サイドチャネル攻撃耐性の効率的な評価が可能なフレームワーク
高位合成による AES 回路の評価をケーススタディとして

小島　拓也†

† 東京大学 情報理工学系研究科
E-mail: †tkojima@hal.ipc.i.u-tokyo.ac.jp

あらまし　消費電力や電磁波, 処理時間の違いなどのサイドチャネル情報を用いて, 暗号処理モジュールの内部状態を予測し鍵などの機密情報を盗み出すサイドチャネル攻撃は暗号技術を中核とした情報の保護や認証技術を脅かす存在である. 安全な暗号処理モジュールの開発を行うには, Test Vector Leakage Assessment (TVLA) などの評価に加えて, 100 万ほどの大量のトレースを用いたとしても相関電力解析による攻撃が成功しないことを確認する必要がある. 暗号処理モジュールの評価を効率的に行うためには, 対象モジュールの実装に加えて暗号処理中の電力波形を測定する実験環境の構築や, 大量のトレースデータを高速に解析するソフトウェアが必要不可欠である. 本研究では, この評価プロセスにおける実装コストを最小化するために, 1) サイドチャネル攻撃の研究で標準的に用いられる評価ボード SAKURA-X を対象に, AXI バスを持つ暗号処理モジュールの接続が可能な設計テンプレート, および, 2) 設計テンプレートを利用可能なドライバ, GPU やメニーコアプロセッサを活用して高速に解析可能なソフトウェアツールを含むフレームワークを提案する. 本研究で実装したソフトウェアは, オープンソースの解析ツールである ChipWhisperer のプラグインとして提供しているため, 本研究で利用する評価環境以外でも有用である. 相関電力解析に関しては, GPU を用いることで, ChipWhisperer の実装を用いた場合と比べて最大で 450 倍程度の高速化を達成した. さらに, ケーススタディとして, Vitis HLS を用いて設計された AES 処理回路を評価し, 提案した設計テンプレートの有用性の確認を行った.

1. はじめに

近年の情報通信において暗号技術は機密情報の保護や認証技術に必要不可欠である. 例えば, Windows 11 では, Trusted Platform Module (TPM) 2.0 チップを必須の要件としている. これは鍵の保存, 暗号化, 復号などの暗号処理に必要な機能を提供する専用のハードウェアモジュールである. ハードウェアによる信頼の基点 (root of trust) を提供することで, 悪意のあるソフトウェアによるリスクを軽減する. しかし, 暗号処理モジュールには物理的なセキュリティリスクが存在する. 特に, サイドチャネル攻撃は暗号処理の内容と相関のある物理情報を利用して, 暗号処理モジュールの内部状態を予測し, 鍵などの機密情報を盗み出す攻撃手法である. 漏洩する物理情報としては, 消費電力や電磁波などが一般的に考慮される [1].

取得された物理情報はさまざまな方法で解析されるが, 特に相関電力解析 (Correlation Power Analysis: CPA) は効率的な攻撃手法として知られている [2]. CPA は, 他のプロファイリングベースの攻撃手法 (例: テンプレート攻撃 [3]) のように攻撃者が鍵操作などの完全な制御権限を持つ必要がないため, より現実的な攻撃手法である. CPA 攻撃では, はじめに攻撃者が暗号処理中の電力消費や放射される電磁波の強弱を波形として取得し, トレースデータとして記録する. 数千から数百万のトレースを取得した後, 攻撃者はこれらを解析して秘密情報を復元する. 解析フェーズでは, 攻撃者は暗号処理の予測中間値と電力消費の相関を計算し, 最も可能性の高い鍵を推測する.

しかし, 取得した波形の長さ (サンプル数) やトレース数の数が大きくなるにつれて, 相関係数の計算にかかる時間が増加する. したがって, 鍵を復元するために必要なトレース数は, 暗号処理モジュールのセキュリティ評価の指標の一つとして考えられる [4]. 設計した暗号処理モジュールがサイドチャネル攻撃へ耐性を有するかどうかを評価するには, 取得したトレースに対して, ノイズ除去などの様々な前処理を試行したり, 異なる予測モデルを用いて解析を繰り返す必要がある. 評価の結果, 十分な耐性が確認されない場合は, 暗号処理モジュールに施した対策を改良し改めて評価を行う必要がある. 効率的な評価サイクルを実現するには, 解析ソフトウェアの高速化は重要である. 加えて, 評価対象の暗号処理モジュールを実装するだけでなく, オシロスコープなどの波形取得装置と連動してトレースデータを取得する実験環境の整備も評価サイクルの効率化を阻害する要因である.

