Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5

Cortex-M0 プロセッサ上の暗号ソフトウェアに対する多重故障注入攻撃の検討

遠藤翔[†1] 梨本翔永[†1] 本間尚文[†1] 林優一[†1] 高橋順子[†2] 冨士仁[†2] 青木孝文[†1]

本稿では，Cortex-M0 コアを搭載した組込みマイコンに対する多重故障注入攻撃について述べる．組込みマイコンにおける暗号ソフトウェアでは，故障注入を複数回行い故障注入攻撃への検算型対策を回避する多重故障注入攻撃が問題となっており，現在までに，攻撃者がプログラムを知らない条件でも攻撃できる手法が知られている．上記手法では，適切な故障注入タイミングを探索することにより，攻撃者は誤った暗号文を得るために必要な故障と，対策を回避するための故障の注入タイミングを知ることができる．しかし，上記の探索手法では，命令の種類により実行サイクル数が変化する組込みマイコンにおいて，分岐命令の実行タイミングを知ることができないことから，上記マイコンにおける脅威は未知であった．本稿では，暗号ハードウェアの実行時間に基づく探索手法を提案し，上記マイコンにおいても多重故障注入を適用可能とすることにより，上記マイコンでも攻撃の脅威があることを指摘する．さらに，上記マイコンの中で特に広く用いられている Cortex-M0 コアを用いて実験を行う．AES プログラムに対して提案手法により故障注入タイミングを探索する実験を通して，提案攻撃の有効性を示す．

A Multiple-fault Injection Attack Against Cryptographic Software on Cortex-M0 Processor

SHO ENDO[†1] SHOEI NASHIMOTO[†1] NAOFUMI HOMMA[†1]
JUNKO TAKAHASHI[†2] HITOSHI FUJI[†2] TAKAFUMI AOKI[†1]

This paper presents a multiple-fault injection attack against the cryptographic software running on Cortex-M0 processor core. Multiple-fault injection attacks have been a problem on software running on embedded microcontroller equipped with countermeasures based on recalculation. The attacks can circumvent the countermeasure through an adaptive fault injection even when cryptographic program is unknown to the attackers. However, the possibility of the attack described above is unknown on the microcontrollers which have instructions with different execution cycles because the attackers cannot identify the execution timing of branch instruction. This paper shows a scanning method against the cryptographic software, which can be applied to the microcontroller described above. Validity of proposed attack is demonstrated through the experiment using AES program running on the Cortex-M0 core, which have been widely used for cryptographic modules.

1. まえがき

近年，暗号システムに対する故障注入攻撃が問題となっている．故障注入攻撃は，暗号システムに意図的に故障を注入し，秘密情報を抽出する攻撃である．現在までに，公開鍵暗号に対する Boneh らの攻撃[5]や，共通鍵暗号に対する差分故障解析(Differential Fault Analysis: DFA) [4][13] などの攻撃手法が提案されている． Boneh らの攻撃では，誤った暗号文をもとに素因数分解を行う．DFA では，正しい暗号文と誤りを含む暗号文（以下，誤り暗号文）の差分を取ることにより秘密鍵を絞り込む．このように，故障注入攻撃では通常誤り暗号文が必要となる．そこで，暗号文に含まれる誤りを検知して，誤り暗号文の出力を停止する対策がしばしば取られている．故障検知の手法としては検算処理が一般的である．検算型の対策には，同じ暗号化処理を複数回実行して出力を比較するもの[2][15]や，暗号化後に復号やチェックサムの計算を行って出力と入力（もしくは入力から得られる値）を比較するもの[2][6]がある．

一般に，検算型対策では，故障モデルとして 1 回の暗号化（復号）処理に対して 1 回の故障注入しか想定していない場合が多い．このため，暗号化処理部と検算部にそれぞれ故障を注入することにより，検算の動作を回避して誤り暗号文を出力できる可能性がある．以下では，この攻撃を多重故障注入攻撃と呼ぶ．実際に多重故障注入攻撃の実験が行われており，誤り暗号文が得られたとする結果が報告されている[16][17].

従来の多重故障注入攻撃では，攻撃者が対象となるプログラムの詳細を知っていることを前提としているものの他に，適応的にタイミングを制御した故障注入により実行中の暗号プログラムの詳細を知らないという条件下においても適用可能な多重故障注入攻撃が提案されている[10]．この攻撃では，故障注入を行うタイミングをクロックサイク

†1 東北大学
Tohoku University
†2 NTT セキュアプラットフォーム研究所
NTT Secure Platform Laboratories

