Notes in Computer Science*, pp. 77–88, Springer-Verlag Berlin Heidelberg, Sept. 2003.

[21] J. M. Schmidt and H. Christoph, “A practical fault attack on square and multiply,” in *Proceedings of the 5th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Washington, DC, USA), pp. 53–58, Oct. 2008.

[22] T. Fukunaga and J. Takahashi, “Practical fault attack on a cryptographic LSI with ISO/IEC 18033-3 block ciphers,” *Proceedings of the 6th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pp. 84–92, Sept. 2009.

[23] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall, and C. Whelan, “The sorcerer’s apprentice guide to fault attacks,” *Proc. of the IEEE*, Vol. 94, No. 2, pp. 370 –382, Feb. 2006.

[24] Y. Hayashi, N. Homma, T. Sugawara, T. Mizuki, and H. Sone, “Non-invasive trigger-free fault injection method based on intentional electromagnetic interference,” Sept. 2011.

[25] S. P. Skorobogatov and R. J. Anderson, “Optical fault injection attacks,” in *Cryptographic Hardware and Embedded Systems - CHES 2003*, vol. 2779 of *Lecture Notes in Computer Science*, pp. 2–12, Springer-Verlag Berlin Heidelberg, 2003.

[26] Marc Joye and Michael Tunstall, eds., *Fault Analysis in Cryptography.* Springer-Verlag.

[27] M. Medwed, F. X. Standaert, J. Großschädl, and F. Regazzoni, “Fresh re-keying:

Security against side-channel and fault attacks for low-cost devices," *Progress in Cryptology - AFRICACRYPT 2010*, Vol. 6055, pp. 279–296, 2010.

[28] A. Hariri and A. Reyhani-Masoleh, "Fault detection structures for the montgomery multiplication over binary extension fields," *Proc. 4th Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 37–46, Sept. 2007.

[29] M. Joye and S. M. Yen, "The montgomery powering ladder," in *Cryptographic Hardware and Embedded Systems - CHES 2002*, vol. 2523 of *Lecture Notes in Computer Science*, pp. 291–302, Springer-Verlag Berlin Heidelberg, Aug. 2002.

[30] N. Selmane, S. Bhasin, S. Guilley, and J.-L. Danger, "Security evaluation of application-specific integrated circuits and field programmable gate arrays against setup time violation attacks," *Information Security, IET*, Vol. 5, No. 4, pp. 181–190, December 2011.

[31] Weste Neil H.E. and David Harris, *CMOS VLSI Design*, p. 388. Pearson Education, 2005.

[32] T. Sugawara, D. Suzuki, and T. Katashita, "Circuit simulation for fault sensitivity analysis and its application to cryptographic LSI," in *Proceedings of the 9th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Leuven, Belgium), pp. 16–23, Sept 2012.

[33] Riscure BV, "VC glitcher datasheet." `https://www.riscure.com/documents/datasheet_vcglitcher.pdf?1376577654`.

[34] Yang Li, Kazuo Ohta, and Kazuo Sakiyama, "Toward effective countermeasures against an improved fault sensitivity analysis," *IEICE Trans. Fundamentals*, Vol. E95–A, No. 1, No. 1, pp. 234–241, 2012.

[35] Amir Moradi, Oliver Mischke, and Thomas Eisenbarth, "Correlation-enhanced power analysis collision attack," in *Cryptographic Hardware and Embedded Systems - CHES 2010*, vol. 6225 of *Lecture Notes in Computer Science*, pp. 125–139, Springer-Verlag Berlin Heidelberg, 2010.

[36] Akashi Satoh, Takeshi Sugawara, Naofumi Homma, and Takafumi Aoki, "High-performance concurrent error detection scheme for aes hardware," in *Cryptographic Hardware and Embedded Systems - CHES 2008*, vol. 5154 of *Lecture Notes in Computer Science*, pp. 100–112, Springer-Verlag Berlin Heidelberg, 2008.

[37] Amir Moradi, Oliver Mischke, and Christof Paar, "Collision timing attack when breaking 42 AES ASIC cores," *Cryptology ePrint Archive, Report 2011/162*, 2011. `http://eprint.iacr.org/`.

[38] Minoru Saeki, Daisuke Suzuki, Koichi Shimizu, and Akashi Satoh, "A design methodology for a DPA-resistant cryptographic LSI with RSL techniques," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, vol. 5747 of *Lecture Notes in Computer Science*, pp. 189–204, Springer-Verlag Berlin Heidelberg, 2009.

[39] A. Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-Box optimization," in *Advances in Cryptology - ASIACRYPT 2001*, vol. 2248 of *Lecture Notes in Computer Science*, pp. 239–254, Springer-Verlag Berlin Heidelberg, dec 2001.

