暗号モジュールへの故障利用攻撃に対する安全性評価手法に関する研究

修 士 論 文

暗号モジュールへの故障利用攻撃に対する安全性評価手法に関する研究

情報基礎科学専攻

遠藤 翔

Security Evaluation Method for Fault Injection Attacks Against Cryptographic Modules

by

Sho ENDO

Graduate School of Information Sciences, Tohoku University, 2011

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，あらゆる機器がネットワークを介して結びつくユビキタス情報社会が構築されつつある．その中の基盤技術として，暗号技術がある．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，サービスの正真性の保証などの機能は，暗号技術により実現される．具体的な応用例としては，電子商取引などに用いられる 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，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．これまで，暗号技術は主にネットワーク上での利用を想定されてきたが，近年では携帯電話やスマートカードを始めとする組み込み用途への応用が進んでいる．組み込み機器では限られた電力，リソースで高速な暗号処理が求められるため，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．近年普及が進んでいる非接触 IC カードや RFID においては，消費電力およびリソースへの制約が特に厳しいため，暗号ハードウェアへの需要はさらに高まると考えられる．

一方，近年暗号装置に対する物理攻撃の危険性が指摘されている．物理攻撃とは，暗号アルゴリズムを実装した装置 (暗号モジュール) から物理的な方法で情報を抽出し，暗号解読を試みる攻撃である．このような攻撃は暗号アルゴリズムの設計段階では考慮されないため，一般に安全とされているアルゴリズムであっても解読される可能性がある．物理攻撃には様々な手法が存在し，LSI のパッケージを開封して内部信号および回路の観察を行う開封攻撃や，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) を計測するサイドチャネル攻撃や，暗号処理に誤りを注入し秘密情報の漏えいを誘発する故障利用攻撃などが代表的である．中でもサイドチャネル攻撃および故障利用攻撃は，暗号モジュールの設計に関する詳細な知識なしで実行可能な場合があり，攻撃に要する設備も比較的安価であることから，現実的脅威であると考えられる．特に組み込み機器では，コストのかかる対策を取ることが難しい上に，暗号モジュールが攻撃者自身により所持されることが多いため，攻撃の実行が非常に容易である．

第 1 章 緒言

暗号モジュールの安全性を確保するためには，物理攻撃に対する安全性を正しく評価することが重要である．物理攻撃は暗号モジュールの物理的な脆弱性を利用するため，攻撃の危険性および対策の効果は実装に大きく依存する．例えば暗号モジュールの消費電流を測定・解析することにより行われる電力解析攻撃は，攻撃の成否が暗号モジュールを搭載するチップの種類 (FPGA，ASIC など) や，チップを搭載するボードにより変化するため，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，物理攻撃の評価手法としては，理論的な攻撃の可能性だけでなく，実際の暗号装置における攻撃の可能性を明らかにすることが重要である．以上のことから，物理攻撃に対する暗号モジュールの安全性を評価するためには，実モジュールを用いた実験的な評価の実験技術が不可欠と考えられている．

しかしながら，従来から用いられている安全性評価手法では，安全性の定量的な評価は困難であるという問題があった．従来では，各研究者が独自に構築した実験環境を用いて，暗号モジュールに対する攻撃を試行し，秘密情報の推定に必要なコスト (暗号処理の試行回数，電流波形の数，攻撃時間など) を測定することにより安全性を評価していた．この方法では，安全性評価の結果が実験環境に依存するため，統一的な基準で安全性を評価することができなかった．そこで，サイドチャネル攻撃の分野では，東北大学および産業技術総合研究所により，サイドチャネル攻撃標準評価ボード (Side-channel Attack Standard Evaluation BOard: SASEBO) [1] が開発された．SASEBO は暗号モジュールと制御用 FPGA を搭載した基板であり，暗号モジュールに流れる電流波形取得用の端子を備える．暗号モジュールとしては ASIC，FPGA，スマートカードが利用可能である．SASEBO を用いることで実験環境を共通化することができ，統一的な基準で安全性を評価することが可能となる．SASEBO は現在，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．

一方で，故障利用攻撃に対しては，現状では統一的な実験環境は存在しないため，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，ある暗号モジュールに対して，ある条件下での攻撃可能性を評価することは可能であるが，実験を通してどの程度攻撃が容易であるかといった評価はできないのが現状である．

そこで本論文では，故障利用攻撃に対する暗号モジュールの安全性評価手法を確立することを目的に，故障利用攻撃の統一的な実験環境に関する検討を行う．具体的には，故障注入の実験を行うための単一の LSI 上に実装可能な故障注入回路を提案する [2]．本回路は単一の FPGA チップ上に実現されるため， FPGA を搭載した SASEBO を想定した場合に，外部機器を用いずに故障利用攻撃の実験が可能である．また，故障注入のタイミングを精密に制御することが容易である．これにより，再現性の高い統一的な故障利用攻撃の安全性評価環境を実現できる．本回路ではセットアップ時間違反による故障を再現す

るため，制御性および再現性の高いクロックグリッチを用いる．従来のクロックグリッチを用いた安全性評価実験では，外部の信号発生器により生成したクロック信号を用いて位相の異なるクロック信号を生成し，それらを合成することによりグリッチを生成していた．本回路では現在の FPGA に広く搭載されているクロック管理機能を用いて位相の異なるクロック信号を生成することにより，外部機器を不要としている．本論文では SASEBO にオンチップ故障注入回路を実装し，評価を行う．生成されたグリッチの時間分解能やジッタの測定を通し，故障利用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．

本論文は，以上の内容を取りまとめたものであり，以下に示す章で構成する．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，暗号モジュールの安全性について述べる．また，物理攻撃およびその一種である故障利用攻撃について述べる．第 3 章では，オンチップで実装できる故障注入回路を用いた安全性評価手法を提案する．まず，故障注入の原理およびオンチップ故障注入回路の設計を示す．次に，実装したグリッチ挿入回路の性能評価を行い，安全性評価に適することを示す．第 4 章では，オンチップ故障注入回路を用いた故障利用攻撃に対する安全性評価について述べる．ここでは安全性評価の具体例として，RSA 暗号に対する Safe error 攻撃を想定した実験を行う．まず，べき乗剰余演算アルゴリズムおよびそれらに対する攻撃について述べる．そして，実装した回路を用いて安全性評価実験を行う．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

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，暗号モジュールに対する物理攻撃について，その分類と対策について述べる．最後に，サイドチャネル攻撃標準評価ボード (SASEBO) を用いた安全性評価について述べる．

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づ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 [3]．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．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，通信内容の秘匿が可能となる．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 (RNG: 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．第三者による鍵の推定を防ぐため，暗号技術における RNG が出力する系列には，値に偏りが無いだけではなく，それまでの履歴から次の値が予測できないという性質が求められる．

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．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．これは，公開鍵から秘密鍵を求めることは数学的未解

決問題を解くことと等価であることを根拠としている．現在の公開鍵暗号方式で使用されている主な数学的未解決問題としては，素因数分解問題，有限体上の離散対数問題，および楕円曲線上の離散対数問題がある．これらの問題を用いた暗号としては，それぞれRSA暗号 [4]，楕円曲線暗号 [5] が挙げられる．

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，DaemenとRijmenにより提案された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 [6]，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．2000年にNISTによって標準アルゴリズムに策定されており，次世代の標準アルゴリズムとして最も有力なアルゴリズムである．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，4種類の基本演算である *SubBytes, ShiftRows, MixColumns,* および *AddRoundKey* を行列に対して繰り返し適用することで行う．鍵長128ビットでは，暗号化・復号に10ラウンドを要する．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による鍵加算である．一方鍵スケジュールは，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．*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 [4] はRivest，Shamir，Adlemanにより考案された公開鍵暗号である．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．

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

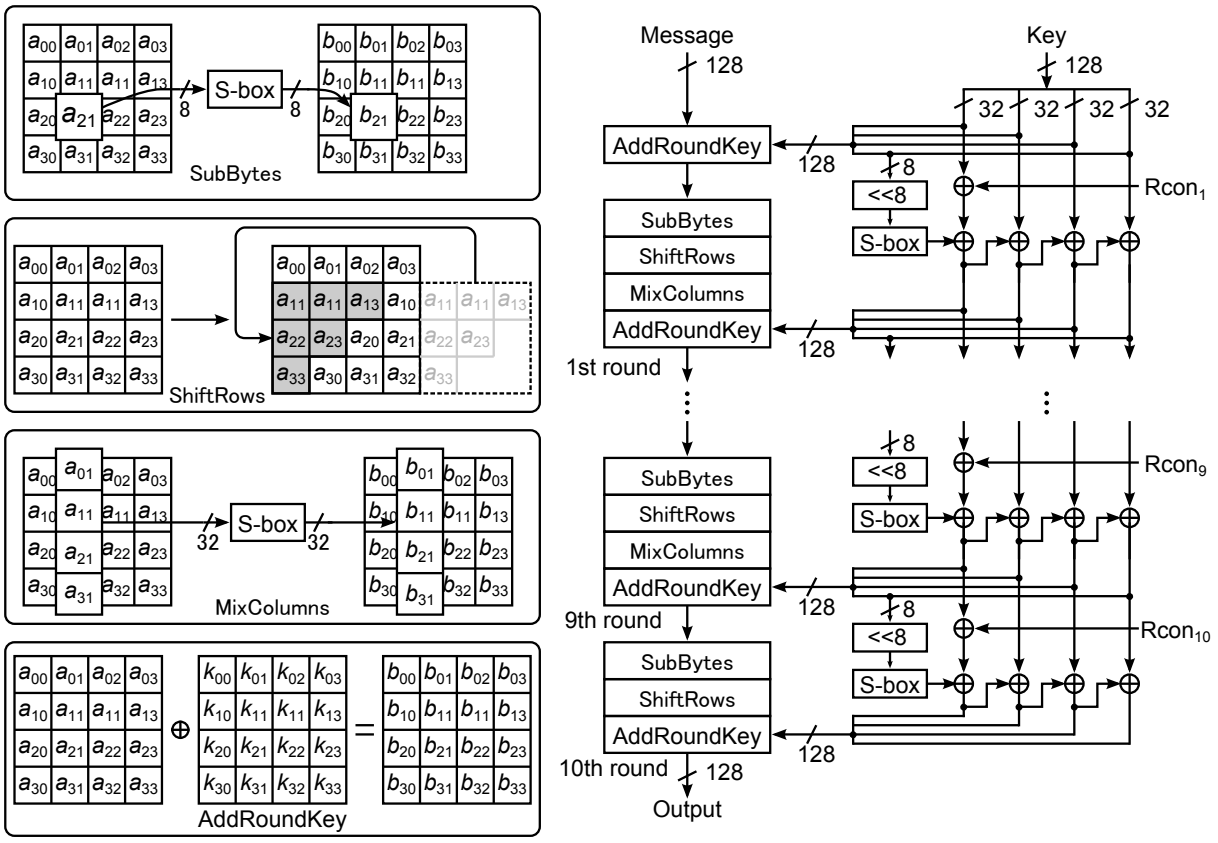


図 2.1　AES の暗号化アルゴリズム

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

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．既知の攻撃方法に対して，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．