この課題に対処するために, 本研究では暗号処理モジュールの効率的な評価を支援するフレームワークを提案する. 相関係数の計算には大量のデータ並列性が存在する. そこで, 提案フレームワークにはメニーコアプロセッサや GPU を活用した高速な解析を可能にするソフトウェアツールが含まれる. メニーコアプロセッサ向けの並列処理には OpenMP を, GPU 向けの

並列処理には OpenCL と CUDA を採用した．このため, 本フレームワークは主要なベンダー (NVIDIA, AMD, Intel, Apple) の GPU をサポートする．また, 提案フレームワークのソフトウェア部はサイドチャネル攻撃の研究で広く利用されるオープンソースの解析ツールである ChipWhisperer [5] のプラグインとして提供するため, ChipWhisperer と互換する評価環境でも利用できる．

また, サイドチャネル攻撃の研究で標準的に利用される SASEBO/SAKURA プロジェクトの評価ボードである SAKURA-X ボードを対象に, 設計テンプレートを作成した．SAKURA-X には暗号処理モジュール用と制御モジュール用の 2 つの FPGA が搭載されており, 暗号処理モジュール部のみ消費電力を測定できるように設計されている．設計テンプレートには, 外部の PC から鍵の設定や, 平文の入力, 暗号文の出力などのデータ交換や, 暗号処理の開始などを行うための制御レジスタへのアクセスを可能にするインターフェースを備える．また, 制御用のソフトウェアは前述のプラグインに含まれる．このテンプレートと暗号処理モジュールの接続には AXI バスを利用しているため, これに互換するプロセッサのソフトコアや高位合成 (High-Level Synthesis: HLS) により設計した暗号処理モジュールの接続が容易である．

本研究では, 提案フレームワークの有用性を示すために, AES-128 の鍵を復元する CPA 攻撃に対する影響を評価する．取得したトレースデータをいくつか構成の異なるシステムで解析し, 提案フレームワークによる高速化の効果を評価する．また, 設計テンプレートがトレースデータに与える影響を確認するために, テンプレートを利用せずた行った RTL 実装と, テンプレートを用いて実装した RTL 実装および HLS 実装をそれぞれ比較する．最後に, ケーススタディとして HLS コンパイラよって施される最適化が攻撃容易性に与える影響を検証する．

2. 背　　景

2.1 消費電力とリークモデル

一般に, 消費電力をモデル化する上で精度と計算複雑性のトレードオフが存在し, 用途に応じてアナログレベルや論理レベルなどのモデリングが行われる．しかし, サイドチャネル攻撃においては消費電力の絶対値を見積もることが目的ではなく, 相対関係が満たされていれば十分であるため, より簡略されたモデルが用いられることが多い．例えば, ハミング重みモデルでは, 中間データのハミング重み, つまり, ビット列の中の'1' の総数を消費電力としてモデル化する [1]．同様に, ハミング距離モデルでは, 中間データの変化による消費電力を, 変化前と変化後の中間データのハミング距離としてモデル化する．ある中間データ v_0 から v_1 に変化した場合の消費電力は, ハミング距離関数 $HD(v_0, v_1)$ で表される．ハミング距離関数は, ハミング重み関数 HW と XOR 演算によって $HD(v_0, v_1) = HW(v_0 \oplus v_1)$ と表されるため, ハミング距離モデルはハミング重みモデルと同様に取り扱うことができる．

AES 処理モジュールに関しては, 攻撃を行う中間データの候補がいくつか存在するが, 最も一般的なのは非線形置換関数である S-Box の出力である．128 ビットの鍵長の場合, AES は合計 10 ラウンドの処理を繰り返して暗号化を行う．*KeySchedule* は各ラウンドを用いるラウンド鍵を元の鍵から生成する．生成されたラウンド鍵は各ラウンドの最初に XOR され, その後 S-Box によって各バイトが変換される．各ラウンドの S-Box 出力のうち, 通常は第 1 ラウンドか最終ラウンドの S-Box 出力が攻撃の対象となる．例えば, 第 1 ラウンドを攻撃対処とするとき, ある部分鍵を $\hat{k}$ と推定した場合, 予測リーケージは平文 p と推定鍵 $\hat{k}$ から以下のように計算できる．

$$h_{\hat{k}} = HW(SBox(p \oplus \hat{k})) \tag{1}$$

2.2 CPA 攻撃

CPA では, ピアソンの相関係数を用いて消費電力と予測リーケージの間の相関を計算する．サンプルベクトル X と Y の相関係数 ρ は以下のように定義される．

$$\rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X)Var(Y)}} \tag{2}$$

ここで, $Cov(X, Y)$ は X と Y の共分散, $Var(X)$ は X の分散を表す．

CPA 攻撃においては X と Y を消費電力と予測リーケージとして考える．各トレースの波形は L 個のサンプルを持ち, $w_{i,t}$ は i 番目のトレースの t 番目のサンプルを表す．また, 前述の通り, リークモデルには予測した部分鍵 $\hat{k}$ が必要であり, AES-128 の場合は 256 個の部分鍵候補が存在する．よって, 鍵候補 $\hat{k}$ とサンプル位置 t ごとに相関係数 $r_{\hat{k},t}$ を計算する．$r_{\hat{k},t}$ は以下のように計算される．

