IPA Better Life with IT

ハードウェア脆弱性評価の最新技術動向に関するセミナー

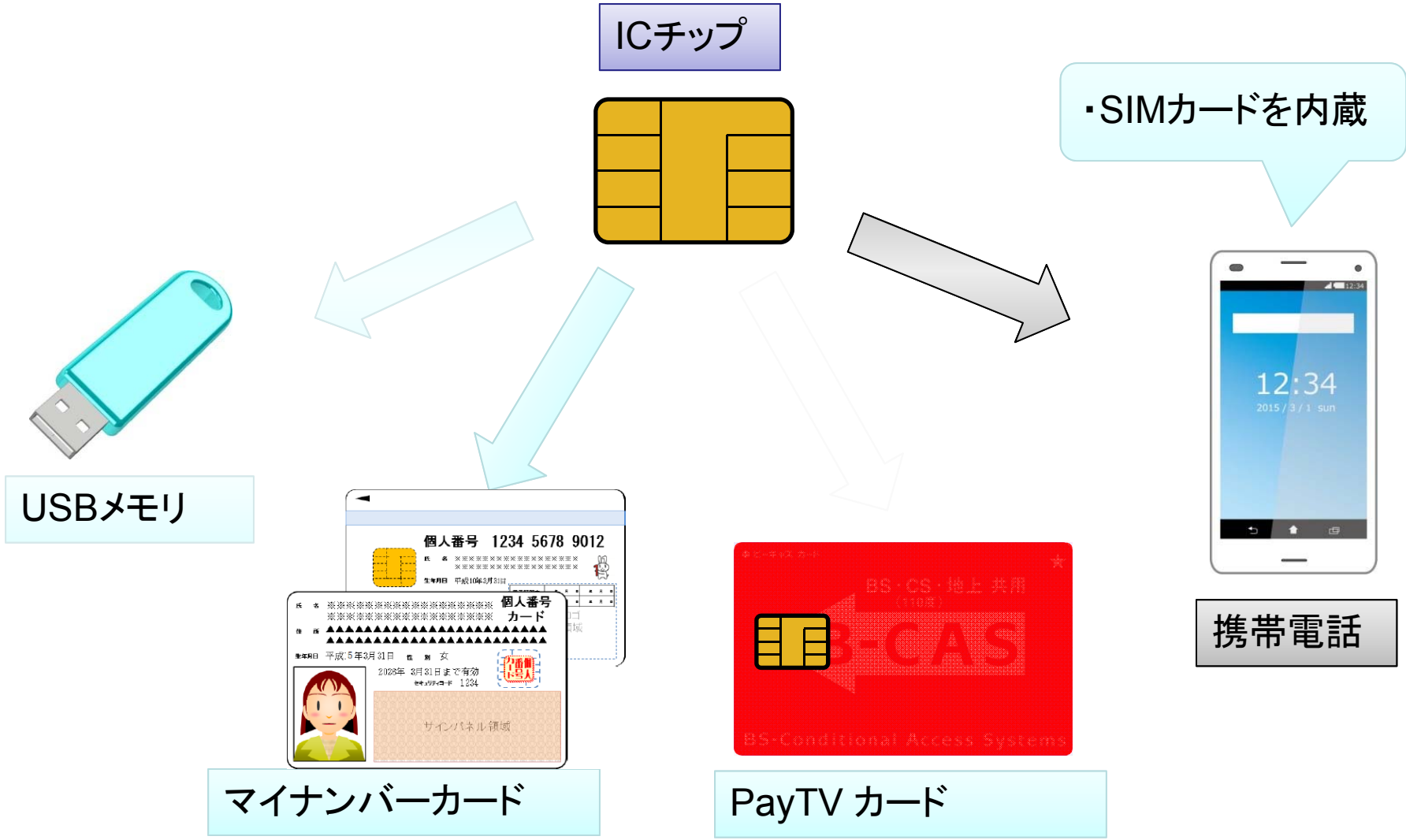
― CARDIS参加報告 ―

2016年2月4日

独立行政法人　情報処理推進機構
技術本部 セキュリティセンター

ハードウェアセキュリティトピックの紹介

様々な情報機器に用いられる セキュリティICチップ



利用環境によって異なる脅威


サーバ


デスクトップPC


複合機

オフィス等物理アクセスが制限された環境に設置される。
部外者が直接アクセスする可能性は低い。（適切に管理されていれば）

ICカード


USBメモリ

携帯電話

どこにでも自由に持ち運べる。
様々な環境で利用されるデバイス。（紛失・盗難等のリスク、通信データの盗聴等のリスクは固定環境より増大）

想定される利用環境における脅威・攻撃の分析が重要。
脅威・攻撃に対抗できるセキュリティ対策が必要となる。

攻撃方法の分類

- Non-Invasive Attack
  - チップ内部への物理的侵入を伴わない攻撃
  - 例: サイドチャネル解析
- Invasive Attack
  - チップ内部への物理的侵入を伴う攻撃
  - 例: プロービング、回路改変
- Semi-Invasive Attack
  - パッケージの開封(穴開け)程度は行うが、パシベーション層までは破壊しない
  - 例: レーザー攻撃

サイドチャネル攻撃 (Side Channel Analysis)

- 暗号機能を実装したハードウェア(スマートカード等)の動作中に、そのハードウェアの状態を観測することで得られる情報を利用して、暗号鍵といった秘密情報の復元を試みる
  - 消費電力 → 電力解析 (Power Analysis)
  - 電磁場 → 電磁解析 (Electromagnetic Analysis)
  - 処理時間 → タイミングアタック
  - その他
    - キャッシュヒット/ミス
    - 分岐予測

AESアルゴリズム

IPA

- 暗号化処理の流れ

```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin   平文              暗号文              拡大鍵
     byte state[4,Nb]                  //内部変数(4行, Nb列の行列)

     state = in

     AddRoundKey(state, w[0, Nb-1]) //

     for round = 1 step 1 to Nr-1
          SubBytes(state) //
          ShiftRows(state) //
          MixColumns(state) //
          AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
     end for

     SubBytes(state)
     ShiftRows(state)
     AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

     out = state
end
```

Nb:4,
Nr:10,12,14
　for 128, 192, 256-bit key,
w:拡大鍵, 要素数Nb *(Nr+1)

