暗号集積回路に対するスキャンベース サイドチャネル攻撃に関する研究

Scan-based side-channel attacks on cryptographic integrated circuits using scan signatures

2016年2月

早稲田大学大学院 基幹理工学研究科

情報理工・情報通信専攻 情報システム設計研究

藤代　美佳

Mika FUJISHIRO

目 次

図 目 次

表 目 次

第1章　序論

1.1　本論文の背景と意義

近年ではICTの発展によりあらゆる情報をデータとして扱うようになり，扱う情報の価値は大きくなっている．情報の改ざん，漏洩が多発する中，想定される攻撃対象は金銭や個人情報から社会インフラまで及び，人身や国家の安全性をも揺るがす事態になっている．そのため多種多様な攻撃を想定したセキュアなシステムの構築が必要である．

Suicaやクレジットカード等のスマートカードは，交通，金融，行政等多くの分野において日常的によく使われている．また2016年1月にはマイナンバーカードの導入が決定している．これらスマートカードに対し，ハードウェアの特性を利用したサイドチャネル攻撃の危険性が指摘されている．サイドチャネル攻撃には，暗号回路の消費電力を計測し秘密情報を取得する電力解析攻撃や故障情報を利用するフォールト解析攻撃，タイミング攻撃，キャッシュ攻撃，スキャンベース攻撃等がある．今日では情報を確実に保護するためには暗号技術だけでなくハードウェアの特性も考慮しなければならない．スマートカードは価値・機密性が高い情報を扱うため，機密情報を確実に保護する安全な暗号集積回路の設計が求められており，暗号技術とハードウェアの特性のセキュリティの研究は必須である．

安全な暗号集積回路を設計するためには，回路の脆弱性を解明する必要がある．これまでに暗号集積回路に対する多様な攻撃が検討されており，テスト用のスキャンチェインを利用したスキャンベース攻撃の危険性が指摘されている．

スキャンチェインはLSI中のレジスタを直列に接続してシフトレジスタを形成し，外部からレジスタを直接制御・観測可能にしたテスト技術である．このスキャンチェインを利用することで，LSIをテストする際にレジスタを自由に制御できテスト効率を高められる．近年のLSIの大規模化や微細化，高性能化により，スキャンチェインは，LSIの動作のテストや検証のコストを削減するための重要なテスト容易化設計 (DFT: Design for Test) 技術になっている．

スキャンチェインを利用したスキャンパステストには，通常動作を実行するノーマルモードとスキャン・イン，スキャン・アウトを利用するテストモードがある．

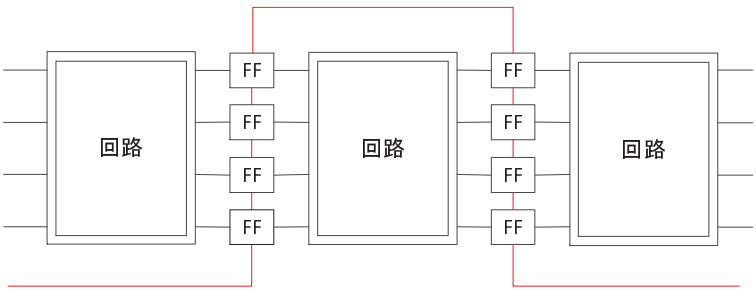


図 1.1: スキャンチェイン.

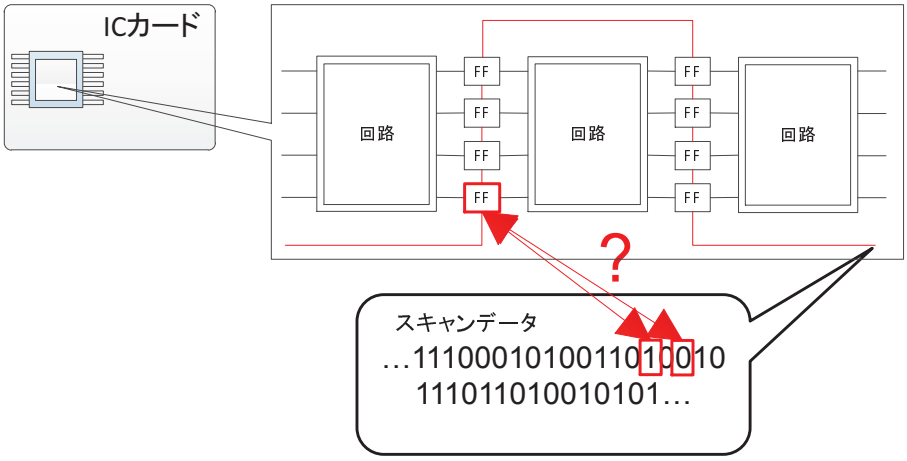


図 1.2: スキャンチェインとスキャンデータ.

スキャン・インでレジスタに値を設定，スキャン・アウトでレジスタの値を読み出す．スキャンパステストでは，テストモード時にスキャン・インで LSI 内部のレジスタにテストパターンを設定し，ノーマルモードで LSI を動作させた後，テストモードに再度切り替えスキャン・アウトでレジスタ値を取得する．スキャン・アウトで取得したレジスタ値をスキャンデータといい，スキャンデータと期待値を比較する．一般に大規模な LSI では回路内の全 FF をスキャンチェインでテスト可能にするフルスキャン方式が用いられている [38]．図 1.1 にスキャンチェインの構造を示す．

スキャンチェイン上でのレジスタの接続の順番は，通常，総配線長が最も短くなるよう決定するため，スキャンデータ上のレジスタ値と実際のレジスタとの対応関係は設計者以外には分からない (図 1.2)．スキャンベース攻撃では，スキャンデータとレジスタの対応関係を求めることが最大の鍵となる．

スキャンベース攻撃における従来手法はスキャンチェインに接続されたレジスタが特定の構成になっていることを前提としている場合が多い．しかし，通常 LSI 上のスキャンチェインは様々な回路のレジスタを接続している．このように攻撃手法

が特定の条件下でのみ有効であっても，スキャンベース攻撃の危険性を完全には指摘しきれていない．脆弱性を解明するためには，攻撃手法が有効になる条件を限定せず，現実的な条件を設定した上で攻撃手法を検討すべきである．

本論文では暗号集積回路のセキュア設計を目的としている．安全な暗号集積回路は,「強固な暗号アルゴリズム」を「情報を漏えいしない適切な仕組みで実装」していることが求められるため,「暗号アルゴリズム」の数理的な性質・脆弱性と「実装法」における脆弱性を評価する．ストリーム暗号，ブロック暗号アルゴリズム，ハッシュを実装した暗号集積回路に対してスキャンチェインの構造に依存しないスキャンベース攻撃手法を提案することで，暗号集積回路の「暗号アルゴリズム」における脆弱性,「実装法」における脆弱性を指摘する．多様な攻撃を想定することで暗号集積回路における脆弱性を解明し，防御設計における必要十分条件を明らかにすることができる．安全なスマートカードの実現に繋がる研究である．

1.2　本論文の概要

本論文では，ストリーム暗号，ブロック暗号，ハッシュへのスキャンベース攻撃手法を提案し，ソフトウェアによる評価実験の結果を報告することを目的とする．

以下に本論文の構成を示す．

第 2 章「サイドチャネル攻撃に関する研究動向」では，サイドチャネル攻撃に関する研究を紹介する．暗号アルゴリズムの特性だけでなくハードウェアの特性を利用したサイドチャネル攻撃が注目されている．既存研究として，暗号 LSI の消費電力を計測，解析する差分電力解析を扱った Kocher らの手法，McEvoy らの手法，Belaid らの手法，桶屋らの手法がある．故障を発生させることで得られた出力を利用するフォールト解析攻撃に関する研究として，Boneh らの手法，Biham らの手法がある．キャッシュを利用したキャッシュ攻撃については，Kelsey らの手法，角尾らの手法がある．スキャンベース攻撃においてはストリーム暗号に対する手法として，Agrawal らの手法，Liu らの手法，Mukhopadhyay らの手法がある．またブロック暗号に対する手法として Yang らの手法，奈良らの手法，小寺らの手法がある．またその他サイドチャネル攻撃としてタイミング攻撃，電磁波解析攻撃を紹介する．

第 3 章「ストリーム暗号へのスキャンベースサイドチャネル攻撃」では，ストリーム暗号に対するスキャンベース攻撃手法を提案し，評価する．ストリーム暗号評価プロジェクトで推奨暗号に認定された Trivium を対象にスキャンベース攻撃手法を提案する．Trivium は 3 本のシフトレジスタから構成され，内部の演算はビット同士の AND 演算と XOR 演算のみであるため，構造が単純で高速に動作する．秘密鍵 K (80bit) と IV (initialization vector: 初期化ベクトル) (80bit) により 288 個の内部状態レジスタが初期化され，内部状態を更新しながらキーストリームのビットを生成する．Trivium LSI から取得したスキャンデータを用いて攻撃者は暗号文を解読する．Trivium の性質上，攻撃者が解読対象の暗号文を出力した直後の Trivium LSI の内部状態値を取得した場合，過去のいかなる内部状態も算出でき，キーストリームを復元できる．得られたキーストリームと暗号文を順に排他的論理和することで元の平文を取得できる．暗号文から平文への復元は，いかに暗号文が出力された直後の内部状態値を取得するかに還元される．そこでストリーム暗号 LSI に任意の秘密鍵と IV を入力できることを利用する．秘密鍵と IV の値を入力ペアとして多数用意し，各入力に対し Trivium LSI を数サイクル動作させるとき，ある 1 ビットレジスタの入力に対する値の変化，動作させたサイクル数に対する値の変化は，そのレジスタ固有の値になる．この固有の値をスキャンシグネチャと呼ぶ．Trivium の内部状態レジスタに対しそれぞれスキャンシグネチャを

シミュレータで計算しておき，同様の条件下で実際のLSI回路から取得したスキャンデータと比較することでレジスタとスキャンデータ上のビットの対応を解析できる．この手法では，スキャンデータにTriviumのレジスタ以外のレジスタの値が含まれていても，高々1ビットの値の変化にのみ着目しているため，内部状態レジスタのビットの位置を特定できる．スキャンデータのビット対応が一度求まれば，Trivium LSIが暗号文を出力した直後のスキャンデータからTriviumの内部状態レジスタの各値を求められる．Triviumの内部状態レジスタ値が求まれば，過去の内部状態，キーストリームを復元でき，Trivium LSIが出力した暗号文と排他的論理和することで，平文を復元できる．評価実験により，提案手法はスキャンデータに他の回路のビットが含まれていても，ビット対応解析可能と確認した．また，Triviumの内部状態レジスタ288個のビット対応解析には，特定の入力を設定してキーストリーム生成フェーズからサイクル毎にスキャンデータを13個取得すれば良く，解析時間は0.139秒で済み，最も効率的に求められることを確認した．

第4章「ブロック暗号へのスキャンベースサイドチャネル攻撃」では，ブロック暗号に対するスキャンベース攻撃手法を提案し，評価する．64ビットブロック暗号LEDを対象にスキャンベース攻撃手法を提案する．LEDは64ビットから128ビットの秘密鍵を用いて副鍵を生成し，分割・転置を実行するラウンド処理と副鍵との排他的論理和を繰り返す．演算処理単位は4ビットであり，各4ビットを1要素としてカウントする．秘密鍵長が64ビットの場合，秘密鍵を解読するためには0番目の副鍵SK^0を解読すればよい．LEDの性質より，ラウンド処理実行前の値の任意の1要素はラウンド処理実行後の値の4つの要素に影響を及ぼしており，他の要素とは独立である．また，ラウンド処理実行後の値の任意の1要素はラウンド処理実行前の値の4つの要素に依存しており，他の要素とは独立の関係にある．これらの関係より0番目の要素のみ異なり，他の要素は等しい2つの平文をLED暗号LSIに入力し，1ラウンド目処理後の値をスキャンデータとして取得し，排他的論理和する時，4つの要素，つまり，ある16個のビットは副鍵SK^0の中の0番目の要素 (4ビット) のみに依存した値になる．よって，副鍵SK^0の0番目の要素の全パターンについて，これら16個のビットのスキャンシグネチャを求め，スキャンデータと比較することで，副鍵SK^0の0番目の要素を解読できる．同様に，副鍵SK^0の他の要素についてもこれらの手法で解読可能である．提案手法は，スキャンチェインに他の回路が含まれていても秘密鍵を解読可能であり，スキャンチェイン長，秘密鍵長に非依存という特長がある．スキャンチェイン長が3万ビット以上の場合には副鍵SK^0を2要素ずつ順番に求めればよい．また，秘密鍵長が64ビットより大きい場合，提案手法を用いてSK^0を求めることで，秘密鍵の上位64ビットの値が判明する．秘密鍵の残りの部分は，1番目の副鍵SK^1を求めることで判

明する．SK^1 を求めるために，SK^1 と排他的論理和する演算に対し特定の入力を与える必要があるが，既に求めた SK^0 を用いてそのような入力を与える平文を計算することで解読可能である．計算機実験では，提案手法を用いて平均 73 個の平文で 64 ビットの秘密鍵を 0.290 秒で復元可能と確認した．また平均 145 個の平文で 128 ビットの秘密鍵を 0.468 秒で復元可能と確認した．スキャンチェインに他の回路が含まれていることを想定し，スキャンデータにランダムなビット値を付加してスキャンチェイン長を 13 万ビット程度まで変化させた場合にも，137 個の平文を用いて副鍵 SK^0 を 2 要素ずつ順番に求めることで，64 ビットの秘密鍵を 2 時間半程度で解読できることを確認した．

第 5 章「ハッシュへのスキャンベースサイドチャネル攻撃」では，ハッシュ関数 PGV の性質とアルゴリズム，ハッシュ関数の実装法 HMAC の性質とアルゴリズムを示し，スキャンシグネチャを用いた HMAC-PGV へのスキャンベース攻撃手法を提案する．HMAC (Hash-based Message Authentication Code) はハッシュ関数を用いたメッセージ認証コード (MAC: Message Authentication Code) で，ハッシュ関数を 2 回実行するという特徴がある．ANSI, IETF ISO, NIST により標準化されており，SSL, TLS, SSH, Ipsec 等に使われている．ハッシュ関数 PGV はブロック暗号を利用したハッシュ関数である．HMAC-PGV に実装されたブロック暗号へのスキャンベース攻撃が可能な時，HMAC-PGV に対してもスキャンベース攻撃可能である．提案手法は，特定のメッセージを入力した LSI から取得したスキャンデータにおいて，特定のビット列に着目することで秘密鍵を解読する手法である．また，提案手法のソフトウェアによる評価実験結果を示す．ハッシュに対してもスキャンベース攻撃可能なことを確認した．

第 6 章「結論」では，本論文の内容をまとめ，今後の課題を示す．

第2章　サイドチャネル攻撃に関する研究動向

2.1　本章の概要

本章ではサイドチャネル攻撃に関する既存研究を紹介する．

以下に本章の構成を示す．

第 2.2 節「サイドチャネル攻撃に関する既存研究」では，サイドチャネル攻撃の既存研究を紹介する．サイドチャネル攻撃はハードウェアの特性を利用した攻撃である．攻撃に利用されるサイドチャネル情報は，消費電力，故障情報，キャッシュ情報，タイミング情報等，多岐にわたる．これらを利用したサイドチャネル攻撃の脅威を紹介する．

第 2.3 節「スキャンベースサイドチャネル攻撃に関する既存研究」では，スキャンベースサイドチャネル攻撃の既存研究を紹介する．スキャンベースサイドチャネル攻撃はテスト用スキャンチェインを悪用したサイドチャネル攻撃であり，単にスキャンベース攻撃とも呼ばれる．スキャンチェインから取得したレジスタ値の情報を解析し，暗号回路の秘密情報を復元する．攻撃対象暗号アルゴリズム毎にスキャンベース攻撃の既存研究を紹介する．

第 2.4 節「本章のまとめ」では，本章の内容をまとめる．

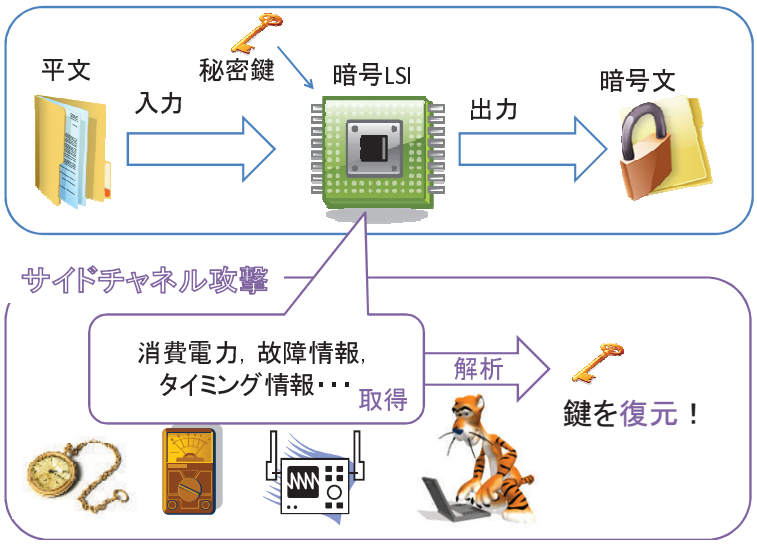


図 2.1: サイドチャネル攻撃.

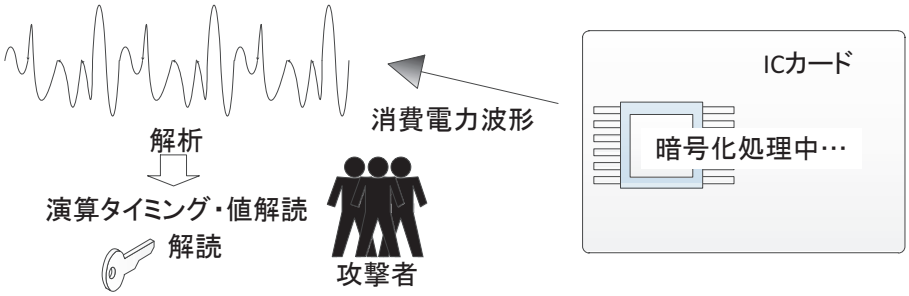


図 2.2: 電力解析攻撃.

2.2　サイドチャネル攻撃に関する既存研究

暗号アルゴリズムの特性だけでなくハードウェアの特性を利用したサイドチャネル攻撃 (図 2.1) が注目されている．サイドチャネル攻撃が悪用するサイドチャネル情報は消費電力，故障情報等，多岐にわたる．暗号回路の消費電力を計測し秘密情報を取得するサイドチャネル攻撃を電力解析攻撃，故障情報を利用する攻撃をフォールト解析攻撃といい，他にキャッシュ攻撃，タイミング攻撃，電磁波解析攻撃，スキャンベース攻撃等のサイドチャネル攻撃が報告されている．

電力解析攻撃は暗号 LSI の消費電力を計測，解析する攻撃である．消費電力は暗号処理中間値，処理中演算に関連があるため，暗号処理中の LSI の電力を測定することで，条件付きジャンプ等の暗号内部処理のタイミングを判別可能である．図 2.2 に概略を示す．単純電力解析 (SPA: Simple Power Analysis) は消費電力の変化を解析に用いる．電力の差が小さい場合，単純電力解析の利用は困難である．差

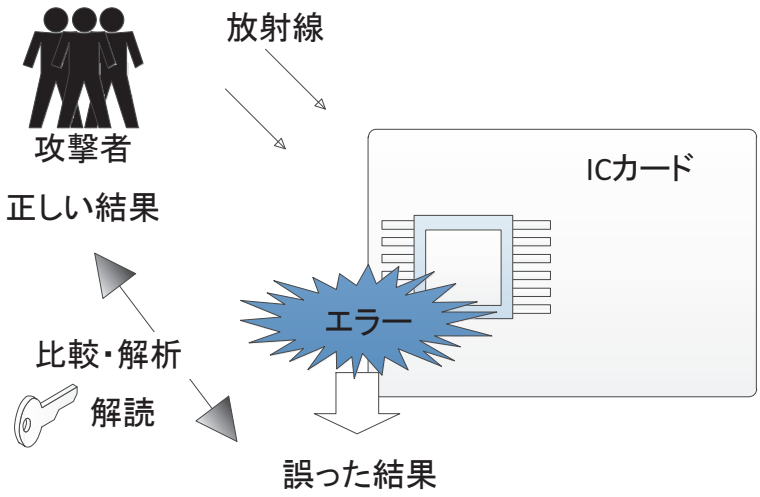


図 2.3: フォールト解析攻撃.

分電力解析 (DPA: Differential Power Analysis) は，多数の電力測定値の平均との差分により電力変化を測定する．

差分電力解析には DES 等への攻撃を示した [20] がある．鍵のあるビット値を予測し，電力を測定する．電力差分に着目することで正しい鍵の予想値を発見することが出来る．

[23] は HMAC-SHA-2 に対する差分電力解析を示している．ハッシュ値を計算中の LSI の電力を測定し，内部中間処理値のハミング距離との相関から内部中間処理値を順に計算する．この手法では秘密鍵そのものは復元できないが SHA-256 における内部中間処理値を復元でき，これらの値より任意のメッセージに対する MAC の偽造が可能になる．差分電力解析に限らず電磁波解析攻撃の適用も可能な手法である．これに対しマスク処理を利用した防御法を提案している．一方，[7] は HMAC-SHA-2 に対しハミング重みを利用した差分電力解析を示している．[23] と異なり，攻撃者は HMAC の実装の状態の知識なしに攻撃可能である．

ブロック暗号を用いた HMAC-PGV に対する差分電力解析には [28] がある．PGV における圧縮関数 12 種の内 11 種に対して HMAC の偽造が可能なことを示している．

フォールト解析攻撃は故障を発生させることで得られた出力を利用する．攻撃者は放射線照射，電圧，クロック周波数の操作等により，意図的に LSI にエラーを発生させる．エラーにより LSI では予期しない動作や結果が得られ，本来の結果と比較・解析することで攻撃者は暗号回路の秘密情報を復元する．図 2.3 に概略を示す．[9] は RSA に対するフォールト解析攻撃を示した．共通鍵暗号に対する攻撃では，DES に対するフォールト解析攻撃 [8] がある．

キャッシュを利用したキャッシュ攻撃は最初に [18] で攻撃可能性が示唆され，[35]

が DES に対するキャッシュ攻撃を示した.

タイミング攻撃 [19] は暗号処理実行時間と鍵の値の関係を利用するサイドチャネル攻撃で，電磁波解析攻撃 (EMA: ElectroMagnetic Analysis) は暗号処理中に放射された電磁波を測定することで，内部演算，内部値を求め解読する攻撃である.