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5

ル単位で変化させることにより，誤った暗号文を出力させるための故障注入のタイミングを探索する．探索はいくつかのステップで行い，各ステップにおいて故障が所望のタイミングで注入されるまで故障注入の試行を繰り返す．このような探索では，命令単位の探索を可能とするため，特定のクロックサイクルに正確に故障を注入できるクロックグリッチを故障注入に用いている．しかしながら，文献[10]の攻撃では，命令ごとに実行サイクル数の異なるプロセッサにおいて探索に失敗する場合があるという問題があった．具体的には，攻撃に利用できる暗号文を得るための探索（文献[10]のステップ III）において，暗号文に誤りを引き起こすための故障（故障 B）を注入する命令を変更すると同時に分岐命令の実行タイミングも変化することにより，故障 B が分岐命令とは異なるタイミングで注入され，結果としてエラー信号が出力される．

そこで本稿では，実行時間に応じて故障 B の注入タイミングを変化させることにより，上記プロセッサ上に実装された検算型対策においても適応的な故障注入を可能とする手法を提案する．また，現在広く用いられている Cortex-M0 プロセッサ上で検算型の故障注入攻撃対策を施した AES ソフトウェアを用いた実験を行い，提案する攻撃により対策を無効化しつつ差分故障解析に必要な誤り暗号文が得られることを示す．

本稿の構成を示す．第 2 章では関連研究について述べる．第 3 章では，文献[10]で示されている攻撃の詳細とその問題点を述べ，これを解決した探索アルゴリズムを提案する．第 4 章では，提案攻撃を Cortex-M0 プロセッサ上で実行される暗号ソフトウェアに適用し，対策を回避して誤った暗号文が取得できることを示す．第 5 章ではまとめと今後の課題を述べる．

2. 関連研究

2.1 差分故障解析(Differential Fault Analysis: DFA)

差分故障解析(Differential Fault Analysis: DFA)は，正しい暗号文と誤った暗号文の差分を元に，秘密鍵の候補を絞り込む解析手法である．ここでは AES に対する DFA[13]について述べる．まず，攻撃者は攻撃対象のデバイスを用いて正しい暗号文 C と誤りを含む暗号文 C^*を計算し，これらを用いて鍵候補を絞り込む．C^*の計算の際，AES の 9 ラウンド目の入力 I_9 のうち 1 バイトに誤りが発生すると仮定する．以降，C と C^*の組を暗号文ペアと呼ぶ．図 1 に，AES の 9 ラウンド目以降のデータの流れを示す．ここでは，10 ラウンド目の入力を上流と下流の両方から計算し，両者の取りうる範囲が異なることを利用して鍵候補を絞り込む．ここでは，図 1 において，最も左側の MixColumns に入力される 4 バイトの値のうちいずれかに故障が注入されたと仮定し，その 4 バイトに注目する．

ここで，ラウンド入力の i バイト目を $I_n(i)$とする．誤りがないときおよび誤りを含む 9 ラウンド目の入力をそれぞれ I_9, I_9^*とする．$\oplus$は排他的論理和である．

$$\Delta_9 = I_9 \oplus I_9^* \tag{1}$$

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

次に，10 ラウンド目の入力における正しい暗号文と誤った暗号文との差分 I_{10} を求める．MixColumns は線形演算であることから式(2)が成り立つ．なお，9 ラウンド目のラウンド鍵は，正しい暗号文と誤った暗号文の差分には影響しないことから式中には含まれない．