$$r_{\hat{k},t} = \frac{\sum_{i=1}^{N}[(h_{i,\hat{k}} - \bar{h}_{\hat{k}}) \cdot (w_{i,t} - \bar{w_t})]}{\sqrt{\sum_{i=1}^{N}(h_{i,\hat{k}} - \bar{h}_{\hat{k}})^2}\sqrt{\sum_{i=1}^{N}(w_{i,t} - \bar{w_t})^2}} \tag{3}$$

$$= \frac{N\sum_{i=1}^{N} h_{i,\hat{k}} w_{i,t} - \sum_{i=1}^{N} h_{i,\hat{k}} \sum_{i=1}^{N} w_{i,t}}{\sqrt{(\sum_{i=1}^{N} h_{i,\hat{k}})^2 - N\sum_{i=1}^{N} h_{i,\hat{k}}^2}\sqrt{(\sum_{i=1}^{N} w_{i,t})^2 - N\sum_{i=1}^{N} w_{i,t}^2}} \tag{4}$$

N はトレース数を表す．式 (3) は単純な形式であるが, 平均値の計算のために全てのトレースと予測リーケージの和を先に計算する必要があり, データアクセスの効率が悪い．そこで, 式 (4) のように最適化された形式が一般に用いられる [11]．また, この形式は, 一部のトレースを用いて部分和を計算しながら相関係数の更新が可能であるため, メモリのサイズが制限される GPU であっても効率的に計算が行うことができる．

2.3 関 連 研 究

ChipWhisperer はサイドチャネル攻撃研究用のオープンソースフレームワークとして広く利用されている [5]．大部分が Python で記述されており, 解析結果を Jupyter Notebook などで可視化するための機能も提供している．また, 解析ソフトウェアだけでなく, このフレームワークがサポートする市販の評価ボードを用意することで, 特殊なハードウェアの開発を行わずにサイドチャネル攻撃の研究を行うことができる．例えば, ChipWhisperer-Nano ボードは, 波形取得用のアナログデジタルコンバータと攻撃対象となる ARM Cortex-M0 を搭載しており, ソフトウェア実装の暗号処理を評価できる．Kintex-7 FPGA を搭載した CW310 ボードや Artix-7 FPGA を搭載した CW305 ボードなども利用できる．ところが, 解析ソフトウェアの実装が Python であるため, CPA 攻撃の実行時間が長くなるという問題がある．

図 1: サイドチャネル攻撃研究用フレームワークの比較

フレームワーク	OSS	ソフトウェア			ハードウェア	
		予測モデル	実装言語	高速化	ターゲット	トレース取得
RamDPA [6]	✓	S-Box 出力のみ	C++	マルチスレッド	n/a	
DareDevil [7]	✓	LUT ファイル	C++	マルチスレッド	n/a	
[8]	×	最終ラウンド	CUDA	GPU	n/a	
[9]	×	最終ラウンド	OpenCL,CUDA	GPU	n/a	
FOBOS3 [10]	✓	別途計算	Python	n/a[1]	FOBOS DUT	FOBOS Shield
SASEBO/SAKURA	✓	最終ラウンド	C#	マルチスレッド	SASEBO/SAKURA シリーズ	VISA 互換 オシロスコープ
ChipWhisperer [5]	✓	複数利用可能	Python	n/a	CW シリーズ	ChipWhisperer-Nano など
提案フレームワーク	✓[1]	CW に互換	Python,C++	OpenMP OpenCL,CUDA	設計テンプレート	VISA 互換 オシロスコープ

[1] 公開予定

FOBOS3 [10] も ChipWhisperer と同様に, 評価用ハードウェアとしての FPGA とそれらを制御するソフトウェアや解析ツールを提供している. 評価環境としては, FOBOS DUT ボードと呼ぶ Artix-7 FPGA を搭載した評価ボードと, FOBOS Shield と呼ぶ波形取得ボードをサポートし, 別途制御用ボードとして PYNQ-Z2 ボードを用いる. これらのボードは KiCad で設計されており, オープンソースで公開されている. しかし, 市販されていないため, 利用者は自分で必要な実装部品を調達し, 基板を製造する必要がある点でハードウェアの準備に手間がかかる. また, 解析ソフトウェアは ChipWhisperer と同じく Python で記述されているため, 同様に解析時間の問題を抱えている.

SASEBO プロジェクトとその後継の SAKURA プロジェクト [12] も同様に, サイドチャネル攻撃に関する評価用のボードを開発している. 搭載している FPGA が異なるいくつかのシリーズがあり, 中には DPA コンテスト用に採用されたものもある. 制御用のソフトウェアや VISA(Visual Instrument Software Architecture) に互換するオシロスコープを扱うことができる波形取得用のソフトウェア, またコンテスト向けに CPA ツールなどが提供されている. CPA ツールについてはマルチスレッド実行に対応しているが, Windows 環境での実行を前提としている.