SubBytes:行列要素の置換

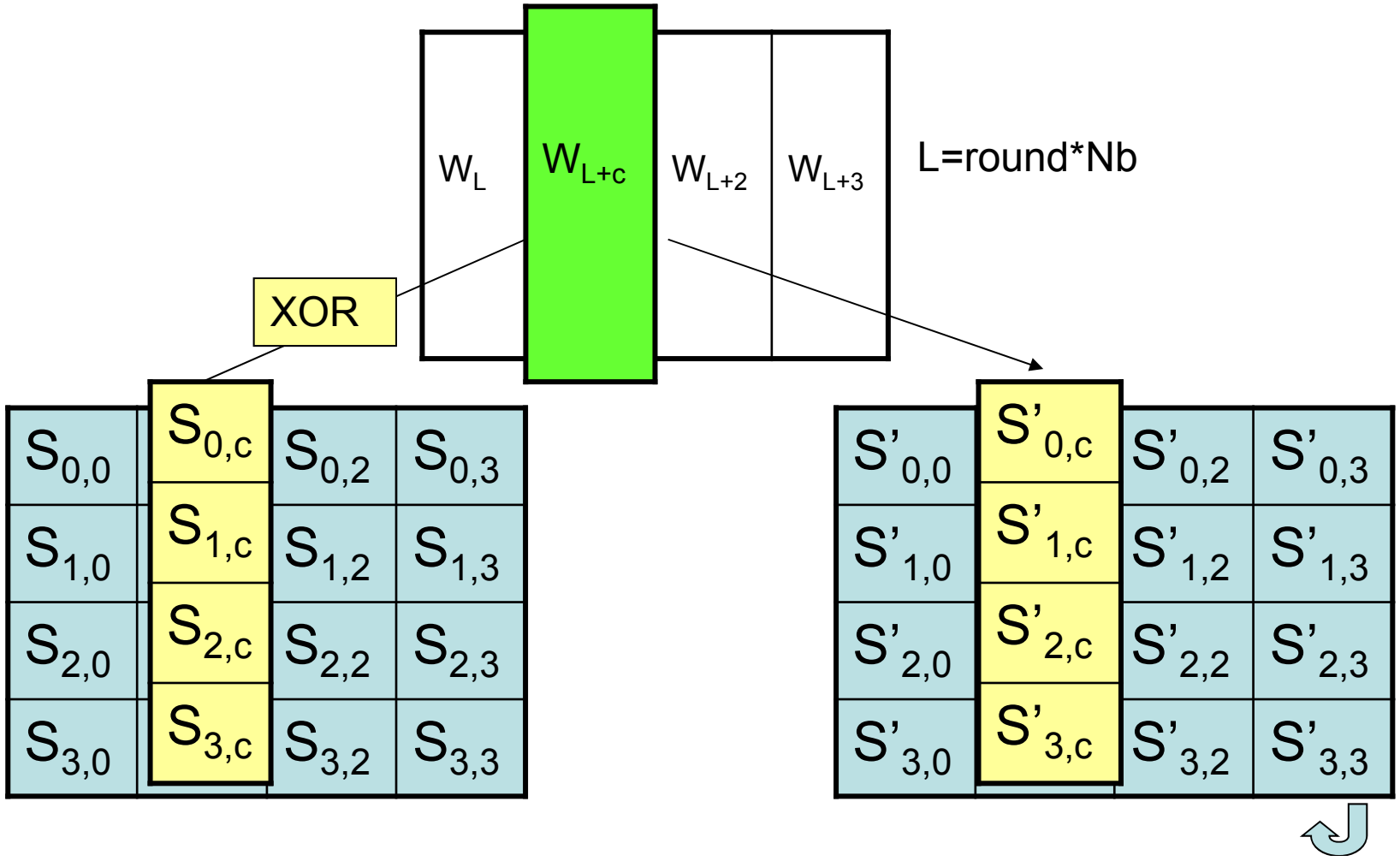
ShiftRows:
行単位の左シフト処理

MixColumns:
列ベクトル単位のデータの変換

AddRoundKey:
列ベクトルと拡大鍵wとのXOR演算

AES：AddRoundKeyの処理

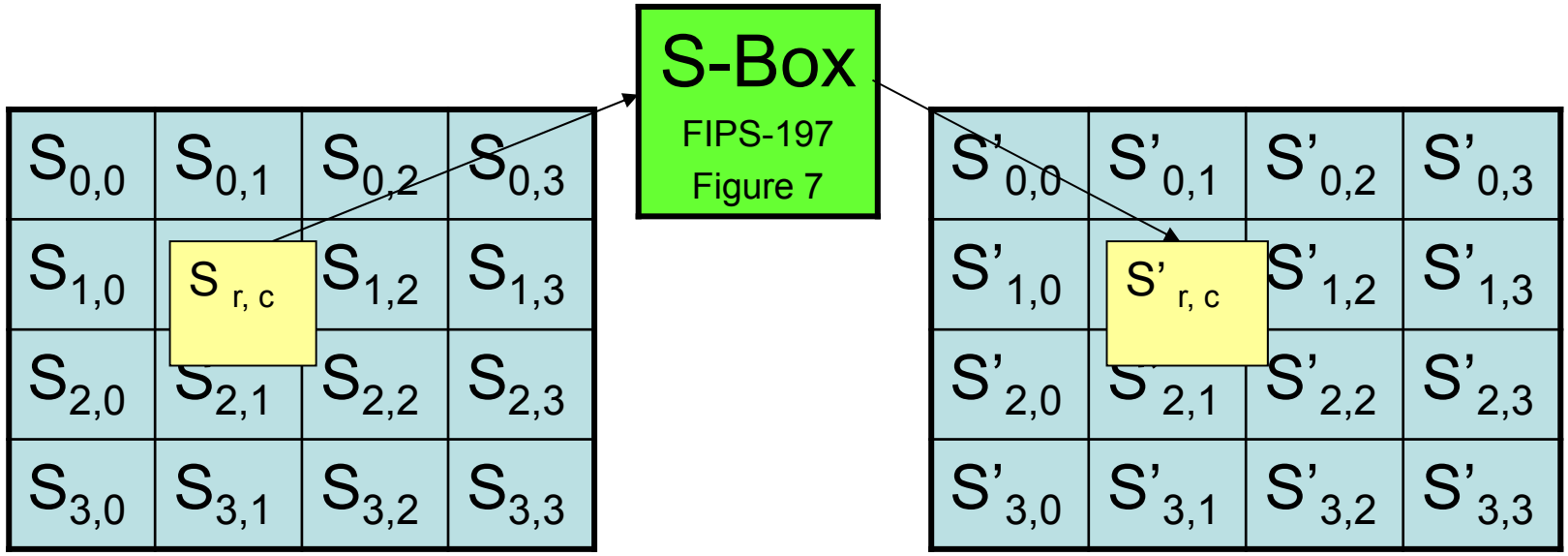
- 4ブロックを1列として，列ごとに拡大鍵とXOR処理。



AES：SubBytesの処理

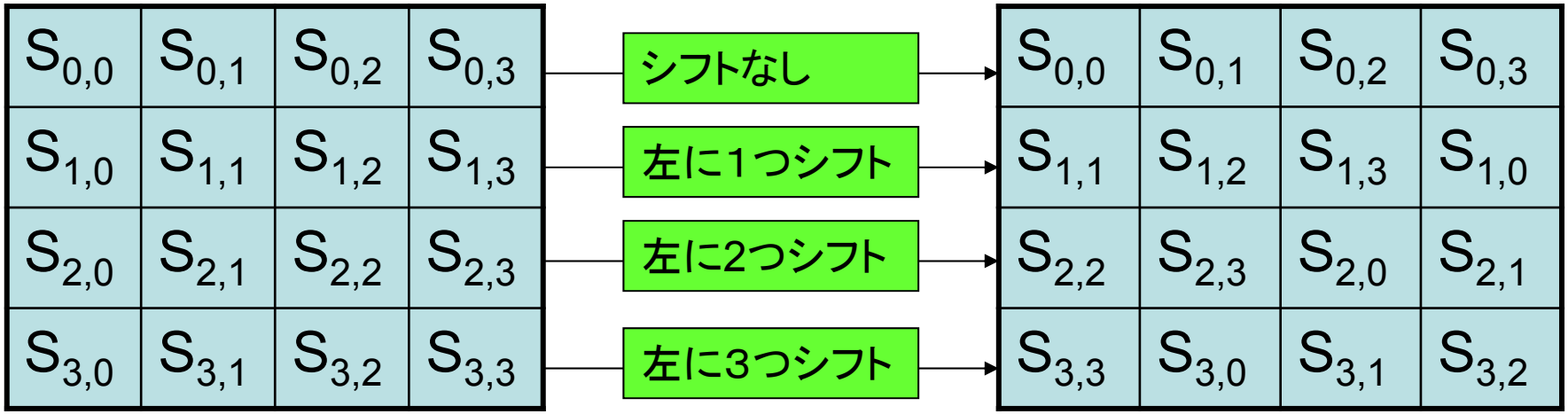
IPA