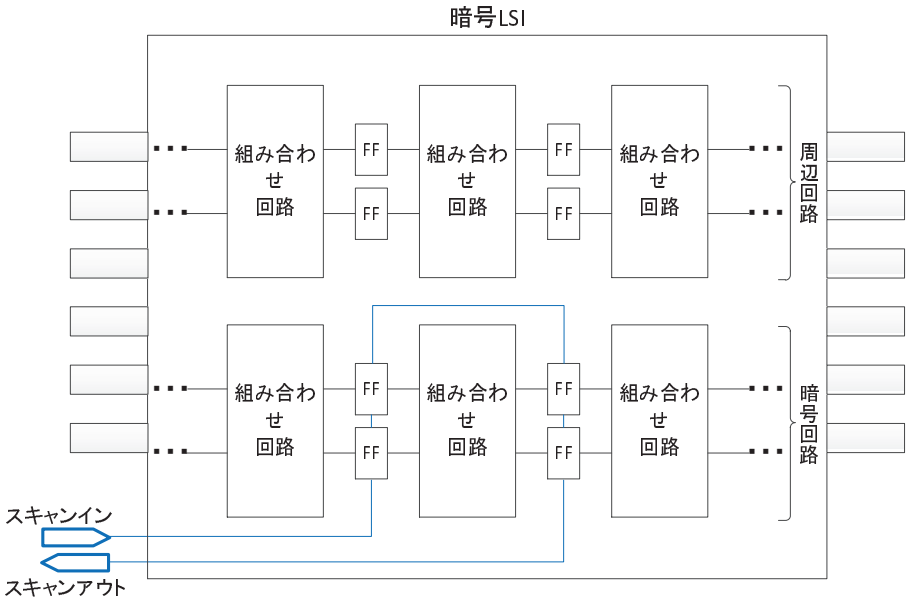


図 2.4: 暗号回路のみがスキャンチェインに接続されている場合.

2.3 スキャンベースサイドチャネル攻撃に関する既存研究

スキャンベース攻撃は，スキャンチェインを実装した暗号 LSI から暗号化処理中のレジスタの値をスキャンデータとして取得・解析し，暗号回路の秘密情報を取得するサイドチャネル攻撃である．テスト容易化のために重要な役割を持つスキャンチェインは大多数の LSI においてテスト用に実装されている．また元々スキャンデータは機密情報ではないため，スキャンベース攻撃は攻撃者がスキャンチェインにアクセスできることを前提としている．

スキャンチェイン上でのレジスタの接続順は，通常，総配線長が最も短くなるよう決定するため，スキャンデータ上のレジスタ値と実際のレジスタとの対応関係は設計者以外には分からない．そのため，スキャンベース攻撃では，スキャンデータとレジスタの対応関係を求めることが重要である．

スキャンベース攻撃の既存研究としてストリーム暗号 Trivium への攻撃手法 [2] がある．この手法では，Trivium の暗号回路の内部レジスタを解析し暗号解読する．内部レジスタのスキャンデータ上でのビット位置を特定の秘密鍵・IV を設定することで求める．ビット位置を求めるレジスタ毎に秘密鍵・IV を用意する必要がある．また，暗号回路の内部レジスタのみがスキャンチェインに含まれていること (図 2.4) を前提としており，周辺回路のレジスタがスキャンチェインに含まれている場合，内部レジスタの解析は困難なため暗号解読できない．しかし一般に，LSI

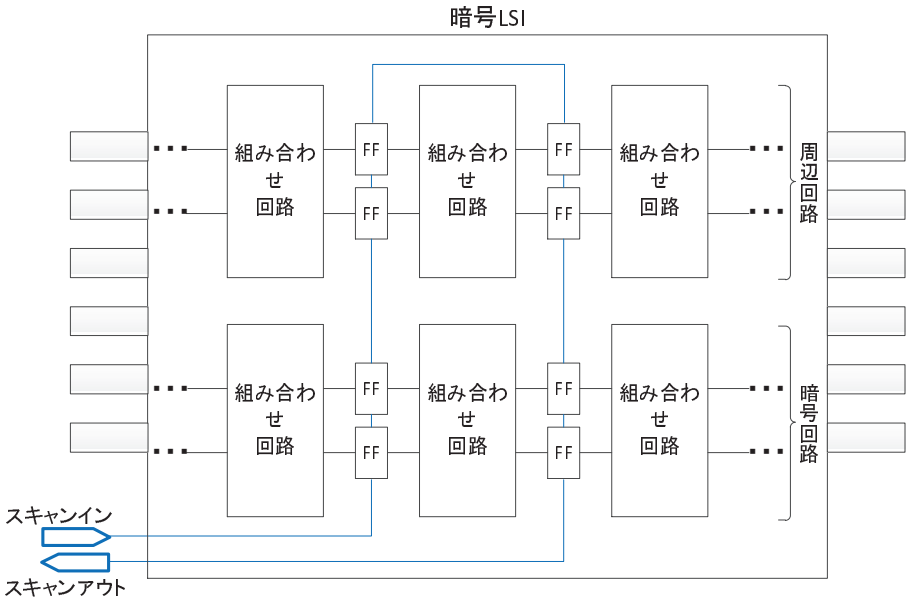


図 2.5: 全 FF がスキャンチェインに接続されている場合.

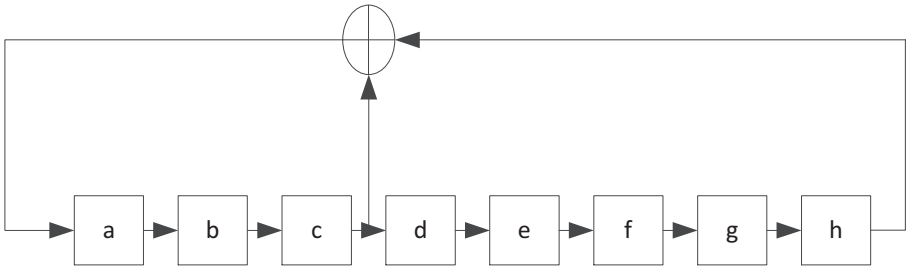
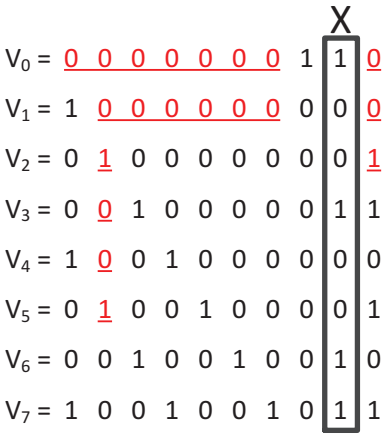


図 2.6: 入力なしフィボナッチ LFSR.

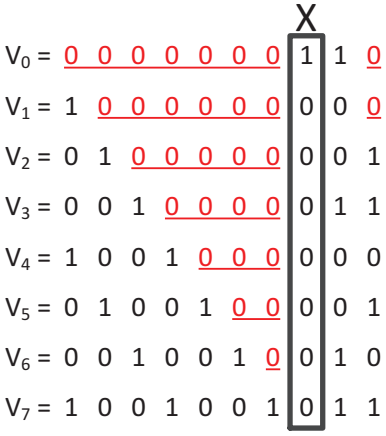
チップ上の暗号回路以外の複数の周辺回路が同一スキャンチェインに接続されること (図 2.5) は多い.

[22] は線形フィードバックシフトレジスタ (LFSR: Linear Feedback Shift Register) を用いたストリーム暗号への攻撃手法を提案しており，一般的なスキャンベース攻撃で用いるスキャン・インに入力するテストデータや回路への入力を必要としない．LFSR の種類毎にスキャンチェインの構造を決定する手法を示しており，スキャンチェインの構造が特定できれば，取得したスキャンデータから暗号回路の内部レジスタ値が判明する．スキャンチェインの構造の特定を 6 つのストリーム暗号アルゴリズム DECIM [10], Pomaranch, A5/1, A5/2, w7, LILI II に適用している．

図 2.6 に示す入力なしフィボナッチ LFSR の場合のスキャンチェインの構造の特定法を示す．この手法ではスキャンデータをサイクル毎に取得する．サイクル毎に取得したスキャンデータを比較し，ある 1 つの特定の値について前のサイクルで



(a) ミス時.



(b) ヒット時.

図 2.7: スキャンデータ上での値の探索 ([22] より作成).

スキャンデータ上のどこに存在するかを探索する (図 2.7(a)). 常に 1 つ前のサイクルで同じ値を示す箇所が存在すれば (図 2.7(b)), その 2 つの箇所はフィボナッチ LFSR 上のレジスタの値であり, 2 つは隣通しで位置している. ここでこれらのレジスタを X, W と名付けることとする (レジスタ X の値が 1 サイクル前のレジスタ W の値と常に一致している).

次に 1 サイクル前のスキャンデータ上でレジスタ W の値と常に一致している箇所を探索する. 存在すればその箇所はレジスタ W の左隣のレジスタの値を示している. また同様に 1 サイクル後のスキャンデータ上でレジスタ X の値と常に一致している箇所を探索する. 存在すればその箇所はレジスタ X の右隣のレジスタの値を示している. これらを順に探索することで, LFSR の右端, 左端のレジスタまでスキャンデータ上の位置を特定できる. レジスタ W の左隣 4 つのレジスタ (順に

レジスタ V，U，T，S と名付ける)，レジスタ X の右隣 2 つのレジスタ (順に Y，Z と名付ける) が探索により発見された場合，図 2.6 のレジスタ a〜h のスキャンデータ上での位置はレジスタ S〜Z の位置に対応していることが判明する．

他にストリーム暗号に対するスキャンベース攻撃手法として [24] がある．この手法では，暗号化に用いるシード (seed) を特定の値に設定することで，スキャンチェインの構造を求める．シードの構造をシフトレジスタ (SR: Shift Register) を初期化する部分，LFSR の原始多項式の係数を設定する部分の 2 つに特定した後，原始多項式の係数を格納するレジスタ (CR: Configurable Register) とシフトレジスタのスキャンデータ上のビット位置を特定する．シフトレジスタのスキャンデータ上のビット位置が判明すれば，過去の内部状態を順に求められる．この時求めた内部状態をそれぞれスキャン・インを用いて入力し，キーストリームを 1 ビット出力させることでキーストリームが復元できる．

ブロック暗号に対するスキャンベース攻撃手法として AES へのスキャンベース攻撃手法 [37] がある．この手法では，平文の特定部分が排他的論理和した暗号処理中データの特定の部分 (A とする) にのみ影響を与えるという AES の特性を利用する．特定の平文を複数入力し取得したスキャンデータを排他的論理和し，変化を観測することでスキャンデータ上の A のビット位置を求める．A のハミング重みを元に秘密鍵を部分ごとに順に解読する．しかし，この手法ではスキャンチェインはデータレジスタのみを接続していることを前提としているため，周辺回路が含まれている場合の動作は不明である．

[25] はスキャンチェインの構造に依存しない AES へのスキャンベース攻撃手法を提案している．スキャンデータを排他的論理和することでスキャンデータへの秘密鍵の影響を削減している．

[3–6] はテストモードのみを使用する AES に対するスキャンベース攻撃手法を示している．テストモードでは平文や途中入力値をテストベクトルとして入力し，出力をキャプチャできる．[3–6] では始めにスキャンチェインにおける AES のラウンドレジスタ 128 ビットの位置を特定する．一部のみ異なるテストパターンを 2 つ入力し，各出力のハミング距離と差分を比較することで，AES のラウンドレジスタ 128 ビット全体の位置を特定する．AES のアルゴリズムの特性から，これら 128 ビット内における 32 ビットブロックの位置を順に特定し，更にその内部の 8 ビットブロックの位置を順に特定する．最後に 8 ビットブロック内における各ビットの位置を特定する．これらの情報を元に秘密鍵を復元する．

また，DES へのスキャンベース攻撃手法 [21, 36] がある．[36] は複数の平文を DES 暗号 LSI に入力することでレジスタ位置とスキャンデータのビットの対応を特定する．攻撃対象のスキャンチェインが暗号 LSI の特定のレジスタのみで構成さ

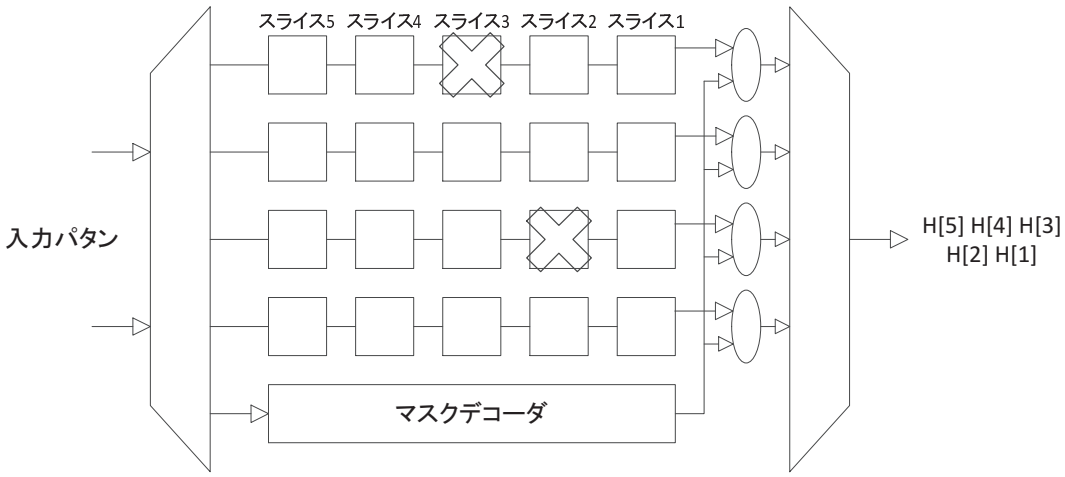


図 2.8: 空間圧縮・マスク技術を実装した回路 ([30] より作成).

れていることを仮定している．一方で，[21] はスキャンチェインのレジスタの構成に依存しない手法を提案している．また暗号 LSI が動作するタイミングが不明の場合にも有効である．

[30–34] はスキャンデータの圧縮やマスクを行う暗号 LSI へのスキャンベース攻撃手法を示している．[30–34] が対象とする回路を図 2.8 に示す．LSI に複数スキャンチェインが実装され，スキャン・インからの入力パタンは内部で展開され，スキャン・アウトの出力は圧縮されて出力される．またマスクデコーダによりスキャンチェイン中の値が除去される構造になっている．図 2.8 の例では，20 個のスキャン FF が 4 つのスキャンチェインに接続されており，5 つのスライスからなっている．ここでスライスとは各スキャンチェインの中で同じ位置にある FF を指す．この例では，各スライスの FF の値は XOR 演算による圧縮で 1 つのビット値 $H[i]$ になっている．

直接の出力値ではなく，値の差異に注目することで，圧縮の影響を除去し，スキャンデータ中で秘密情報に関連するビットを見つけることができる．複数の平文ペアについてその出力 $H[i]$ の差分を求め，鍵を予測して求めた値と比較し，一致したものを正しい秘密鍵の値とする．但し，これらはスライス内のレジスタが暗号回路の内部レジスタと暗号回路とは独立したレジスタのみから構成される場合のみ適用可能である．スライス内に暗号回路に依存する外部レジスタが接続されている LSI に対しては，鍵の値が既知で攻撃対象 LSI と構造が同一の回路が 2 つ存在すると仮定した場合，解読可能である．これらの回路の出力差分を比較することで求められる．

[30–34] は MISR (Multiple Input Signature Register) を用いた時間圧縮するス

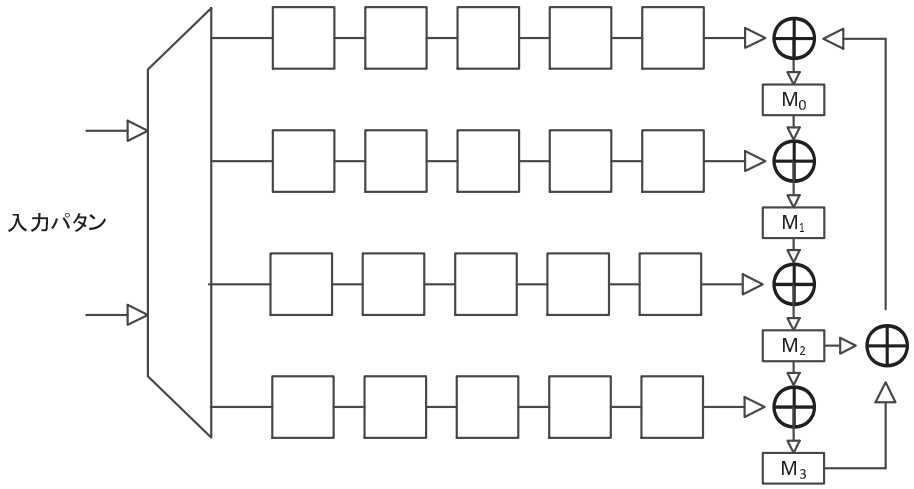


図 2.9: MISR を用いた時間的な圧縮 ([30] より作成).

キャンチェイン (図 2.9) についても対象としている．MISR ベースの時間圧縮により複数スライスから 1 つの出力値が得られる．図 2.9 の例では 4 つのスライスから 4 ビットの出力が得られる．攻撃者が任意のタイミングで MISR の FF を観察できる，または更新度に観察できる場合，同じ手法が適用できる．一方で，MISR が BIST (Built-In Self-Test) とともにに用いられている場合は，攻撃者はテスト機能を利用できないため，スキャンベース攻撃できない.

公開鍵暗号に対するスキャンベース攻撃の既存研究としては，RSA へのスキャンベース攻撃 [26] がある．この手法では，右向きバイナリ法 (図 2.10) を用いて復号処理する RSA において秘密鍵のビットを MSB から順に解読する．秘密鍵のビット値を予想して中間値を計算し，スキャンデータと比較することで，秘密鍵のビット値を求める.

図 2.10 は秘密鍵が 2 進数で 1011 の時を表している．右向きバイナリ法では秘密鍵の MSB から順にビットの値に応じて処理を行う．ビットが 0 の時，2 乗剰余演算のみを実行し，ビットが 1 の時，平方剰余演算と乗算剰余演算を実行する．図 2.10 では秘密鍵の MSB が 1 であるため，平方剰余演算と乗算剰余演算を実行し，秘密鍵の次のビットが 0 であるため 2 乗剰余演算を実行する．[26] では秘密鍵の MSB を初めに予測し，スキャンデータと比較・解析することで解読できるとしている.

また他に離散対数問題を利用した楕円曲線暗号 (ECC: Elliptic Curve Cryptosystems) に対するスキャンベース攻撃 [27] が報告されている．[27] では，点の乗算を効率的に行うアルゴリズム Montgomery 手法に着目している．秘密鍵の 1 ビットの値により，べき乗計算ループ内における途中演算値は異なるため，シミュレータにより計算した値と実際のスキャンデータを比較することで，秘密鍵のビット値を順

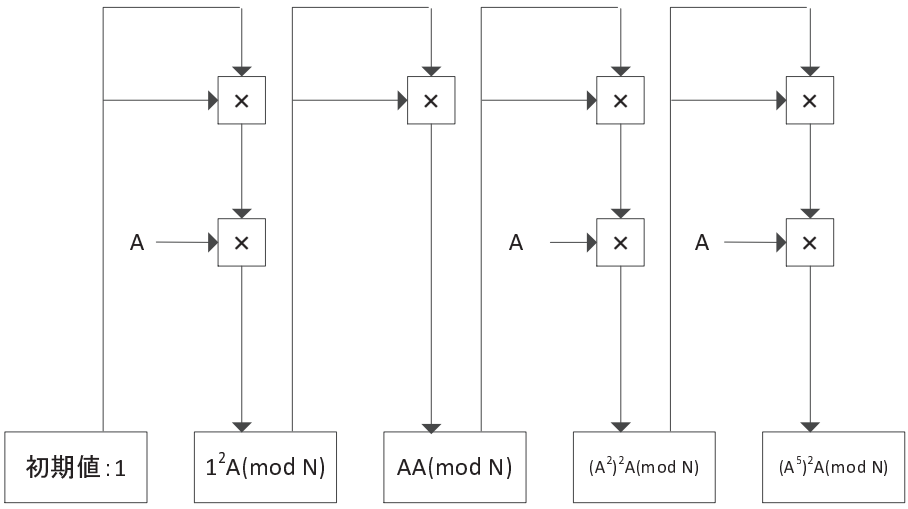


図 2.10: 右向きバイナリ法 ([26] より作成).

に求めることができる.

ハッシュを利用する LSI に対するスキャンベース攻撃はこれまでに報告されていない.

表 2.1 に対象としている暗号アルゴリズム毎にスキャンベース攻撃の既存研究一覧と本論文に示す提案手法の位置付けを示す. 表 2.1 に示すように, スキャンベース攻撃の既存研究の多くは AES, DES を対象にしており, 他のブロック暗号やストリーム暗号に関する安全性の研究は少ない. またハッシュへのスキャンベース攻撃の既存研究は存在しない. 攻撃手法が特定の条件下でのみ有効であるなら, スキャンベース攻撃の危険性を完全には指摘しきれていない.

限られた暗号アルゴリズムや実装法ではなく, 幅広い暗号方式に対してスキャンベース攻撃の危険性を求める必要があると考え, 本論文ではブロック暗号, ストリーム暗号, ハッシュを対象としたスキャンベース攻撃手法を示している. またスキャンチェインにおけるレジスタの構成や接続順, 周辺回路, 秘密鍵長等に依存していない手法となっており, 一般的な暗号 LSI に対して有効といえる.

表 2.1: 既存研究と本研究の位置づけ.