2.2.5 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により，

CPU や Digital Signal Processor (DSP) 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．本稿では，専用 LSI (ASIC) だけでなく，Field Programmable Gate Array (FPGA) などの再構成型 LSI への実装もハードウェア実装とよぶ．以下では両者の特徴について述べる．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，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．これについては 2.3 節で詳細に述べる．

2.3　物理攻撃に関する基礎的考察

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．本節では，暗号モジュールの物理攻撃について，概要，分類，対策法の順に述べる．

2.3.1　物理攻撃の概要

現在製品に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．しかしながら，暗号アルゴリズムの安全性が保証されていたとしても，暗号モジュールの実装によっては物理攻撃を用いた解読の危険性がある．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．しかしながら，物理攻撃では，攻撃者が物理的な暗号装置に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，消費電流やチップからの漏えい電磁波の測定により，解読に必要な情報を得る．現状では，攻撃者の物理的なアクセスに関しては暗号モジュールの設計段階で考慮されていないため，上述した推奨暗号であっても物理攻撃に対して脆弱になりうる．

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

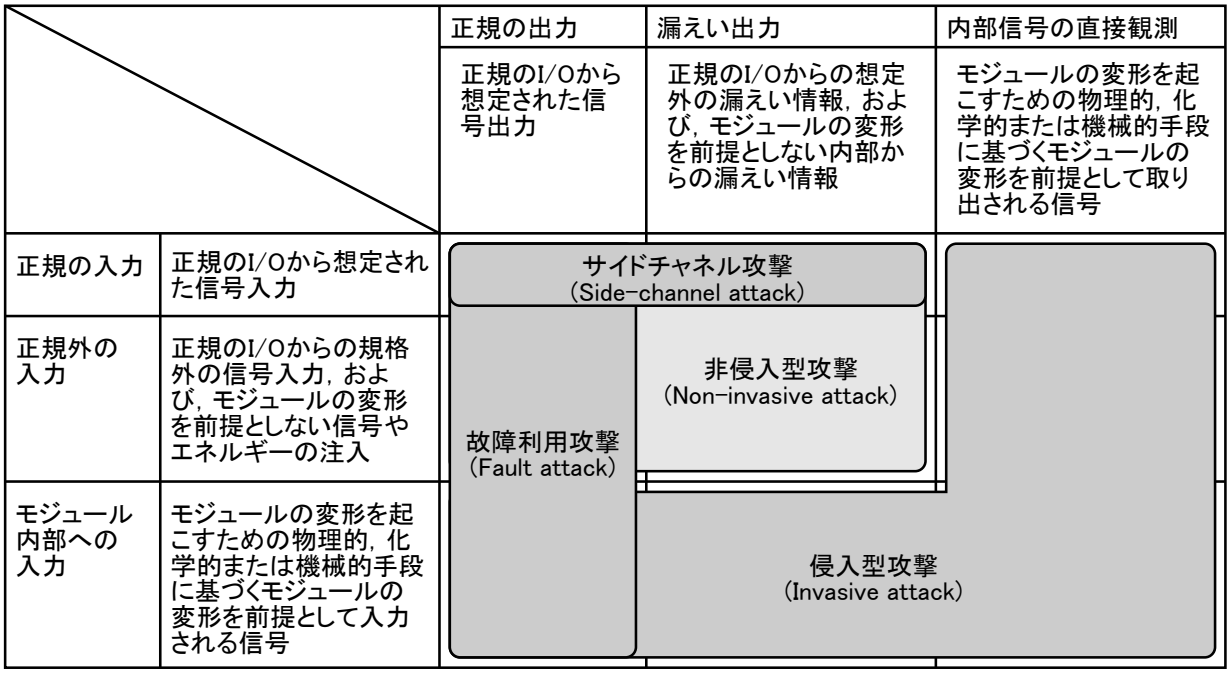


図 2.2 物理攻撃の分類:文献 [7] の表 1.1 および図 1.2 を文献 [3] を参考にして加工したもの

合には，物理攻撃は問題とならない．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．

図 2.2 に，文献 [7] 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ため，重要性の低い秘密情報に関しては攻撃の危険性は低いといえる．重要性の高い情報を扱う機器に関しても，侵入検知メッシュや光センサーを搭載することで攻撃を検知できるため，攻撃された際に秘密情報を消去するなどの対策を比較的容易に取ることができる．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，故障利用攻撃 (Fault attack) ，サイドチャネル攻撃 (Side-channel attack) ，およびこれらを組み合わせた攻撃がある．ただし，故障利用攻撃は非侵入型の攻撃だけでなく，侵入型の攻撃手法も存在する．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，秘密情報の漏えいを誘起する攻撃である．これらの攻撃は，オシロスコープや PC 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 2 回行うのみで秘密情報の奪取が可能である [8] 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．さらに，故障注入により漏えいしたサイドチャネル情報を観測することにより，従来のサイドチャネル攻撃および故障利用攻撃に対する対策法が無効化される恐れもある．このことから，故障利用攻撃は近年注目を集めている．

2.3.2 故障利用攻撃の分類

故障利用攻撃は，故障の種類および注入手法によりそれぞれ分類される．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．また，過渡故障は一般に検知が困難であり，攻撃の痕跡も残らないことから，故障利用攻撃の脅威としてしばしば想定される．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．

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 [9]， (ii) クロック周波数の変化 [10] ， (iii) 温度変化 [11] ， (iv) 電磁波照射 [12]， (v) 光照射 [13]，(vi) レーザ照射 [11] が挙げられる．表 2.1 に故障注入手法の比較を示す．

この中で，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，電磁波，光，レーザ照射では，電磁波によりメモリまたは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

表 2.1 故障注入手法の比較

故障注入手法	制御性		コスト
	時間的	空間的	
(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	×	×	○
(v) 白色光照射	○	◎	○
(vi) レーザ照射	○	◎	×

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非侵襲の故障注入手法である．(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．これを利用して，クリティカルパスを外部から測定することにより，クリティカルパスのデータ依存性を利用した攻撃 [14] も可能となる．このため，故障利用攻撃の中で最も現実的な攻撃といえる．さらに，クロック周波数を変化させる故障注入回路は，オンチップで実現可能である．そこで，本論文では，クロック周波数を変化させて故障を注入するオンチップ故障注入回路を実装する．

2.3.3 故障利用攻撃への対策

故障利用攻撃への対策としては，(i) プロトコルレベルでの対策，(ii) アルゴリズムレベルでの対策，(iii) 回路レベルでの対策がある．(i) 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 [15] 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．(ii) 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 [16] 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 [17] などが挙げられる．(iii) 回路レベルでの対策としては，クロック周波数の異常を検知する回路 [18] が挙げられる．

2.4 サイドチャネル攻撃標準評価ボード SASEBO による暗号モジュールの安全性評価

本論文では，オンチップ故障注入回路をサイドチャネル攻撃標準評価ボード (Side-channel Attack Standard Evaluation BOard: SASEBO) [1] に実装し，生成した波形の再現性を示す．本節では，SASEBO の概要について述べる．

2.4.1 暗号モジュールの安全性評価の概要

暗号モジュールの物理攻撃に対する耐性は，アルゴリズムだけでなく実装に依存する．このため，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．

実機による安全性評価の方法としては，脅威として想定した攻撃を実際に試行し，秘密情報の推定に成功したかどうかにより安全性を判定することが多い．このため，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．

2.4.2 SASEBO の概要

SASEBO はサイドチャネル攻撃に対する安全性評価を目的として開発されたボードである．現在 SASEBO–R，G，B，GII，W の 5 種類のボードが存在する．表 2.2 に示すように，ボードの種類により搭載する LSI が異なり，あらゆるプラットフォームを想定した実験が可能である．

SASEBO はレベルの高い攻撃者を想定した安全性評価を行うことを可能にするため，サイドチャネル情報を高精度に測定するための設計となっている．まず，暗号処理用と制御用の 2 つの LSI を備える．制御用は主に PC との通信に用いられ，暗号処理用の 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．また，サイドチャネル情報を高精度に測定す

表 2.2 SASEBO ボードの仕様

SASEBO の種類	暗号 LSI の種類
SASEBO–G	FPGA (Xilinx XC2VP7)
SASEBO–R	ASIC (着脱可能)
SASEBO–B	Altera EP2S30
SASEBO–GII	Xilinx XC5VLX30
SASEBO–W	IC カード (着脱可能)

るため，ノイズを低減する設計となっている．これにより，安全性評価において特にレベルの高い攻撃者を想定した実験が可能である．

2.4.3 SASEBO を用いた故障利用攻撃実験

SASEBO はサイドチャネル攻撃を想定して設計されているが，故障利用攻撃の実験装置として用いた場合にも利点がある．制御用 FPGA にオンチップ故障注入回路を実装することにより，外部機器なしで実験が可能となる．さらに，SASEBO は本来サイドチャネル攻撃を評価するためのボードであるため，サイドチャネル攻撃と故障利用攻撃を組み合わせた攻撃に対する安全性評価実験にも適している．