一方で, 解析の高速化を図ったオープンソースのフレームワークもいくつか存在する. RamDPA [6] は解析の過程で必要な中間データを最小化し, データの並びを最適化することでキャッシュミスを削減している. また, マルチスレッディングを用いて実行時間の短縮を行なっている. しかし, 計算方法が式 (3) の形式であるため, 計算時間の短縮は限定的である. また, 解析可能なリークモデルも S-Box 出力のハミング重みモデルしか対応しておらず, 他のリークモデルに対応するためにはソースコードの変更が必要である. DareDevil [7] は, RamDPA と同様にマルチスレッディングを用いて CPA 攻撃を高速化するツールである. RamDPA と異なり, 式 (4) の形式で計算を行うように最適化されている. また, 相関を隠蔽する対策手法として知られるマスキングに対して有効な攻撃手法である High-Order CPA にも対応している. リークモデルは, Look-Up Table (LUT) ファイルを用いて指定することができ, S-Box の出力などのよく知られたものは予め用意されているが, LUT で表現できないリークモデルには対応できない. また, これらの二つのツールは独自の形式でトーレスデータを扱うため, ユーザーはトレースデータをそれぞれのフォーマットに変換する必要がある.

表 1: プロファイリング結果

処理	実行時間 (ms)	割合 (%)
サンプルの和, 2 乗和	72.80	0.3772
予測リーケージの計算	25.04	0.1297
予測リーケージの和, 2 乗和	6.384	0.0330
予測リーケージとサンプルの内積	19,083	98.89
相関係数の計算	80.53	0.4173
その他	28.67	0.1486

オープンソースの解析ツールではないが, GPU による高速化の試みはいくつか報告がある [8], [9]. Swamy [8] らは, CUDA と OpenCL を用いた CPA 攻撃の実装を提供している. ところが, 実装されている解析アルゴリズムは 2.2 節で説明する一般的な CPA とは異なる. まず, 各トレースの消費電力波形を最大値に変換してから相関係数を計算する. この方法は, 波形中の正しいピーク点を特定できれば攻撃に成功する可能性があるものの, 情報がリークするピーク点が常に波形の最大値であるとは限らないため, 解析の有効性は限定的である. また, Gamaarachchi ら [9] は, NVIDIA の GPU を用いた CPA 攻撃の実装を提供している. CUDA 実装の方法は本提案と類似しているが, 本研究の提案では, GPU と OpenMP によるマルチスレッディングを併用している点で異なる. CPA の計算においては, GPU で実行するには適さない計算や高速化への貢献があまり期待できない箇所がある. 本研究ではプロファイリングを行い, GPU で実行すべき箇所を選定することで, 効率的な高速化を実現している.

最後にこれらの先行研究と本研究の提案フレームワークとの比較を表 1 にまとめる. 提案フレームワークは, ChipWhisperer のプラグインとして提供されるため, ChipWhisperer の機能も利用可能である. つまり, CW シリーズのハードウェアを用いて取得したトレースデータを形式変換などの処理を行わずに, 提案フレームワークの高速化 CPA で解析することが可能である. 提案する設計テンプレートは, 他のフレームワークで提供される FPGA 向け設計サンプルとは異なり, 標準化された AXI バスを用いているため, 簡単に他のモジュールと接続することができる. したがって, 高位合成ツールを用いて設計した暗号処理モジュールであれば, Verilog などの RTL 記述を一切行わずに評価を行うことも可能である.

3. 提案フレームワークの実装

3.1 CPA の高速化実装

3.1.1 プロファイリング

図 2 に AES-128 の鍵 16 バイトを復元する CPA 攻撃の過程

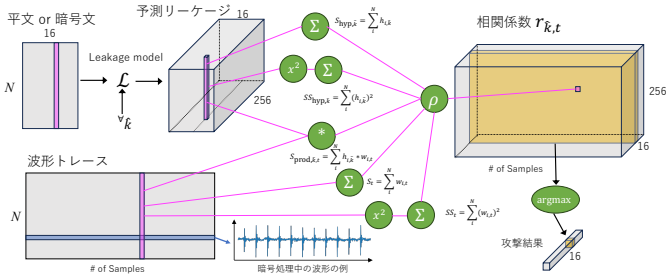


図 2: AES-128 における相関係数の計算と鍵推定の流れ

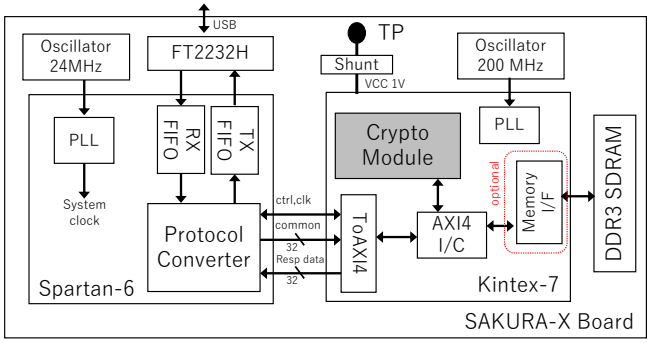


図 3: 設計テンプレートの構成

で生成される中間データを示す．トレースデータとして N セットの平文 (または暗号文) と波形データが与えられる．平文から有効なリークモデルを用いて，部分鍵位置ごとに予測リーケージを計算する．予測リーケージはトレース数方向に和と 2 乗和が計算される．この結果は波形位置に依存しないため，相関係数の計算に再利用できる．波形データもサンプル位置ごとに和と 2 乗和が計算される．この結果は予測鍵や部分鍵に依存しないため，相関係数の計算に再利用できる．一方で，波形トレースと予測リーケージの内積を計算する部分は，波形位置，予測鍵，部分鍵位置の 3 つの要素に依存するため，再利用できない．最終的に，$16\times256\times L$ の相関係数が計算される．鍵の復元を行う際は，部分鍵ごとに絶対値が最大の相関係数となった予測鍵を選択する．