方式	対象	手法	特徴
ブロック暗号	AES	Yang et al. [37]	
		Nara et al. [25]	スキャンチェインの構造に非依存
		Rolt et al. [30–32]	XOR ベースの空間圧縮，マスク，MISR ベースの時間圧縮する LSI へ攻撃可能
		Ali et al. [3–6]	テストモードのみ使用で攻撃可能 XOR ベースの空間圧縮する LSI へ攻撃可能
	DES	Yang et al. [36]	
		Kodera et al. [21]	スキャンチェインの構造に非依存
		Rolt et al. [30]	XOR ベースの空間圧縮，マスク，MISR ベースの時間圧縮する LSI へ攻撃可能
	LED	本論文の提案手法	LED を対象とした既存研究は存在しない スキャンチェインの構造に非依存
ストリーム暗号	LFSR ベース	Mukhopadhyay et al. [24]	
		Liu et al. [22]	スキャンチェインの構造に非依存
	Trivium	Agrawal et al. [2]	スキャンチェインの構造に依存
		本論文の提案手法	スキャンチェインの構造に非依存
公開鍵暗号	RSA	Nara et al. [26]	スキャンチェインの構造に非依存
		Rolt et al. [30, 34]	XOR ベースの空間圧縮，マスク，MISR ベースの時間圧縮する LSI へ攻撃可能
	ECC	Nara et al. [27]	スキャンチェインの構造に非依存
		Rolt et al. [30, 33]	XOR ベースの空間圧縮，マスク，MISR ベースの時間圧縮する LSI へ攻撃可能
ハッシュ	HMAC	本論文の提案手法	ハッシュを対象とした既存研究は存在しない スキャンチェインの構造に非依存

2.4 本章のまとめ

本章ではサイドチャネル攻撃に関する既存研究を紹介した．

各節の内容を以下にまとめる．

第 2.2 節「サイドチャネル攻撃に関する既存研究」では，サイドチャネル攻撃の既存研究を紹介した．サイドチャネル攻撃はハードウェアの特性を利用した攻撃であり，攻撃に利用される情報は，電力，故障情報，キャッシュ情報，タイミング情報等，多岐にわたる．これらを利用したサイドチャネル攻撃の種類と脅威を紹介した．

第 2.3 節「スキャンベースサイドチャネル攻撃に関する既存研究」では，スキャンベースサイドチャネル攻撃の既存研究を紹介した．スキャンベースサイドチャネル攻撃はテスト用スキャンチェインを悪用したサイドチャネル攻撃で，スキャンチェインから取得したレジスタの情報を解析し，暗号回路の秘密情報を復元する攻撃である．対象としている暗号アルゴリズム毎にスキャンベース攻撃の既存研究を紹介した．

第3章　ストリーム暗号へのスキャンベースサイドチャネル攻撃

3.1　本章の概要

本章では，ストリーム暗号 Trivium の性質とアルゴリズムを示し，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案する．

以下に本章の構成を示す．

第 3.2 節「ストリーム暗号 Trivium」では，ストリーム暗号 Trivium のアルゴリズムを説明する．ストリーム暗号では，ビット毎あるいはバイト毎にキーストリームと排他的論理和し暗号化・復号する．Trivium は同期式ストリーム暗号で，3 本のシフトレジスタから構成され，内部の演算はビット同士の AND 演算と XOR 演算のみであるため，構造が単純で高速に動作する．

第 3.3 節「Trivium に対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案する．提案手法は，1 ビットレジスタ値の入力・動作サイクル数に対する変化がそのレジスタ固有の値になることを利用した手法である．提案手法では，スキャンチェインに Trivium の内部状態レジスタ 288 個が含まれていれば，スキャンチェインの構造によらず，スキャンベース攻撃できる．

第 3.4 節「評価実験」では，提案手法のソフトウェア実験の結果を示し，提案手法の有効性を評価する．

第 3.5 節「本章のまとめ」では，本章の内容をまとめる．

本章は [13, 15, 39, 40] で発表した内容から構成される．

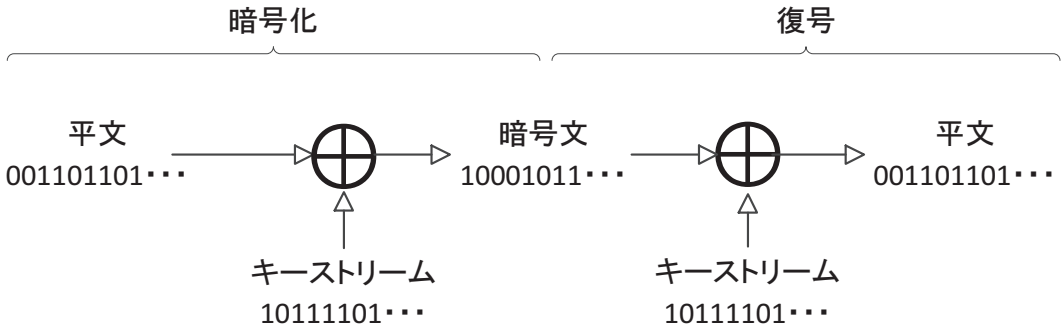


図 3.1: 同期式ストリーム暗号の暗号化・復号.

3.2　ストリーム暗号 Trivium

ストリーム暗号は，共通鍵暗号方式の暗号で，ビット毎あるいはバイト毎に順次暗号化・復号する．一般に，キーストリームと呼ばれる平文と同じ長さを持つ乱数列を用い，平文の 1 桁毎にキーストリームと排他的論理和して暗号化し，暗号文の 1 桁毎にキーストリームと排他的論理和して復号する．平文のビットを P_i，キーストリームのビットを K_i，暗号文のビットを C_i としたときの暗号化の式を式 3.1 に，復号を式 3.2 に示す.

$$P_i \oplus K_i = C_i \tag{3.1}$$

$$C_i \oplus K_i = (P_i \oplus K_i) \oplus K_i = P_i \oplus (K_i \oplus K_i) = P_i \tag{3.2}$$

ストリーム暗号は，暗号・復号回路が同一で済み，構造が単純なため，高速に動作する．ストリーム暗号には，非同期式ストリーム暗号と同期式ストリーム暗号の 2 種類がある．非同期式ストリーム暗号は平文や暗号文の系列に依存してキーストリームを生成する．同期式ストリーム暗号は平文や暗号文とは独立にキーストリームを生成し暗号化する．一般に，同期式ストリーム暗号のキーストリームの生成は，秘密鍵と IV(initialization vector: 初期化ベクトル) のみに依存する．図 3.1 に同期式ストリーム暗号の暗号化・復号の様子を示す.

ストリーム暗号のアルゴリズムは，KSA(Key Scheduling Algorithm: 鍵スケジューリングアルゴリズム) と PRGA(Pseudo-Random Generation Algorithm: 疑似乱数生成アルゴリズム) からなる．KSA では秘密鍵と IV を入力として内部状態を初期化し，PRGA では初期化された内部状態を入力として内部状態を更新しキーストリームを生成する．図 3.2 にキーストリーム生成と暗号化の様子を示す.

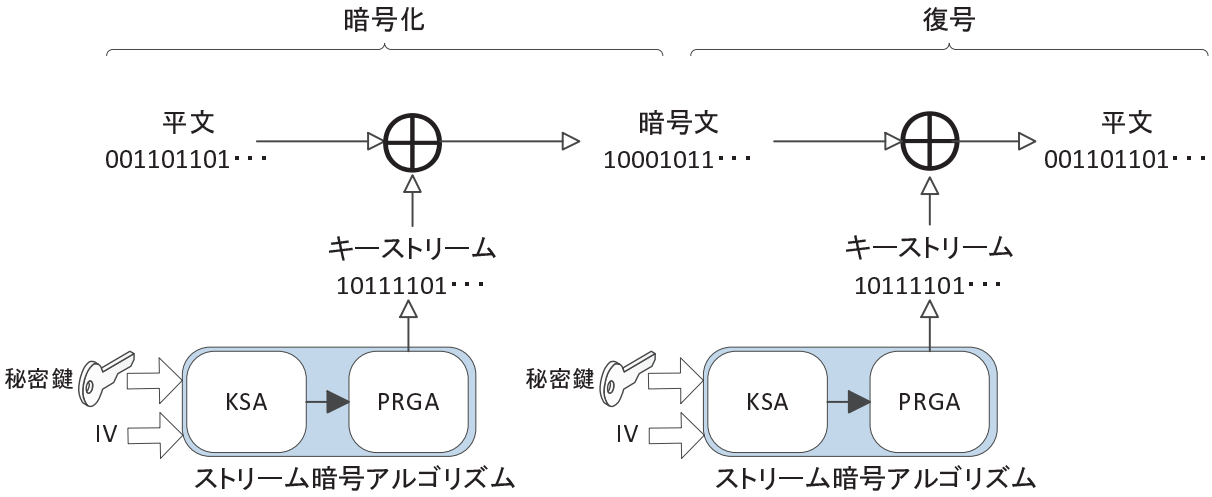


図 3.2: 同期式ストリーム暗号のキーストリーム生成と暗号化.

Trivium のアルゴリズム

Trivium [12] は Cannière らが提案した同期式ストリーム暗号である．ストリーム暗号の安全性の評価の研究は少ないが，近年実施されたストリーム暗号評価プロジェクト eSTREAM では Trivium は推奨暗号に認定された．また軽量で動作が高速であり，スマートカードでの実装に適しているため，本章では Trivium を攻撃対象暗号アルゴリズムとする．

以降，[12] に従い Trivium のアルゴリズムを紹介する．3 本のシフトレジスタから構成され，内部演算はビット同士の AND 演算と XOR 演算のみであるため，構造が単純で高速に動作する．秘密鍵 K (80 ビット) と IV (Initialization Vector: 初期化ベクトル) (80 ビット) でキーストリームを 2^{64} ビットまで生成する．図 3.3 に表すようにこのキーストリームと平文とを 1 ビット毎に XOR 加算することで暗号化する．同様に暗号文とキーストリームを 1 ビット毎に XOR 加算することで平文を復元する．

図 3.4 に Trivium のハードウェア構造を示す．図 3.4 の太枠で囲った四角形の部分が Trivium の内部状態レジスタを示している．1bit レジスタ 288 個が円状に並び，シフトレジスタを形成している．サイクル毎にレジスタの値がシフトされ，s_1, s_{94}, s_{178} に非線形関数の演算結果が設定される (詳細は後に記述する)．Trvium LSI は内部状態レジスタ 288 個がスキャンチェインで接続されている．

Trivium のキーストリーム生成は 2 つのフェーズからなる (図 3.3)．秘密鍵と IV を入力として内部状態を初期化する初期化フェーズと内部状態を入力として内部状態を更新しキーストリームのビットを生成するキーストリーム生成フェーズである．

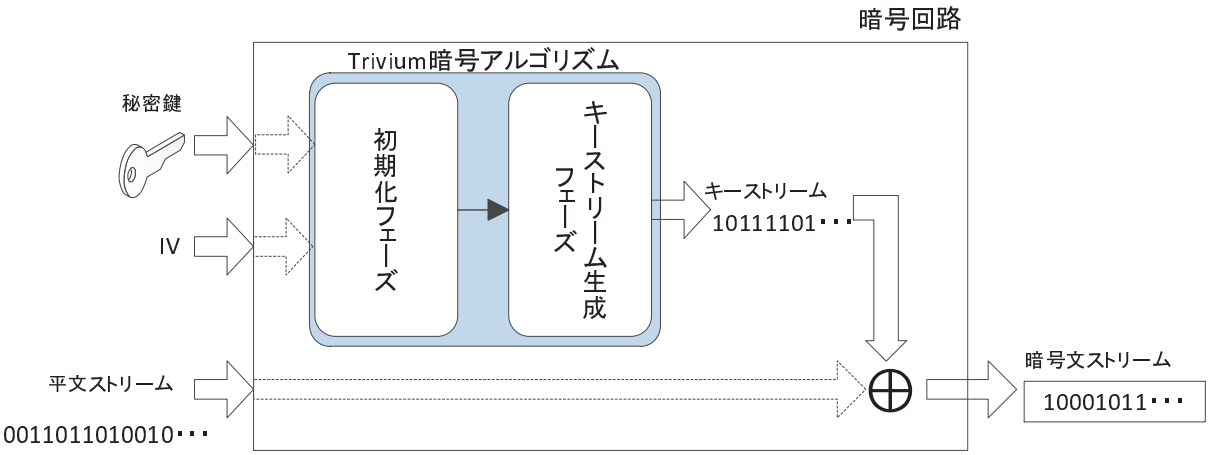


図 3.3: Trivium の暗号回路の概略図.

初期化フェーズ

Trivium には 1 ビットの内部状態レジスタが 288 個ある．これらを $s_1, \ldots, s_{288}$ とする．初期化フェーズでは，レジスタへの初期値設定作業とレジスタ値の更新と循環作業を順に実行する．初期値設定作業では，80 ビットの秘密鍵と 80 ビットの IV を 288 ビットの内部状態レジスタに設定し初期化する．秘密鍵 K の各ビットを $K_1, \ldots, K_{80}$, IV の各ビットを $IV_1, \ldots, IV_{80}$ とする．Algorithm 3.1 に示す.

Algorithm 3.1 初期化フェーズ (初期値設定)

$(s_1, s_2, \ldots, s_{93}) \leftarrow (K_1, \ldots, K_{80}, 0, \ldots, 0)$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (IV_1, \ldots, IV_{80}, 0, \ldots, 0)$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (0, \ldots, 0, 1, 1, 1)$

Algorithm 3.1 は 1 クロックサイクルで実行される.

続いて更新と循環作業では，内部状態レジスタ中の特定の 15 個のビットの値を用いて内部状態レジスタ値を更新し，循環させる．これを 4 回繰り返す．Algorithm 3.2 に示す.

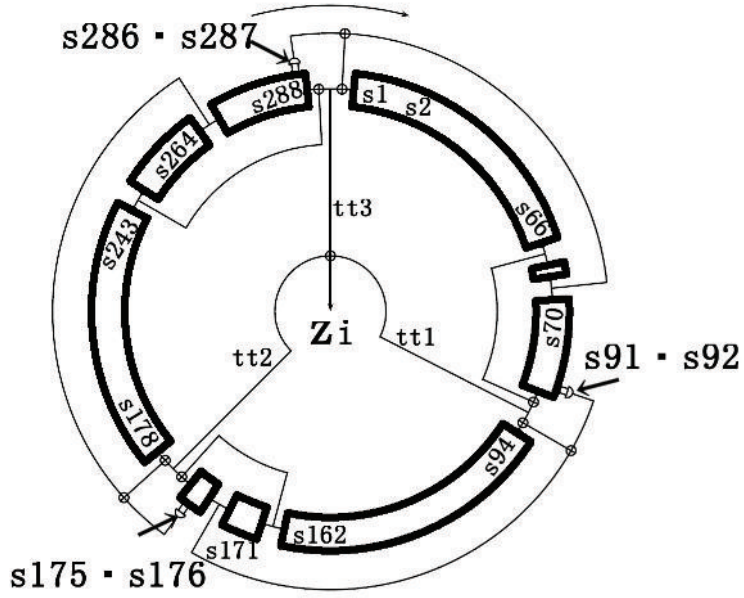


図 3.4: Trivium の構造 [12].

Algorithm 3.2 初期化フェーズ (更新と循環)

for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} \oplus s_{91} \cdot s_{92} \oplus s_{93} \oplus s_{171}$

$t_2 \leftarrow s_{162} \oplus s_{175} \cdot s_{176} \oplus s_{177} \oplus s_{264}$

$t_3 \leftarrow s_{243} \oplus s_{286} \cdot s_{287} \oplus s_{288} \oplus s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Algorithm 3.2 は $4 \cdot 288 = 1152$ クロックサイクルで実行される.

キーストリーム生成フェーズ

キーストリーム生成フェーズでは，内部状態レジスタ中の特定の 15 個のビットの値を用いて 3 個のビットを更新し，キーストリーム z_i を 1bit ずつ算出する．内部状態レジスタのビットは循環し，キーストリームの全ビット N $(\leq 2^{64})$ の生成が終了するまで繰り返す．キーストリーム生成フェーズで実行する作業を Algorithm 3.3 に示す.

Algorithm 3.3 キーストリーム生成フェーズ

for $i = 1$ to N **do**

$tt_1 \leftarrow s_{66} \oplus s_{93}$

$tt_2 \leftarrow s_{162} \oplus s_{177}$

$tt_3 \leftarrow s_{243} \oplus s_{288}$

$z_i \leftarrow tt_1 \oplus tt_2 \oplus tt_3$

$t_1 \leftarrow tt_1 \oplus s_{91} \cdot s_{92} \oplus s_{171}$

$t_2 \leftarrow tt_2 \oplus s_{175} \cdot s_{176} \oplus s_{264}$

$t_3 \leftarrow tt_3 \oplus s_{286} \cdot s_{287} \oplus s_{69}$

$(s_1, s_2, \ldots, s_{93}) \leftarrow (t_3, s_1, \ldots, s_{92})$

$(s_{94}, s_{95}, \ldots, s_{177}) \leftarrow (t_1, s_{94}, \ldots, s_{176})$

$(s_{178}, s_{179}, \ldots, s_{288}) \leftarrow (t_2, s_{178}, \ldots, s_{287})$

end for

Algorithm 3.3 は N クロックサイクルかけて実行され，クロックサイクル毎にキーストリームの各ビット z_i が出力される．そして，キーストリーム生成フェーズで生成したキーストリームと平文とを1ビット毎にXOR加算することで暗号化する．同様に暗号文とキーストリームを1ビット毎にXOR加算することで平文を復元する．

3.3 Trivium に対するスキャンベース攻撃手法

ストリーム暗号に対するスキャンベース攻撃手法として，2008 年に Agrawal らが提案した Trivium への攻撃手法 [2] がある．この手法では，Trivium の暗号回路の内部レジスタを解析し暗号解読するが，暗号回路の内部レジスタのみがスキャンチェインに含まれていることを前提とし，周辺回路のレジスタがスキャンチェインに含まれている場合，暗号解読できない．一般に，LSI チップには暗号回路と共に複数の回路が同一スキャンチェインに含まれることが多く，この手法を実際の攻撃手法として利用することは難しい．

本節では，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案する．提案手法では，1 ビットレジスタ値の入力・動作サイクル数に対する変化がそのレジスタ固有の値になることを利用し，スキャンチェインの構造を求める．提案手法を用いることで，周辺回路のレジスタがスキャンチェインに含まれている場合にも平文を復元できる．計算機実験の結果，他の回路のビットが含まれていても，Trivium の内部状態を復元でき，元の平文を復元できた．

Trivium 攻撃の前提条件

Trivium をはじめ同期式ストリーム暗号では平文や暗号文とは独立にキーストリームを生成し暗号化・復号する．一般に，同期式ストリーム暗号のキーストリームの生成は，秘密鍵と IV のみに依存し，暗号化側と復号側で同期をとる必要がある．暗号化側と復号側で事前に秘密鍵・IV を共有し，それぞれ秘密鍵・IV からキーストリームを生成し，同期をとりながら平文/暗号文と排他的論理和することで暗号化/復号する．

ここで [2] にならい，Trivium へのスキャンベース攻撃において，攻撃者が知っていることを以下に示す．

(K1) Trivium LSI が出力した暗号文 C

(K2) Trivium LSI が暗号文 C を出力した直後のスキャンデータ

攻撃者が分からないことを以下に示す．

(U1) 暗号文 C を生成するために使用した秘密鍵・IV・キーストリーム KS

(U2) Trivium LSI のスキャンチェインに含まれるレジスタの数や種類，接続順

攻撃者ができることを以下に示す．

(A1) Trivium LSI に任意の秘密鍵と任意の IV を与え任意ビットのキーストリームが生成できる

(A2) 任意のタイミングで Trivium LSI のスキャンチェインにアクセスでき，スキャンデータを得られる

攻撃者は任意の秘密鍵と IV を Trivium LSI に設定し，暗号処理中のレジスタの値をスキャンデータとして取得できる (A1, A2) が，スキャンチェインに含まれるレジスタの種類・接続順は分からない (U2)．このままでは，Trivium LSI が出力した暗号文，暗号化後のスキャンデータを取得 (K1, K2) しても，平文は復元できない．

内部状態の復元

暗号化に用いられたキーストリームが不明であっても，暗号文とそのときのスキャンデータから元の平文を復元することを考えよう．今，次の仮定をおく．

仮定：暗号文を出力した直後の Trivium の内部レジスタ 288 個の値が全て分かる

以下，まずこの仮定をおくと，過去の Trivium の内部状態を復元できることを示す．

表 3.1 に Trivium における現在の内部状態 (時刻 T) と 1 サイクル前の内部状態 (時刻 $T-1$) の関係を示す．ここで，$s_1^T, \ldots, s_{288}^T$ は時刻 T の内部状態レジスタ $s_1, \ldots, s_{288}$ の値を示す．仮定から $s_1^T, \ldots, s_{288}^T$ は既知とする．表 3.1 の左の式から，

$$
\begin{aligned}
s_1^{T-1} = s_2^T, \ldots, s_{92}^{T-1} = s_{93}^T \\
s_{94}^{T-1} = s_{95}^T, \ldots, s_{176}^{T-1} = s_{177}^T \\
s_{178}^{T-1} = s_{179}^T, \ldots, s_{287}^{T-1} = s_{288}^T
\end{aligned}
$$

となり，時刻 $T-1$ の内部状態レジスタの値は，$s_{93}^{T-1}, s_{177}^{T-1}, s_{288}^{T-1}$ を除き，直ちに求めることが出来る．つまり，$s_{93}^{T-1}, s_{177}^{T-1}, s_{288}^{T-1}$ が算出できれば，時刻 T の内部状態レジスタの値から時刻 $T-1$ の内部状態レジスタの値を復元できることになる．

時刻 $T-1$ において Algorithm 3.3 より，

$$
\begin{aligned}
tt_1 &\leftarrow s_{66}^{T-1} \oplus s_{93}^{T-1} && (3.3) \\
t_1 &\leftarrow tt_1 \oplus s_{91}^{T-1} \cdot s_{92}^{T-1} \oplus s_{171}^{T-1} && (3.4) \\
(s_{94}^T, s_{95}^T, \ldots, s_{177}^T) &\leftarrow (t_1, s_{94}^{T-1}, \ldots, s_{176}^{T-1})
\end{aligned}
$$

であるから，式 (3.3)，式 (3.4) を統合して，

$$
t_1 \leftarrow s_{66}^{T-1} \oplus s_{93}^{T-1} \oplus s_{91}^{T-1} \cdot s_{92}^{T-1} \oplus s_{171}^{T-1}
$$

表 3.1: 内部状態の現在と 1 サイクル前の関係.

現在の内部状態 (時刻 T)	1 サイクル前の 内部状態 (時刻 T-1)
$(s_1^T, \ldots, s_{93}^T) = (t_3, s_1^{T-1}, \ldots, s_{92}^{T-1})$	$(s_1^{T-1}, \ldots, s_{93}^{T-1})$
$(s_{94}^T, \ldots, s_{177}^T) = (t_1, s_{94}^{T-1}, \ldots, s_{176}^{T-1})$	$(s_{94}^{T-1}, \ldots, s_{177}^{T-1})$
$(s_{178}^T, \ldots, s_{288}^T) = (t_2, s_{178}^{T-1}, \ldots, s_{287}^{T-1})$	$(s_{178}^{T-1}, \ldots, s_{288}^{T-1})$

と表せる．ここで，$t_1 = s_{94}^T, s_{66}^{T-1} = s_{67}^T, s_{91}^{T-1} = s_{92}^T, s_{92}^{T-1} = s_{93}^T, s_{171}^{T-1} = s_{172}^T$ より，

$$s_{94}^T = s_{67}^T \oplus s_{93}^{T-1} \oplus s_{92}^T \cdot s_{93}^T \oplus s_{172}^T$$

s_{93}^{T-1} について解けば以下のようになる．

$$s_{93}^{T-1} = s_{94}^T \oplus s_{67}^T \oplus s_{92}^T \cdot s_{93}^T \oplus s_{172}^T \tag{3.5}$$

同様に，s_{177}^{T-1}，s_{288}^{T-1} を以下のように算出できる．

$$s_{177}^{T-1} = s_{178}^T \oplus s_{163}^T \oplus s_{176}^T \cdot s_{177}^T \oplus s_{265}^T \tag{3.6}$$

$$s_{288}^{T-1} = s_1^T \oplus s_{244}^T \oplus s_{287}^T \cdot s_{288}^T \oplus s_{70}^T \tag{3.7}$$

暗号文が出力された直後の内部状態値を取得できれば，過去のいかなる内部状態も式 (3.5), (3.6), (3.7) と表 3.1 より算出できる．内部状態値が分かれば，Algorithm 3.3 よりキーストリームを復元でき，得られたキーストリームと暗号文を 1 ビットずつ順に排他的論理和すれば元の平文が取得できる．

以上の議論より，暗号文から平文への復元は，いかに上述の仮定を満足するか，すなわち，いかに暗号文が出力された直後の内部状態値を取得するかに還元される．これを内部状態値取得問題と呼ぶことにする．以降，Trivium LSI が暗号文を出力した直後のスキャンデータを用いて，(A1)・(A2) を利用して内部状態値取得問題を解法することを考える．

内部状態値取得問題の解法

内部状態値取得問題に対して，[2] はスキャンチェインに内部状態レジスタのみを含むことを前提に，その解決手法を提案した．しかし，暗号回路以外のレジスタがスキャンチェインに含まれている場合，[2] は内部状態値取得問題を解法できない．一般に，LSI チップには暗号回路と共に複数の回路が同一スキャンチェインに含まれることが多く，この手法を実際の攻撃手法として利用することは難しい．

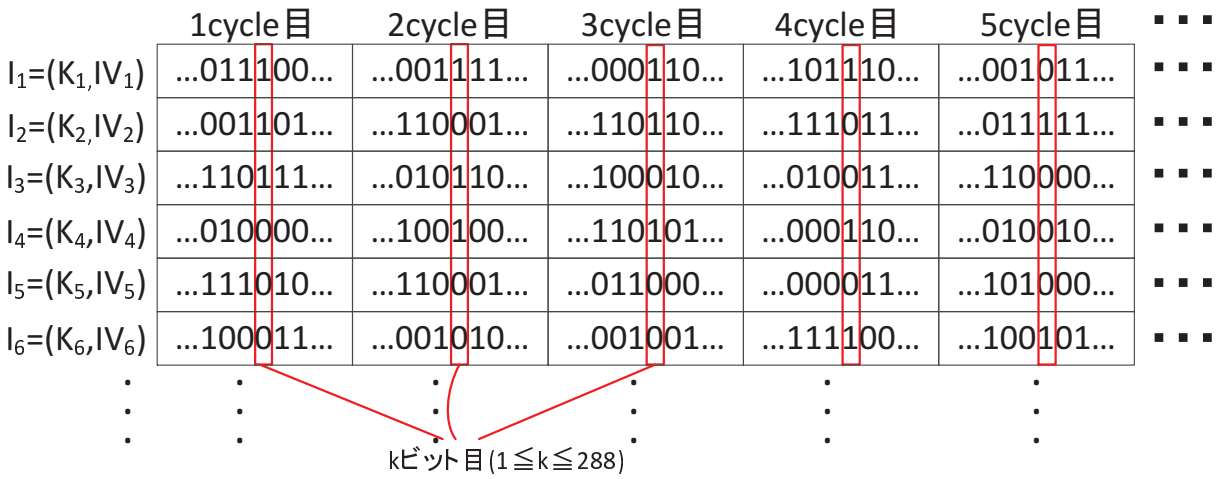


図 3.5: スキャンシグネチャ.