- 128ビットのデータを1バイト（8ビット）ごとに16のサブブロックに分割。
- 各ブロックでは1バイトの入力データを1バイトの出力データへ置換。



AES：ShiftRowsの処理

- 4ブロックを1行として，行ごとに左シフト処理。

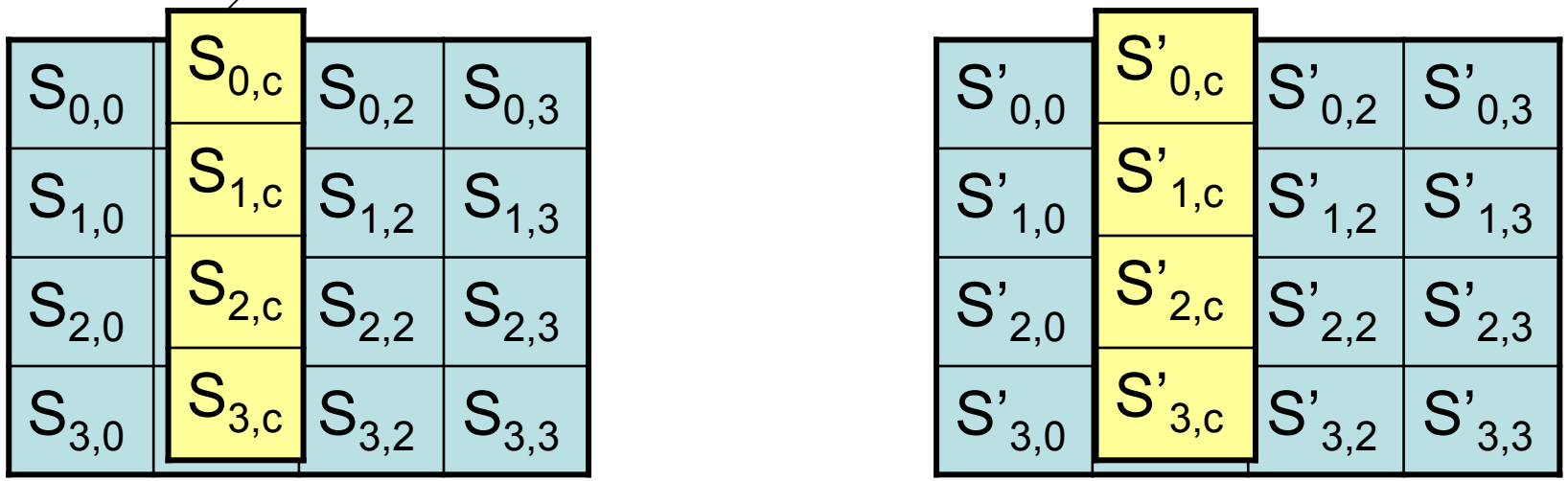


AES：MixColumsの処理

IPA

•4ブロックを1列として，列ごとに列ベクトルの変換。

$$\begin{pmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{pmatrix}$$



AES: S-Boxの定義の詳細

- 1バイト(8ビット)の値aに対し、
  - 逆元: $c = a^{-1}, a$の$GF(2^8)$における乗法の逆元
    (ただし、$a = 0$のときは$c = 0$)
  - アフィン変換: 出力$s = Mc \oplus b$:

$$\begin{pmatrix} s_7 \\ s_6 \\ s_5 \\ s_4 \\ s_3 \\ s_2 \\ s_1 \\ s_0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{pmatrix} \oplus \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