表 1 に相関係数の計算にかかる処理時間の内訳を示す．この結果は N=10000，波形長=4000 のデータを後述のシステム No. 1 で単スレッド実行時の結果である．実行時間の 98.89% が波形トレースと予測リーケージの内積計算に費やされている．よって，GPU による高速化はこの部分を対象とすべきである．また，波形サンプルと予測リーケージの和，2 乗和の計算に必要なデータも必然的に GPU に転送するため，これらの計算も GPU で行う．一方で，予測リーケージの計算はモデルによっては複雑な計算を必要とするため，GPU による計算の対象外とする．さらに，相関係数の計算については後述の精度の問題があるため，ホストプロセッサで計算する．

3.1.2 相関係数計算の精度

式 (4) に示した計算方法はメモリアクセスの回数を削減できる一方で，減算における桁落ちの可能性がある．通常，予測リーケージは整数値であるためこの問題は発生しない．一方で，平均化やローパスフィルタなどを用いた波形データは浮動小数点数値となる．ChipWhisperer の実装では，相関係数の計算に `numpy.longdouble` を用いて，この問題を可能な限り回避している．しかし，このデータ型はプラットフォームやコンパイラに依存するため，実際の精度は異なる．例えば，x86-64 の GCC では `long double` の精度は仮数部が 64 ビットの拡張倍精度であるが，AppleSilicon の Clang では `double` と同等の精度である．このため，`long double` の精度が十分でない場合，減算結果が 0 となり，相関係数の計算結果が無限大となることがある．本提案の実装では，ChipWhisperer の実装と同じ精度を保つために，桁落ちが発生しうる部分の計算のみに拡張倍精度を利用し，それ以外の部分は倍精度で計算する．

AppleSilicon の CPU のように倍精度以上の精度で計算ができないプロセッサでは，ソフトウェアエミュレーションを用いることで拡張倍精度を実現できる [13]．例えば，2 つの倍精度変数を 1 つの四倍精度変数として扱い，複数回の演算を組み合わせてエミュレートすることが可能である．パフォーマンスの上では，エミュレーションによるオーバーヘッドが発生するが，拡張倍精度が必要な箇所のみで使用するため，全体の実行時間に与える影響は限定的である．同様に，AppleSilicon の GPU [14] のようにいくつかの GPU は倍精度演算をサポートしていないため，同様に単精度演算によるエミュレーションを用いて倍精度演算を実現する OpenCL 実装を行う．

3.2 設計テンプレートと制御ソフトウェア

本研究で設計した SAKURA-X 向けのテンプレートは，前述の通りコントローラ部である Spartan-6 FPGA の設計と暗号処理モジュールを実装する Kintex-7 FPGA の設計に分かれる．構成を図 3 に示す．コントローラ部は USB 変換モジュールである FT2232H を介してホスト PC と通信し，データアクセスの内容をデコードして Kintex-7 FPGA にリクエストを送信する．Spartan-6 と Kintex-7 FPGA はボード上で直接接続されている．ただし，利用可能な IO ピンの制約から，アドレスバス 32 ビットと書き込みデータバス 32 ビットは共用する．Kintex-7 FPGA はコントローラ側からのアクセスリクエスト信号を AXI4-Lite に変換するモジュールを持つ．評価対象の暗号処理モジュールは AXI インターコネクトを介して接続される想定である．Vivado Design Suite のブロックデザインによる設計が開始できるように，テンプレート生成 TCL スクリプトを用意している．オプションで MIG(Memory Interface Generator) を用いた DDR3 メモリコントローラを含むテンプレートの生成もサポートする．

コントローラ部を制御するソフトウェア実装は，ChipWhisperer の `TargetTemplate` クラスを継承したクラスを提供している．ゆえに，ChipWhisperer の波形取得 API にそのまま利用することができる．このクラスをさらに継承して，鍵，平文，暗号文の読み書きを行う 3 つのメソッドと，暗号処理を開始するメソッドを継承するだけで独自の暗号処理モジュールが利用可能になる．例えば，4. 節で評価に用いる RTL 実装モジュールの制御用コードは 30 行で実装できる．

4. 評　　価

4.1 CPA 高速化の結果

SAKURA-X のリファレンスデザイン [15] を用いて取得したトレースデータを用いて提案実装の高速化を評価する．Keysight Infiniviion MSO-X 4104A 5GS/s のオシロスコープを用いて，暗号処理の全区間 (4us, 20000 サンプル) の波形を取得し合計 20000 トレースを保存した．表 2 に示す 5 つの異なる実行環境で，CPA における相関係数の計算に要した時間を図 4 に示す．ChipWhisperer の Python 実装に加えて，マルチスレッド