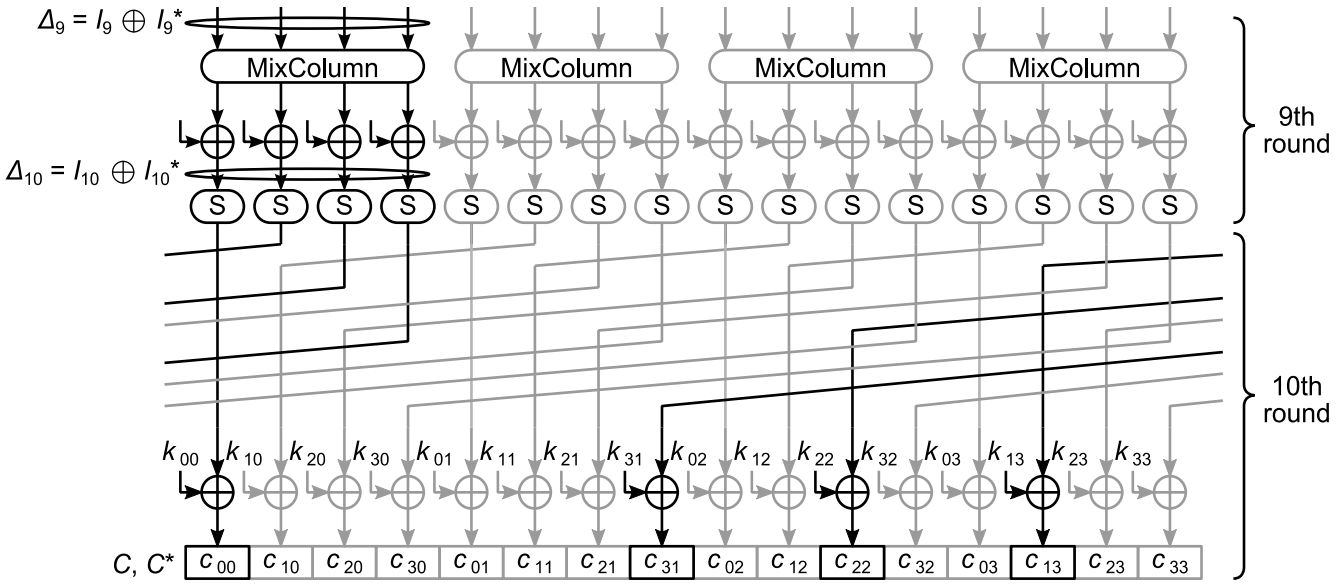


図 1 差分故障解析の流れ

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5

$$\begin{aligned}\Delta_{10} &= MixColumns(I_9) \oplus MixColumns(I_9^*) \\ &= MixColumns(I_9 \oplus I_9^*) \\ &= MixColumns(\Delta_9)\end{aligned} \quad (2)$$

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

続く SubBytes の演算は非線形演算であることから計算を続けることができない．そこで，暗号文から 10 ラウンド目の入力を逆算する．ここでは，暗号文 C と推定鍵 k' を用いて逆算した 10 ラウンド目の入力を $J_{10}(C, k')$と表記し，$J_{10}(C, k')$は式(3)のように求められる．

$$J_{10}(C, k') = InvSubBytes(InvShiftRows(C \oplus k')) \quad (3)$$

これを用いると，10 ラウンド目の入力における差分は式(4)で表される．

$$\Delta J_{10}(k') = J_{10}(C, k') \oplus J_{10}(C*, k') \quad (4)$$

DFA では，全ての鍵候補 ($0 \leqq k' \leqq 2^{32}$) に対し，式(2)の計算を行い，$\Delta J_{10}(k')$が Δ_{10} の取りうる値に含まれるかどうかを判定する．含まれる場合，k'を鍵候補に追加する．Δ_{10} は 2^{10} 個であることから，鍵候補を 2^{10} 個に絞り込むことができる．ただし，ここで得られる鍵候補は 2^{10} 通りの部分鍵であり，鍵全体を復元するにはさらなる絞り込みが必要である．平文を変更して暗号文ペアを取得し，上記の手順を複数回繰り返すことにより，鍵候補を 1 つに絞り込むことができる．なお，この攻撃は，1 つの暗号文ペアに対して $O(2^{32})$の計算量を要する．

2.2 関連研究

組込みマイコンでは，実行中の命令を故障注入によりスキップできることが知られており，故障注入攻撃における故障注入の手段として検討されている[1,17,8,12,13]．命令スキップこれは電源電圧の一時的な低下[16]，電磁パルス[8,12], チップ表面へのレーザー照射[2,6]により引き起こすことができる．命令スキップの故障モデルは，メインメモリ内のプログラムは改変せずに，命令がメインメモリからプロセッサ内にロードされる時に故障注入を行うことから，メモリ内のプログラムの改変に対する対策では防ぐことが困難である．攻撃者は，暗号処理に関わる特定の命令をスキップさせることで暗号処理に誤りを発生させ，DFA などの攻撃を実施する．

上記の故障注入攻撃へのハードウェア的な対策としては，故障を検知した際に暗号処理を停止する回路が用いられている[2]．一方組込みプロセッサにおいては故障検知用のハードウェアを追加できない場合が多いため，ソフトウェアによる対策が検討されている． 典型的な対策として，検算処理を用いて誤りを検知し，暗号文の出力を停止する対策が挙げられる．検算型対策は，複数回同一の暗号化処理を行うもの[2][3]と，暗号化処理と故障検知のための異なる演算を行うもの[15][6]とに分類できる．異なる演算を用いる手法としては，暗号化を行った後に復号を行うもの

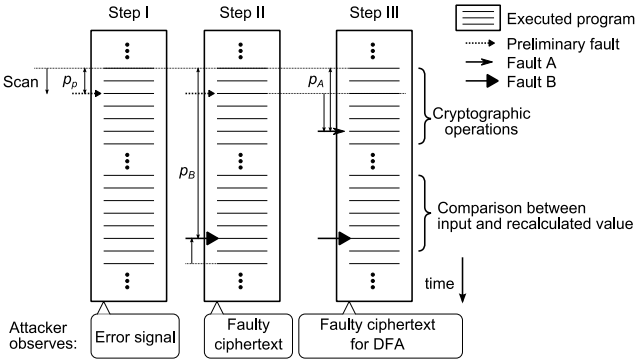


図 2 文献[10]の攻撃の基本コンセプト

[15]や，AN+B 符号と呼ばれる符号を用いる手法[11]などが挙げられる． 同一の演算を用いる手法では，多重故障注入により全ての暗号演算に同じ故障を注入することにより無効化される問題点がある．一方，異なる演算を用いる手法では，暗号化処理に注入した故障と同一の故障を検算用の処理に注入することが困難であることから，上記の問題点は解消される．しかしながら，異なる演算を用いる手法においても，故障を検知して条件分岐を行うコードそのものをスキップする攻撃に対しては脆弱である．