AES: S-Boxの実装

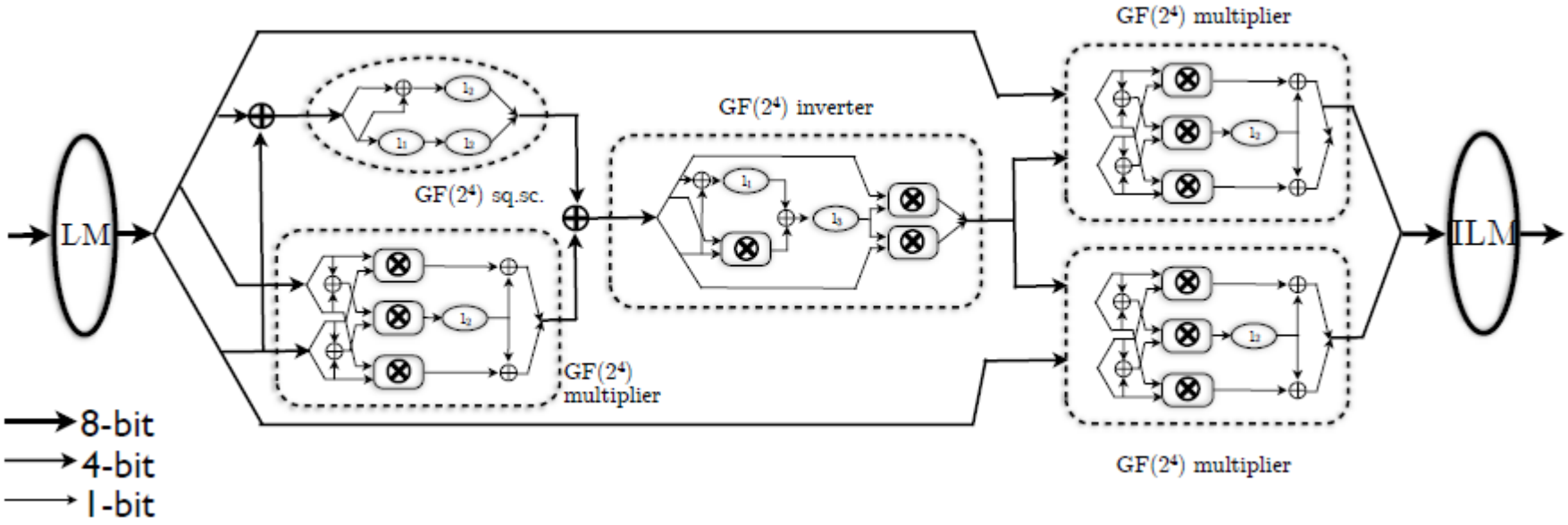
- テーブル参照
  - ソフトウェア向き

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

[FIPS197]より

AES: S-Boxの実装

- 数学的定義にしたがって逆元計算やアフィン変換を実装
  - GF(2^8)をGF((2^4)^2)やGF((((2^2)^2)^2)と見て逆元演算回路を構成
  - ハードウェア向き



S-Boxの実装例: [CD05] Canright's Very Compact AES S-box

公開鍵暗号の例: RSA暗号の原理

- *p, q:* 巨大な素数
- *n=pq:* 巨大な素数の積
- *e*: 公開鍵, *d*: 秘密鍵
- 暗号化: $c \equiv m^e \pmod{n}$
- 復号: $m \equiv c^d \pmod{n}$
- 署名: $s \equiv m^d \pmod{n}$
- 署名検証: $m \equiv s^e \pmod{n}$

巨大な数のべき乗剰余演算が必要

べき乗剰余演算の実装

- バイナリ法(Square-and-Multiply)アルゴリズム

入力: M, d
$d = d_1 d_2 \cdots d_n$: dの2進数表現 $(d_i = 0\ or\ 1)$

$S \leftarrow M$
for i from 1 to n-1 do
  $S \leftarrow S * S \bmod N$
  if $d_i = 1$ then
    $S \leftarrow S * M \bmod N$
  end
end
return S