そこでスキャンチェイン上に暗号回路以外の周辺回路が含まれる場合にも内部状態値取得問題を解法できる手法を提案する.

(A1) より，Trivium LSI に秘密鍵と IV を任意に入力できる．そこで図 3.3 のように入力ペア (秘密鍵と IV) を多数用意し，各入力ペアに対し Trivium LSI を数サイクル動作させ，クロックサイクル毎に内部状態レジスタの値を図 3.5 のように横に並べることにする．このとき，各クロックサイクルにおいて，$k(1 \leq k \leq 288)$ ビット目に注目すれば，これはある 1 ビットの内部状態レジスタの値の変化を表し，入力ペア数，サイクル数を十分に大きくとれば，その内部状態レジスタ固有の値になることが予想される．この固有の値をスキャンシグネチャと呼ぶ.

図 3.5 にスキャンシグネチャの例を示す．図 3.5 において，縦軸は入力ペア，横軸は動作させたサイクル数で，それぞれの入力ペア・サイクル数において，内部状態レジスタ 288 個がとるビット値を示している．枠で囲った値はある内部状態レジスタのとる値である．入力ペアの数，サイクル数を十分に大きくとれば枠で囲った値は特定の 1 つの内部状態レジスタに固有の値になり，これがその内部状態レジスタのスキャンシグネチャとなる.

そこで予め Trivium の内部状態レジスタの 1 つ 1 つに対しそれぞれシミュレーションによってスキャンシグネチャを計算しておき，同様の条件下で実際の LSI 回路から取得したスキャンデータとの対応を解析すればビット対応が求まることになる．つまり，内部状態値取得問題が解法できることになる.

提案手法のアルゴリズムを以下に示す.

1. 入力ペア (秘密鍵と IV) のパターンを $I_1, \ldots, I_L$ の L 個用意する.

2. I_1 に対する Trivium の内部状態値をシミュレーションにより M サイクル分計

算する．

3. 同様に $I_2, \ldots, I_L$ について M サイクル分求める．

4. (2), (3) で求めた値に対して，Trivium の 1 つの内部状態レジスタ s_k $(1 \leq k \leq 288)$ のスキャンシグネチャを E_{s_k} とする．

5. 入力ペア $I_1, \ldots, I_L$ を実際の Trivium LSI に入力し，それぞれ M サイクル分スキャンデータを取得する．入力ペア I_i，サイクル数 j の時に取得したスキャンデータを $V_{i,j}$ とする

6. スキャンデータ $V_{1,1}, \ldots, V_{L,1}$ を縦に並べたものを $S_1 = (V_{1,1}, \ldots, V_{L,1})^t$ とする．M サイクル分 $S_1, \ldots, S_M$ を横に並べると，図 3.6(b) のようになる．このとき，スキャンシグネチャ E_{s_k} が各スキャンデータ $S_1, \ldots, S_M$ の何列目に存在するか調べる．

7. (6) で，E_{sk} が各スキャンデータ $S_1, \ldots, S_M$ の p 列目にのみ存在するとき，内部レジスタ s_k のビット位置がスキャンデータの p ビット目であることが分かる．

上記アルゴリズムにおいて，L および M を十分大きくとれば，p の値は一意に定まり，スキャンデータ中のビット位置と内部状態レジスタの対応が一意に定まる．

この手法では，スキャンデータに Trivium の内部状態レジスタ以外のレジスタの値が含まれていても，高々1 ビットの値の変化にのみ着目しているため，内部状態レジスタのビットの位置を特定できる．スキャンデータのビット対応が一度求まれば (K2)，Trivium LSI が暗号文を出力した直後のスキャンデータから Trivium の内部状態レジスタの各値を求められる．Trivium の内部状態レジスタ値が求まれば，前の議論により，過去の内部状態，キーストリームを復元でき，Trivium LSI が出力した暗号文と排他的論理和することで，平文が復元できる．

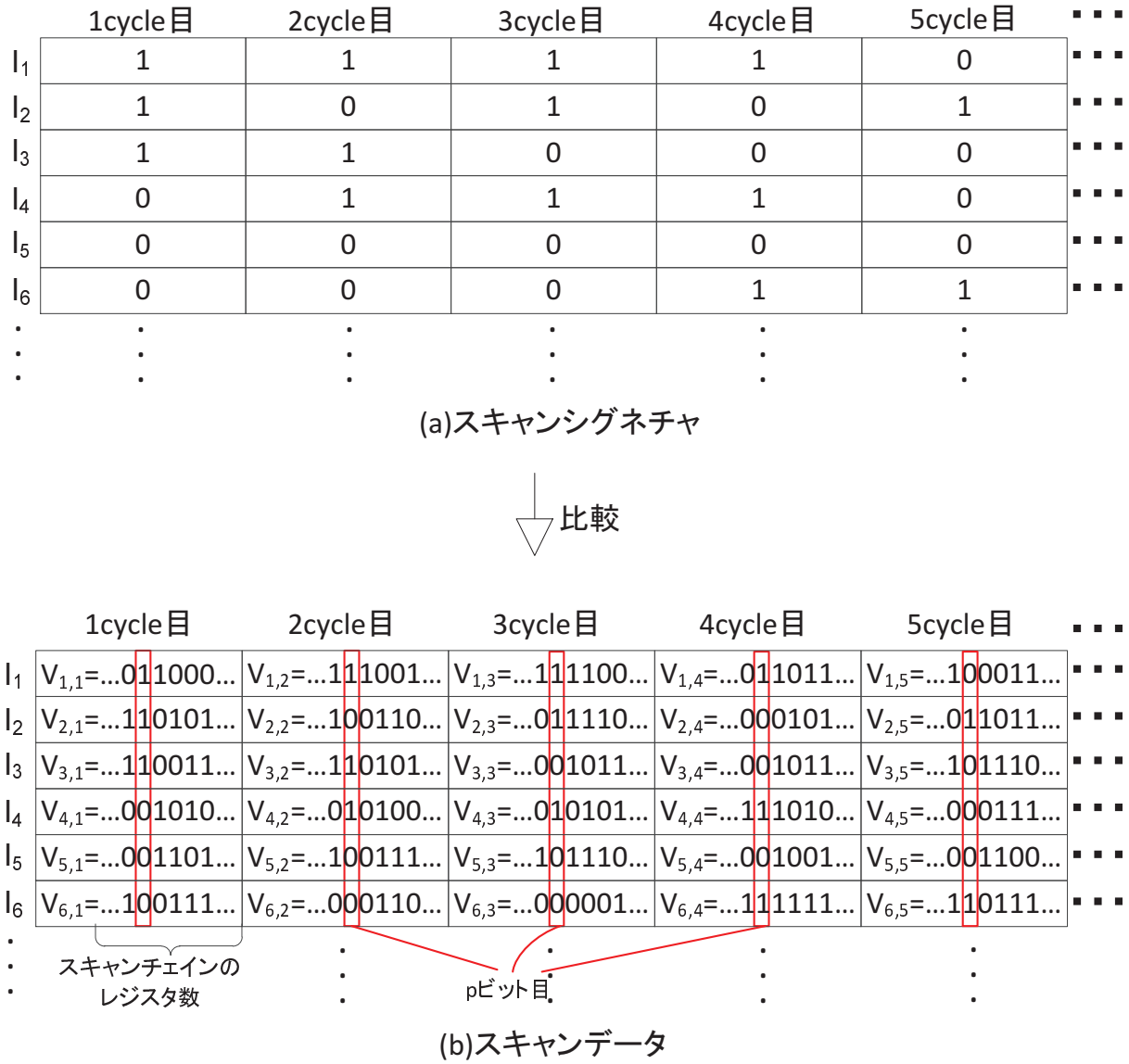


図 3.6: スキャンシグネチャとスキャンデータの比較 (発見時).

3.4 評価実験

本節では，提案手法を用いて Trivium の内部状態レジスタとスキャンデータのビット対応を求め平文を復元する実験の結果を説明する．実験では，提案手法を C 言語で実装し，暗号回路のシミュレータは文献 [12] のコードを使用した．暗号文・キーストリームは，512 ビットを想定した．Trivium の全内部状態レジスタ (288 個) についてスキャンデータ上のビット位置を特定する (これをビット対応解析と呼ぶ) ために必要な入力ペア数・サイクル数の最小値，その時の入力ペア，解析時間を求める．本実験は，CPU が Intel(R) Core(TM) i7-2620M 2.70GHz × 4，メモリが 8GB の計算機を用い，コンパイラは gcc を使用した．

実験方法

実験は，以下の条件で実行した．

1. スキャンデータの取得開始タイミングとスキャンチェイン長を変化させる実験

スキャンチェインに周辺回路のレジスタが含まれている場合と Trivium の内部状態レジスタ 288 個のみを含む場合の 2 つを想定する．今回は，スキャンデータのサイズが 512，1024，2048，4096 ビットになるようにランダムなビット値を加えた．スキャンデータは，キーストリーム生成の初期化フェーズの 1 サイクル目からサイクル毎に取得，キーストリーム生成フェーズの 1 サイクル目からサイクル毎に取得の 2 パターンを考え，ビット対応解析に必要な入力数とサイクル数，解析時間をそれぞれ求める．

2. 入力ペアの値と入力ペア数を変化させる実験

入力ペア数を L 個 $(1 \leq L \leq 7)$ としたとき，入力ペアの値を変化させ，ビット対応解析に必要な最小サイクル数，その時の入力ペア，解析時間を求める．

取得サイクル数を 1 として入力ペア数，入力ペアの値を変化させ，ビット対応解析に必要な最小入力ペア数，その時の入力ペア，解析時間を求める．

実験結果

実験結果を以下に示す．

1. スキャンデータの取得開始タイミングとスキャンチェイン長を変化させる実験

キーストリーム生成の初期化フェーズの 1 サイクル目からサイクル毎にスキャンデータを取得した場合の結果を表 3.3 に示す．また，この時の秘密鍵・IV の値を表

3.4 に示す．*

スキャンチェイン長を 288 から 4096 に変化しても，入力ペア数 1，サイクル数 108 でビット対応解析に成功した．これは，Algorithm 3.1，Algorithm 3.2 より，秘密鍵，IV に関わらず s_{284} と s_{285} が 107 サイクル目までともに 0 であり，初期化フェーズからスキャンデータを取得した場合，入力ペア数，入力ペアの値に関わらず，107 サイクル以下でビット対応は解析できないためである．初期化フェーズに取得したスキャンデータからビット対応解析することは，効率的でないと分かる．

スキャンデータにランダムなビット値を加え解析した場合でも，必要なサイクル数は変化せず 108 サイクルであった．

キーストリーム生成フェーズの 1 サイクル目からサイクル毎にスキャンデータを取得した場合の結果を表 3.5 に示す．この実験でも秘密鍵・IV は表 3.4 の値を使用した，スキャンチェイン長が 4096 ビットまでの場合，30 サイクル程度までスキャンデータを取得すればビット対応解析が成功する．キーストリーム生成フェーズでは，シフトレジスタを初期化してから既に 4× 288 回更新と循環を繰り返しているため，初期化フェーズと比較してレジスタ値の変化の頻度は平均して多い．そのため，解析に必要なサイクル数は初期化フェーズより少ない．効率的にビット対応を解析するためには，キーストリーム生成フェーズからスキャンデータを取得する必

*表 3.4 の値以外に 100 個のランダムな入力ペアを用いて実験を実行した結果，表 3.4 や表 3.2 の値の時，最小サイクル数になることを確認した．

より小さなサイクル数になる入力が他に存在する可能性はあるが，表 3.3, 3.5 の解析時間からいって，表 3.4 の値は十分に効率的な入力ペアであるといえる．

表 3.2: 入力ペア．

秘密鍵 (80 ビット)	IV (80 ビット)
84327C64B0AA55E6DA55	1EF269B92FC8898D884D
56889874D3CE461ECF0F	08BA882D7A4CEE3FB5A0
9C3B3FA28CD8E2D52284	1D15A9EA8CE1A30F3541
AA4523DB2A486FBEEBD4	5453856D0A68B5DC48BD
F5BD474DC1875D41A9DA	F452D557DBF5316D7E30
C3970430E7757B3CE34C	E98B93553DCF279DD45D
B3CB3ADAC955DF017FD4	D90A0BB3F6A360F6258A
9433A72F8D14A88BB8E6	B808FB71684DD1A8C6E3
ECB5AFDAC2C322F63E3B	8F4528DDB76160060FD2

表 3.3: 初期化フェーズからスキャンデータを取得した場合の結果.

スキャンチェイン長	追加ビット数	入力ペア数	必要サイクル数 (平均)	必要サイクル数 (最悪)	解析時間 [s]
288	0	1	108	108	1.231
512	224	1	108	108	2.074
1024	736	1	108	108	3.977
2048	1760	1	108	108	7.862
4096	3808	1	108	108	15.522

表 3.4: スキャンデータの取得開始タイミングを変化させる実験の入力ペア.

秘密鍵 K (80 ビット)	IV (80 ビット)
FFFFFFFFFF FFFFFFFFFF	FFFFFFFFFF FFFFFFFFFF

要がある.

この実験で，スキャンデータに他のビットが含まれる場合でも，提案手法により，ビット対応が解析できることが確認できた.

2. 入力ペアの値と入力ペア数を変化させる実験

入力ペア数を 1 から 7 まで変化させ，それぞれの入力ペア数で入力ペアの値を 100 通り用意しビット対応解析した．100 個の入力ペア中，最小サイクルでビット対応解析に成功したものについて結果を表 3.6 に示す．今回の実験では，入力ペア数 1 の時 100 入力ペア中 9 パターンの入力ペアで最小サイクル数 13 になった．入力ペア数 2 の時は 11 パターンの入力ペアで最小サイクル数 7 に，入力ペア数 3 の時は 39 パターンの入力ペアで最小サイクル数 5 に，入力ペア数 4 の時は 57 パターンの入力ペアで最小サイクル数 4 に，入力ペア数 5 の時は 25 パターンの入力ペアで最小サイクル数 3 に，入力ペア数 6 の時は 87 パターンの入力ペアで最小サイクル数 3 に，入力ペア数 7 の時は 5 パターンの入力ペア (表 3.7 に一例を示す) で最小サイクル数 2 になった.

取得サイクル数を 1 として入力ペア数を変化させて，それぞれの入力ペア数で入力ペアの値を 100 通り用意しビット対応解析した．今回の実験では，サイクル数 1 の時にビット対応解析に必要な最小入力ペア数は 14 になり，100 入力ペア中 10 パターンの入力ペア (表 3.8 に一例を示す) でビット対応解析が成功し，解析時間は 0.187 秒だった.

表 3.5: キーストリーム生成フェーズからスキャンデータを取得した場合の結果.

スキャンチェイン長	追加ビット数	入力ペア数	必要サイクル数 (平均)	必要サイクル数 (最悪)	解析時間 [s]
288	0	1	13	13	0.139
512	224	1	17.49	23	0.339
1024	736	1	19.09	25	0.709
2048	1760	1	20.24	25	1.501
4096	3808	1	21.88	30	3.215

表 3.6: 入力ペア数を変化させた時の最小サイクル数 (100 データ中) と比較時間.

スキャンチェイン長	追加ビット数	入力ペア数	必要サイクル数	解析時間 [s]
288	0	1	13	0.139
288	0	2	7	0.155
288	0	3	5	0.171
288	0	4	4	0.186
288	0	5	3	0.171
288	0	6	3	0.218
288	0	7	2	0.171
288	0	14	1	0.187

提案手法の性質

[2] では，全内部状態レジスタについて，各 1 回ずつスキャンデータを取得してビット対応を求めている．そのため，Trivium の内部状態レジスタ 288 個のビット位置を解析するためには，スキャンデータを 288 個取得しなければならない．また，スキャンチェインに他の回路のレジスタが含まれている場合，ビット対応解析に成功するとは限らない.

一方，提案手法では，スキャンデータに他の回路のビットが含まれていても，ビット対応解析できることを実験で確認できた．また，表 3.5 より Trivium の内部状態レジスタ 288 個のビット対応解析には，スキャンデータを 13 個取得すればよい．入力ペア数 1 で，表 3.4 の入力ペアを入力として設定し，スキャンデータをキーストリーム生成フェーズからサイクル毎に 13〜30 個取得すれば良く，解析時間は 0.139 秒で済み，最も効率的に求められることを確認した.

表 3.7: 入力ペア数 7 の時の入力ペアの値.

	秘密鍵 (80 ビット)	IV (80 ビット)
$I_1 =$	2710244320 5F6BF0ABA6	0940C7CBA0 740464E9B8
$I_2 =$	CDDD4C4521 AF96123310	6601A3FB55 AE408A48F1
$I_3 =$	57FE370926 843C8711F9	3EC6002A23 F02F4B8E07
$I_4 =$	FEF1DA04DD 22BC002126	BB64FAE4F1 B1C2366E8C
$I_5 =$	23BE7948E0 DCDC354A61	EF74E34F04 3FE7DD27C2
$I_6 =$	4A0FFC1797 307ED087EE	5B32C1EC2D D2E975E422
$I_7 =$	A505D9DE61 BA2061C9B8	71DA855D46 D2C41C0A0E

提案手法は単純な比較から構成されるため，スキャンチェインにおけるレジスタの接続順に解読コストは影響を受けない．スキャンチェイン長に対しては比較時間は比例する性質であり，スキャンチェイン長を n とすると O(n) となる．

計算機実験において，ランダムなビット値を加えることでスキャンチェイン長を増加させているが，実際の回路のレジスタ値には偏りがある．そのため，実験結果に示す入力ペア数・サイクル数では解読に十分でない場合がある．その場合，入力ペア数・サイクル数を増加させることで対応可能である．

提案手法は Trivium に限らず他のストリーム暗号に対しても内部状態を把握するために適用可能である．内部状態の復元については暗号アルゴリズム毎に求めることが必要である．

表 3.8: 入力ペア数 14 の時の入力ペアの値.

	秘密鍵 (80 ビット)	IV (80 ビット)
$I_1 =$	4CAADE4A78 F255AA3289	DD51F2604A 449AB1A17B
$I_2 =$	AC86295962 659C663192	1A4CE909AB 2BC2E11C28
$I_3 =$	2A460E9FB3 B52309294B	066FA69657 DE3DF40453
$I_4 =$	1D7C2C27A0 57DC47B1B8	AEF95F085F 87D7304671
$I_5 =$	8D0F62C503 63D552525E	FDC09A4EDD 7F3248904A
$I_6 =$	F7281C92B1 3FE8608989	D2B230E681 CEB26F9A76
$I_7 =$	9A2FBE9AF9 4285774A1F	A22CCEBCE1 D73D6856E0
$I_8 =$	C362E4DF79 F18F8424E7	20873E049E ED3DAC7079
$I_9 =$	62BAB5A4B1 3DD0F2FBD0	BA54CB6E44 812909A97D
$I_{10} =$	918ACF93AC CD5C980FA7	0F519B9242 A42118022B
$I_{11} =$	5B5B58D8B2 961429B84C	69E2328E6C CFBC1473FF
$I_{12} =$	C6B4213B9C C5A8CB60C6	C5BBE34C72 89D01E7F08
$I_{13} =$	B79A746580 878AF0CD44	61C85E65E5 04731C35DA
$I_{14} =$	2D3089A5CF 0719C045CE	6388282408 35714CF17C

3.5 本章のまとめ

本章では，ストリーム暗号 Trivium の性質とアルゴリズムを示し，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案した．

各節の内容を以下にまとめる．

第 3.2 節「ストリーム暗号 Trivium」では，ストリーム暗号 Trivium のアルゴリズムを説明した．ストリーム暗号では，ビット毎あるいはバイト毎にキーストリームと排他的論理和し暗号化・復号する．Trivium は同期式ストリーム暗号で，3 本のシフトレジスタから構成され，内部の演算はビット同士の AND 演算と XOR 演算のみであるため，構造が単純で高速に動作する．

第 3.3 節「Trivium に対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案した．提案手法は，1 ビットレジスタ値の入力・動作サイクル数に対する変化がそのレジスタ固有の値になることを利用した手法である．

第 3.4 節「評価実験」では，提案手法のソフトウェア実験の結果を示した．実験より，スキャンデータに他の回路のビットが含まれていても，提案手法はビット対応解析できることを確認した．また，暗号回路のみをスキャンチェインに含む場合，Trivium の内部状態レジスタ 288 個のビット対応解析には，特定の入力を設定してキーストリーム生成フェーズからサイクル毎にスキャンデータを 13 個取得すれば良く，解析時間は 0.139 秒で済み，最も効率的に求められることを確認した．

提案手法で，手間がかかるがビット対応解析できる条件を以下に示す．

- 暗号回路のアーキテクチャが不明
- 暗号化処理のタイミングが不明

暗号回路のアーキテクチャが不明の場合，考え得るアーキテクチャの全パターンについて手法を適用すれば，ビット対応が解析できる．暗号化処理のタイミングが不明の場合，大まかに予測した範囲でスキャンデータとスキャンシグネチャを比較すれば良い．スキャンデータの取得回数は 3.4 節で示した値より増加すると考えられるが，その場合でもビット対応解析可能と予想できる．

今後の課題として，以下を考えている．LSI アーキテクチャの構造によっては全内部レジスタの値を保持していない場合や Trivium の内部状態レジスタの一部のみがスキャンチェインに接続されている場合がある．提案手法でビット対応解析は可能であるが，そのままでは内部状態の復元ができない．全レジスタを把握しなくてもキーストリームが復元可能か今後考察予定である．

第4章　ブロック暗号へのスキャンベースサイドチャネル攻撃

4.1　本章の概要

本章では，LED 暗号の性質とアルゴリズムを示し，スキャンシグネチャを用いた LED 暗号へのスキャンベース攻撃手法を提案する．

以下に本章の構成を示す．

第 4.2 節「ブロック暗号 LED」では，LED 暗号のアルゴリズムを説明する．LED 暗号は 64 ビットブロック暗号で，秘密鍵長は 64 ビットから 128 ビットである．ブロック暗号の中でも最軽量であり，ハードウェアへ小面積で実装可能である．

第 4.3 節「LED に対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しない LED 暗号へのスキャンベース攻撃手法を提案する．提案手法は，特定の平文を入力した LSI から取得したスキャンデータを XOR 加算し，特定のビット列に着目することで秘密鍵を部分ごとに解読する手法である．

第 4.4 節「評価実験」では，提案手法のソフトウェアによる評価実験結果を示す．

第 4.5 節「本章のまとめ」では，本章の内容をまとめる．

本章は [14, 16, 41–43] で発表した内容から構成される．

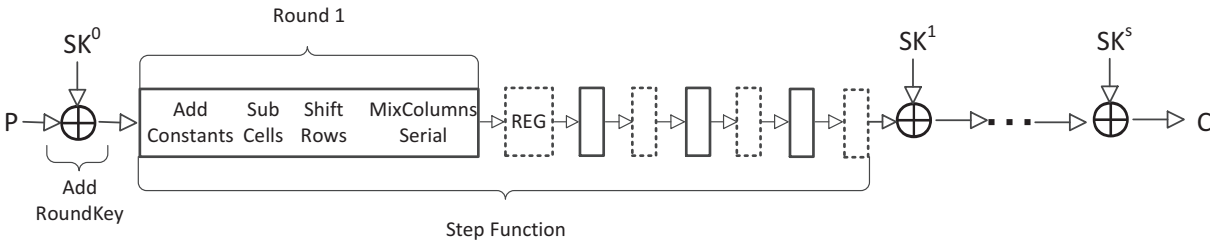


図 4.1: LED 暗号処理.