検算処理の単位についても複数の手法が存在し，暗号化処理が完結してから検算用演算を行うものや[2]，命令ごとに検算するものがある[3]．命令を複製する対策[3]では，複製する個数より少ない回数の故障注入による誤りの発生を防止できるが，それ以上の回数の故障注入により無効化できる．

検算型対策とは異なるアプローチとして，プログラムにランダムな遅延を挿入する対策も報告されている[7]．この対策では，暗号処理の前後にランダムな遅延を挿入することにより，命令の実行タイミングが毎回変化する．これにより，故障注入のタイミングが一定の攻撃を防ぐことができる．

3. 適応的にタイミングを制御した多重故障注入攻撃

3.1 適切な故障注入タイミングの探索手法

ここでは，文献[10]に示されている，適切な故障注入タイミングの探索手法について述べる．図 2 にこの攻撃の基本コンセプトを示す．ここでは，暗号化の後に検算を行い，故障を検知しない場合に暗号文，故障を検知した場合にエラー信号を返すプログラムを想定する．ステップ I では，故障 A と B の注入タイミングを探索するための予備的な処理として，対策を動作させるために必要な故障（予備故障）の注入タイミングを探索する．探索はプログラムの開始（基準信号以降）から順に命令を一つずつスキップさせながら行う．まず，暗号文に何らかの故障を注入して対策を動作させる．プログラムの最初から探索を開始し，エラ

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5

Algorithm 1 Encryption with Recalculation (EncWithRecal)

Input: Plaintext P
Output: Ciphertext C

```
1: C ← Encryption(P)
2: P_2 ← Decryption(C)
3: if P = P_2 then
4:     return C
5: end if
6: return Error signal
```

Algorithm 2 Activating the countermeasure

Input: Key k, plaintext P, maximum fault injection position p_{max}
Output: Correct ciphertext C, position of preliminary fault p_p

```
1: C ← EncWithErrorDetection(P) # Correct ciphertext
2: for p_p = 0 to p_max do
3:     SetPreliminaryFaultPosition(p_p)
4:     C_f ← EncWithErrorDetection(P)
5:     if Error signal is observed then
6:         return p_p
7:     end if
8: end for
```

Algorithm 3 Injecting a fault to circumvent the countermeasure

Input: Random plaintext P, correct ciphertext C, maximum glitch position p_{max}, position of preliminary fault p_p
Output: Position of Fault B p_B

```
1: SetPreliminaryFaultPosition(p_p)
2: for p_B = p_max to p_p do
3:     SetFaultBPosition(p_B)
4:     C_f ← EncWithErrorDetection(P)
5:     if C_f ≠ C and Error signal is not observed then
6:         return p_B
7:     end if
8: end for
```

Algorithm 4 Obtaining an faulty ciphertext for attacks

Input: Random plaintext P, correct ciphertext C, maximum glitch position p_{max}, position of preliminary fault p_p
Output: Position of Fault B p_B, faulty ciphertext for attacks C_f

```
1: SetFaultBPosition(p_B)
2: for p_A = p_p to p_max do # Glitch position
3:     SetFaultAPosition(p_A)
4:     C_f ← EncWithErrorDetection(P)
5:     d ← C_f ⊕ C
6:     if d has four non-zero bytes then
7:         return C_f
8:     end if
9: end for
```

一信号が観測されるとステップ I を終了する．ステップ II では，1 回の暗号処理の間に複数回故障を注入し，対策を回避する．ステップ I で得られたタイミングで予備故障を注入すると同時に，それ以降のタイミングで故障 B を注入し，故障 B のタイミングをプログラムの最後から探索する．対策の無効化に成功したことは誤り暗号文が得られることから判定できる．ステップ III では，対策を無効化した状態で，DFA 等の攻撃に利用可能な故障 A の注入タイミングを探索する．

以下では，Algorithm 1 に示す検算型対策における具体的な探索手法を述べる．以降，基準信号から故障注入の実行までの時間を故障位置と呼ぶ．Algorithm 1 は，暗号化の後に同じ鍵を用いて復号し，復号結果が入力と等しい場合に正常とする． ステップ I, II, III はそれぞれ Algorithm 2, 3, 4 で実現される．ここで，最大の注入位置 p_{max} は 1 回の暗号処理に要するクロックサイクル数とする． まず，Algorithm 2 を実行し，対策を動作させるための予備故障の注入タイミングを求める．故障注入の位置 p_p は $0 \leqq p_p \leqq p_{max}$ の範囲となる．ここではプログラムの先頭($p_p = 0$)より探索を開始する．プロセッサがエラー信号を出力したとき Algorithm2 は完了となる．