べき乗剰余演算に対するSPA

- SquareとMultiplyのパターンから指数が分かる

入力: M, d
$d = d_1 d_2 \cdots d_n$: dの2進数表現 $(d_i = 0\ or\ 1)$

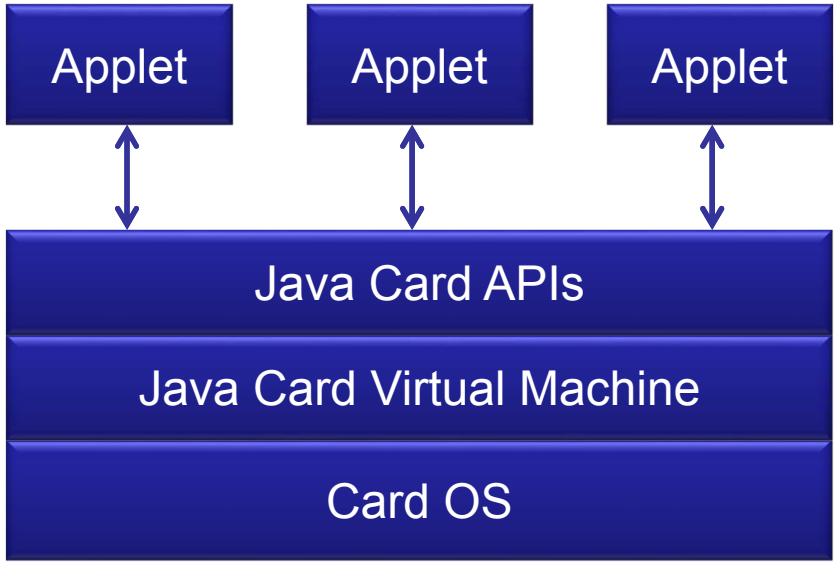
$S \leftarrow M$
for i from 1 to n-1 do
  $S \leftarrow S * S \bmod N$ S
  if $d_i = 1$ then
    $S \leftarrow S * M \bmod N$ M
  end
end
return S

サイドチャネル攻撃対策: message blindingとexponent blinding

- $c = m^d \mod N$
  - $m_r = mr^{-e} \mod N$ message blinding
  - $d_r = d + r\varphi(N)$ exponent blinding
  - $c_r = {m_r}^{d_r} \mod N$ blinded exponentiation
  - $c = c_r r \mod N$ message “unblinding”
- SとMの出現順序は、指数に依存する
  - dがランダムならば、指数のビットの出現順はRSAの実行ごとに異なる
  - mがランダムならは、中間値がランダムになり、予測困難になる
- DPAは中間値の予測に基づく
- 中間値が予測できないと、DPAは有効でない

Java Card

- Javaテクノロジに基づく
- アプレットと呼ばれる、Javaベースのアプリケーションを搭載できる
- 複数のアプレットを搭載できる



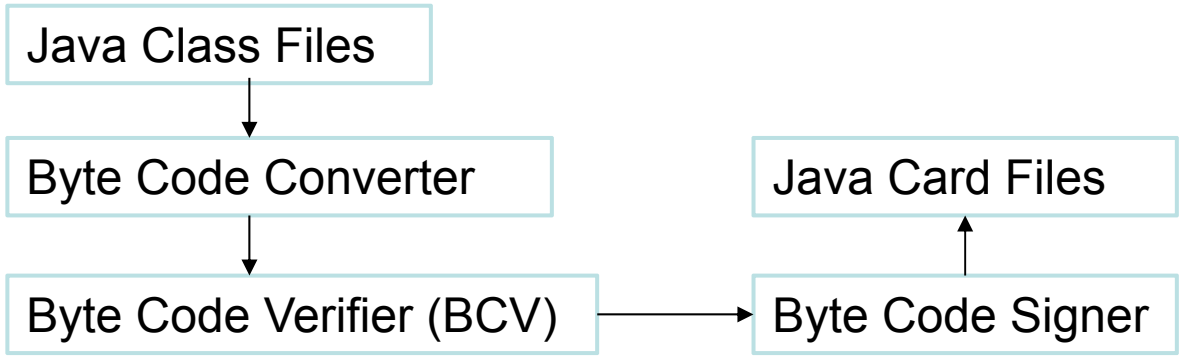
Java Cardのセキュリティ機構

IPA

- Type Safety
  - 例えば、整数(int)型の値を異なる型に再解釈すること(type confusion)は禁止される
- Byte-code Verifier
  - 不正なバイトコードを検出する
  - カード上に実装される場合と、カード外で実行される場合がある
- Defensive Virtual Machine
  - 不正なバイトコードの実行を防ぐ
- Firewall
  - あるアプレット上のデータを他のアプレットから保護する

Java Card Byte Code Verifier (BCV)