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，暗号とその実装技術に関する概要を述べ，暗号アルゴリズムのハードウェア実装の必要性を述べた．続いて，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．最後に，SASEBO の概要を述べ，SASEBO は故障利用攻撃の実験にも有用であることを述べた．

第3章

オンチップ故障注入回路の提案

3.1　はじめに

本章では，オンチップで実装できる故障注入回路を提案する．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る．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す．さらに，提案手法の評価として，SASEBO 上にオンチップ故障注入回路の特性を測定し，本回路が安全性評価実験の要件を満たすことを示す．さらに，加算回路に対する故障注入実験を行い，提案回路により正確な故障注入が可能であることを示す．

3.2　クロック周期の操作による故障注入の原理

本論文で提案する故障注入回路では，クロック周期を変化させることにより故障を注入する．同期式の論理回路は，図 3.1 に示すように，組み合わせ回路と順序回路から構成される．ここで，あるフリップフロップから次のフリップフロップまでデータが伝わる時間は，$T_{prop} = t_c + t_s$ で表される．ただし，t_c は組み合わせ回路のクリティカルパスの遅延時間，t_s はフリップフロップのセットアップ時間である．このため，クロック周期 T_{clock} は $T_{clock} > T_{prop}$ となるよう設定される．$T_{clock} \leq T_{prop}$ とすると，D フリップフロップが入力をラッチするために必要な時間が確保されないため，記憶内容に誤りが生じる．クロック周期を変化させる故障利用攻撃では，これを利用してフリップフロップに誤りを注入する．クロック周期を操作する方法の中でも，クロック信号にグリッチを挿入する方法が広く用いられている [11]．本論文ではグリッチを用いた故障注入を行う．

図 3.2 にグリッチのパラメータを示す．図 3.2 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，それぞ

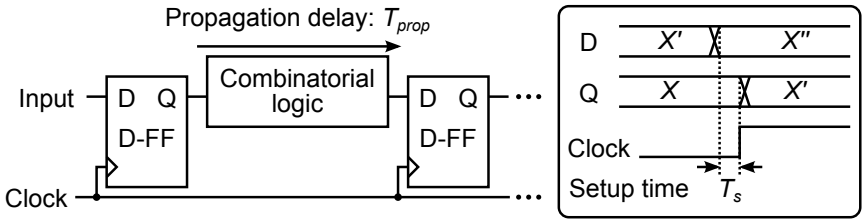


図 3.1 同期式論理回路

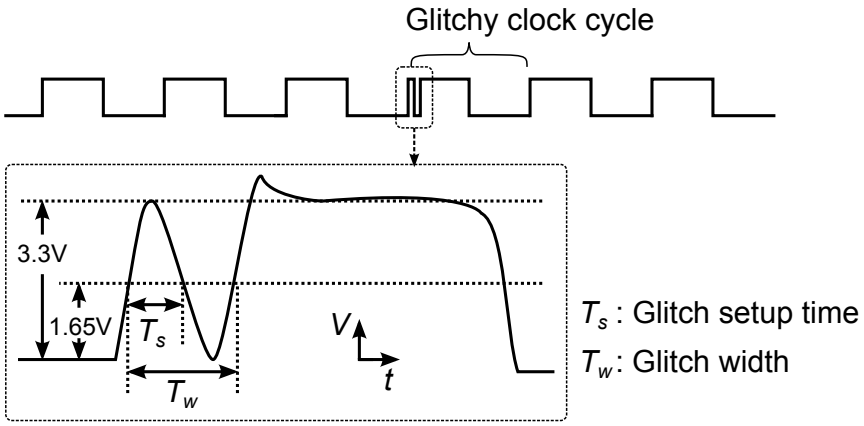


図 3.2 グリッチのパラメータ

れグリッチ整定時間，グリッチ幅とよぶ．

3.3 従来のグリッチ生成手法

本節では，文献 [10] で提案されているグリッチ生成手法（以下，従来手法）について述べる．従来手法では，図 3.3 に示すように，位相の異なる 2 つのクロック信号を合成することにより，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 3.4 が提案されている．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’ 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を FPGA へ出力する．これにより，グリッチ生成のタイミングを制御することができる．信号 S は，オシロスコープの出力に同期して立ち上がり，クロック信号に同期して立ち下がる信号であり，FPGA 内部で生成される．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，グリッチが形成される．

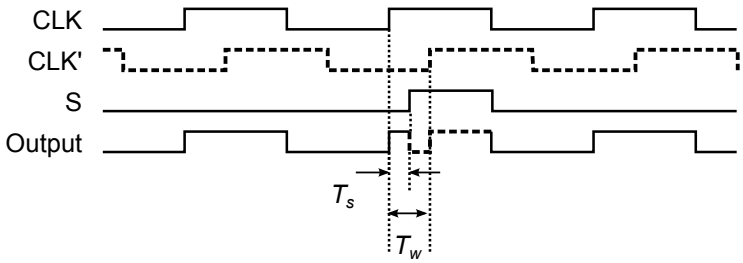


図 3.3　従来のグリッチ生成方法

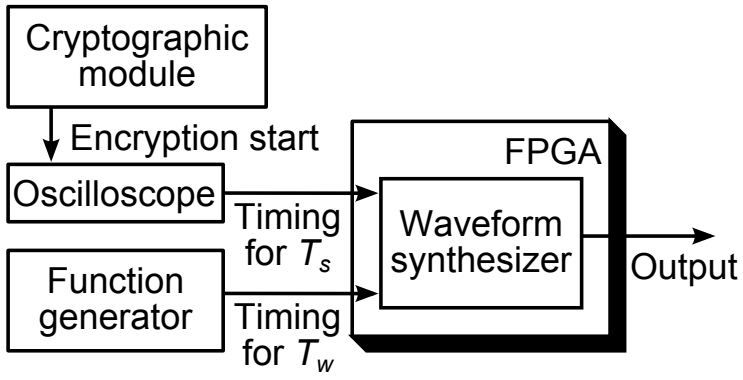


図 3.4　文献 [10] におけるグリッチ生成回路

3.4　オンチップ故障注入回路の実装

3.4.1　故障注入の要件

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 [14] に対応した回路を実装することを考える．

故障感度解析の実験では，グリッチ幅の分解能 Δ が 0.035 ns 以下 [19]，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 [20]．そこで，本論文では，以上の条件を満たすオンチップ故障注入回路を実装する．

3.4.2　グリッチ生成回路の実装

本節では，単一の FPGA 上に実装可能なグリッチ生成回路を提案する．最も基本的な実装として，図 3.4 における外部機器を FPGA 上の DLL (Delay Locked Loop) へ置き換えた構成が考えられる． 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

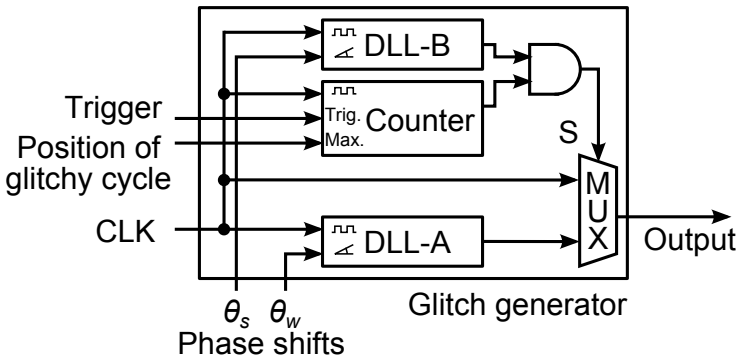


図 3.5 基本的なグリッチ生成回路

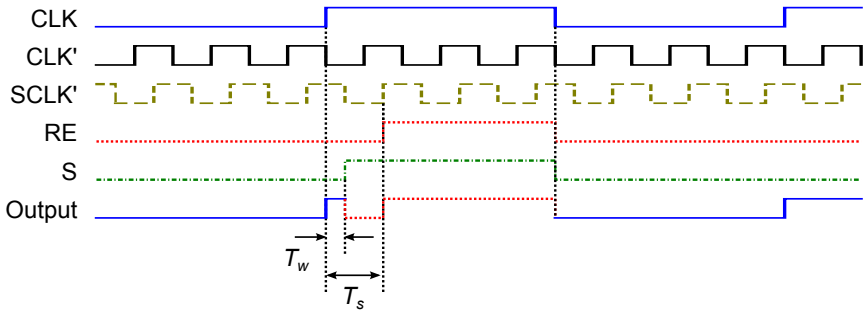
も本回路の実装は容易である．

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．本回路では，カウンタにより一定時間を作り出し，DLL–B により生成された位相の異なるクロックとの論理積を取ることにより，信号 S を生成する．DLL を設定することにより，位相シフトのパラメータ θ_w および θ_s を制御できる．

しかしながら，図 3.5 の回路には，DLL の仕様により出力周波数の範囲が制限されるという問題がある．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分解能 100 ps 以下を達成することができない．

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，タイミングの制御のみに用いることにより，任意の周波数におけるグリッチ生成を可能とする．図 3.6(a) 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 3.8 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



(a) タイミング図

図 3.6　提案手法のタイミング図

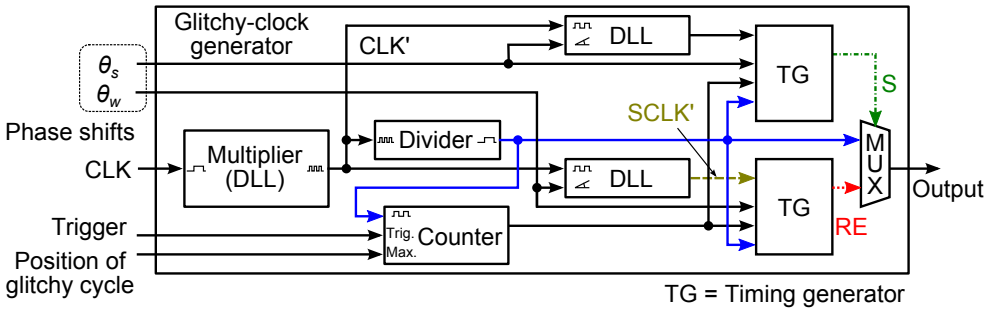


図 3.7　オンチップ故障注入回路のブロック図

同じタ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．

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

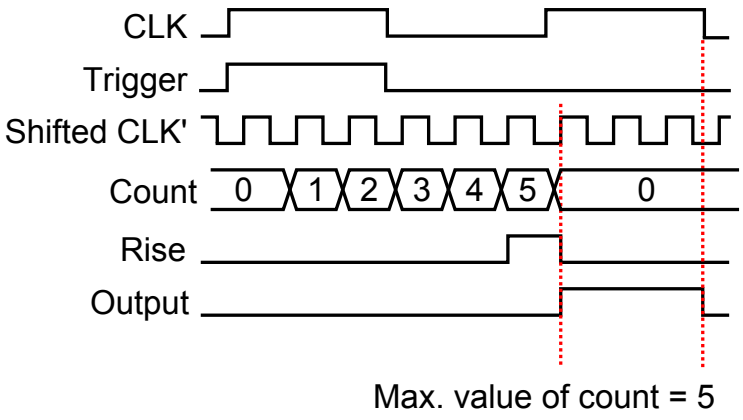


図 3.8 タイミング生成器内部のタイミングチャート

とした.