次に，Algorithm 3 を用いて，故障 B の注入タイミングを決定する．ここでは，Algorithm 2 で求めた位置 p_p に予備故障を注入しつつ，プログラムの末尾より故障 B の注入タイミング p_B を探索する．故障 B を適切な位置で注入すると，条件分岐に故障が注入され，誤り暗号文が得られる．

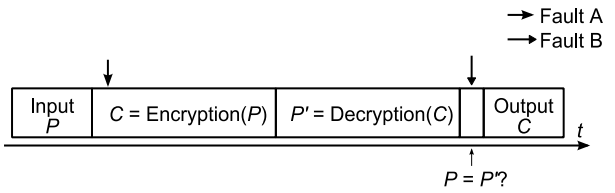
最後に，Algorithm 4 を用いて，DFA に用いることができる誤り暗号文を得るため，故障 A の注入タイミング p_A を探索する． 同アルゴリズムは DFA 等の攻撃に利用できる誤り暗号文が得られた時点で完了となる. Algorithm4 では，例として AES に対する Piret らの DFA[3]を適用する場合に必要な誤り暗号文を得た時に同アルゴリズムの完了としている．この攻撃では，AES の 9 ラウンド目の入力に 1 バイトの故障を注入することから，暗号文は 4 バイトの誤りを含む．従って，4 バイトの誤りを含む暗号文が得られたとき，攻撃が成功したといえる．以上の手順により，検算型対策を回避しつつ DFA 等の攻撃を行うことができる．

上記の探索アルゴリズムは，故障 A の位置を変化させたときに分岐命令の位置が変化することを想定していないことから，命令ごとにサイクル数が異なるプロセッサで探索に失敗する問題がある．図 3(a)にステップ II 完了時の故障位置を示す. 攻撃に使用できる暗号文の探索を開始すると，故障 A の位置を変化させた時にプログラムの実行時間が変化してしまい，それに伴い図 3(b)のように分岐命令のタイミングも変化する．この時故障 B の位置はプログラム開始からの位置で決定されるため，分岐命令から外れてしまい，出力はエラー信号となる．

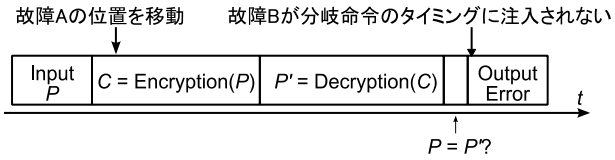
3.2 提案手法

本稿では，故障 B のタイミングを分岐命令に追従させることにより，命令ごとに実行サイクル数が異なるプロセッサにおいても探索を可能とする手法を提案する．提案手法

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5



(a) ステップ II の終了時の状態



(b) 故障 A を移動した状態

図 3 文献[10]の攻撃手法における問題点

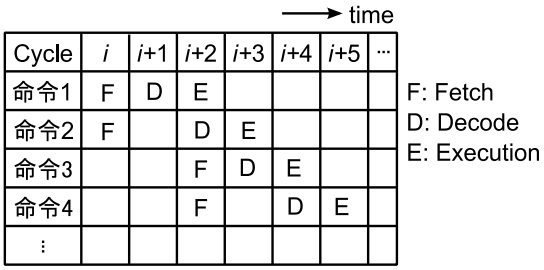


図 4 Cortex-M0 プロセッサのパイプライン

では文献[10]の手法と同様に 3 つのステップで探索を行う．ステップ I，II は従来手法と同様であり，ステップ III において故障 B の位置を分岐命令の実行タイミングへ適応させる．Algorithm 5 に，提案攻撃におけるステップ III のアルゴリズムを示す．ステップ III において故障 A の位置を変更すると，(a)プログラムの実行時間（サイクル数 c_p），(b)分岐命令の実行からプログラム終了までの時間（サイクル数 c_b）の 2 つのパラメータが変化するため，提案手法はこれらに適応するアルゴリズムとする．これらの変化が発生する原因は，(a)はスキップされる命令のサイクル数が変化するため，(b)はパイプラインの状態が変化するためと考えられる．(a)プログラムの実行時間については，故障 A の位置を変更するごとに測定して取得する．(b)分岐命令の実行からプログラム終了までの時間は，直接測定することができないため探索により調べる．

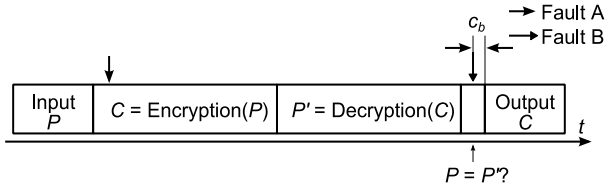
さらに，プロセッサのアーキテクチャによっては，(b)のサイクル数の変化が発生する．例えば，Cortex-M0 のアーキテクチャでは，命令フェッチサイクルの仕様に起因して(b)が発生する．図 4 に Cortex-M0 プロセッサのパイプラインの模式図を示す．図 4 では命令 1 から 4 は全てシングルサイクルの命令であるとする．Cortex-M0 プロセッサでは，フェッチは 2 命令同時に行われることから，パイプラインの状態に依存してフェッチから実行までのサイクル数が変化する．従って分岐命令からプログラム終了までのサイクル数もパイプラインの状態に応じて変化することが予想される．ここでは分岐命令の位置を数サイクルに渡り探索する．ここではサイクル数を c_o とする．