[40] Dhiman Saha, Debdeep Mukhopadhyay, and Dipanwita RoyChowdhury, "A diagonal fault attack on the advanced encryption standard." Cryptology ePrint Archive, Report 2009/581, Nov. 2009.

[41] Amir Moradi, MohammadT.Manzuri Shalmani, and Mahmoud Salmasizadeh, "A generalized method of differential fault attack against AES cryptosystem," in *Cryptographic Hardware and Embedded Systems - CHES 2006*, vol. 4249 of *Lecture Notes in Computer Science*, pp. 91–100, Yokohama, Japan: Springer-Verlag Berlin Heidelberg, Oct. 2006.

[42] Amine Dehbaoui, Jean-Max Dutertre, Bruno Robisson, and Assia Tria, "Electromagnetic transient faults injection on a hardware and a software implementations of AES," in *Proceedings of the 9th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Leuven, Belgium), pp. 7–15, Sept. 2012.

[43] N. Moro, A. Dehbaoui, K. Heydemann, B. Robisson, and E. Encrenaz, "Electromagnetic fault injection: Towards a fault model on a 32-bit microcontroller," in *Proceedings of the 10th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Santa Barbara, CA, USA), pp. 77–88, Aug 2013.

[44] Mathieu Ciet and Marc Joye, "Practical fault countermeasures for Chinese remaindering based RSA," in *Proceedings of the 3rd Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 124–131, Sept. 2005.

[45] Subidh Ali, Debdeep Mukhopadhyay, and Michael Tunstall, "Differential fault analysis of aes using a single multiple-byte fault." Cryptology ePrint Archive, Report 2010/636, Dec. 2010.

[46] M. Witteman, "Secure application programming in the presence of side channel attacks." https://www.riscure.com/benzine/documents/Paper_Side_Channel_Patterns.pdf, Aug. 2013.

研究業績

学術論文

1. Sho Endo, Yang Li, Naofumi Homma, Kazuo Sakiyama, Kazuo Ohta, Daisuke Fujimoto, Makoto Nagata, Toshihiro Katashita, Jean-Luc Danger and Takafumi Aoki, "A silicon-level countermeasure against fault sensitivity analysis and its evaluation," *IEEE Trans. on Very Large Scale Integration Systems* (採録決定済).
2. Sho Endo, Naofumi Homma, Yu-ichi Hayashi, Junko Takahashi, Hitoshi Fuji, and Takafumi Aoki, "An adaptive multiple-fault injection attack on microcontrollers and a countermeasure," *IEICE Trans. Fundamentals*, vol. E98-A, no. 1, Jan. 2015.
3. Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh, "A configurable on-chip glitchy-clock generator for fault injection experiments," *IEICE Trans. Fundamentals*, vol. E95–A, no. 1, pp. 263–266, Jan. 2012.
4. Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh, "An on-chip glitchy-clock generator for testing fault injection attacks," *Journal of Cryptographic Engineering*, vol. 1, no. 4, pp. 265–270, Dec. 2011.

国際学会

5. Hajime Uno, Sho Endo, Yu-ichi Hayashi, Naofumi Homma, and Takafumi Aoki, "Electromagnetic analysis against cryptographic software on embedded OS," *2014*

International Symposium on Electromagnetic Compatibility, no. 14P2-A5, (Tokyo, Japan), pp. 313–316, May 2014.

6. Sho Endo, Naofumi Homma, Yu-ichi Hayashi, Junko Takahashi, Hitoshi Fuji, and Takafumi Aoki, "A multiple-fault injection attack by adaptive timing control under black-box conditions and a countermeasure," *5th International Workshop on Constructive Side-channel Analysis and Secure Design(COSADE)*, (Paris, France), Apr. 2014.
7. Yang Li, Sho Endo, Nicolas Debande, Naofumi Homma, Takafumi Aoki, Thanh-Ha Le, Jean-Luc Danger, Kazuo Ohta, and Kazuo Sakiyama, "Exploring the relations between fault sensitivity and power consumption," *4th International Workshop on Constructive Side-channel Analysis and Secure Design(COSADE)*, (Paris, France), Mar. 2013.
8. Sho Endo, Yang Li, Naofumi Homma, Kazuo Sakiyama, Kazuo Ohta, and Takafumi Aoki, "An efficient countermeasure against fault sensitivity analysis using configurable delay blocks," *Proc. of the 9th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, (Leuven, Belgium), pp. 95–102, Sept. 2012.
9. Sho Endo, Yu-ichi Hayashi, Naofumi Homma, Takafumi Aoki, Toshihiro Katashita, Yohei Hori, Kazuo Sakiyama, Makoto Nagata, Jean-Luc Danger, Thanh-Ha Le, and Pirouz Bazargan-Sabet, "Measurement of side-channel information from cryptographic devices on security evaluation platform: Demonstration of SPACES project," *Proc. of SICE Annual Conference*, (Akita, Japan), pp. 313–316, Aug. 2012.
10. Sho Endo, Takeshi Sugawara, Naofumi Homma, Takafumi Aoki, and Akashi Satoh, "An on-chip glitchy-clock generator and its application to safe-error attack," *2nd International Workshop on Constructive Side-channel Analysis and Secure Design (COSADE)*, (Darmstadt, Germany), pp. 175 – 182, Feb. 2011.