4.2 ブロック暗号 LED

低消費電力が望まれ，リソースに高い制約があるセンサ等の装置においては，軽量ブロック暗号が望まれる．軽量暗号は低コスト，低消費電力であるため，スマートカードや RFID タグ等にも用いられる．最軽量ブロック暗号の 1 つに LED 暗号 [17] がある．スキャンベース攻撃の既存研究は大半がブロック暗号 AES，DES を対象としているが，新しいブロック暗号に対しても研究の必要があるため，本章では，スキャンベース攻撃に対する耐性の研究事例がない軽量ブロック暗号 LED を攻撃対象としている．

本節では LED 暗号の概要とアルゴリズムを示す．LED (Light Encryption Device) 暗号は 2011 年に Guo らが提案した 64 ビットブロック暗号である．ブロック暗号の中でも最軽量でハードウェアへの実装面では小面積で済む点が利点である．秘密鍵長は 64 ビットから 128 ビットである．LED では秘密鍵長をハイフン以下に記す，つまり秘密鍵長 64 ビットの LED を LED-64，秘密鍵長 128 ビットの LED を LED-128 と表すこととする．LED 暗号は分割・転置等を実行するラウンド処理と鍵との加算を繰り返す．AES に似た構造であるが AES より小面積であり，AES-256 等の暗号への攻撃手法として効果的な関連鍵攻撃 [1, 11] に対し，耐性がある．

LED 暗号化処理

LED 暗号の演算処理単位は 4 ビットであるため，64 ビットのデータを 4 ビットで 1 要素とした 4×4 行列でデータを表現する．64 ビットの平文ブロックを $m_0 \parallel m_1 \parallel \ldots \parallel m_{15}$ とする時，以下のように表せる．

$$\begin{bmatrix} m_0 & m_1 & m_2 & m_3 \\ m_4 & m_5 & m_6 & m_7 \\ m_8 & m_9 & m_{10} & m_{11} \\ m_{12} & m_{13} & m_{14} & m_{15} \end{bmatrix}$$

LED 暗号ではラウンド処理を繰り返し実行する．4 ラウンドで 1 ステップとし，ステップ毎に 64 ビットの副鍵 SK^i と暗号処理データを排他的論理和する Add RoundKey を実行する．SK^0 は平文と排他的論理和され，SK^1 は 1 ステップ終了後のデータと排他的論理和され，SK^2 は 2 ステップ終了後のデータと排他的論理和される．暗号化処理するステップ数は秘密鍵長によって決定される．秘密鍵長が 64 ビットの時には 8 ステップ，65 ビットから 128 ビットの時には 12 ステップ暗号化処理を実行する．図 4.1 に概略を示す．図 4.1 において P は平文，C は暗号文を示す．

副鍵 SK^i は 64 ビットであり i 番目の副鍵 SK^i は以下のように表せる．

$$\begin{bmatrix} sk_0^i & sk_1^i & sk_2^i & sk_3^i \\ sk_4^i & sk_5^i & sk_6^i & sk_7^i \\ sk_8^i & sk_9^i & sk_{10}^i & sk_{11}^i \\ sk_{12}^i & sk_{13}^i & sk_{14}^i & sk_{15}^i \end{bmatrix}$$

ここで l ビットの秘密鍵 K の各ビットを $k_0, k_1, \ldots, k_{l-1}$ とすると，

$$sk_j^i = k_{j+i\times 16 mod\ l}$$

である．秘密鍵が 64 ビットの時，副鍵 $SK^i(0 \leq i \leq 8)$ は全て秘密鍵と等しくなる．秘密鍵が 128 ビットの時，副鍵 $SK^i(0 \leq i \leq 12)$ は秘密鍵の前半，後半の値と交互に等しくなる．

ラウンド処理

LED 暗号処理において繰り返し実行されるラウンド処理を説明する．図 4.2 に LED 暗号のラウンド処理を示す．ラウンド処理では，Add Constants, Sub Cells, Shift Rows, Mix Columns Serial を実行する．

- Add Constants ではラウンド定数を XOR 加算する．ラウンド定数 $AC_0, \ldots AC_{15}$ を以下に示す．

$$\begin{bmatrix} 0\oplus(ks_7 \parallel ks_6 \parallel ks_5 \parallel ks_4) & (rc_5 \parallel rc_4 \parallel rc_3) & 0 & 0 \\ 1\oplus(ks_7 \parallel ks_6 \parallel ks_5 \parallel ks_4) & (rc_2 \parallel rc_1 \parallel rc_0) & 0 & 0 \\ 2\oplus(ks_3 \parallel ks_2 \parallel ks_1 \parallel ks_0) & (rc_5 \parallel rc_4 \parallel rc_3) & 0 & 0 \\ 3\oplus(ks_3 \parallel ks_2 \parallel ks_1 \parallel ks_0) & (rc_2 \parallel rc_1 \parallel rc_0) & 0 & 0 \end{bmatrix}$$

  $(rc_5, rc_4, rc_3, rc_2, rc_1, rc_0)$ は最初に 0 に初期化される．ラウンド毎に左に 1 シフトし，rc_0 は $rc_5 \oplus rc_4 \oplus 1$ の値を設定することで更新する．$ks_7 ks_6 \ldots ks_0$ は秘密鍵長を 8 ビットで表した値である．

表 4.1: Sbox.

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S [x]	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

- Sub Cells ではSboxで換字処理を行う．表4.1にSboxの動作を16進数で示す．
- Shift Row では行列の $i(0 \leq i \leq 3)$ 行目を左に i シフトする．
- Mix Columns Serial では行列 M と列ごとに乗算する．以下に行列 M を示す．

$$M = \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix}$$

図4.3にLED暗号のハードウェアアーキテクチャの例を示す．State, Key State, MCS, AK, AC, SC, Controllerの7つのモジュールから構成されている．Stateでは4ビットのフリップフロップが 4×4 行列を構成しており，各行はフィードバックシフトレジスタを形成している．Shift Rows と Add Constant の1列目のXOR加算を実行する．Key Stateでは秘密鍵の値を保持する．MCSではMix Cplmuns Serialを列ごとに順に実行し結果をStateに格納する．AKではAdd RoundKeyを実行する．ACではAdd Constantの0列目のXOR加算を実行する．SCではSub Cellsを実行する．Controllerは各モジュールを制御する．

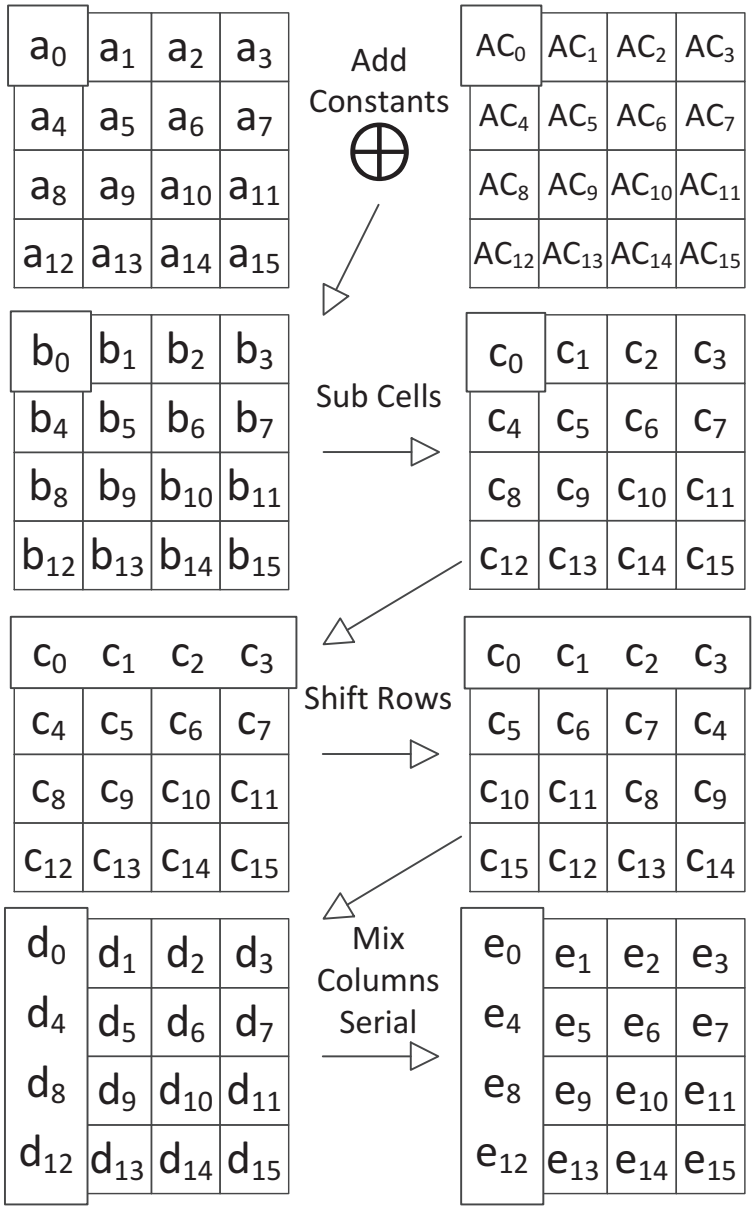


図 4.2: LED 暗号のラウンド処理.

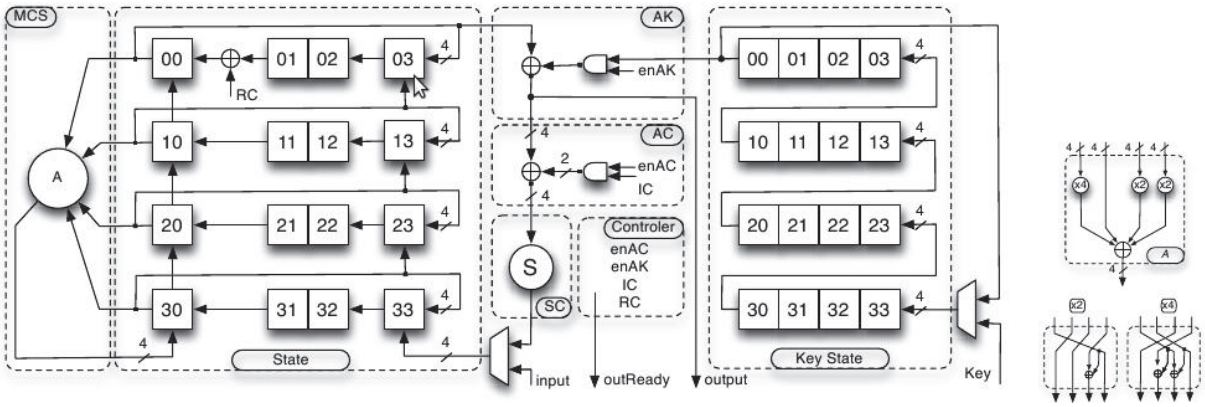


図 4.3: LED 暗号のハードウェアアーキテクチャ[17].

4.3 LED に対するスキャンベース攻撃手法

本節では，LED 暗号に対するスキャンベース攻撃手法を提案する．まず，秘密鍵長 64 ビットの LED 暗号を想定する．攻撃対象の LSI のスキャンチェインは，Mix Columns Serial 後のデータを格納するレジスタを含むものとする．秘密鍵の値を保持するレジスタは含まないものとする．

前提条件

LED 暗号へのスキャンベース攻撃において，攻撃者が出来ることを以下に示す．

- 暗号化アルゴリズムを知っている
- 暗号回路で任意の平文を暗号化できる
- 任意のタイミングでスキャンチェインへアクセスできる

攻撃者は任意の平文を暗号 LSI に入力し，任意のタイミングでスキャンデータを取得できる．

攻撃者が分からないことを以下に示す．

- スキャンチェインのレジスタ接続順
- スキャンチェインに含まれるレジスタの数や種類

スキャンチェインは総配線長が短くなるように接続されるため，レジスタの接続順は攻撃者には分からない．そのため，暗号回路のレジスタの値がスキャンデータ上のどのビット位置にあるか攻撃者には不明である．また，スキャンチェインには通常，LSI チップ上の他の周辺回路のレジスタも含まれており，スキャンチェインで接続されているレジスタの種類や数は攻撃者には分からない．スキャンチェインに含まれるレジスタの接続順，種類が分からない場合でも秘密鍵を解読できる手法を提案する．

スキャンシグネチャを用いた解析

多数の平文を想定し，それぞれを攻撃対象 LSI に設定し，暗号化処理中にそれぞれ同じタイミングでスキャンデータを取得したとする．図 4.4 下のように用いた平文毎に取得したスキャンデータを縦に並べる．スキャンチェイン長を k ビット，入力した平文数を n 個とすると，横に k 個，縦に n 個のビットが並ぶことになる．

これらを n ビットの列データが k 個横に並んでいるものとして見ると，それぞれの列データは攻撃対象 LSI 中のある 1 ビットレジスタの平文に対する値の変化を表していることに気付く．攻撃対象 LSI に入力する平文数 n が大きいとき，この n ビットの列データの値はその 1 ビットレジスタ固有の値になる．これをスキャンシグネチャという．

次に，攻撃対象 LSI に入力した平文 n 個について，それぞれ暗号化シミュレータで暗号化処理を行い，スキャンデータを取得したタイミングと同じ時点のレジスタ値を求める．但し，秘密鍵のとりうる全ての値について想定し，それぞれシミュレータでレジスタ値を計算するものとする．これらシミュレータで求めた値を平文毎に縦に並べる．そして，秘密鍵を K_1 と想定し，n 個の平文を暗号シミュレータで暗号化したときに得られたレジスタ r のスキャンシグネチャが LSI から取得したスキャンデータ中に存在するかを探索する (図 4.4)．秘密鍵を K_1 と想定したときのレジスタ r のスキャンシグネチャがスキャンデータに存在すれば，秘密鍵の値は K_1 と考えることができる．レジスタ r のスキャンシグネチャがスキャンデータに存在しなければ，予想した秘密鍵の値は誤っていることが分かる．

ところがここで大きな問題が起こる．秘密鍵長が 64 ビットの LED 暗号では副鍵 SK^0 を解読できれば直ちに秘密鍵 K を解読可能であるが，副鍵 SK^0 は 64 ビットであるため，その候補は 2^{64} 個となり，これらを総当たりで試行し，スキャンデータと比較することは事実上不可能である．副鍵 SK^0 の 64 ビットのデータ全部でなく，その一部のみが影響を与えるように，データを加工する必要がある．

副鍵の影響の削減

副鍵 SK^0 の 64 ビットのデータ全部でなく，その一部のみが影響を与えるように，データを加工することを考える．

ラウンド処理では，Add Constants, Sub Cells, Shift Rows, Mix Columns Serial を順に実行し，レジスタに値が格納される．図 4.2 において，a_0 の値は e_0, e_4, e_8, e_{12} の値に影響を及ぼし，他の e_i の値は a_0 の値とは独立である．同様に，$a_i (1 \leq i \leq 15)$ の値に Mix Columns Serial 後の 4 つの要素の値がそれぞれ依存しており，残りの 12 の要素は a_i の値とは独立になっている．

また，図 4.2 において，e_0 の値は a_0, a_5, a_{10}, a_{15} に依存している．同様に，$e_i (1 \leq i \leq 15)$ の値は Add Constants 実行前の 4 つの要素の値にそれぞれ依存している．

ここで，先頭の 4 ビット a_0 と a'_0 のみが異なり，他の要素は同じ 64 ビットの 2 つの数値 a, a' を用意する．これらに対し，それぞれ Add Constants, Sub Cells, Shift Rows, Mix Columns Serial を順に実行して e, e' を計算する．求めた e, e' の排他的

論理和をとって $e \oplus e'$ を求める．e と e' は 0 列目のみが異なり，他の要素は同じ値になるため，$e \oplus e'$ は 0 列目以外は 0 になる．$e \oplus e'$ の 0 列目は以下のようになる．

$$\begin{aligned}\begin{bmatrix} e_0 \oplus e'_0 \\ e_4 \oplus e'_4 \\ e_8 \oplus e'_8 \\ e_{12} \oplus e'_{12} \end{bmatrix} &= \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix} \begin{bmatrix} d_0 \oplus d'_0 \\ d_4 \oplus d'_4 \\ d_8 \oplus d'_8 \\ d_{12} \oplus d'_{12} \end{bmatrix} \\ &= \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix} \begin{bmatrix} d_0 \oplus d'_0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{4.1}$$

式 4.1 より，$e \oplus e'$ の 0 列目は d_0, d'_0 を用いて表せることが分かる．d_0, d'_0 は a_0, a'_0 に依存する．a_0, a'_0 は 1 つ前のラウンド処理で実行された Mix Columns Serial の出力値の 0 番目の要素，または直前に実行された Add RoundKey の出力値の 0 番目の要素である．

さてここで，1 ラウンド目を考えよう．つまりこれら 64 ビットの数値 a, b, c, d, e と a', b', c', d', e' が 1 ラウンド目の値であるとする．64 ビットの 2 つの平文をそれぞれ m，m' とすると a_0, a'_0 は，

$$\begin{aligned} a_0 &= m_0 \oplus SK^0_0 \\ a'_0 &= m'_0 \oplus SK^0_0 \end{aligned}$$

である．ここで SK^0_0 は 64 ビットの副鍵 SK^0 の上位 4 ビット，つまり第一要素のみを表している．よって $e \oplus e'$ の 0 列目は平文 m_0, m'_0 と副鍵 SK^0 の第一要素 SK^0_0 のみに依存する値になっていることが分かる．つまり，この値は副鍵 SK^0 全体 64 ビットに依存する値でなくその第一要素 SK^0_0 の 4 ビットにのみ依存する値であり，この値は SK^0_0 の全数探索によって，十分現実的にそのすべての値を試行することができる．

そして，0 番目の要素のみ値が異なる 2 つの平文 m, m' を攻撃対象の LSI に設定し，1 ラウンド目の Mix Columns Serial 後の値をスキャンデータとしてそれぞれ取得する．取得したスキャンデータを排他的論理和したデータ中で $e \oplus e'$ の 0 列目に当たる部分は，平文 m_0, m'_0 と副鍵 SK^0_0 のみに依存する．つまり，排他的論理和したデータ中で $e_0 \oplus e'_0$, $e_4 \oplus e'_4$, $e_8 \oplus e'_8$, $e_{12} \oplus e'_{12}$ の 16 個のビットは副鍵 SK^0_0 に依存しており，他の副鍵の要素 $SK^0_i (i \neq 0)$ とは独立である．

全て 0 の平文 $m^{(0)}$ と 0 番目の要素のみ値が異なる多数の平文 $m^{(1)}, \ldots, m^{(n)}$ をそれぞれ攻撃対象の LED 暗号 LSI に入力し，1 ラウンド目の Mix Columns Serial 後

の値をスキャンデータとしてそれぞれ取得し，$sd^{(0)}, \ldots, sd^{(n)}$ とする．次に

$$\begin{gathered} sd^{(0)} \oplus sd^{(1)} \\ sd^{(0)} \oplus sd^{(2)} \\ \vdots \\ sd^{(0)} \oplus sd^{(n)}. \end{gathered}$$

を求める．これらのデータ中で

$$\begin{array}{cccc} e_0^{(0)} \oplus e_0^{(1)}, & e_4^{(0)} \oplus e_4^{(1)}, & e_8^{(0)} \oplus e_8^{(1)}, & e_{12}^{(0)} \oplus e_{12}^{(1)} \\ e_0^{(0)} \oplus e_0^{(2)}, & e_4^{(0)} \oplus e_4^{(2)}, & e_8^{(0)} \oplus e_8^{(2)}, & e_{12}^{(0)} \oplus e_{12}^{(2)} \\ \vdots & \vdots & \vdots & \vdots \\ e_0^{(0)} \oplus e_0^{(n)}, & e_4^{(0)} \oplus e_4^{(n)}, & e_8^{(0)} \oplus e_8^{(n)}, & e_{12}^{(0)} \oplus e_{12}^{(n)} \end{array}$$

は副鍵 SK_0^0 に依存している．これら 16 個の列データをスキャンシグネチャ SS_0 とする．

スキャンシグネチャ SS_0 は副鍵の要素の中で SK_0^0 のみに依存している．4.3 節より，副鍵 SK_0^0 の全パターンについてシミュレータでスキャンシグネチャ SS_0 を求め，スキャンデータを排他的論理和したデータ中にこのスキャンシグネチャ SS_0 があるかを探索する．スキャンシグネチャ SS_0 が存在すれば予想した SK_0^0 の値が正しいと分かる．同様に，$SK_0^0, \ldots SK_{15}^0$ を解読できれば，秘密鍵 K を解読できる．