Algorithm 5 Obtaining an faulty ciphertext for DFA

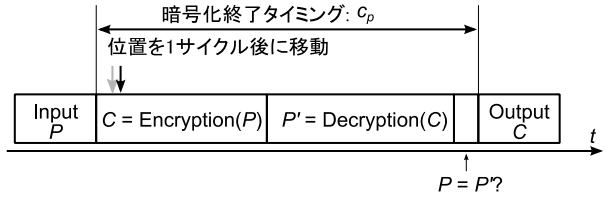
Input: Random plaintext P, correct ciphertext C, maximum glitch position p_{max}, position of preliminary fault p_p, position of Fault B p_B, execution cycles of program c_p

Output: Faulty ciphertext for DFA C_f

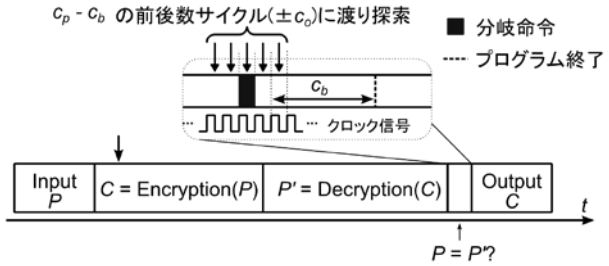
```
c_b = c_p − p_B
for p_A = p_p to p_max do
    SetFaultAPosition1(p_A)
    c_p = CycleCount(Encryption(P)) # for measurement of calculation time
    for c_o = o_max downto −o_max do
        SetFaultBPosition(c_p − c_b + c_o)
        C_f ← Encryption(P)
        d ← C_f ⊕ C
        if d = 0 then
            Continue
        end if
        if d has four non-zero bytes then
            return C_f
        end if
    end for
end for
```



(a) ステップ II の終了時の状態



(b) 位置変更と時間の測定

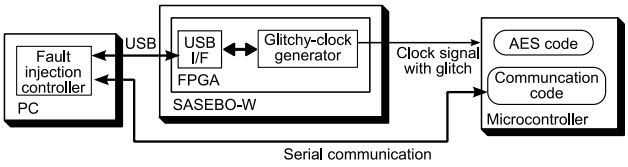


(c) 故障 B の位置の探索

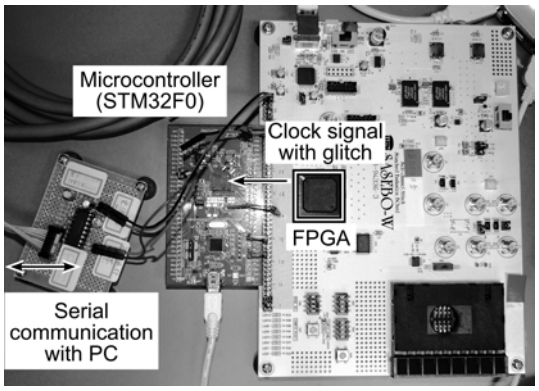
図 5 Algorithm 5 の手順のイメージ図

Algorithm 5 に，提案攻撃におけるステップ III のアルゴリズムを示す．また，図 5 に Algorithm 5 の手順のイメージ

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5



(a) ブロック図



(b) 外観

図 6 実験環境

図を示す．Algorithm 5 において，c_o は故障 B の位置を微調整するためのパラメータ，o_{max} はサイクル数である．Algorithm5 は，文献[10]のステップ III と同様に故障 A の位置 p_A を探索するが，故障 B の位置 p_B も同時に微調整し，故障 B が分岐命令から外れることを防止する．3 行目の関数 CycleCount は実行時間をサイクル単位で返す関数である．ステップ II の終了後，Algorithm 5 の 1 行目および図 5(a)に示すように，最初に故障 B の位置からプログラム終了位置までの間隔 c_b を計算する．p_A を変更する度に実行時間を測定し，これに応じて故障 B を注入する．故障 B の位置は概ね(c_p - c_b)となるが，パイプラインの状態変化に対応するため，(c_p - c_b)から周辺の±o_{max} サイクルを探索する．ステップ II の終了後，次に，図 5(b)のように，故障 A の位置を 1 サイクル移動させ，故障 B は注入せずに実行時間 c_p を測定する．故障 B の位置は(c_p - c_b)となる．以降，図 5(b)と(c)を繰り返して探索を行う．