3.5 オンチップ故障注入回路の評価

オンチップ故障注入回路を SASEBO 上に実装し，基本的な特性を評価する．ここでは SASEBO–G の制御用 FPGA (Xilinx 社製 XC2VP30) に提案回路を実装し，オシロスコープ (Tektronix 社製 DPO7254) で測定を行った.

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．以降の実験および波形の取得は，すべて図 3.9 の実装を用いて行う.

3.5.1 グリッチ幅 T_w の設定範囲および分解能

ここでは，グリッチ整定時間 T_s を固定し，T_w を変化させたときの提案回路の出力波形を示し，T_w の設定範囲および分解能を評価する．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表示している.

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．オシロスコープを用いて，T_w を 4.9 ns –17.9 ns まで変化させたときのグリッチの波形を観

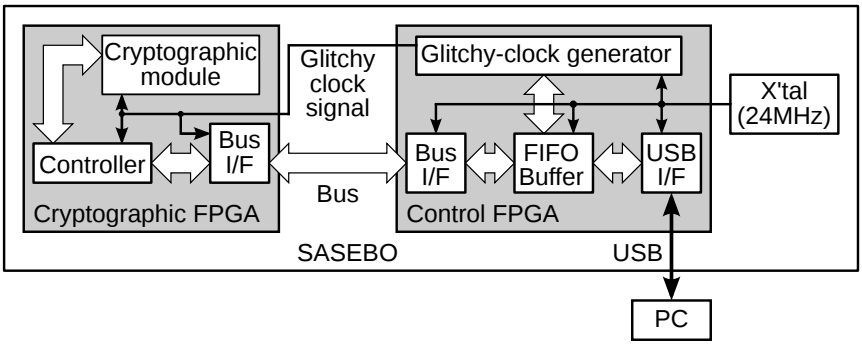


図 3.9　SASEBO–G 上に実装したオンチップ故障注入回路

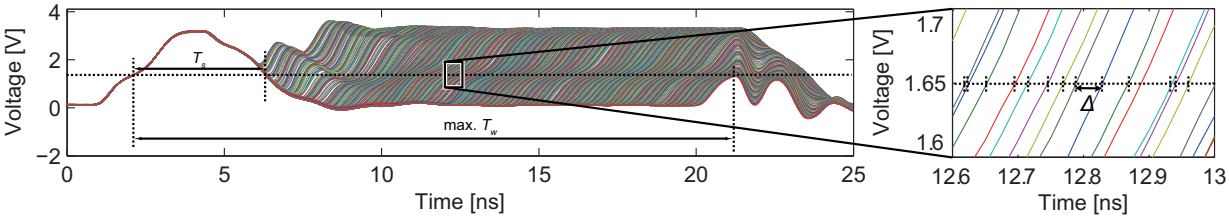


図 3.10　異なるグリッチ幅の設定値におけるオンチップ故障注入回路の出力波形

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り，仕様上 0.020 – 0.040 ns となっているため，妥当な値である．

図 3.10 では，T_w が大きい場合と小さい場合に波形が乱れている．これは，SASEBO 上のクロック信号の配線が共振を起こしているためと考えられる．波形の乱れは，波形の立ち上がりエッジおよび立ち下がりエッジが連続する場合にのみ起こる．このときの波形は 500 MHz の正弦波の一部が現れるため，500 MHz 付近の周波数成分を多く含んでいる．このことから，SASEBO 上では 500 MHz で共振を起こすと考えられる．

3.5.2　グリッチ幅の再現性の評価

本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

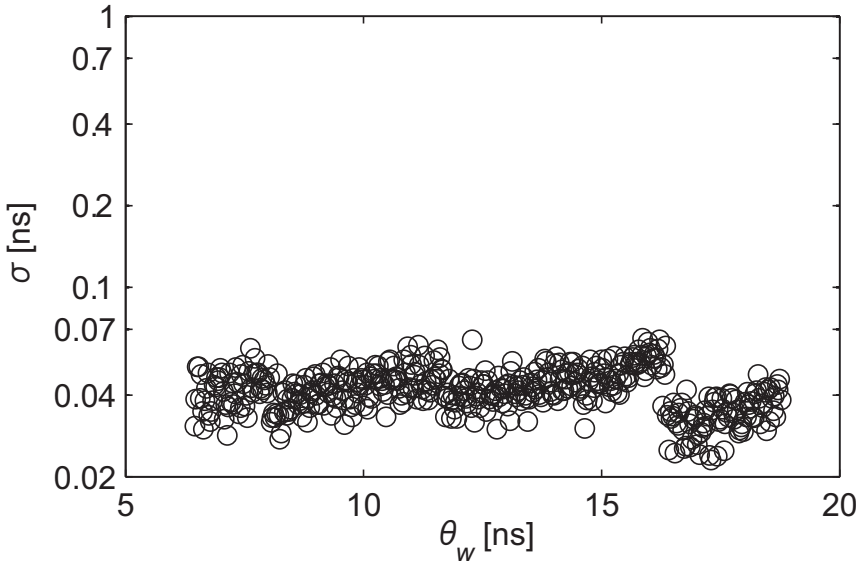


図 3.11 $T_s = 4.12$ ns における標準偏差 σ の変化

差 σ を用いる．

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．σ は概ね 0.050 ns 以下に収まっている．この値は要件である 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．

3.6 提案回路による FPGA への故障注入実験

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う．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

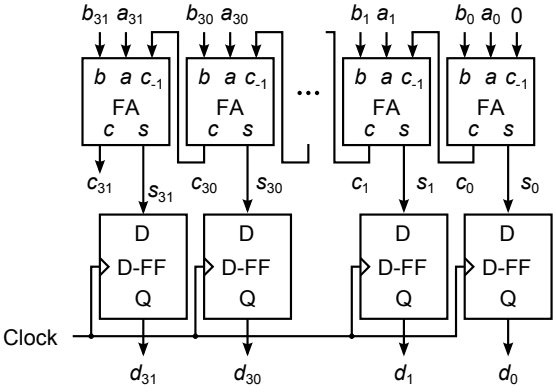


図 3.12　故障注入実験に用いる加算回路

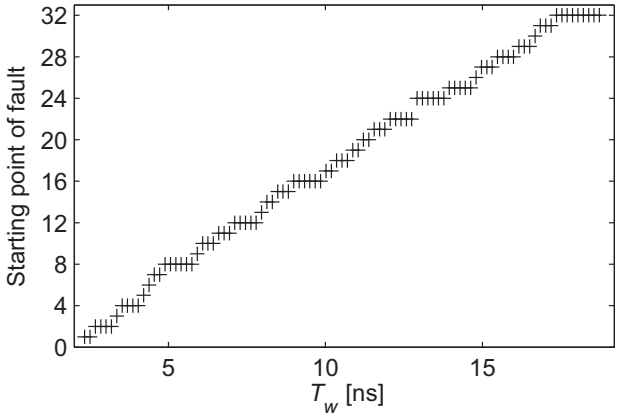


図 3.13　グリッチ幅 T_w と誤り開始点の関係

位置が下位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 ns – 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．

3.7　結び

本章では，オンチップで実装できる故障注入回路を提案した．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 上にオンチップ故障注入回路の特性を測定し，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

RSA暗号モジュールの安全性評価実験

4.1 はじめに

本章では，第3章で実装したオンチップ故障注入回路を用いて，RSA暗号モジュールの安全性評価実験を行う．具体的には，RSA暗号に対するSafe error攻撃を脅威として想定し，この攻撃により秘密情報の推定を試みる．本論文では，攻撃が比較的容易であり，攻撃の検知が難しいことから，危険性が高いと考えられるSafe error攻撃を取り上げる．また，RSA暗号は暗号化に要する時間が長いため，オンチップ故障注入回路による暗号モジュールとの同期機能が最大限に活用される．

本章では，まずRSA暗号の中心的な演算であるべき乗剰余演算アルゴリズムについて述べる．その後，べき乗剰余演算に対するSafe error攻撃，電力解析を用いたSafe error攻撃，実験条件および結果について述べる．以下では，乗剰余算を乗算，自乗剰余算を自乗算とよぶ.