副鍵を 1 要素ずつ解読する手法のアルゴリズム

SK_0^0 を解読する手法をまとめる．

1. **攻撃に使用する平文を生成し，スキャンデータを取得**
   全て 0 の平文 $m^{(0)}$ と，0 番目の要素をランダムに生成し，0 番目の要素 (m_0) 以外はすべて 0 である n 個の平文を生成する．この平文群を $m^{(0)}, \ldots, m^{(n)}$ とする．平文群を攻撃対象の LED 暗号 LSI に入力し，1 回目のラウンド処理を終えた時点のスキャンデータ $sd^{(0)}, \ldots, sd^{(n)}$ を取得する．

2. **スキャンデータの XOR 演算**
   $sd^{(0)} \oplus sd^{(1)}, sd^{(0)} \oplus sd^{(2)}, \ldots, sd^{(0)} \oplus sd^{(n)}$ を計算し，解析データとする．

3. **副鍵を予想してシミュレーション**
   副鍵 SK^0 の第一要素 SK_0^0 について $2^4 = 16$ 通りの値をそれぞれ想定し，平文群 $m^{(0)}, \ldots, m^{(n)}$ を入力としたときの $e^{(0)}, \ldots, e^{(n)}$ を暗号シミュレータから求める．

4. **シミュレーション値の XOR 演算**
   $e^{(0)} \oplus e^{(1)}, e^{(0)} \oplus e^{(2)}, \ldots, e^{(0)} \oplus e^{(n)}$ を計算し，比較データとする．

5. **スキャンシグネチャを利用した比較**
   解析データの列データと比較データを比較し，比較データ中の 16 個の列データ，つまりスキャンシグネチャ SS_0 が解析データの列データ中に存在するとき，予想した SK_0^0 が正しい値であることが分かる (図 4.5)．

同様に $SK_1^0, \ldots, SK_{15}^0$ を求めれば，秘密鍵全体が解読できる．

スキャンチェイン長の増加

上記の手法はスキャンチェイン長が増加した場合，以下の問題が生じる．副鍵の要素を 1 つ求めるために用いることのできる平文数は，LED 暗号が 1 要素 4bit であるため，最大で 16 個である．生成した平文のラウンド 1 処理後の値を XOR 加算して比較データを作るため，副鍵の要素を 1 つ求めるために用いる比較データの行数は最大で 15 になる．よって，比較データの列の値 (スキャンシグネチャのパターン) は最大 $2^{15} = 32768$ 通りになる．スキャンチェイン長が 32768bit 以上の時，誤って副鍵を予想して算出した比較データの列データが解析データ中に存在する可能性は高くなる．実際には，スキャンチェインに含まれるレジスタ値に偏りがあるため，これより小さいスキャンチェイン長でもこの問題は生じると考えられる．

Add Constants 実行前と Mix Columns Serial 後の値に依存関係がある要素の位置に着目し，副鍵 SK^0 を同時に 2 要素ずつ順に解読することで，上記の問題を解決できる．

副鍵の要素を求めるために用いることが出来る平文数を増加させることを考える．既に示した通り，平文のある 4 つの要素は 1 ラウンド目の Mix Columns Serial 後の値の特定の 4 要素に影響を与えている．ここで，0 番目と 5 番目の要素のみ値が異なり他の値は全て等しい 64 ビットの平文 m, m' ($a_0 \neq a'_0$, $a_5 \neq a'_5$, $a_i = a'_i$ for $i = 1, \cdots, 4, 6, \cdots, 15$) を用意する．これらに対し，それぞれ Add RoundKey, Add Constants, Sub Cells, Shift Rows, Mix Columns Serial を順に実行して e, e' を計算する．求めた e, e' の排他的論理和をとって $e \oplus e'$ を求める．e と e' は 0 列目のみが異なり，他の要素は同じ値になるため，$e \oplus e'$ は 0 列目以外は 0 になる．$e \oplus e'$ の 0

列目は以下のようになる．

$$\begin{aligned}\begin{bmatrix} e_0 \oplus e'_0 \\ e_4 \oplus e'_4 \\ e_8 \oplus e'_8 \\ e_{12} \oplus e'_{12} \end{bmatrix} &= \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix} \begin{bmatrix} d_0 \oplus d'_0 \\ d_4 \oplus d'_4 \\ d_8 \oplus d'_8 \\ d_{12} \oplus d'_{12} \end{bmatrix} \\ &= \begin{bmatrix} 4 & 1 & 2 & 2 \\ 8 & 6 & 5 & 6 \\ B & E & A & 9 \\ 2 & 2 & F & B \end{bmatrix} \begin{bmatrix} d_0 \oplus d'_0 \\ d_4 \oplus d'_4 \\ 0 \\ 0 \end{bmatrix} \end{aligned} \tag{4.2}$$

式4.2より，$e \oplus e'$ の0列目は d_0, d'_0, d_4, d'_4 を用いて表せることが分かる．これらは平文 m_0, m_5, m'_0, m'_5 と sk^0_0, sk^0_5 に依存する．よって，$e \oplus e'$ の0列目の値は sk^0_0, sk^0_5 のみに依存し，他の副鍵の要素とは独立であるから，$e \oplus e'$ の0列目のスキャンシグネチャを用いて sk^0_0, sk^0_5 の値を同時に解読すればよい．

副鍵を2要素ずつ解読する手法のアルゴリズム

以下に sk^0_0, sk^0_5 を解読する手法をまとめる．

1. **攻撃に使用する平文を生成し，スキャンデータを取得**
   全て0の平文 $m^{(0)}$ と，0番目の要素と5番目の要素をランダムに生成し，0番目の要素 (m_0) と5番目の要素 (m_5) 以外はすべて0である n 個の平文を生成する．この平文群を $m^{(0)}, \ldots, m^{(n)}$ とする．平文群を攻撃対象のLED暗号LSIに入力し，1回目のラウンド処理を終えた時点のスキャンデータ $sd^{(0)}, \ldots, sd^{(n)}$ を取得する．

2. **スキャンデータのXOR演算**
   $sd^{(0)} \oplus sd^{(1)}, sd^{(0)} \oplus sd^{(2)}, \ldots, sd^{(0)} \oplus sd^{(n)}$ を計算し，解析データとする．

3. **副鍵を予想してシミュレーション**
   sk^0_0 と sk^0_5 について $2^8 = 256$ 通りの値をそれぞれ想定し，平文群 $m^{(0)}, \ldots, m^{(n)}$ を入力としたときの $e^{(0)}, \ldots, e^{(n)}$ を暗号シミュレータから求める．

4. **シミュレーション値のXOR演算**
   $e^{(0)} \oplus e^{(1)}, e^{(0)} \oplus e^{(2)}, \ldots, e^{(0)} \oplus e^{(n)}$ を計算し，比較データとする．

5. **スキャンシグネチャを利用した比較**
   解析データの列データと比較データを比較し，比較データ中の16個の列デー

タ，つまりスキャンシグネチャが解析データの列データ中に存在するとき，予想した sk_0^0, sk_5^0 が正しい値であることが分かる (図 4.6).

同様に SK^0 の残りの要素を 2 要素ずつ順に求めれば，秘密鍵全体が解読できる．

この手法では，副鍵 2 要素を同時に求める．そのため，使用できる平文数は最大で $2^8 = 256$ 個である．比較データの行数は最大で 255 になり，比較データの列の値 (スキャンシグネチャのパターン) は最大 2^{255} 通りになる．スキャンチェイン長が増加しても十分に対応可能なことが分かる．

他の秘密鍵長の時の解読

これまで秘密鍵長が 64 ビットの時の秘密鍵解読手法を示したが，秘密鍵長が 64 ビットより大きい場合も提案手法で解読可能である．最初の Add RoundKey で用いられる副鍵 SK^0 は秘密鍵の上位 64 ビットである．提案手法を用いて SK^0 を求めれば，秘密鍵の上位 64 ビットの値が判明する．秘密鍵の残りの部分は，SK^1 を求めることで判明する．

式 4.1 より，$e \oplus e'$ の 0 列目は a_0, a_0' に依存することを示した．ここでこれらが 5 ラウンド目の値であるとすると，$e \oplus e'$ の 0 列目は 4 ラウンド目の出力値の 0 番目の要素，副鍵 SK^1 の第一要素 sk_0^1 のみに依存する値になっている．つまり，この値は副鍵 SK^1 全体 64 ビットに依存する値でなくその第一要素 sk_0^1 の 4 ビットにのみ依存する値であり，この値は sk_0^1 の全数探索によって，十分現実的にそのすべての値を試行することができる．

ここで，4 ラウンド目の出力値は 0 番目の要素のみ異なる値に設定する必要がある．求めた SK^0 を用いて，そのような値を与える平文を計算すればよい．

sk_0^1 を解読する手法をまとめる．

1. **攻撃に使用する平文を生成し，スキャンデータを取得**
   4 ラウンド目の出力値をそれぞれ，全て 0 にする平文，0 番目の要素のみ異なり，0 番目の要素以外全て 0 にする平文，n 個を SK^0 を用いて逆演算し求める．この平文群を $m^{(0)}, \ldots, m^{(n)}$ とする．平文群を攻撃対象の LED 暗号 LSI に入力し，5 回目のラウンド処理を終えた時点のスキャンデータ $sd^{(0)}, \ldots, sd^{(n)}$ を取得する．

2. **スキャンデータの XOR 演算**
   $sd^{(0)} \oplus sd^{(1)}, sd^{(0)} \oplus sd^{(2)}, \ldots, sd^{(0)} \oplus sd^{(n)}$ を計算し，解析データとする．

3. **副鍵を予想してシミュレーション**
   副鍵 SK^1 の第一要素 sk_0^1 について $2^4 = 16$ 通りの値をそれぞれ想定し，平文

群 $m^{(0)}, \ldots, m^{(n)}$ を入力としたときの $e^{(0)}, \ldots, e^{(n)}$ を暗号シミュレータから求める．

4. **シミュレーション値の XOR 演算**
   $e^{(0)} \oplus e^{(1)}, e^{(0)} \oplus e^{(2)}, \ldots, e^{(0)} \oplus e^{(n)}$ を計算し，比較データとする．

5. **スキャンシグネチャを利用した比較**
   解析データの列データと比較データを比較し，比較データ中の 16 個の列データ，つまりスキャンシグネチャ SS_0 が解析データの列データ中に存在するとき，予想した sk_0^1 が正しい値であることが分かる．

同様に $sk_1^1, \ldots, sk_{15}^1$ を求めれば，副鍵 SK^1 全体が解読できる．

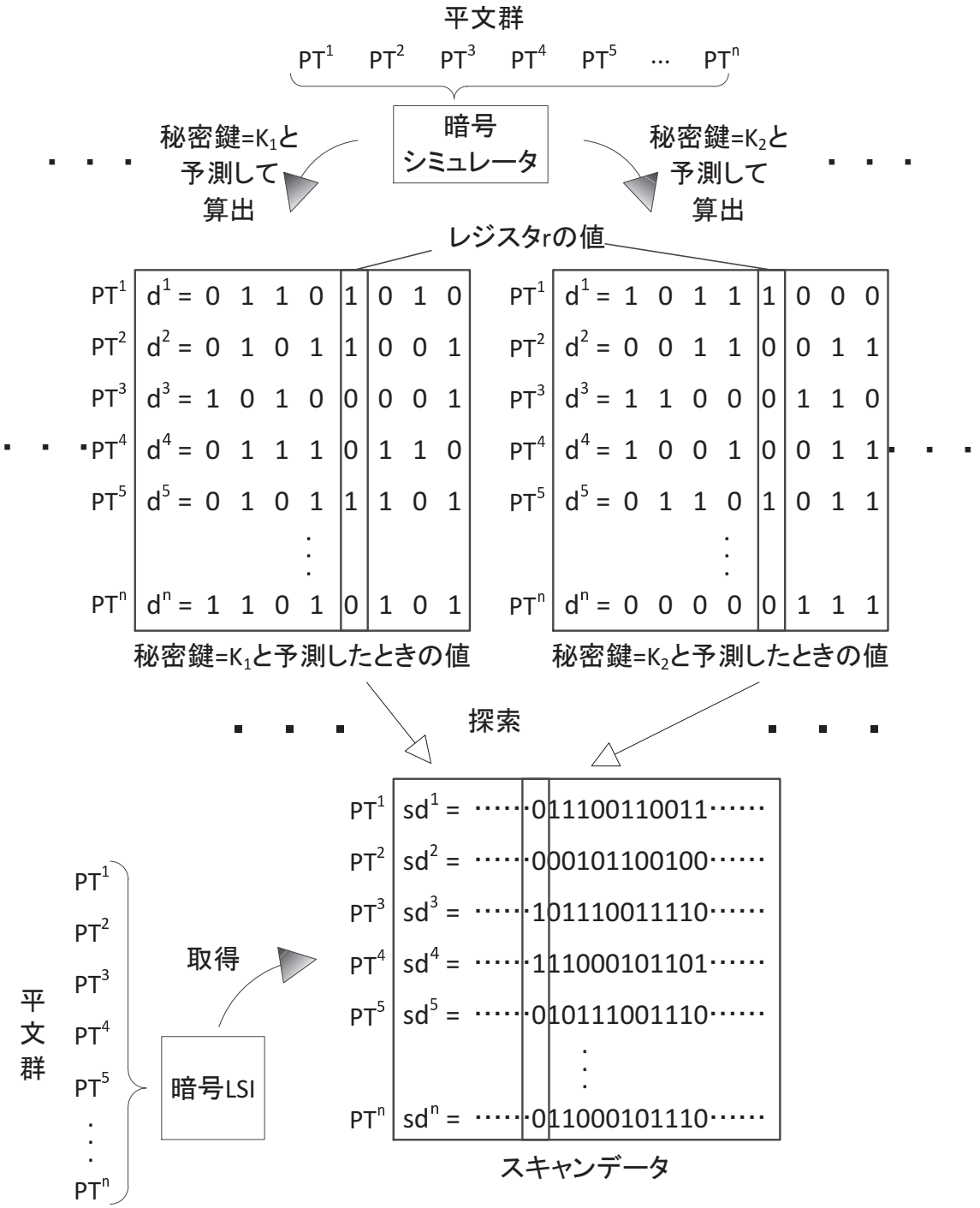


図 4.4: スキャンシグネチャと秘密鍵の予想.

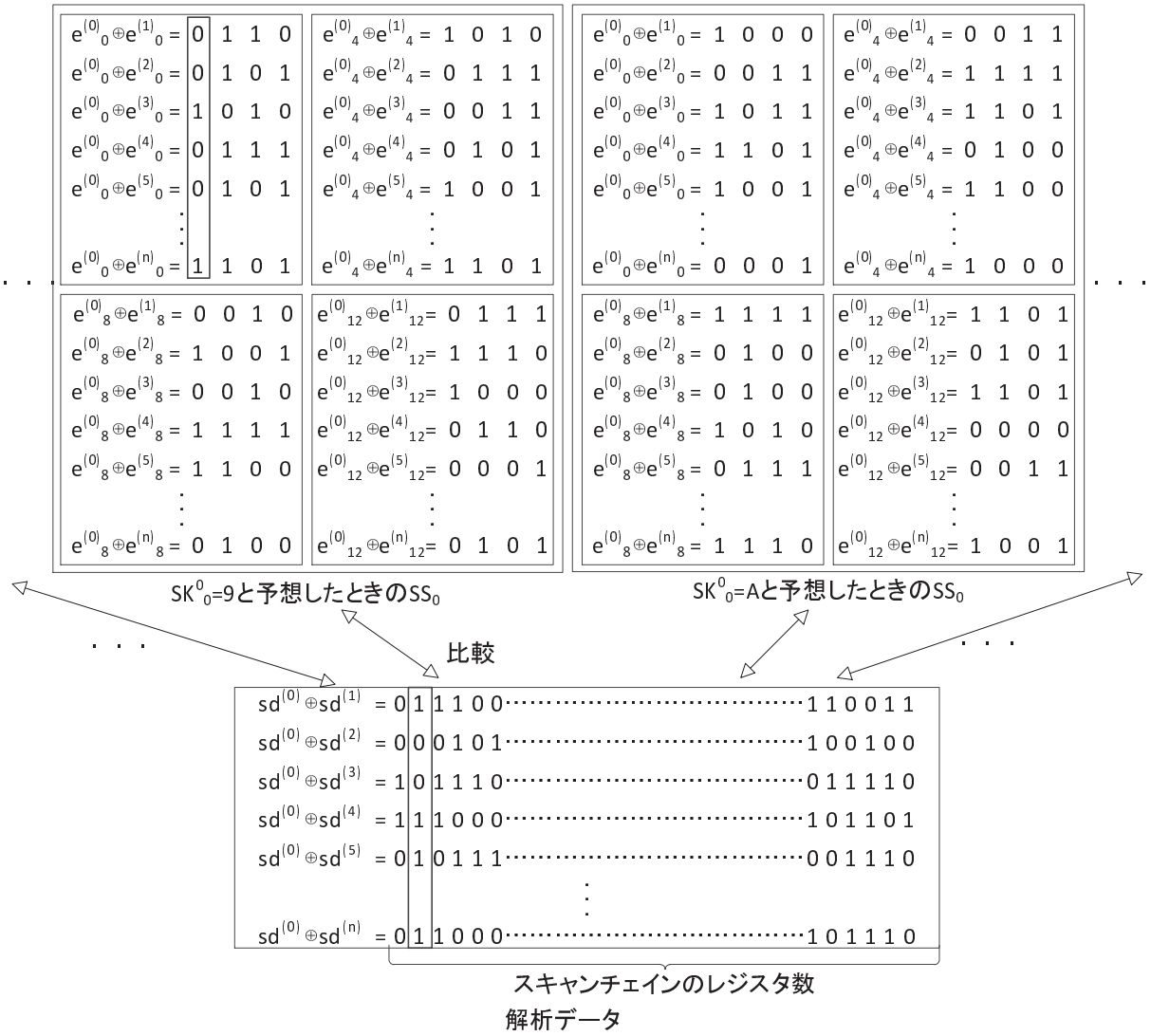

SK⁰₀=9と予想したときのSS₀
SK⁰₀=Aと予想したときのSS₀
比較
スキャンチェインのレジスタ数
解析データ

図 4.5: SK_0^0 の解読.

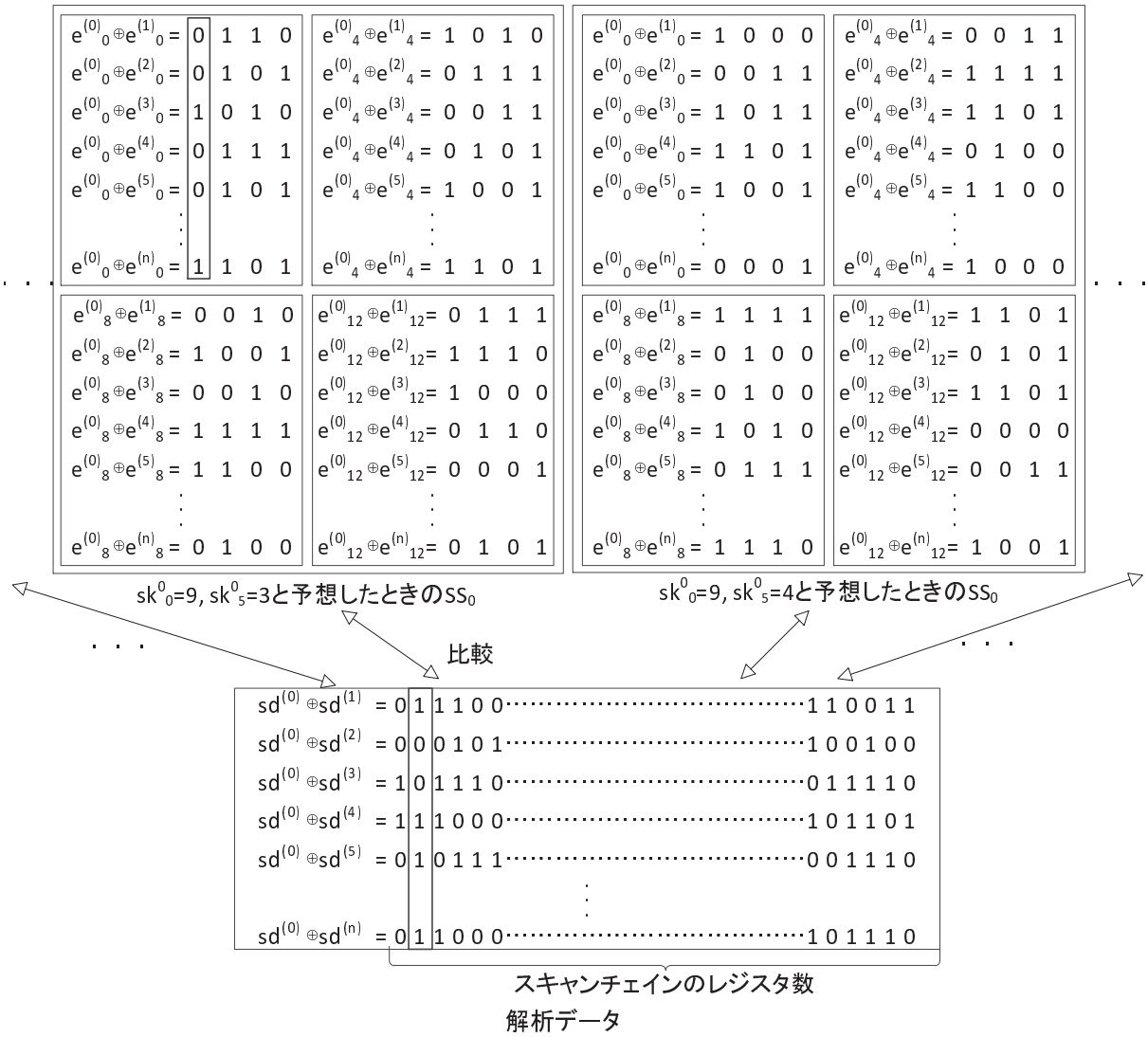


図 4.6: SK^0_0, SK^0_5 の解読.

4.4　評価実験

本節では，提案手法のソフトウェアシミュレーション結果を説明する．実験では，提案手法 (副鍵を 1 要素ずつ解読する手法と副鍵を 2 要素ずつ解読する手法) を C 言語で実装し，暗号回路のシミュレータは [17] のコードを使用した．1 回目と 5 回目のラウンド処理後のレジスタ値をシミュレータにより取得し，スキャンデータと想定する．スキャンデータと想定したデータに対し，提案手法 (副鍵を 1 要素ずつ解読する手法と副鍵を 2 要素ずつ解読する手法) のシミュレータを実行して秘密鍵を解読する．ランダムに生成した 64 ビットの秘密鍵 100 個の復元に必要な平均平文数，最悪平文数，平均解析時間，最悪解析時間を計測した．本実験は，CPU が Intel(R) Core(TM) i7-2620M 2.70GHz × 4，メモリが 8GB の計算機を用い，コンパイラは gcc を使用した．

実験方法

1. 副鍵を 1 要素ずつ解読する手法

本実験では，暗号回路のレジスタ 64 個のみがスキャンチェインに接続されている場合と暗号回路以外のレジスタがスキャンチェインに接続されている場合の 2 つを想定した．スキャンデータと想定するデータにランダムなビット値を加えることでスキャンチェイン長を変化させた．今回は，スキャンチェイン長が 256 ビット，1024 ビット，4096 ビットになるようにランダムなビット値を加え，秘密鍵の解読を行った．

2. 副鍵を 2 要素ずつ解読する手法

上記の実験と同様に，スキャンチェイン長を 256 ビット，1024 ビット，4096 ビットに増加し秘密鍵の解読を行った．また，1 要素ずつ解読する手法の解読可能なスキャンチェイン長は大よそ 2^{15} ビット未満と考えられるため，スキャンチェイン長を $2^{15} = 32768$ ビット，$2^{16} = 65536$ ビットに増加させ，ランダムに生成した 64 ビットの秘密鍵 10 個の復元に必要な平均平文数，最悪平文数，平均解析時間，最悪解析時間を計測した．また，スキャンチェイン長が 10 万ビット以上の場合でも 2 要素ずつ解読する手法で現実的な時間内に解読可能であるかを確認するため，スキャンチェイン長を $2^{17} = 131072$ ビットになるようにランダム値を加え 1 つの 64 ビット秘密鍵を解読した．

3. 128 ビットの秘密鍵の解読

本実験では，暗号回路のレジスタ 64 個のみがスキャンチェインに接続されている場合を想定し，1 要素ずつ解読する手法を用いて，ランダムに生成した 128 ビッ

表 4.2: 副鍵を 1 要素ずつ解読する手法による秘密鍵解読結果.