4. 実験

4.1 実験の概要

本章では，提案攻撃を Cortex-M0 プロセッサ上で実行される暗号ソフトウェア上で実行し，誤った暗号文が得られることを示す．図 6 に実験環境，表 1 に実験条件を示す．組込みマイコンボード STM32F0Discovery 上に，Algorithm 1 に示す検算型対策を施した AES ソフトウェアを実装した．暗号モジュールへの物理攻撃の標準的な評価環境として開発されたサイドチャネル攻撃標準評価ボード SASEBO-W[14]上の FPGA に実験システムを実装した．FPGA にはクロックグリッチ発生器[9]を実装し，クロックグリッチを含むクロック信号を IC カードへ供給する．表 1 に実験条件を示す．攻撃実験では，マイクロプロセッサ ATmega163 向けに gcc 4.8.3 でコンパイルしたバイナリを使用している．ここでは攻撃アルゴリズムとして，ステップ I，II，III（それぞれ Algorithm 2，3，5 で表される）の順に実行し，探索に要した故障注入の試行回数を示す．攻撃の終了条件は，Piret らの DFA を実行するために必要な 4 バイトの誤りを含む暗号文が得られた時とする．また，

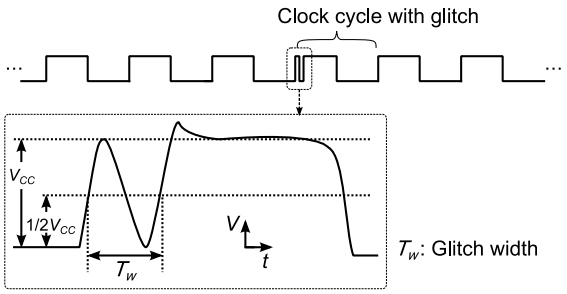


図 7 クロックグリッチの模式図

表 1 実験条件

暗号アルゴリズム	128 ビット AES
言語	C 言語（暗号演算用ライブラリは用いずに記述）
コンパイラ	gcc 4.8.3（コードサイズで最適化するオプションを設定）
組込みマイコンボード	STM32F0Discovery
FPGA	Xilinx XC6SLX150
平文	(00112233445566778899aabbccddeeff)$_{16}$
秘密鍵	(000102030405060708090a0b0c0d0e0f)$_{16}$

表 2 実験結果

Step	開始パラメータ	終了パラメータ	故障注入の試行回数
I	p_p = 0	p_p =19	20
II	p_B = 8050	p_B = 8046	5
III	p_A = 19 c_o = 2	p_A = 3097 c_o = 1	14163
合計			14188

表 3 Algorithm 5 における c_o のループの試行回数

探索終了時の c_o	2	1	0	-1	-2
試行回数	1	2	3	4	5
発生回数	6	98	388	138	2449

Vol.2015-DPS-162 No.15
Vol.2015-CSEC-68 No.15
2015/3/5

Algorithm 5 のパラメータとして，故障 B の探索範囲 o_{max} は 2 とした．すなわち，Algorithm5 では，探索範囲の中心から -2, -1, 0, 1, 2 の 5 箇所に故障注入を行う．

4.2 クロックグリッチのパラメータを決定する予備実験

クロックグリッチにより故障を注入する場合，クロックグリッチのパラメータにより発生する故障が変化するため，探索実験の前にパラメータを設定した．図 7 にクロックグリッチの模式図を示す．クロックグリッチには，基準となる信号からグリッチ発生までのサイクル数(故障位置)の他に，最初の立ち上がりエッジから 2 番目の立ち上がりエッジまでの時間を表すグリッチ幅 t_w がある．ここでは故障 A と故障 B のそれぞれで t_w を設定した．故障 A は様々な命令へ故障注入を行うため，故障 A の位置 p_A を 0 から 100 まで変化させて，故障注入に成功した時の t_w の中央値として 6.9ns と設定した．故障 B の t_w は，分岐命令に故障を注入した際に，故障注入に成功した時の t_w の中央値として 9.8ns と設定した．

4.3 実験結果

表 2 に，探索完了までに要した故障注入の試行回数を示す．合計で 14188 回の故障注入により 4 バイトの誤りを含む暗号文が出力され，攻撃は成功した．表 3 にステップ III の内訳を示す．ステップ III では，パイプラインの状態により，Algorithm 5 に示す 5 行目から 15 行目のループ（c_o のループ）の試行回数が変化する．表 3 では，1 行目に探索終了の時の c_o の値，2 行目に上記 c_o の値の場合の故障注入の試行回数，3 行目に上記 c_o の値が発生した回数を示す．合計の試行回数は（試行回数）×（試行回数の発生回数）の和となることから，ステップ III は合計で (6+98×2+388×3+138×4+2449×5) =14163 回の試行となった．この実験は 1 時間程度で実行できた．