国内学会・研究会等

11. 遠藤翔, 本間尚文, 林優一, 高橋順子, 冨士仁, 青木孝文, “適応的にタイミングを制御した多重故障注入攻撃とその対策,” 2014 年暗号と情報セキュリティシンポジウム (SCIS), no. 3A1–1, (鹿児島市), Jan. 2014.
12. 宇野甫, 遠藤翔, 本間尚文, 林優一, 青木孝文, “組込み OS 上に実装された公開鍵暗号ソフトウェアに対する電磁波解析攻撃,” 2014 年暗号と情報セキュリティシンポジウム (SCIS), no. 3A2–3, (鹿児島市), Jan. 2014.
13. ヴィッレウリマウル, 遠藤翔, 本間尚文, 青木孝文, “LED 暗号ハードウェアに対する相関電力解析とその対策,” 情報処理学会第 75 回全国大会, no. 2Z-9, (仙台市), Mar. 2013.
14. 遠藤翔, 李陽, 本間尚文, 崎山一男, 藤本大介, 永田真, 太田和夫, 青木孝文, “故障感度隠蔽のための効率的な対策とその評価,” 2013 年暗号と情報セキュリティシンポジウム (SCIS), no. 1E1–5, (京都市), Jan. 2013.
15. Sho Endo, Naofumi Homma, and Takafumi Aoki, “Efficient countermeasure against fault injection attacks on modular exponentiation algorithms,” 平成 23 年度電気関係学会東北支部連合大会, no. 1A03, (多賀城市，宮城県), p. 3, Aug. 2011.
16. 遠藤翔, 本間尚文, 菅原健, 青木孝文, “べき乗剰余演算に対する故障注入を用いた電力解析攻撃,” 2011 年暗号と情報セキュリティシンポジウム (SCIS), no. 1D2–5, (北九州市), Jan. 2011.
17. 遠藤翔, 本間尚文, 菅原健, 青木孝文, “暗号プロセッサの故障解析システムに関する検討,” 多値論理とその応用研究会, no. MVL-11, (仙台市，宮城県), pp. 62–65, Jan. 2011.

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（青木研究室）において行った研究を取りまとめたものであります．本研究を推し進めるにあたり，多くの方々から御協力と御助言を頂きました．

恩師 青木孝文教授には，熱心な御指導と終始変わらぬ励ましを頂きました．先生の研究・教育に対する真摯な御姿勢から多くを学ばせて頂きたことを銘記し，ここに改めて深く感謝の意を表します．

本論文をまとめるにあたり，曽根秀昭教授ならびに遠藤哲郎教授より，それぞれの御専門の立場から大変有意義な御意見を賜りました．ここに深く感謝いたします．

本研究室の准教授である本間尚文博士には，著者が研究室へ配属されて以来，直接研究に対する熱心な御指導を頂きました．研究に関する様々な御助言や励ましを頂いただけでなく，生活面においても大変お世話になりました．ここに改めて深く感謝の意を表します．

また，在学中には多くの研究者の方々と共同研究させて頂く機会に恵まれました．共同研究を通じ，新たな洞察と有意義な研究成果を得ることが出来ました．皆様には改めて感謝の意を表すとともに，以下に名前を挙げさせていただきます．佐藤証教授（電気通信大学）崎山一男教授（電気通信大学），李陽特任助教（電気通信大学），片下敏宏博士（産業技術総合研究所），永田真教授（神戸大学），藤本大介博士（神戸大学），冨士仁博士，高橋順子博士（NTT セキュアプラットフォーム研究所）林優一准教授（東北大学），菅原健博士（現 三菱電機株式会社），宇野甫氏，上野嶺氏，梨本翔永氏，菅原幸弘氏，Ville Yli-Mayry 氏（青木研究室）．

インターンシップにおいて在籍した Telecom ParisTech の Jean-Luc Danger 教授，Shivam Bhasin 博士には，故障注入攻撃に関する様々なご意見やご協力を頂きました．ここにお礼申

し上げます.

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お礼申し上げます.

日本学術振興会特別研究員制度による助成をして頂いたことを感謝いたします.

最後に，長きにわたる研究生活を応援し支えてくれた両親に感謝の意を表し，本論文を結びます.

2015 年 1 月 20 日