表 2: 評価したシステム

System	CPU	OS	RAM	Compiler	GPU	GPU Toolkit/Runtime
No. 1	Ryzen 7995X	Linux Kernel 5.14.0-284	128 GB	GCC 11.3.1	NVIDIA RTX 4070	CUDA 12.2
No. 2	Intel Core i9-14900KF	Linux Kernel 5.14.0-362	128 GB	GCC 11.4.1	Radeon RX 7900 XTX	ROCm 6.0.1
No. 3	Ryzen7 5700G	Linux Kernel 6.2.0-39	64 GB	GCC 11.4.0	Intel Arc A770	Intel graphics compute runtime 23.22.26516.34
No. 4	Apple M1 Ultra 20 Cores	macOS 12.6.4	128 GB	Clang 14.0.0	38-core GPU	Xcode 14.2
No. 5	Apple M2 Max 12 Cores	macOS 14.4.1	96 GB	Clang 14.0.3	64-core GPU	Xcode 15.2

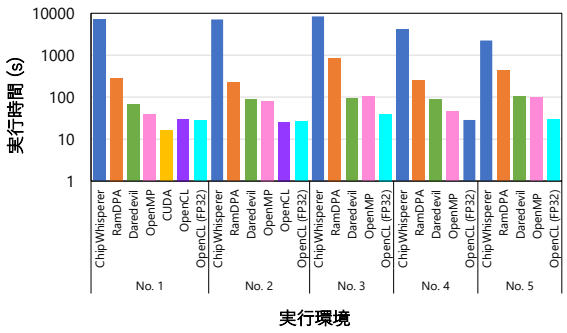


図 4: CPA 攻撃に要する時間の比較

表 3: 各設計の回路規模

	コントローラ (Spartan-6 xc6slx45)			
	LUT	FF	MUXCY	BRAM
[15]	206 (0.377%)	179 (0.656%)	36 (0.264%)	0
提案	850 (1.42%)	775 (3.11%)	60 (0.440%)	0
	AES 回路 (Kintex-7 xc7k160t)			
	LUT	FF	DSP	BRAM
[15]	952 (0.938%)	3137 (1.55%)	0	0
コアなし	141 (0.139%)	594 (0.292%)	0	0
RTL	2,152 (2.12%)	1,973 (0.973%)	0	0
HLS	4,785 (4.72%)	4,392 (2.17%)	0	6 (1.85%)

実装である RamDPA [6] と DareDevil [7] の実行時間も計測を行い比較している．本研究の実装は OpenMP,CUDA, および OpenCL であり, 計測時間には Python による呼び出しなどの時間も含まれる．AppleSiliocn の CPU の環境である System No. 4 と No. 5 における ChipWhisperer の実装は, 前述の通り精度が他のものと異なることに注意されたい．

ChipWhisperer の実装が 1 時間から 2 時間ほどの時間を要するのに対し, 提案している OpenMP 実装は最大で 185 倍 (No. 1) の高速化を達成した．また, 実装方法が類似する DareDevil と比較して, No. 3 を除きわずかに短い時間で完了し, 最大で 1.7 倍の高速化を達成している (No. 1)．この結果から, ChipWhisperer のプラグインとして実装するのに必要なラップ処理などのオーバーヘッドは極めて小さいことがわかる．一方,DareDevil は前述の通り最適化された計算方法ではないため, OpenMP 実装や DareDevil より 2 倍から 8 倍ほど遅い結果となっている．GPU による高速化の効果として, OpenMP による並列実行からさらに 2 倍から 3 倍程度の高速化を達成している．NVIDIA 製 GPU 向け CUDA 実装が最も高速で約 15 秒で処理が完了した．これは ChipWhisperer の実装に対して約 450 倍の高速化を達成している．

4.2 設計テンプレートを用いた評価

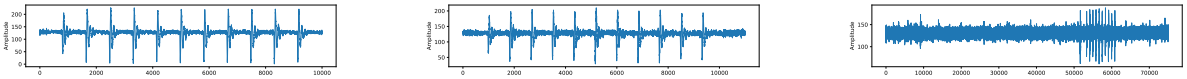
次に, 設計テンプレートが相関係数解析に与える影響を確認する．評価には 1) 前述のリファレンスデザイン, 2) 設計テンプレート +RTL 記述の AES 暗号化コア, 3) 設計テンプレート +HLS による AES 暗号化コアをそれぞれ用いて, 波形を取得する．使用したオシロスコープは 4.1 節と同じである．ただし, ボード上のトレースポイント J19 からインピーダンス 50Ω の同軸ケーブルと Langer 社のプリアンプ PA303(ゲイン 30dB) を介してオシロスコープに接続した．条件を統一するために, 暗号化回路の動作周波数は 6MHz とした．一方で, テンプレートにおけるコントローラ部および AXI 変換部の動作周波数は 100MHz とした．コントローラ部の設計とリファレンスデザインは ISE 14.7 を使用してビットストリームを作成した．一方で, 設計テンプレートを用いた Kintex-7 側の設計には Vivado 2023.2, 高位合成には Vitis HLS 2023.2 を使用した．