4.2 べき乗剰余演算アルゴリズム

べき乗剰余演算は，入力をX，指数をE，法をNとすると，式(4.1)で表される．

$$Y = X^E \bmod N \tag{4.1}$$

べき乗剰余演算は，乗算および自乗算の繰り返しにより求めることができる．古典的なアルゴリズムである左バイナリ法を**アルゴリズム 4.1**に示す．左バイナリ法は指数ビットの左側から1ビットごとに剰余を求めていく方法である．ここで，指数Eは式(4.2)で表される．

$$E = (e_{k-1}, ..., e_1, e_0)_2 = \sum_{i=0}^{k-1} e_i 2^i \tag{4.2}$$

アルゴリズム 4.2
左バイナリ法

Input:	X, N, $E = (e_{k-1}, ..., e_1, e_0)_2$
Output:	$X^E \bmod N$

```
1 :  R := 1;
2 :  for i = k − 1 downto 0 do
3 :     R := R · R mod N;          — squaring
4 :     if e_i = 1 then            — multiplication
5 :     R := R · X mod N;          — multiplication
6 :     end if
7 :  end for
8 :  return R
```

アルゴリズム中の R は中間変数を表す．本アルゴリズムでは，最上位ビット ($i = k-1$) から順に指数ビットをスキャンし，現在の指数ビット e_i が 0 ならば自乗算，1 なら自乗算および乗算を行う．指数ビット全てについて上記の処理を行った後，R に格納されている値が最終出力となる．

しかしながら，指数が秘密情報である場合，本アルゴリズムは電力解析に対して脆弱であるという問題がある．左バイナリ法では，指数ビットと演算の種類 (自乗算および乗算) に依存性があるため，演算の種類を同定することができれば秘密の指数を推定することが可能である．本アルゴリズムの FPGA 実装に対して，単純電力解析 (Simple Power Analysis: SPA) を行った例を図 4.1 に示す．S は自乗算，M は乗算を表す．この例では，高基数モンゴメリ乗算に基づく RSA 暗号モジュール [21] を SASEBO–G の暗号用 FPGA (XC2VP7) に実装し，指数の上位 7 ビットを $(1110010)_2$，平文として $X = 2^{-1023}$ を入力として与えたときの消費電流の波形を示している．電力解析という名称は，文献 [22] で用いられて以来慣習的に用いられているが，実際には暗号モジュールに流れる電流波形を用いる．この平文の値は，高基数モンゴメリ乗算回路において，乗算および自乗算の間で消費電流に大きな差が生じる条件である．図 4.1 では，演算の種類に応じて消費電流が変化しており，秘密の指数が推定できていることが分かる．

この攻撃への対策として，Squaring-and-multiply always 法 が提案されている．これは左バイナリ法において指数ビットが $e_i = 0$ のとき，出力結果に影響しないダミーの乗算を実行するアルゴリズムである．**アルゴリズム 4.2** に Squaring-and-multiply always 法を示す．本アルゴリズムでは秘密の指数に関係なく常に自乗算および乗算が実行されるため，SPA による指数の推定を防ぐことができる．

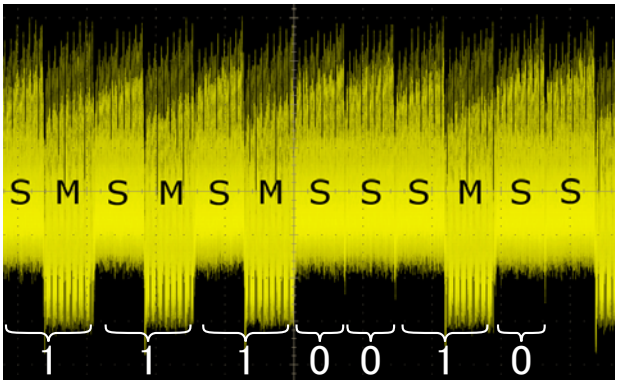


図 4.1 左バイナリ法に対する単純電力解析の例

アルゴリズム 4.1

Square-and-multiply always 法

Input:	X, N, $E = (e_{k-1}, ..., e_1, e_0)_2$
Output:	$X^E \bmod N$

1 : $R := 1$;
2 : **for** $i = k - 1$ **downto** 0 **do**
3 : $R_0 := R \cdot R \bmod N$; — squaring
4 : $R_1 := R_0 \cdot X \bmod N$; — multiplication
5 : $R := R_{e_i}$
6 : **end for**
7 : **return** R

4.3 RSA 暗号における Safe error 攻撃

Safe error 攻撃 [23] は，Squaring-and-multiply always 法に対する故障利用攻撃である．Safe error 攻撃では，攻撃者は乗算を行っている間に，暗号プロセッサと同期したタイミングで誤りを注入する．誤りが注入された演算がダミーの場合，中間結果は破棄されるため，誤りを含む中間結果は後の演算に伝搬しない．従って，攻撃者は出力結果を確認することにより，通常の乗算およびダミーの乗算を識別することができる．結果として，攻撃者は 1 回の故障注入に対して秘密の指数を 1 ビット得ることができる．これを k 回繰り返すことにより，すべての指数を得ることができる．

本論文では，Safe error 攻撃の拡張として，出力結果の代わりに暗号モジュールへの電流波形を用いてダミーの乗算を識別する攻撃を考える．攻撃者は，乗算への誤りの注入を伴う電流波形と伴わない波形をそれぞれ観測し，2 つの電流波形の間の差異を調べる．乗

表 4.1 実験条件

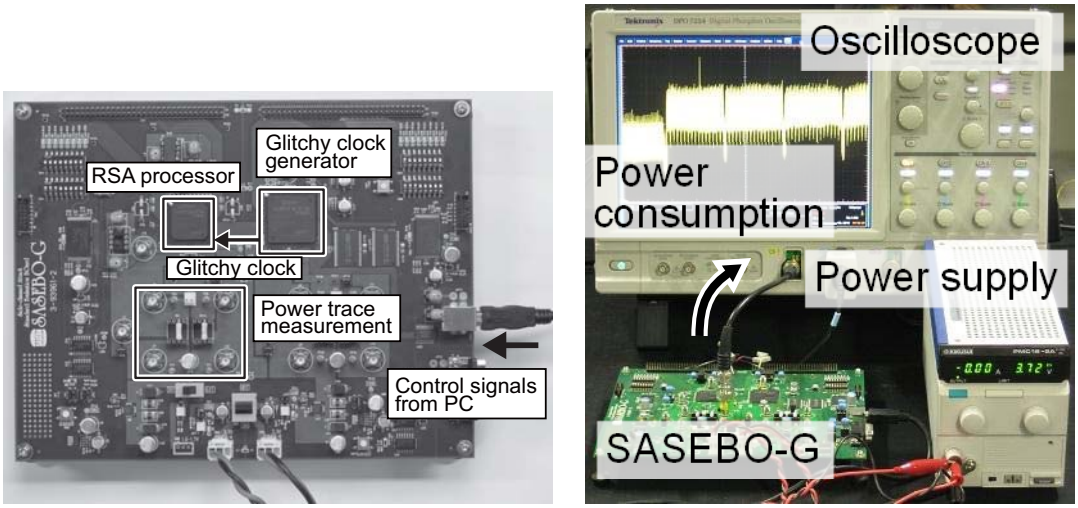
Experimental FPGA board (SASEBO)	
FPGA(cryptography)	Virtex-II Pro XC2VP7
FPGA(glitchy-clock generator)	Virtex-II Pro XC2VP30
Experimental setting	
Digital oscilloscope	Tektronix DPO7254
Sampling frequency	1 G samples/s
Probe	Coaxial cable (50-Ohm)
Stabilized power supply	3.3 V
Clock frequency	24 MHz
Measuring point	Resistor (1 Ohm)
	attached to GND line

算がダミーであれば，続く演算は同一であり，2 つの電流波形の間の差異は小さくなる．乗算がダミーでない場合は，故障注入後の演算は誤りを含むため，大きな差異が観測される．結果として，攻撃者は差分波形を観測することにより，出力結果を観測することなくダミーの乗算を識別できる．

4.4 RSA 暗号への故障利用攻撃に対する安全性評価実験

4.4.1 実験条件

本実験では高基数モンゴメリ乗算に基づく RSA 暗号モジュール [21] を用いる．データパスは Square-and-multiply always 法に従い乗算および加算を繰り返す 512 ビットの乗算器を含む．32 ビットのデータパスはレジスタに格納された 32 ビットのオペランドを用いて乗算および加算を繰り返す．それぞれの乗算および自乗算には 578 サイクルを要する．安全性評価実験の前に，グリッチ幅 T_w を掃引することにより故障が注入される T_w の条件を調べた．グリッチ整定時間 T_s を 4.2 ns に固定し，T_w を 6.0 ns – 13.5 ns の間で掃引しながら RSA 暗号の誤り率を測定した．それぞれのグリッチ周期に対し 100 回の試行を行い，誤りを含む出力が現れた回数を 100 で割った値を誤り率とした．結果として，$6.5 \leq T_w \leq 12.6$ [ns] の範囲で誤り率 100%を得た．T_w が 6.5 ns より短い場合，2 番目のクロック信号の立ち上がりまでの間に十分な電圧低下が得られなかった．反対に，T_w が 12.6 ns より長い場合，セットアップ時間違反が起こらないため，誤りは注入されない．以降の実験では，確実に誤りを注入するため，T_w を誤り率 100%の範囲の中央値である 9.6 ns とした．



(a) SASEBO–G　　(b) 実験装置

図 4.2 実験環境

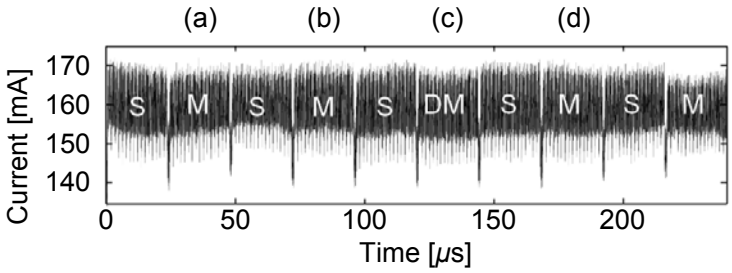


図 4.3 RSA 暗号モジュールの電流波形

表 4.1 に実験条件，図 4.2 に実験環境をそれぞれ示す．SASEBO–G，オシロスコープ，PC から構成される．本実験では 4.4.1 で述べたように，誤りを注入した場合とそうでない場合の電流波形を観測し，その差分を取った．図 A.4 に，RSA 暗号モジュールの電流波形を示す．文字 S，M，DM はそれぞれ自乗算，乗算およびダミー乗算を表す．上記の攻撃は 2 つの波形の差分を観測するため，クロックジッタの蓄積の影響を取り除くための波形の精密な位置合わせが重要である．本実験では，図 4.4 に示すように，位相を元にした波形マッチング手法 [24] を用いて，サンプリング分解能より高い分解能での波形の位置合わせを行なっている．波形の一部に対して正確に位置合わせを行った後，差分を求める．