スキャンチェイン長	追加ビット数	平均平文数	最悪平文数	平均解析時間 [s]	最悪解析時間 [s]
64	0	72.48	79	0.290	0.376
256	192	122.74	128	1.125	1.293
1024	960	159.34	165	5.534	5.788
4096	4032	192.41	197	25.125	25.802

トの秘密鍵 100 個の復元に必要な平均平文数，最悪平文数，平均解析時間，最悪解析時間を計測した.

評価実験結果

1. 副鍵を 1 要素ずつ解読する手法

表 4.2 にスキャンチェイン長を変化させたときの秘密鍵解読に必要な平文数の平均値と最悪値，解析時間を示す．スキャンチェイン長を 4 倍にすると平均必要平文数が約 32 個増えている．これは，スキャンチェイン長が 4 倍になると解析データの列データ数が 4 倍になる．今回の実験ではランダムな値を付加してスキャンチェイン長を増加させているため，副鍵の 1 要素当たりの解読に必要な平文数が 2 個増える．全体で 16 要素あるため，$2 \times 16 = 32$ 個増加するためということが分かる.† また，スキャンチェイン長が 4 倍になると解析時間も約 4 倍になっていることが確認できる．探索する解析データの列データ数が 4 倍になっているためである.

2. 副鍵を 2 要素ずつ解読する手法

表 4.3 にスキャンチェイン長を変化させたときの秘密鍵解読に必要な平文数の平均値と最悪値，解析時間を示す．スキャンチェイン長を 4 倍にすると平均必要平文数が約 16 個増えている．これは，スキャンチェイン長が 4 倍になると解析データの列データ数が 4 倍になる．今回の実験ではランダムな値を付加してスキャンチェイン長を増加させているため，一度に解読する要素数を 1 ブロックとすると，1 ブロック (2 要素) 当たりの解読に必要な平文数が 2 個増える．全体で 8 ブロックあるため，$2 \times 8 = 16$ 個増加するためということが分かる.†また，スキャンチェイン長

†この平文数の増分の理論値は，スキャンデータがランダムな値のみで構成される場合の理論値である．スキャンチェイン長が 64 ビットや 128 ビットの時のスキャンデータは主に暗号回路のビットで構成されており，ランダムな値ではない．そのため平文数の増分の理論値とは必ずしも一致しない．スキャンチェイン長が増大するにつれ理論値に近くなる.

表 4.3: 副鍵を 2 要素ずつ解読する手法による秘密鍵解読結果.

スキャン チェイン長	追加 ビット数	平均 平文数	最悪 平文数	平均解析 時間 [s]	最悪解析 時間 [s]
64	0	35.94	40	1.846	2.402
256	192	64.23	65	8.983	9.187
1024	960	80.83	83	43.970	45.288
4096	4032	97.30	100	205.318	214.175

表 4.4: スキャンチェイン長が増大した時の解読結果.

スキャン チェイン長	追加 ビット数	平均 平文数	最悪 平文数	平均解析 時間 [s]	最悪解析 時間 [s]
$2^{15} = 32768$	32704	121.3	124	2085.098	2169.678
$2^{16} = 65536$	65472	129.2	131	4509.054	4574.105

が 4 倍になると解析時間も約 4 倍になっていることが確認できる．探索する解析データの列データ数が 4 倍になっているためである．

図 4.7 に 1 要素ずつ解読する手法と 2 要素ずつ解読する手法の平文数を示す．1 要素ずつ解読する手法と 2 要素ずつ解読する手法では平文数は約半分になっている．1 要素ずつ解読する手法では解読するブロック数は 16 であるが，2 要素ずつ解読する手法では 8 ブロック，つまり半分になるためである．2 要素ずつ解読する手法では，解読するブロック数は半分になるが，副鍵のブロックの予想が 2^4 通りから 2^8 通りに 16 倍に増加するため，解析時間は 1 要素ずつ解読する手法の $16/2 = 8$ 倍になることが分かる．

表 4.4 にスキャンチェイン長が $2^{15} = 32768$ ビット，$2^{16} = 65536$ ビットの時の結果を示す．2 要素ずつ解読する手法で解読可能なことを確認した．また，スキャンチェイン長が $2^{17} = 131072$ ビットの時，平文数は 137 個，解析時間は 9176.964 秒となった．スキャンチェイン長が 13 万ビットの場合も 2 時間半程度で解読可能なことが確認できた．解析時間は 1 要素ずつ解読する手法と比較して $16/2 = 8$ 倍になるが，十分に現実的な時間で解読可能なことを確認した．

3. 128 ビットの秘密鍵の解読

表 4.5 に，128 ビットの秘密鍵 100 個の復元に必要な平均平文数，最悪平文数，平均解析時間，最悪解析時間を示す．提案手法で副鍵 SK^0，SK^1 を順に解読できることを確認した．

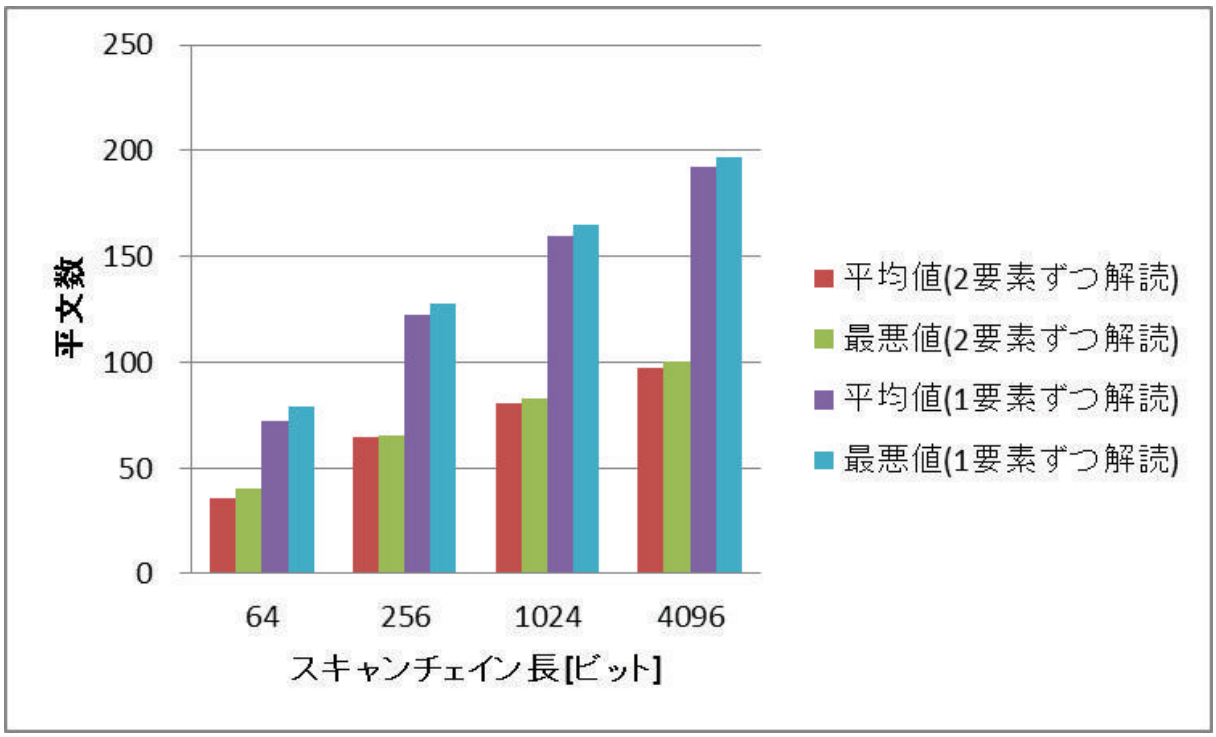


図 4.7: 秘密鍵解読に必要な平文数の比較.

表 4.5: 128 ビット秘密鍵解読結果.

	平均 平文数	最悪 平文数	平均解析 時間 [s]	最悪解析 時間 [s]
SK^0 の解読	71.98	79	0.234	0.268
SK^1 の解読	72.15	78	0.234	0.268

提案手法の性質

提案手法は単純な比較から構成されるため，解読コストはスキャンチェインにおけるレジスタの接続順に影響を受けない，スキャンチェイン長に対しては比例するといった性質を持つ．スキャンチェイン長 n の場合，解読時間は O(n) となる．

本実験において，ランダムなビット値を加えることでスキャンチェイン長を増加させている．しかし実際の回路のレジスタ値には偏りがあり，実験結果に示す入力ペア数・サイクル数では解読に十分でない場合も想定される．その場合，入力ペア数・サイクル数を増加させることで解読可能である．

4.5　本章のまとめ

本章では，LED 暗号の性質とアルゴリズムを示し，スキャンシグネチャを用いた LED 暗号へのスキャンベース攻撃手法を提案した．

各節の内容を以下にまとめる．

第 4.2 節「LED 暗号」では，LED 暗号のアルゴリズムを説明した．LED 暗号は 64 ビットブロック暗号で，秘密鍵長は 64 ビットから 128 ビットである．ブロック暗号の中でも最軽量であり，ハードウェアへ小面積で実装可能である．

第 4.3 節「LED 暗号に対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しない LED 暗号へのスキャンベース攻撃手法を提案した．提案手法は，特定の平文を入力した LSI から取得したスキャンデータを XOR 加算し，特定のビット列に着目することで秘密鍵を部分ごとに解読する手法である．

第 4.4 節「評価実験」では，提案手法のソフトウェアによる評価実験結果を示した．計算機実験より，暗号回路のみをスキャンチェインに含む場合，提案手法を用いて平均 73 個の平文で 64 ビットの秘密鍵を 0.290 秒で復元可能と確認した．秘密鍵を 8bit ずつ順に求める手法では，暗号回路のみをスキャンチェインに含む場合，平均 36 個の平文で 64 ビットの秘密鍵を 1.846 秒で解読可能と確認した．スキャンチェインに他の回路が含まれていることを想定し，スキャンデータにランダムなビット値を付加してスキャンチェイン長を 13 万ビットまで変化させた場合にも，秘密鍵が解読できることを確認した．秘密鍵長が 64 ビットより大きい場合にも提案手法で解読可能であり，暗号回路のみをスキャンチェインに含む場合，平均 145 個の平文で 128 ビットの秘密鍵を 0.468 秒で解読可能と確認した．

今後の研究課題として以下が挙げられる．現在，暗号化処理のタイミングが分かっていることを前提としているが，暗号化処理タイミングが不明の時でも提案手法は有効であると考えている．ラウンド 1 処理後のタイミングが分からない時に提案手法で秘密鍵の解析ができるか今後調査予定である．また，LED 暗号回路をハードウェア実装し，ハードウェアから取得した暗号化処理中の値に対して提案手法を適用する実験が考えられる．

第5章　ハッシュへのスキャンベースサイドチャネル攻撃

5.1　本章の概要

第4章で提案したブロック暗号LEDへのスキャンベース攻撃は，ハッシュに対するスキャンベース攻撃に拡張可能である．そこで本章では，ハッシュ関数PGVの性質とアルゴリズム，ハッシュ関数の実装法HMACの性質とアルゴリズムを示し，圧縮関数としてPGVを利用しているHMAC-PGVへのスキャンシグネチャを用いたスキャンベース攻撃手法を提案する．

以下に本章の構成を示す．

第5.2節「HMACとハッシュ関数PGV」では，HMAC，PGVのアルゴリズムを説明する．HMACはハッシュ関数を用いたメッセージ認証コードで，ハッシュ関数を2回実行するという特徴がある．ハッシュ関数PGVはブロック暗号を利用したハッシュ関数である．

第5.3節「HMAC-PGVに対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しないHMAC-PGVへのスキャンベース攻撃手法を提案する．HMACの2回のハッシュの実行で用いられる2つの鍵を復元する．HMAC-PGVに実装されたブロック暗号へのスキャンベース攻撃が可能な時，HMAC-PGVに対してもスキャンベース攻撃可能なことを示す．提案手法は，特定のメッセージを入力したLSIから取得したスキャンデータにおいて，特定のビット列に着目することで秘密鍵を解読する手法である．

第5.4節「評価実験」では，提案手法のソフトウェアによる評価実験結果を示す．ハッシュに対してもスキャンベース攻撃可能なことを確認した．

第5.5節「本章のまとめ」では，本章の内容をまとめる．

5.2 HMACとハッシュ関数PGV

HMAC

HMAC (Hash-based Message Authentication Code) はメッセージ認証コード (MAC: Message Authentication Code) の一種で，ハッシュ関数を用いる．ANSI, IETF ISO, NIST により標準化されており，SSL, TLS, SSH, Ipsec 等に使われている．HMAC では，メッセージ $Text$ の認証コードを以下のように生成する．

$$HMAC(Text, K) = H(K \oplus opad, H(K \oplus ipad, Text))$$

ここで，K は鍵，H はハッシュ関数，$ipad$, $opad$ は定数である．

[7,23,28] によると，HMAC に対するサイドチャネル攻撃では以下に示す 2 つのハッシュ関数の秘密値を復元する．

$$\begin{aligned} K_{in} &= f(IV, K \oplus ipad) \\ K_{out} &= f(IV, K \oplus opad) \end{aligned}$$

但し，f は圧縮関数，IV はハッシュ関数 H への初期入力値である．図 5.1 に HMAC のアルゴリズムを示す．メッセージ $Text$ を $m_1, \ldots, m_n$ に分割し，圧縮関数 f を $n+3$ 回実行する．図 5.1 において，1 行目が 1 回目のハッシュ関数，2 行目が 2 回目のハッシュ関数を示している．

K_{in}, K_{out} は HMAC の鍵として考えることができる．メッセージ $Text$ は攻撃者が自由に設定できる．つまり，図 5.1 において，m_i は任意に設定できるが，h_{i-1} は攻撃者にとって不明な値である．攻撃により K_{in}, K_{out} が復元された場合，任意のメッセージの HMAC を求められるようになる．これにより偽造が可能になる．

PGV [29]

ブロック暗号を元にしたハッシュ関数 (PGV: Preneel-Govaerts-Vandewalle) [29] を示す．図 5.2 に圧縮関数一覧を示す．これら 12 種の圧縮関数について，h_{i-1} はブロック暗号における秘密鍵の入力，m_i は平文に該当する．

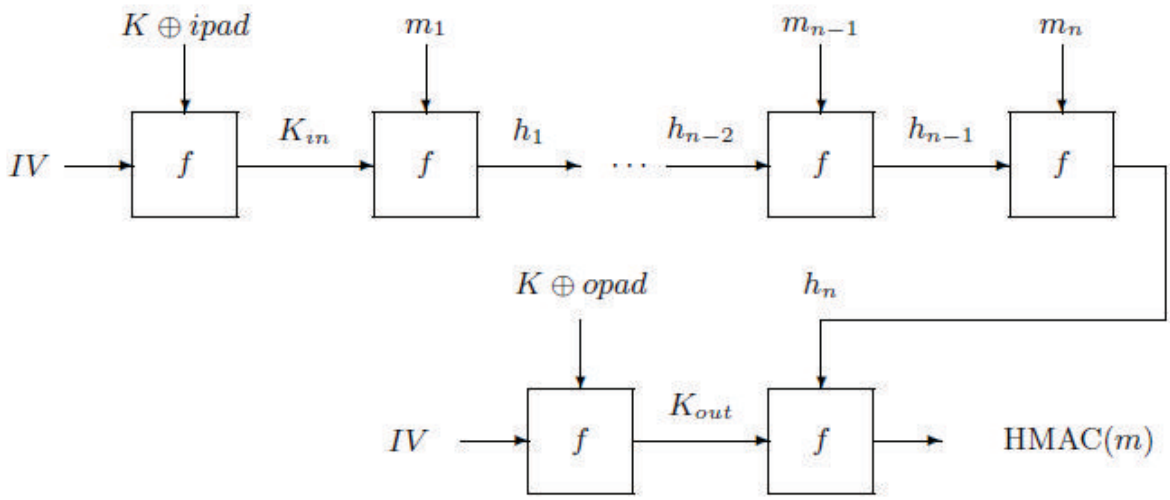

$K \oplus ipad$
m_1
m_{n-1}
m_n
IV
f
K_{in}
h_1
h_{n-2}
h_{n-1}
$K \oplus opad$
h_n
K_{out}
HMAC(m)

図 5.1: HMAC [28].

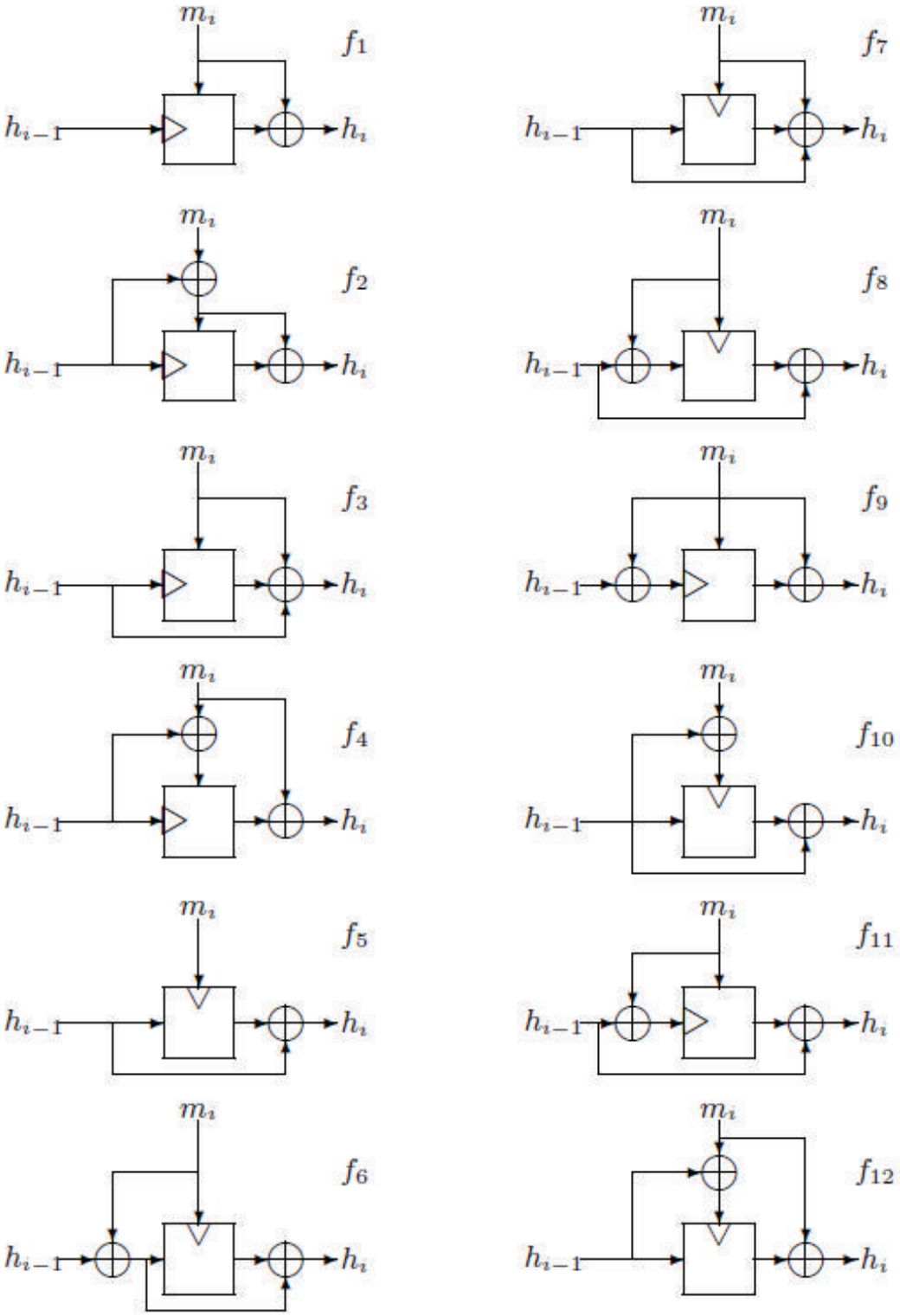


図 5.2: PGV 圧縮関数 [28].

5.3 HMAC-PGV に対するスキャンベース攻撃手法

本節では，HMAC-PGV に対するスキャンベース攻撃手法を提案する．ハッシュについてはスキャンベース攻撃の既存研究は存在していないが，ハッシュに対するスキャンベース攻撃の危険性は調べる必要がある．本章ではハッシュを利用した最も主要なメッセージ認証法 HMAC を攻撃対象としている．4 章で示したブロック暗号 LED へのスキャンベース攻撃が，HMAC に対しても適用可能であることを示すために，対象を HMAC-PGV-f_1-LED とした．

HMAC へのスキャンベース攻撃では，攻撃者が任意のメッセージ $m_1, \ldots, m_n$ を入力することで，K_{in}, K_{out} を解読することを目的とする．K_{in}, K_{out} の復元方法は，HMAC に実装されたハッシュ関数による．

ブロック暗号を元にしたハッシュ関数 PGV の 1 つである f_1 (図 5.3) を実装した HMAC-pgv-f_1 に対するスキャンベース攻撃を示す．

以下に攻撃の前提条件を示す．

- 圧縮関数として PGV 圧縮関数のうちの f_1 を利用している
- ブロック暗号として LED-64 を利用している
- ブロック暗号のレジスタにはスキャンチェインが接続されている

また，攻撃者は

- 任意のメッセージを入力し MAC 値を求められる
- 任意のタイミングでスキャンデータを取得できる

とする．

図 5.3 より，ブロック暗号における平文に対応する入力 m_i は可変で，攻撃者が自由に設定でき，ブロック暗号における鍵に対応する入力 h_{i-1} は 2 回目の圧縮関数実行時には K_{in} になっている．実装されているブロック暗号に対してスキャンベース攻撃可能な場合，2 回目の圧縮関数実行時にスキャンベース攻撃することで K_{in} をを解読可能である．よって 4.3 節に示した手法により K_{in} を解読可能である．

一方，K_{out} を復元する際は，異なる手法になる．入力 $m_1, \ldots, m_n$ は攻撃者が任意に設定でき，K_{in} の値は判明していることから，ブロック暗号における平文に対応する入力 h_n の値は容易に計算できる．また HMAC はループ構造で実装されていることを想定しているため，K_{in} 解読完了時に LED の暗号レジスタ 64 ビット全てについて位置が特定できている．HMAC-PGV-f_1-LED-64 において，LED 暗号処理における最終ラウンド (31 ラウンド目) のレジスタの値 (図 4.1) は副鍵と XOR

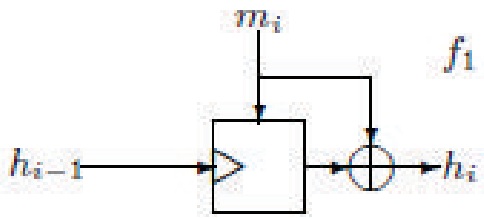


図 5.3: PGV Construction の関数 f_1 [28].