AES において回路の大部分を占める S-Box の実装には LUT 実装よりも省面積であると知られる合成体による実装を用いた．設計テンプレートとともに用いられる RTL 記述はリファレンスデザインと同じコアを AXI4 Lite のスレーブインターフェースを持つようにラップしたものである．暗号化コアは AES の処理の各ラウンドを 1 サイクルで各バイトごとに並列計算するように設計されている．一方で, HLS による実装では, 鍵, 平文, 暗号文の入出力ポートに m_axi モードを使用して設計した．これらのデータは Block Memory Generator で生成したメモリを AXI BRAM Controller を介してアクセスするようにブロックデザインを行なった．RTL 設計と同様に, バイト方向のループはアンローリングするようにプラグマを設定した．

表 3 に各設計の回路規模を示す．テンプレートにおけるコントローラ部は機能が増えたため, リファレンスデザインに比べて LUT と FF の使用率が増加しているが, FPGA 全体の規模に対しては十分な余裕がある．暗号処理モジュール部については, テンプレート自体が消費する最低限のリソースを確認するために, AES コアを含まない設計も示している．表に示す通り LUT と FF の使用率はいずれも 1% 未満であり, リソース消費の観点で与える影響は極めて小さい．また, 同等の計算回路を実装しているがインターフェースの違いや高位合成のオーバーヘッドにより LUT と FF の使用率は HLS 設計が RTL 設計に比べて 2 倍程度大きくなっている．一方で, RTL 設計とリファレンスデザインは同じ暗号処理コアを用いているが, 設計ツールが ISE と Vivado で異なるため, 各々使用するリソース量が異なっていると考えられる．

これらの回路を用いて取得した波形を図 5 に示す．テンプレートには 100MHz で動作するインターフェース回路などが含まれるため, リファレンスデザインと比べノイズが含まれていることがわかる．また, HLS 設計では暗号処理中であることを外部に示すためのトリガー信号を出力するのが容易ではなかったため, ブロックメモリへのアクセス信号をトリガー信号として波形を取得している．そのため, 他の 2 つ比べ取得された波形区間が長い．しかし, いずれの設計も AES の各ラウンドに対応する 11 ヶ所のピークが観測できる．

この取得された波形に対して最終ラウンドの S-Box 入力と



(a) リファレンスデザイン　(b) RTL 設計　(c) HLS 設計

図 5: 取得した波形の比較

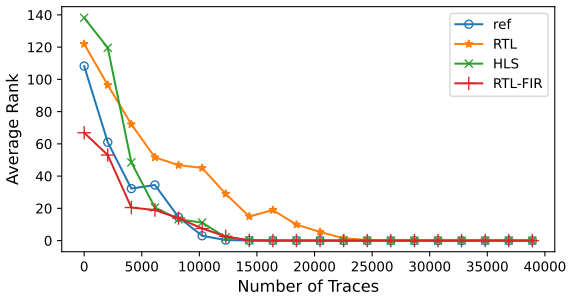


図 6: 解析結果の比較

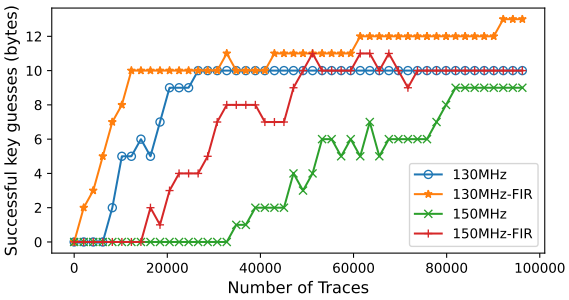


図 7: 自動パイプライン化による影響

出力のハミング距離をリークモデルとして解析を行なった結果が図 6 である．部分鍵ごとに正解鍵の相関係数が何番目に高いかをランクとし (ランク 0 は正しく鍵が推定されたことを意味する), 16 バイトの各ランクを平均したものを示している．リファレンスデザインと HLS 設計はほぼ同じ結果を示しているが, RTL 設計では推定に必要なトレース数が若干多い．そこで, インターフェース部の周波数である 100MHz をカットオフ周波数として FIR フィルタを適用した波形に対して解析を行なった結果も示す．結果から分かる通り, いずれのケースも 4000 トレース程度ですべての部分鍵を正しく推定できている．以上のことから, 本提案の設計テンプレートが解析容易性に大きく影響しないと言える．

4.3 HLS による最適化の影響

高位合成を用いる利点としてターゲットの周波数に応じて自動でパイプライン化などを行う強力な最適化が挙げられる．最後に, 高いターゲット周波数としたときに行われる最適化が AES 回路のサイドチャネル耐性に与える影響を評価する．ターゲット周波数を 10MHz から 150MHz まで 10MHz 刻みで変化させ, 生成される RTL 記述の変化を確認した．その結果, 60MHz で 1 ラウンドを 2 サイクルで, 120MHz で 1 ラウンドを 3 サイクルで, 150MHz で 1 ラウンドを 5 サイクルで処理するようにパイプライン化された．そこで, 120MHz と 150MHz の設計を各々その周波数で動作させた時の電力波形を 100,000 トレース取得して解析を行なった．