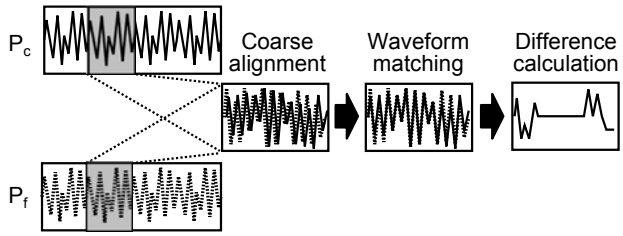


図 4.4 波形マッチングを用いた差分の計算

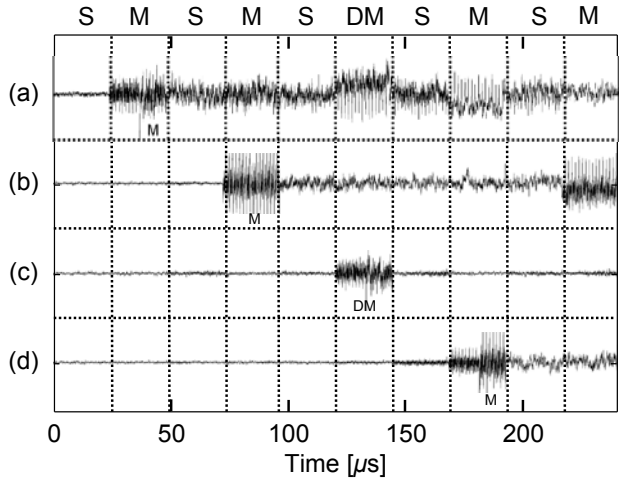


図 4.5 差分波形

4.4.2 実験結果

図 A.5 に，波形の組 (誤りを含むものと含まないもの) の間の差分波形を示す．図 A.5 (a)–(d) は，最初から 4 番目の乗算に誤りを注入した場合に対応する．ここでは，3 番目の演算のみダミー乗算である．誤りが注入された乗算の差分は特に大きくなっている．誤り注入後の演算の差分は相対的に小さいが，通常の乗算とダミーの乗算の間には明らかな差異がある．図 A.5 (c) はダミーの乗算に誤りを注入した場合の差分波形であり，その振幅は図 A.5 (a)，(b) および (d) と比較して小さい．故に，攻撃者は 3 番目の演算がダミーであり，指数が 0 であることを推定できる．対照的に，図 A.5 (a)，(b) および (d) は元の波形と誤りを含む波形が一致しないことを示している．これは誤りを注入した演算が通常の乗算であることを意味しており，最初，2 番目および 4 番目の指数ビットが 1 であることが分かる．結果として，攻撃者は最初の 4 つの指数ビット $E = (1101)_2$ を得ることができる．

以上の実験により，暗号モジュールに Safe error 攻撃に対する脆弱性があることが示さ

れた．

4.5 Safe error 攻撃への対策

Safe error 攻撃への対策法を以下に示す．べき乗剰余演算のアルゴリズムレベルでの対策としては，Montgomery powering ladder 法 [17] や Square-multiply exponentiation 法 [25] が提案されている．これらの対策では，すべての自乗算および乗算の結果が最終的な結果に影響するアルゴリズムとなっており，ダミー乗算が存在しないため，Safe error 攻撃を防ぐことができる．より下位のレベルの対策も考えられる．各乗算ごとに検算を行い，誤りを検出した場合に出力を停止する方法 [16] や，クロック信号そのものを監視し，異常なクロック信号を検知した場合に出力を停止する手法 [18] などが有効である．

4.6 結び

第 4 章では，第 3 章で提案したオンチップ故障注入回路を用いた安全性評価実験について述べた．まず，RSA 暗号の中心的な演算であるべき乗剰余演算アルゴリズムについて述べた．その後，Square-and-multiply always 法に対する Safe error 攻撃について述べた．また，電力解析を用いた演算の正誤判定について述べた．そして，RSA 暗号プロセッサに対する Safe error 攻撃を想定した安全性評価実験を行い，秘密情報を推定できることを示した．これにより，オンチップ故障注入回路により安全性評価実験が可能であることを示した．

第 5 章

結言

以上，第 2 章から第 4 章まで，故障利用攻撃に対する定量的な安全性評価を目的として，オンチップ故障注入回路を用いた故障利用攻撃に対する安全性評価手法について述べた．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，暗号アルゴリズムのハードウェア実装の必要性を述べた．続いて，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．最後に，SASEBO の概要を述べ，SASEBO は故障利用攻撃の実験にも有用であることを述べた．

第 3 章では，オンチップ故障注入回路を用いた安全性評価手法について述べた．まず，クロック周期の操作による故障注入の原理について述べ，オンチップ故障注入回路の構成を示した．その後，オンチップ故障注入回路を SASEBO–G へ実装し，パラメータの調整範囲および設定値に対するパラメータの線形性，出力信号の再現性を評価した．この結果，提案回路は安全性評価実験の要件を満たしていることを示した．さらに，加算回路に対する故障注入実験を行い，提案回路により正確な故障注入が可能であることを示した．

第 4 章では，第 3 章で提案したオンチップ故障注入回路を用いた安全性評価実験について述べた．まず，RSA 暗号の中心的な演算であるべき乗剰余演算アルゴリズムについて述べた．その後，Square-and-multiply always 法に対する Safe error 攻撃について述べた．また，電力解析を用いた演算の正誤判定について述べた．そして，RSA 暗号プロセッサに対する Safe error 攻撃を想定した安全性評価実験を行い，秘密情報を推定できることを示した．これにより，オンチップ故障注入回路により安全性評価実験が可能であることを示した．

今後の展望としては，提案回路を用いて，故障利用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．さらに，出力信号のジッタを調整する機能を付加することにより，想定する攻撃者のレベルを変化させた実験も可能となる．さらに，安全性の尺度を明らかにすることも重要な課題である．組み込みシステムにおいては効率的な故障利用攻撃対策が求められており，あらゆる攻撃の可能性の中から真に脅威となる攻撃を見極め，それに対して対策を施すことが重要である．提案手法により網羅的な実験を行うことにより，どの脆弱性が秘密情報の奪取につながり得るかを明らかにすることで，効率的な故障利用攻撃対策が可能となる．

付録 A

べき乗剰余演算に対する高度な故障利用攻撃実験

A.1　はじめに

本付録では，第 4 章で行った攻撃を発展させ，Safe error 攻撃への対策アルゴリズムに対する攻撃を行う．まず，べき乗剰余演算における物理攻撃対策について述べ，この対策を無効化する攻撃手法を示す．そして，SASEBO–G 上での実験を通し，提案攻撃の有効性を示すとともに，本攻撃に対する対策法を示す．

A.2　べき乗剰余演算における物理攻撃対策

第 4 章で示したように，Square-and-multiply always 法は Safe error 攻撃に対して脆弱性を持つ．そこで，Safe error 攻撃に耐性を持つアルゴリズムとして，Montgomery powering ladder 法 [17] および Square-multiply exponentiation 法 [25] が提案されている．これらのアルゴリズムは，ダミー演算ではなく常に意味のある演算を行うため，ダミー演算において誤りを挿入しても結果が誤らないことを利用する Safe error 攻撃 を防止することができる．

アルゴリズム A.1 に Montgomery powering ladder 法 のアルゴリズムを示す．このアルゴリズムは，指数の最上位から演算を開始し，指数ビットに応じて乗算 ($R_{\overline{e_i}} = R_{e_i} R_{\overline{e_i}} \bmod N$) と自乗算 ($R_{e_i} = R_{e_i}^2 \bmod N$) を繰り返す．指数ビット長を k ，最上位ビット (k ビット目) から j ビット目までの部分指数を $E_L^{(j)} = (e_{k-1} e_{k-2} ... e_{k-j})_2$ ，入力を X とすると，変数 R_0，R_1 は，j 番目の演算終了時，それぞれ $R_0 = X^{E_L^{(j)}}$，$R_1 = X^{E_L^{(j)}+1}$ と表せる．最終的な出力は R_0 から得られる．

アルゴリズム A.2 に Square-multiply exponentiation 法 のアルゴリズムを示す．このアルゴリズムは，鍵の最下位から演算を開始し，指数ビットに応じて自乗算 ($R_{\overline{e_i}} = R_{\overline{e_i}}^2 \bmod N$) と乗算 ($R_{\overline{e_i}} = R_{\overline{e_i}} R_{e_i} \bmod N$) を繰り返す．最下位ビット (0 ビット目) から j ビット目ま

アルゴリズム A.1

Montgomery powering ladder 法

Input:	X, N, $E = (e_{k-1}, ..., e_1, e_0)_2$
Output:	$X^E \bmod N$

```
1 :  R_0 := 1; R_1 := X;
2 :  for i = k − 1 downto 0 do
3 :    if e_i = 1 then
4 :      R_0 := R_0 R_1 mod N;   — multiplication
5 :      R_1 := R_1 R_1 mod N;   — squaring
6 :    else [e_i = 0]
7 :      R_1 := R_1 R_0 mod N;   — multiplication
8 :      R_0 := R_0 R_0 mod N;   — squaring
9 :    end if
10 :  end for
11 :  return R_0
```