加算後，h_n と XOR 加算され，MAC 値として出力される．攻撃者は取得した MAC 値に h_n の値を XOR 加算後，31 ラウンド目のレジスタの値をスキャンデータを用いて取得し，XOR 加算することで，副鍵を復元できる．この副鍵は K_{out} と一致するため，K_{out} を解読可能である．

表 5.1: 副鍵を 1 要素ずつ解読する手法による秘密鍵解読結果.

スキャンチェイン長	追加ビット数	平均平文数	最悪平文数	平均解析時間 [s]	最悪解析時間 [s]
64	0	72.48	79	0.290	0.376
256	192	122.74	128	1.125	1.293
1024	960	159.34	165	5.534	5.788
4096	4032	192.41	197	25.125	25.802

5.4 評価実験

HMAC に対する攻撃の目的は K_{in}，K_{out} の解読である．以下に解読結果を示す.

K_{in} 解読

HMAC-PGV-f_1-LED-64 に対する K_{in} 解読の結果を以下に示す．これは LED-64 に対するスキャンベース攻撃と同様の手順であるため，結果は同一になる.

1. 副鍵を 1 要素ずつ解読する手法

本実験では，暗号回路のレジスタ 64 個のみがスキャンチェインに接続されている場合と暗号回路以外のレジスタがスキャンチェインに接続されている場合の 2 つを想定した．スキャンデータと想定するデータにランダムなビット値を加えることでスキャンチェイン長を変化させた．スキャンチェイン長が 256 ビット，1024 ビット，4096 ビットになるようにランダムなビット値を加え，100 個の秘密鍵の解読を行った.

表 5.1 にスキャンチェイン長を変化させたときの秘密鍵解読に必要な平文数の平均値と最悪値，解析時間を示す.

2. 副鍵を 2 要素ずつ解読する手法

上記の実験と同様に，スキャンチェイン長を 256 ビット，1024 ビット，4096 ビットに増加し秘密鍵の解読を行った．また，1 要素ずつ解読する手法の解読可能なスキャンチェイン長は大よそ 2^{15} ビット未満と考えられるため，スキャンチェイン長を $2^{15} = 32768$ ビット，$2^{16} = 65536$ ビットに増加させ，ランダムに生成した 64 ビットの秘密鍵 10 個の復元に必要な平均平文数，最悪平文数，平均解析時間，最悪解析時間を計測した．また，スキャンチェイン長が 10 万ビット以上の場合でも 2 要素ずつ解読する手法で現実的な時間内に解読可能であるかを確認するため，スキャンチェイン長を $2^{17} = 131072$ ビットになるようにランダム値を加え 1 つの 64 ビット秘密鍵を解読した.

表 5.2: 副鍵を 2 要素ずつ解読する手法による秘密鍵解読結果.

スキャンチェイン長	追加ビット数	平均平文数	最悪平文数	平均解析時間 [s]	最悪解析時間 [s]
64	0	35.94	40	1.846	2.402
256	192	64.23	65	8.983	9.187
1024	960	80.83	83	43.970	45.288
4096	4032	97.30	100	205.318	214.175

表 5.3: スキャンチェイン長が増大した時の解読結果.

スキャンチェイン長	追加ビット数	平均平文数	最悪平文数	平均解析時間 [s]	最悪解析時間 [s]
$2^{15} = 32768$	32704	121.3	124	2085.098	2169.678
$2^{16} = 65536$	65472	129.2	131	4509.054	4574.105

表 5.2 にスキャンチェイン長を変化させたときの秘密鍵解読に必要な平文数の平均値と最悪値，解析時間を示す.

表 5.3 にスキャンチェイン長が $2^{15} = 32768$ ビット，$2^{16} = 65536$ ビットの時の結果を示す．また，スキャンチェイン長が $2^{17} = 131072$ ビットの時，平文数は 137 個，解析時間は 9176.964 秒となった．スキャンチェイン長が 13 万ビットの場合も 2 時間半程度で解読可能なことが確認できた.

K_{out} 解読

ソフトウェア実験により K_{out} についても正しく解読出来ることを確認した.

5.5 本章のまとめ

本章では，HMAC，PGV の性質とアルゴリズムを示し，スキャンシグネチャを用いた HMAC-PGV へのスキャンベース攻撃手法を提案した．

各節の内容を以下にまとめる．

第 4.2 節「HMAC とハッシュ関数 PGV」では，HMAC，PGV のアルゴリズムを説明した．HMAC はハッシュ関数を用いたメッセージ認証コードである．ハッシュ関数を 2 回実行するという特徴がある．ハッシュ関数 PGV はブロック暗号を利用したハッシュ関数である．

第 4.3 節「HMAC-PGV に対するスキャンベース攻撃手法」では，スキャンチェインの構造に依存しない HMAC-PGV へのスキャンベース攻撃手法を提案した．HMAC-PGV に実装されたブロック暗号へのスキャンベース攻撃が可能な時，HMAC-PGV に対してもスキャンベース攻撃可能なことを示した．提案手法は，特定のメッセージを入力した LSI から取得したスキャンデータにおいて，特定のビット列に着目することで秘密鍵を解読可能な手法である．

第 4.4 節「評価実験」では，提案手法のソフトウェアによる評価実験結果を示した．実装されているブロック暗号に対してスキャンベース攻撃可能な時，ハッシュに対してもスキャンベース攻撃可能なことを確認した．

第6章　結論

本論文では，ストリーム暗号 Trivium，ブロック暗号 LED へのスキャンベース攻撃手法を提案し，ソフトウェアシミュレーション実験の結果を示した．

各章の内容を以下にまとめる．

第 2 章「サイドチャネル攻撃に関する研究動向」では，主にスキャンベース攻撃を中心としたサイドチャネル攻撃の既存研究を示した．

第 3 章「ストリーム暗号へのスキャンベースサイドチャネル攻撃」では，ストリーム暗号 Trivium の性質とアルゴリズムを示し，スキャンチェインの構造に依存しない Trivium へのスキャンベース攻撃手法を提案した．提案手法は，1 ビットレジスタ値の入力・動作サイクル数に対する変化がそのレジスタ固有の値になることを利用した手法である．評価実験により，スキャンデータに他の回路のビットが含まれていても，提案手法はビット対応解析できることを確認した．また，暗号回路のみをスキャンチェインに含む場合，Trivium の内部状態レジスタ 288 個のビット対応解析には，特定の入力を設定してキーストリーム生成フェーズからサイクル毎にスキャンデータを 13 個取得すれば良く，解析時間は 0.139 秒で済み，最も効率的に求められることを確認した．

第 4 章「ブロック暗号へのスキャンベースサイドチャネル攻撃」では，LED 暗号の性質とアルゴリズムを示し，スキャンシグネチャを用いた LED 暗号へのスキャンベース攻撃手法を提案した．提案手法は，特定の平文を入力した LSI から取得したスキャンデータを排他的論理和し，特定のビット列に着目することで秘密鍵を部分ごとに解読する手法である．計算機実験では，暗号回路のみをスキャンチェインに含む場合，提案手法を用いて平均 73 個の平文で 64 ビットの秘密鍵を 0.290 秒で復元可能と確認した．秘密鍵を 8bit ずつ順に求める手法では，暗号回路のみをスキャンチェインに含む場合，平均 36 個の平文で 64 ビットの秘密鍵を 1.846 秒で解読可能と確認した．また，スキャンチェインに他の回路が含まれていることを想定し，スキャンデータにランダムなビット値を付加してスキャンチェイン長を 13 万ビットまで変化させた場合にも，秘密鍵が解読できることを確認した．

第 5 章「ハッシュへのスキャンベースサイドチャネル攻撃」では，HMAC，PGV の性質とアルゴリズムを示し，スキャンシグネチャを用いた HMAC-PGV-f_1 への

スキャンベース攻撃手法を提案した．HMAC はハッシュ関数を 2 回実行するメッセージ認証コードである．PGV はブロック暗号を利用したハッシュ関数である．HMAC-PGV-f_1 に実装されたブロック暗号へのスキャンベース攻撃が可能な時，HMAC-PGV-f_1 に対してもスキャンベース攻撃可能である．提案手法は，特定のメッセージを入力した LSI から取得したスキャンデータにおいて，特定のビット列に着目することで秘密鍵を解読する手法である．計算機実験より，ハッシュに対してもスキャンベース攻撃可能なことを確認した．

これらスキャンベース攻撃に対する防御手法は，各章で示した攻撃の必要条件を満たさないような回路の設計である．攻撃の必要条件の中で最も脅威となる条件はスキャンチェインに攻撃者がアクセスできることである．スキャンチェインを焼き切る等の対策が最も有効と考えられる．

今後の課題として，提案手法を他の暗号アルゴリズムへ適用すること，ハードウェア実装することを考えている．また，スキャンベース攻撃以外のサイドチャネル攻撃についても同様に攻撃・防御手法を考案することを考えている．

謝辞

本研究は，筆者が早稲田大学大学院 基幹理工学研究科 情報理工・情報通信専攻博士後期課程在学中に，同大学院 戸川望教授の指導のもとに行ったものである．

本論文の執筆にあたり多大なる御指導，御助言を賜りました本学情報理工・情報通信専攻 戸川望教授に心より感謝致します．学部，修士課程，博士後期課程の4年間にわたり丁寧にご指導頂き，日々，多くを学び成長することができました．おかげさまで修士課程，博士後期課程の早期修了に至り大変感謝しております．同じく常日頃から的確なご指導を頂きました本学情報理工・情報通信専攻 柳澤政生教授に深く御礼申し上げます．研究者としての姿勢を学ぶことができ，研究の質，研究生活の質を高めることが出来ました．本論文全般に亘り熱心なご指導を頂きました本学情報生産システム研究科 木村晋二教授に深く感謝致します．様々なご指摘を頂き，本論文を修正できました．加えて本論文に関する貴重な御助言を頂きました本学情報理工・情報通信専攻 森達哉准教授に深く感謝致します．お忙しい中多岐に亘るご指導，ご指摘を頂き，本論文の質を高めることができました．

また，研究に関する数多くのアドバイスを頂いた史又華准教授に感謝申し上げます．学部時代より研究をご指導頂き，研究に関する鋭いご意見，ご助言を下さいました小寺博和氏および跡部悠太氏に御礼申し上げます．蒋慧倩さんとは日常の議論を通じ多くの課題を発見できました．研究生活においても多くを学びました．ここに感謝の意を表します．

最後に，日頃より多方面にわたり様々な御意見を頂き支援していただいた戸川研究室，柳澤研究室の皆様に心より感謝致します．

参考文献

[1] M. Ågren, "Some instant- and practical-time related-key attacks on KTAN-TAN32/48/64," in *Proc. International Conference on Selected Areas in Cryptography*, pp. 213–229, 2011.

[2] M. Agrawal, S. Karmakar, D. Saha, and D. Mukhopadhyay, "Scan based side channel attacks on stream ciphers and their counter-measures," *Lecture Notes in Computer Science*, vol. 5365, pp. 226–238, 2008.

[3] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Novel test-mode-only scan attack and countermeasure for compression-based scan architectures," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 808–821, 2015.

[4] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "New scan-based attack using only the test mode," in *Proc. IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 234–239, 2013.

[5] S. S. Ali, O. Sinanoglu, and R. Karri, "Test-mode-only scan attack using the boundary scan chain," in *Proc. IEEE European Test Symposium (ETS)*, pp. 1–6, 2014.

[6] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Scan attack in presence of mode-reset countermeasure," in *Proc. IEEE International On-Line Testing Symposium (IOLTS 2013)*, pp. 230–231, 2013.

[7] S. Belaïd, L. Bettale, E. Dottax, L. Genelle, and F. Rondepierre, "Differential power analysis of HMAC SHA-2 in the hamming weight model," in *Proc. International Conference on Security and Cryptography (SECRYPT 2013)*, pp. 230–241, 2013.

[8] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Lecture Notes in Computer Science*, vol. 1294, pp. 513–525, 1997.

[9] D. Boneh, R. A. DeMillo, and R. J. Lipton, “On the importance of checking cryptographic protocols for faults,” *Lecture Notes in Computer Science*, vol. 1233, pp. 37–51, 1997.

[10] C. Berbain, O. Billet, A. Canteaut, N. Courtois, B. Debraize, H. Gilbert, L. Goubin, A. Gouget, L. Granboulan, C. Lauradoux, M. Minier, T. Pornin, and H. Sibert, “DECIM v2,” http://www.ecrypt.eu.org/stream/decimp3.html.

[11] A. Biryukov and D. Khovratovich, “Related-key cryptanalysis of the full AES-192 and AES-256,” *Lecture Notes in Computer Science*, vol. 5912, pp. 1–18, 2009.

[12] C. D. Cannière and B. Preneel, “Trivium,” http://www.ecrypt.eu.org/stream/triviumpf.html.

[13] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based attack against trivium stream cipher independent of scan structure,” in *Proc. IEEE International Conference on ASIC (ASICON 2013)*, pp. 146–149, 2013.

[14] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based attack on the LED block cipher using scan signatures,” in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2014)*, pp. 1460–1463, 2014.

[15] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based attack against trivium stream cipher using scan signatures,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 7, pp. 1444–1451, 2014.

[16] M. Fujishiro, M. Yanagisawa, and N. Togawa, “Scan-based side-channel attack on the LED block cipher using scan signatures,” *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 12, pp. 2434–2442, 2014.

[17] J. Guo, T. Peyrin, A. Poschmann, and M. Robshaw, “The LED block cipher,” *Lecture Notes in Computer Science*, vol. 6917, pp. 326–341, 2011.

[18] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, “Side channel cryptanalysis of product ciphers,” *Lecture Notes in Computer Science*, vol. 1485, pp. 97–110, 1998.

[19] P. Kocher, "Timing attacks on implementations of Diffie-Hellmann, RSA, DSS, and other systems," *Lecture Notes in Computer Science*, vol. 1109, pp. 104–113, 1996.

[20] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Crypto '99*, pp. 388–397, 1999.

[21] H. Kodera, M. Yanagisawa, and N. Togawa, "Scan-based attack against DES cryptosystems using scan signatures," in *Proc. IEEE Asia Pacifc Conference on Circuits and Systems*, pp. 599–602, 2012.

[22] Y. Liu, K. Wu, and R. Karri, "Scan-based attacks on linear feedback shift register based stream ciphers," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 16, no. 2, pp. 20:1–20:15, 2011.

[23] R. McEvoy, M. Tunstall, C. C. Murphy, and W. P. Marnane, "Differential power analysis of HMAC based on SHA-2, and countermeasures," *Lecture Notes in Computer Science*, vol. 4867, pp. 317–332, 2007.

[24] D. Mukhopadhyay, S. Banerjee, D. RoyChowdhury, and B. B. Bhattacharya, "Cryptoscan: a secured scan chain architecture," in *Proc. Asian Test Symposium (ATS)*, pp. 348–353, 2005.

[25] R. Nara, N. Togawa, M. Yanagisawa and T. Ohtsuki, "A scan-based attack based on discriminators for AES cryptosystems," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E92-A, no. 12, pp. 3229–3237, 2009.

[26] R. Nara, K. Satoh, M. Yanagisawa, T. Ohtsuki and N. Togawa, "Scan-based side-channel attack against RSA cryptosystems using scan signatures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E93-A, no. 12, pp. 2481–2489, 2010.

[27] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "Scan-based attack against elliptic curve cryptosystems," in *Proc. Asia and South Pacific Design Automation Conference (ASP-DAC 2010)*, pp. 407–412, 2010.

[28] K. Okeya, "Side channel attacks against HMACs based on block-cipher based hash functions," *Lecture Notes in Computer Science*, vol. 4058, pp. 432–443, 2006.

[29] B. Preneel, R. Govaerts, and J. Vandewalle, “Hash functions based on block ciphers: a synthetic approach,” *Lecture Notes in Computer Science*, vol. 773, pp. 368–378, 1993.

[30] J. D. Rolt, G. D. Natale, M. Flottes, and B. Rouzeyre, “A novel differential scan attack on advanced DFT structures,” *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 18, no. 4, pp. 58:1–58:22, 2013.

[31] J. D. Rolt, G. D. Natale, M. Flottes, and B. Rouzeyre, “Scan attacks and countermeasures in presence of scan response compactors,” in *Proc. IEEE European Test Symposium (ETS)*, pp. 19–24, 2011.

[32] J. D. Rolt, G. D. Natale, M. Flottes, and B. Rouzeyre, “Are advanced dft structures sufficient for preventing scan-attacks?,” in *Proc. IEEE VLSI Test Symposium (VTS)*, pp. 246–251, 2012.

[33] J. D. Rolt, A. Das, G. D. Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, “A new scan attack on rsa in presence of industrial countermeasures,” *Lecture Notes in Computer Science*, vol. 7275, pp. 89–104, 2012.

[34] J. D. Rolt, A. Das, G. D. Natale, M. Flottes, B. Rouzeyre, and I. Verbauwhede, “A scan-based attack on elliptic curve cryptosystems in presence of industrial design-for-testability structures,” in *Proc. IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, pp. 43–48, 2012.

[35] Y. Tsunoo, E. Tsujihara, K. Minematsu, and H. Miyauchi, “Cryptanalysis of block ciphers implemented on computers with cache,” *Lecture Notes in Computer Science*, vol. 2779, pp. 62–76, 2003.

[36] B. Yang, K. Wu, and R. Karri, “Scan based side channel attack on dedicated hardware implementations of data encryption standard,” in *Proc. International Test Conference*, pp. 339–344, 2004.

[37] B. Yang, K. Wu and R. Karri, “Secure scan: a design-for-test architecture for crypto chips,” *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 10, pp. 2287–2293, 2006.

[38] 日経テクノロジー online, “改訂版 EDA 用語辞典 スキャン・テスト,” http://techon.nikkeibp.co.jp/article/WORD/20090107/163744/.

[39] 藤代美佳, 柳澤政生, 戸川望, “スキャンシグネチャを用いたストリーム暗号 Trivium へのスキャンベース攻撃手法,” 信学技報, VLD2013-8, vol. 113, no. 30, pp. 61–66, 2013.

[40] 藤代美佳, 柳澤政生, 戸川望, “ストリーム暗号 Trivium に対するスキャンチェインの構造に依存しないスキャンベース攻撃手法,” 第 26 回回路とシステムワークショップ論文集, pp. 442–447, 2013.

[41] 藤代美佳, 柳澤政生, 戸川望, “スキャンシグネチャを用いた LED 暗号へのスキャンベース攻撃,” 信学技報, VLD2013-55, vol. 113, no. 235, pp. 47–52, 2013.

[42] 藤代美佳, 柳澤政生, 戸川望, “スキャンチェイン長に依存しない LED 暗号に対するスキャンベース攻撃,” 信学技報, VLD2013-139, vol. 113, no. 454, pp. 31–36, 2014.

[43] 藤代美佳, 柳澤政生, 戸川望, “鍵長に依存しない LED 暗号に対するスキャンベース攻撃,” 情処研報, vol. 2015-SLDM-170, no. 47, pp. 149–154, 2015.

研究業績

論文

1. ◯ **M. Fujishiro**, M. Yanagisawa, and N. Togawa, "Scan-based attack against trivium stream cipher using scan signatures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 7, pp. 1444–1451, 2014.

2. ◯ **M. Fujishiro**, M. Yanagisawa, and N. Togawa, "Scan-based side-channel attack on the LED block cipher using scan signatures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E97-A, no. 12, pp. 2434–2442, 2014.

3. H. Jiang, **M. Fujishiro**, H. Kodera, M. Yanagisawa, and N. Togawa, "Scan-based side-channel attack on the camellia block cipher using scan signatures," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E98-A, no. 12, pp. 2547–2555, 2015.

国際会議 (査読付き)

1. ◯ **M. Fujishiro**, M. Yanagisawa, and N. Togawa, "Scan-based attack against trivium stream cipher independent of scan structure," in *Proc. IEEE International Conference on ASIC (ASICON 2013)*, pp. 146–149, 2013.

2. ◯ **M. Fujishiro**, M. Yanagisawa, and N. Togawa, "Scan-based attack on the LED block cipher using scan signatures," in *Proc. IEEE International Symposium on Circuits and Systems (ISCAS 2014)*, pp. 1460–1463, 2014.

3. **【招待論文】** **M. Fujishiro**, Y. Shi, M. Yanagisawa, and N. Togawa, "Scan-based side-channel attack against symmetric key ciphers using scan signatures," in *Proc. IEEE Conference on Electron Devices and Solid-State Circuits (EDSSC 2015)*, pp. 309–312, 2015.

4. H. Jiang, **M. Fujishiro**, H. Kodera, M. Yanagisawa, and N. Togawa, "Scan-based side-channel attack on camellia cipher using scan signatures," in *Proc. IEEE Asia Pacifc Conference on Circuits and Systems (APCCAS 2014)*, pp. 252–255, 2014.

5. H. Jiang, **M. Fujishiro**, M. Yanagisawa, and N. Togawa, "Scan-based side-channel attack implementation evaluation on the LED cipher using SASEBO-GII," in *Proc. Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2015)*, pp. 433–434, 2015.

国内会議 (査読付き)

1. **藤代美佳**, 柳澤政生, 戸川望, "ストリーム暗号 Trivium に対するスキャンチェインの構造に依存しないスキャンベース攻撃手法," 第 26 回回路とシステムワークショップ論文集, pp. 442–447, 2013.

学会発表

1. **藤代美佳**, 柳澤政生, 戸川望, "スキャンシグネチャを用いたストリーム暗号 Trivium へのスキャンベース攻撃手法," 信学技報, VLD2013-8, vol. 113, no. 30, pp. 61–66, 2013.

2. **藤代美佳**, 柳澤政生, 戸川望, "スキャンシグネチャを用いた LED 暗号へのスキャンベース攻撃," 信学技報, VLD2013-55, vol. 113, no. 235, pp. 47–52, 2013.

3. **藤代美佳**, 柳澤政生, 戸川望, "スキャンチェイン長に依存しない LED 暗号に対するスキャンベース攻撃," 信学技報, VLD2013-139, vol. 113, no. 454, pp. 31–36, 2014.

4. **藤代美佳**, 柳澤政生, 戸川望, "鍵長に依存しない LED 暗号に対するスキャンベース攻撃," 情処研報, vol. 2015-SLDM-170, no. 47, pp. 149–154, 2015.

受賞

1. 2014 年 8 月　DA シンポジウム 2014 アルゴリズムデザインコンテスト特別賞.