5. むすび

本稿では，適応的にタイミングを制御した多重故障注入攻撃の Cortex-M0 プロセッサへの適用について述べた．文献[10]に示されている攻撃は，において，故障 A を注入する命令を変化させたときに探索に失敗するという問題点を解決することにより，Cortex-M0 プロセッサでの探索を可能とした．提案攻撃への対策としては，文献[10]に示されている命令配置の変更が有効と考えられる．今後の課題としては，提案攻撃を他の対策へ適用することや，文献[10]の対策の安全性を検証することが挙げられる．

謝辞 本研究は JSPS 特別研究員奨励費 24・7639 の助成を受けたものである．

参考文献

1) Balasch, J., Gierlichs, B. and Verbauwhede, I.: An Indepth and Black-box Characterization of the Effects of Clock Glitches on 8-bit MCUs, Proceedings of the 8th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Nara, Japan, pp. 105–114 (2011).
2) Bar-El, H., Choukri, H., Naccache, D., Tunstall, M. and Whelan, C.: The Sorcerer's Apprentice Guide to Fault Attacks, Proc. of the IEEE, Vol. 94, No. 2, pp. 370 –382 (2006).
3) Barenghi, A., Breveglieri, L., Koren, I., Pelosi, G. and Regazzoni, F.: Countermeasures against fault attacks on software implemented AES: effectiveness and cost, Proceedings of the 5th Workshop on Embedded Systems Security (WESS), New York, NY, USA, pp. 7:1–7:10 (2010).
4) Biham, E. and Shamir, A.: Differential fault analysis of secret key cryptosystems, Advances in Cryptology, CRYPTO '97, Lecture Notes in Computer Science, Vol. 1294, pp. 513–525 (1997).
5) Boneh, D., DeMillo, R. and Lipton, R.: On the Importance of Checking Cryptographic Protocols for Faults, Advances in Cryptology - EUROCRYPT '97, Vol. 1233, pp. 37–51 (1997).
6) Ciet, M. and Joye, M.: Practical Fault Countermeasures for Chinese Remaindering Based RSA, Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 124–131 (2005).
7) Coron, J.-S. and Kizhvatov, I.: Analysis and Improvement of the Random Delay Countermeasure of CHES 2009, Vol. 6225, pp. 95–109 (2010).
8) Dehbaoui, A., Dutertre, J.-M., Robisson, B. and Tria, A.: Electromagnetic Transient Faults Injection on a Hardware and a Software Implementations of AES, Proceedings of the 9th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Leuven, Belgium, pp.7-15 (2012).
9) Endo, S., Hayashi, Y., Homma, N., Aoki, T., Katashita, T., Hori, Y., Sakiyama, K., Nagata, M., Danger, J.-L., Le, T.-H. and Bazargan-Sabet, P.: Measurement of side-channel information from cryptographic devices on security evaluation platform: Demonstration of SPACES project, Proc. of SICE Annual Conference, Akita, Japan, pp. 313–316 (2012).
10) Endo, S., Homma, N., Hayashi, Y., Takahashi, J., Fuji, H. and Aoki, T.: An Adaptive Multiple-fault Injection Attack on Microcontrollers and a Countermeasure, IEICE Trans. Fundamentals, Vol. E98-A, No.1, pp.171-181 (2015).
11) Medwed, M. and Schmidt, J.-M.: A Continuous Fault Countermeasure for AES Providing a Constant Error Detection Rate, Proceedings of the 7th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Santa Barbara, CA, USA, pp. 66–71 (2010).
12) Moro, N., Dehbaoui, A., Heydemann, K., Robisson, B. and Encrenaz, E.: Electromagnetic Fault Injection: Towards a Fault Model on a 32-bit Microcontroller, Fault Diagnosis and Tolerance in Cryptography (FDTC), 2013 Workshop on, pp. 77–88 (2013).
13) Piret, G. and Quisquater, J.-J.: A Differential Fault Attack Technique against SPN Structures, with Application to the AES and Khazad, Cryptographic Hardware and Embedded Systems - CHES 2003, Lecture Notes in Computer Science, Vol. 2779, Springer Berlin Heidelberg, pp. 77–88 (2003).
14) Research Center for Information Security (RCIS) of National Institute of Advanced Industrial Science and Technology: SASEBO Project Overview, http://satoh.cs.uec.ac.jp/SASEBO/en/board/index.html.
15) Satoh, A., Sugawara, T., Homma, N. and Aoki, T.: High-Performance Concurrent Error Detection Scheme for AES Hardware, Cryptographic Hardware and Embedded Systems - CHES 2008, Lecture Notes in Computer Science, Vol. 5154, pp. 100–112 (2008).
16) Schmidt, J. M. and Christoph, H.: A Practical Fault Attack on Square and Multiply, Proceedings of 5th Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), Washington, DC, USA, pp. 53–58 (2008).

17) Trichina, E. and Korkikyan, R.: Multi Fault Laser Attacks on Protected CRT-RSA, Proceedings of the 7th Fault Diagnosis and Tolerance in Cryptography (FDTC), Santa Barbara, CA, USA, pp. 75–86 (2010).