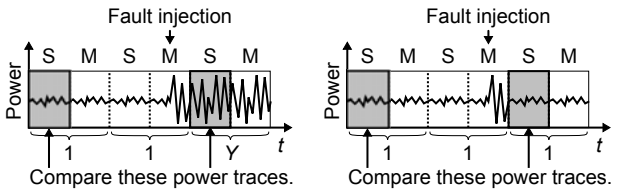
アルゴリズム A.2

Square-multiply exponentiation 法

Input:	X, N, $E = (e_{k-1}, ..., e_1, e_0)_2$
Output:	$X^E \bmod N$

```
1 :  R_0 := 1; R_1 := X;
2 :  for i = 0 upto k − 1 do
3 :    if e_i = 1 then
4 :      R_0 := R_0 R_0 mod N;   — squaring
5 :      R_0 := R_0 R_1 mod N;   — multiplication
6 :    else [e_i = 0]
7 :      R_1 := R_1 R_1 mod N;   — squaring
8 :      R_1 := R_1 R_0 mod N;   — multiplication
9 :    end if
10 :  end for
11 :  return R_0
```

での部分指数を $E_R^{(j)} = (e_{j-1}...e_1e_0)_2$ とすると，変数 R_0，R_1 は，j 番目の演算終了時，それぞれ $R_0 = X^{E_R^{(j)}}$，$R_1 = X^{\{2^{j-1}-E_R(j-1)\}}$ と表せる．最終的な出力は R_0 から得られる．Square-multiply exponentiation 法では，自乗算の入力値には常に乗算の出力結果が用いられる．このため，自乗算のコリジョンを用いて鍵を推定する Doubling 攻撃 [26] および Yen らの攻撃 [27] に対する耐性を有する．



(a) 演算がダミーでない場合　(b) 演算がダミーの場合

図 A.1 Square-and-multiply always 法への適用例 (平文が 1 の場合)

A.3 提案する攻撃手法

A.3.1 基本概念

本稿で提案する攻撃は，ある乗算もしくは自乗算に誤りを注入し，その後の演算誤りの伝搬を電力解析を用いて特定することにより鍵を推定する．本攻撃では，電力解析を用いて特定できる演算誤りやその位置が鍵ビットの値によって異なることを利用する．これにより，誤り暗号文を取得することなく，従来の Safe error 攻撃と同様の攻撃が可能となる．また，SPA および Safe error 攻撃に耐性を持つ Montgomery powering ladder 法や Square-multiply exponentiation 法においても鍵推定が可能となる．

演算の正誤を判定するには，攻撃者が平文を選択できる場合とできない場合の二つのシナリオが考えられる．

まず，平文を選択できる場合，攻撃者は誤り注入の前後で電力波形が大きく変化する平文を入力する．例として，平文を 1 とする Square-and-multiply always 法に対する攻撃を考える．平文が 1 のとき，演算結果は鍵ビットによらず毎回 1 となり，各演算の電力波形は等しくなる．ここで，ある乗算に誤りを入れ，中間結果が変化したとすると，乗算がダミーでない場合は次の自乗算が誤る．一方，乗算がダミーの場合は次の自乗算は正しく行われる．誤りの検出は，誤り注入前後の自乗算の電力波形を比較することにより可能である．同様の攻撃は平文が -1 の場合にも可能である．平文が -1 のとき，演算結果は 1 の場合と -1 の場合があるため，奇数番目の演算の正誤判定には奇数番目の演算，偶数番目の演算には偶数番目の演算を誤り注入の前後で比較する．このような平文選択の手法は， Montgomery powering ladder 法や Square-multiply exponentiation 法においても適用できる．

一方，平文が選択できない場合，攻撃者は正しい演算の波形と誤りを含む演算の波形との差異を観測する．このとき，各鍵ビットに対して，同一の入力で正しい処理と誤りを含

む処理の電力波形をそれぞれ取得する必要がある．

A.3.2 Square-and-multiply always 法への適用

Square-and-multiply always 法への提案手法の適用を考える．ここでは平文を X，法を N，k ビットの指数ビットを $E = (e_{k-1}...e_1e_0)_2$ とする．第 4 章の**アルゴリズム 2** において，j 番目までの部分指数を $E_L(j) = (e_{k-1}...e_{k-j+1}e_{k-j})_2$ とすると，中間変数 R_0 の値 $R_0(j)$ は式 (A.1) のように表せる．

$$R_0(j) = X^{E_L(j)} \mod N \tag{A.1}$$

ここで，j 番目の乗算に誤りを入れ，乗算結果が格納される中間変数の値が $R_{\overline{e_{k-j}}}(j) = Y$ のように誤ったとする．$e_{k-j} = 1$ のときのみ R_0 に乗算結果が格納されるため，R_0 の値は式 (A.2) で表される．

$$R_0(j) = \begin{cases} X^{E_L(j)} \mod N & (e_{k-j} = 0) \\ Y & (e_{k-j} = 1) \end{cases} \tag{A.2}$$

式 (A.2) により，指数ビット $e_{k-j} = 0$ のとき，次の自乗算は正しい値を用いて行われるため，正しい演算となる．一方 $e_{k-j} = 1$ のとき，次の自乗算は誤り (Y) を含む値を用いて行われるため，誤った演算となる．提案手法では式 (A.2) で生じる違いを電力波形から読み取ることで鍵を推定する．

図 A.1 に，平文 1 を用いて演算の違いを判別する場合を示す．通常，入力が 1 のときハードウェアではゲートのスイッチングがほとんど発生しないため，波形の振幅は小さくなる．これに対して故障注入後は，測定波形の振幅に変化が生じるため，故障による値の変化を知ることなく，波形の差異から故障を注入した演算がダミーかどうかを判定することができる．その結果，従来の Safe error 攻撃と比較して，誤り暗号文が不要となる．

A.3.3 Montgomery powering ladder 法への適用

部分指数を $E_L(j) = (e_{k-1}...e_{k-j+1}e_{k-j})_2$ とすると，j 番目の中間変数 R_0, R_1 の値 $R_0(j), R_1(j)$ は，それぞれ式 (A.3)，(A.4) のように表せる．

$$R_0(j) = X^{E_L(j)} \bmod N \tag{A.3}$$

$$R_1(j) = X^{E_L(j)+1} \bmod N \tag{A.4}$$

Montgomery powering ladder 法では，誤りを乗算に入れた場合と自乗算に入れた場合とで 2 通りの攻撃がある．ここでは j 番目の乗算に入れた場合を説明する．

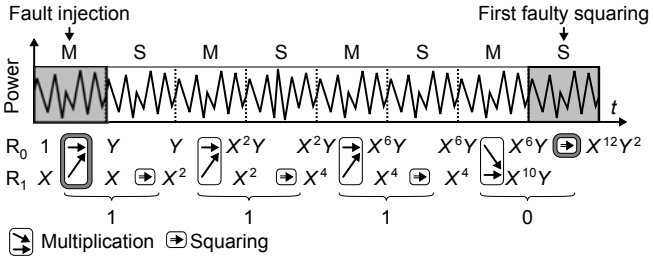


図 A.2 Montgomery powering ladder 法への適用例

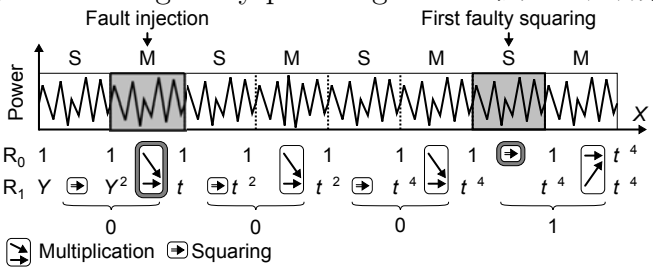


図 A.3 Square-multiply exponentiation 法への適用例

j 番目の乗算結果が格納される変数の値を $R_{\overline{e_{k-j}}}(j) = Y$ とすると，$j+1$ 番目の自乗算に用いられる中間変数 $R_{e_{i+1}}(j+1)$ (ただし，$i = k-j$) の値は式 (A.5) のように表せる．

$$R_{e_i}(j+1) = \begin{cases} \{R_0(j)\}^2 & \bmod N \quad (e_i = e_{i+1} = 0) \\ Y & \bmod N \quad (e_i \neq e_{i+1}) \\ \{R_1(j)\}^2 & \bmod N \quad (e_i = e_{i+1} = 1) \end{cases} \tag{A.5}$$

式 (A.5) より，指数ビットが反転する場合は演算に誤りが伝搬し，反転しない場合は伝搬しないことが分かる．指数ビットが反転しない場合，中間変数 R_1 の値のみが誤り (Y) を含む状態が続くため，$j+2$ 番目以降の自乗算においても式 (A.5) と同様の式が成り立つ．提案手法では，j 番目以降の自乗算に対して順番に正誤判定を行い，誤りを入れてから最初に指数ビットが反転する場所を見つけることで鍵ビットの部分系列を推定する．

図 A.2 に提案攻撃の Montgomery powering ladder 法への適用例を示す．ここでは入力を X ，指数の最上位から 4 ビットの部分指数を $E_L(4) = (1110)_2$ とする．最初の演算に誤りを入れた場合，2 番目以降の自乗算を順番に観測する．ここでは 4 番目の演算でビットが反転しているため，4 番目の自乗算が誤る．この最初に誤りが出現する演算を同定することで，E_L の値が得られる．

A.3.4 Square-multiply exponentiation 法への適用

部分指数を $E_R(j) = (e_{j-1}...e_1e_0)_2$ とすると，j 番目の中間変数 R_0, R_1 の値 $R_0(j), R_1(j)$ はそれぞれ式 (A.6)，(A.7) のように表せる．

$$R_0(j) = X^{E_R(j)} \bmod N \tag{A.6}$$

$$R_1(j) = X^{\{2^{j-1}-E_R(j-1)\}} \bmod N \tag{A.7}$$

ここで，j 番目の乗算もしくは自乗算に誤りを入れ，乗算結果が格納される誤りを含む変数の値を $R_{\overline{e_j}}(j) = Y$ とする．$j+1$ 番目の自乗算に用いられる中間変数の値 $R_{\overline{e_{j+1}}}(j+1)$ は式 (A.8) で表せる．

$$R_{\overline{e_{j+1}}}(j+1) = \begin{cases} Y & \bmod N \quad (e_j = e_{j+1}) \\ R_0(j) & \bmod N \quad (e_j = 0, e_{j+1} = 1) \\ R_1(j) & \bmod N \quad (e_j = 1, e_{j+1} = 0) \end{cases} \tag{A.8}$$