解析の結果正しく鍵を推定できたバイト数と解析に使用したトレース数の関係を図 7 に示す．いずれのケースも FIR フィルタの適用の有無によって差は見られるものの, 16 バイトの鍵すべてを特定することはできなかった．パイプライン化によって内部のレジスタが増加したことにより, 情報がリークする瞬間に別のレジスタで消費する電力も増えたことが原因であると考えられる．また, 動作周波数が図 6 の評価と比べ高いため, 測定装置の限界によるものなのかどうかは今後明らかにする必要がある．しかし, 16 バイト中半分以上の部分鍵が推定されてしまっており, 依然としてサイドチャネル攻撃に対して脆弱であることがわかる．また, 細粒度にパイプライン化されたことにより, また別のリークモデルが有効である可能性もある．

5. おわりに

本研究では, 暗号処理モジュールのサイドチャネル耐性を容易に評価するためのハードウェアおよびソフトウェアを実装した．ハードウェアとしては, AXI インターフェースを持つ暗号処理モジュールを接続可能な設計テンプレートを提供し, ソフトウェアとしては相関係数解析を高速化実装や, 設計テンプレートと通信するためのドライバを ChipWhisperer のプラグインとして実装した．評価の結果, 本フレームワークの有効性を確認した．

謝　辞

本研究は科学技術振興機構戦略的研究推進事業 (JST) さきがけ JPMJPR22P5 の支援を受けたものである．

文　献

[1] S. Mangard, E. Oswald, and T. Popp, Power analysis attacks: Revealing the secrets of smart cards, vol.31, Springer Science & Business Media, 2008.
[2] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” Cryptographic Hardware and Embedded Systems-CHES 2004: 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings 6, pp.16–29, Springer, 2004.
[3] S. Chari, J.R. Rao, and P. Rohatgi, “Template attacks,” Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 13–15, 2002 Revised Papers 4, pp.13–28, Springer, 2003.
[4] K. Papagiannopoulos, O. Glamočanin, M. Azouaoui, D. Ros, F. Regazzoni, and M. Stojilović, “The side-channel metrics cheat sheet,” ACM Computing Surveys, vol.55, no.10, pp.1–38, 2023.
[5] ColinO’ flynn, Z. Chen, “Chipwhisperer: An open-source platform for hardware embedded security research,” Constructive Side-Channel Analysis and Secure Design: 5th International Workshop, COSADE 2014, Paris, France, April 13-15, 2014. Revised Selected Papers 5Springer, pp.243–260 2014.
[6] A.F. Rodríguez, L.H. Encinas, A.M. Muñoz, and B.A. Alcázar, “A modular and optimized toolbox for side-channel analysis,” IEEE Access, vol.7, pp.21889–21903, 2019.
[7] P. Bottinelli and J.W. Bos, “Computational aspects of correlation power analysis,” Journal of Cryptographic Engineering, vol.7, pp.167–181, 2017.
[8] T. Swamy, N. Shah, P. Luo, Y. Fei, and D. Kaeli, “Scalable and efficient implementation of correlation power analysis using graphics processing units (GPUs),” Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy, pp.1–8, 2014.
[9] H. Gamaarachchi, R. Ragel, and D. Jayasinghe, “Accelerating correlation power analysis using graphics processing units (gpus),” 7th International Conference on Information and Automation for SustainabilityIEEE, pp.1–6 2014.
[10] E. Ferrufino, L. Beckwith, A. Abdulgadir, and J.-P. Kaps, “Fobos 3: An open-source platform for side-channel analysis and benchmarking,” Proceedings of the 2023 Workshop on Attacks and Solutions in Hardware Security, pp.5–14, 2023.
[11] Q.L. Meunier, “FastCPA: Efficient correlation power analysis computation with a large number of traces,” Proceedings of the Sixth Workshop on Cryptography and Security in Computing Systems, pp.7–12, 2019.
[12] Y. Nomata, M. Matsubayashi, K. Sawada, and A. Satoh, “Comparison of side-channel attack on cryptographic cirucits between old and new technology fpgas,” 2016 IEEE 5th Global Conference on Consumer ElectronicsIEEE, pp.1–4 2016.
[13] A. Thall, “Extended-precision floating-point numbers for GPU computation,” ACM SIGGRAPH 2006 research posters, pp.52–es ●●, , 2006.
[14] C. Kenyon and C. Capano, “Apple silicon performance in scientific computing,” 2022 IEEE High Performance Extreme Computing Conference (HPEC)IEEE, pp.1–10 2022.
[15] “SAKURA (SASEBO-GIII),” https://satoh.cs.uec.ac.jp/SAKURA/hardware/SAKURA-X.html. Accessed on 2024-05-13.