式 (A.8) の値は，続く自乗算の正誤を左右する．誤りを入れた後の演算で指数ビットが反転する場合には，誤りは伝搬しないため，自乗算は正しい演算となる．一方指数ビットが反転する場合，誤りは伝搬し，自乗算は誤り (Y) を含む値を用いた演算となる．提案手法では，Montgomery powering ladder 法と同様に，このような誤り伝搬の有無を利用する．

図 A.3 に提案攻撃の Square-multiply exponentiation 法への適用例を示す．ここでは入力を X ，指数の最上位から 4 ビットの部分指数を $E_R(4) = (1000)_2$ とする．最初の乗算に誤りを入れた場合，2 番目以降の自乗算を順番に観測する．ここでは 4 番目の演算でビットが反転しているため，4 番目の自乗算が誤る．この最初に誤りが出現する演算を同定することで，E_R の値が得られる．

A.4 実験

本章では，Montgomery powering ladder 法に基づく RSA 暗号プロセッサに対する攻撃実験を行い，提案する攻撃の有効性を示す．実験条件および環境は第 4 章と同様である．

図 A.4 に RSA 暗号プロセッサから取得した電力波形を示す．ここで S, M はそれぞれ自乗算，乗算を表し，記号の添字は鍵ビットの値に対応する．また，インデックス (a)–(d) は誤り (グリッチ) を挿入した 乗算位置を表し，インデックス (i)–(v) は誤り挿入後の自乗算を表す．

実験では，1 回につき 1 カ所にグリッチを挿入して，その電力波形を計測するという手順を (a)-(d) の位置について繰り返し，四つの波形を取得した．

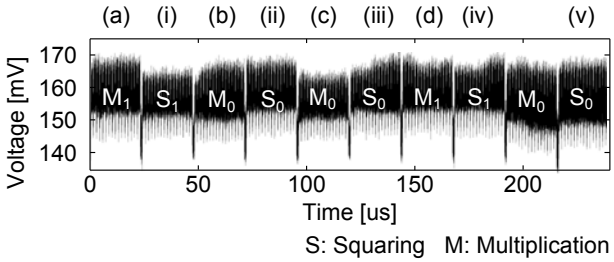


図 A.4 RSA プロセッサの電力波形

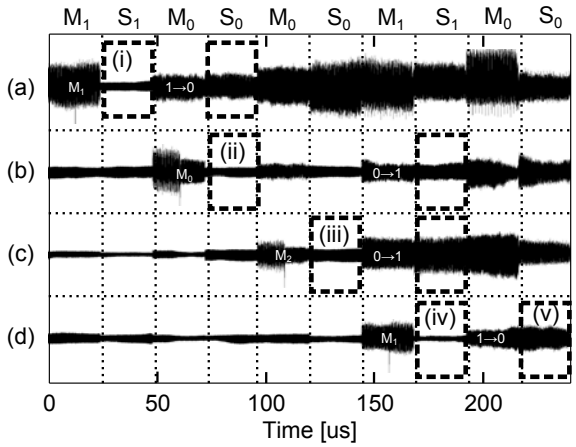


図 A.5 電力の差分波形

図 A.5 は，図 A.4 に対応する乗算インデックス (a)–(d) に誤りを挿入したときの電力波形と正常な演算時の電力波形との差分波形である．3.3 節で述べたように，誤り注入後，鍵ビットの値が反転するところで自乗算に初めて誤りが混入する．そのため，(a) の (ii)，(b) の (iv)，(c) の (iv) および (d) の (v) 番目の自乗算における差分波形の振幅は誤り挿入直後の自乗算部分と比較して大きくなる．一方，鍵ビットの値が反転しない (b) の (iii) 番目の自乗算では差分波形は小さく抑えられる．結果として，最初の鍵ビットを $x(\in \{0,1\})$ とすると，対応する部分の鍵ビットは $E_L(j) = (x\overline{x}x\overline{x})_2$ と表せる．同様の攻撃を最後の鍵ビットまで繰り返すことにより，鍵ビットすべてを $x, \overline{x}$ で表すことができ，鍵を 2 通りまで絞り込むことができる．

A.5 提案攻撃への対策

提案攻撃は，平文選択した波形もしくは同一入力下での差分波形を用いるため，平文や指数に乱数のマスクを施す対策である Data masking [28] が有効である．平文に対する Data masking では，平文 M とランダムな値 v の積をとり，その値 $\hat{M} = vM \bmod N$ を

平文として用いる．マスクされた暗号文 $\hat{C}$ はあらかじめ求めておいた $(v^E)^{-1}$ を乗ずることで暗号文 C に変換できる．平文に乱数マスクを施すことで，1 や -1 を入力しても演算には用いられないため誤り挿入前後の差異を抑制できる．また，1 回ごとにマスクを変更することにより，正しい演算と誤りを含む演算の比較による指数ビットの同定を防ぐことができる．また，指数に対するマスキングも有効である．例えば，法 N における位数 $\Phi(N)$ の整数倍を指数に加え，$E' = E + k\Phi(N)$ を指数として用いることにより指数を隠ぺいする手法がよく知られている [28]．ここで $\Phi(N)$ はオイラー関数である．$M^{k\Phi(N)} \equiv 1 (\mathrm{mod}\ \ N)$ より，E' を用いて演算を行っても E と等しい結果が得られる．1 回ごとにマスクを変更することで，真の鍵ビットが推定されるのを防ぐことができる．

A.6 まとめ

本稿では，暗号プロセッサ上でべき乗剰余演算を行うアルゴリズムに対する実装攻撃として，電力解析攻撃と故障利用攻撃を組み合わせた新たな攻撃を提案した．本攻撃により，単純電力解析および故障利用攻撃（Safe error 攻撃）に耐性をもつアルゴリズムである Montgomery powering ladder 法や Square-multiply exponentiation 法においても鍵推定が可能となることを示した．また，提案した攻撃により Montgomery powering ladder 法を用いた RSA 暗号プロセッサに対する攻撃実験を行い，本攻撃により鍵推定が可能であることを示した．今後の課題としては，本攻撃に対する効率的な対策アルゴリズムの検討が挙げられる．

参 考 文 献

[1] 産業技術総合研究所情報セキュリティ研究センター, “SASEBO プロジェクトの概要.” http://staff.aist.go.jp/akashi.satoh/SASEBO/ja/board/index.html, Aug. 2010.

[2] S. Endo, T. Sugawara, N. Homma, T. Aoki, and A. 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] D. Boneh, R. DeMillo, and R. Lipton, “On the importance of checking cryptographic protocols for faults,” *Advances in Cryptology – EUROCRYPT ’97*, Vol. 1233, pp. 37–51.

[9] J. M. Schmidt and H. Christoph, “A practical fault attack on square and multiply,” *Proc. 5th Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 53–58, Oct. 2008.

[10] T. Fukunaga and J. Takahashi, "Practical fault attack on a cryptographic lsi with iso/iec 18033-3 block ciphers," *Proc. 6th Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 84–92, Sept. 2009.

[11] 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.

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

[13] S. P. Skorobogatov and R. J. Anderson, "Optical fault injection attacks," *Cryptographic Hardware and Embedded Systems – CHES 2003*, pp. 2–12, 2003.

[14] Y. Li, K. Sakiyama, S. Gomisawa, T. Fukunaga, J. Takahashi, and K. Ohta, "Fault sensitivity analysis," *Lecture Notes in Computer Science*, Vol. 6225, pp. 320–334.

[15] 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.

[16] 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.

[17] M. Joye and S. M. Yen, "The montgomery powering ladder," *CHES 2002, Lecture Notes in Computer Science*, Vol. 2523, pp. 291–302, Aug. 2002.

[18] N. Selmane, S. Bhasin, S. Guilley, T. Graba, and J. L. Danger, "Wddl is protected against setup time violation attacks," *Fault Diagnosis and Tolerance in Cryptography*, 2009.

[19] Y. Li, K. Ohta, and K. Sakiyama, "New fault-based side-channel attack using fault sensitivity," *Information Forensics and Security, IEEE Transactions on*, Vol. 7, No. 1, pp. 88–97, Feb. 2012.

[20] 菅原健 鈴木大輔, "Fault Sensitivity Analysis のための回路シミュレーション," 2012 年暗号と情報セキュリティシンポジウム予稿集, No. 3C3-1, Feb. 2012.

[21] A. Miyamoto, N. Homma, T. Aoki, and A. Satoh, "Systematic design of high-radix montgomery multipliers for rsa processors," *Proc. 26th IEEE Int'l Conf. Computer Design*, pp. 416–422, Oct. 2008.

[22] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *CRYPTO '99: Proc. of the 19th Annual International Cryptology Conference on Advances in Cryptology*, pp. 388–397, Springer-Verlag, Aug. 1999.

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

[24] N. Homma, A. Miyamoto, T. Aoki, A. Satoh, and A. Shamir, "Comparative power analysis of modular exponentiation algorithms," *IEEE Trans. on Computers*, Vol. 59, No. 6, pp. 795–807, June 2010.

[25] M. Joye, "Highly regular right-to-left algorithms for scalar multiplication," *CHES 2007, Lecture Notes in Computer Science*, Vol. 4727, pp. 135–147, Sept. 2007.

[26] A. P. Fouque and F. Valette, "The doubling attack -why upwards is better than downwards," *CHES 2003, Lecture Notes in Computer Science*, Vol. 2779, pp. 269–280, Sept. 2003.

[27] S. M. Yen, W. C. Lien, S. J. Moon, and J. C. Ha, "Power analysis by exploiting chosen message and internal collisions - vulnerability of checking mechanism for RSA-decryption.," *Mycrypt 2005, Lecture Notes in Computer Science*, Vol. 3715, pp. 183–195, Sept. 2005.

[28] P. Kocher, "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, pp. 104–113, Springer-Verlag, Aug. 1996.

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・御批判を頂いた曽根秀昭教授ならびに羽生貴弘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.

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.

2012年2月8日