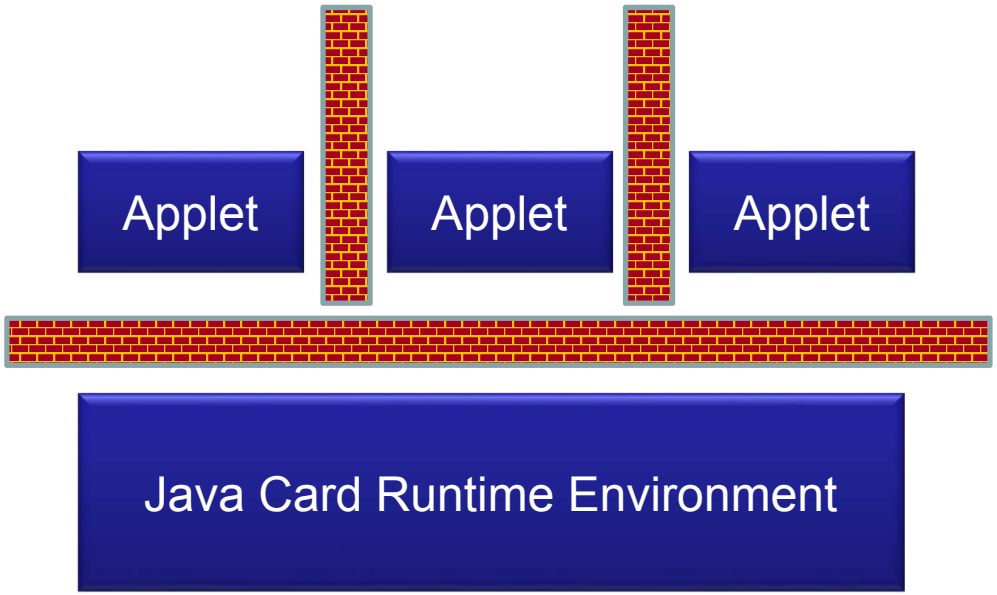
- バイトコードの整合性をチェック
  - 型の整合性
  - スタックオーバーフロー/アンダーフロー
  - オブジェクトの正しい初期化
  - …
- Off-Card BCV



- On-Card BCV

Java Card Firewall

- アプレットが、他のアプレットのデータやメソッドにアクセスすることを禁止する



Java Card: Type confusion

- どんなリファレンスも、元のタイプのリファレンスとして参照することしか許されない
- もしbyteの配列がshortの配列としてアクセスされると?

byte[4]として読み出し

0	1	2	3
00	01	02	03

short[4] として読み出し

0	2	4	6
0001	0203	XXXX	XXXX

- 配列の境界を超えたアクセス!
- この現象をtype confusionと呼ぶ

ワークショップの内容紹介

カンファレンス情報

- CARDIS (14th Smart Card Research and Advanced Application Conference)
  - Bochum, Germany
  - 2015/11/4-11/6
  - 17本の論文が採用
  - 6個のセッション
    - Side-Channel Attacks
    - Java Cards
    - Evaluation Tools
    - Fault Attacks
    - Countermeasures
    - Implementations
  - 2件の招待講演

注: 以下のスライド中の図は、特に指定がない場合はCARDISのスライドが出典である

Higher-Order Threshold Implementation of the AES S-box

Thomas De Cnudde[1], Begül Bilginand[1], Oscar Reparaz[1], Ventzislav Nikov[2], and Svetla Nikova[1]

[1]KU Leuven, ESAT-COSIC and iMinds, Belgium
[2]NXP Semiconductors, Belgium

https://securewww.esat.kuleuven.be/cosic/publications/article-2575.pdf

Higher-Order Threshold Implementation of the AES S-box

- マスキング
  - サイドチャネル攻撃への対策のひとつ
  - 中間値を、ランダムな値で、XORなどの演算でマスクすることにより、リークを防ぐ

例: AESの第1ラウンド (p: 平文、k: 鍵)

マスキングなし

$$x = p \oplus k$$
$$y = SBox(x)$$

マスキングの例 (boolean masking)

$$m = random$$
$$p_m = p \oplus m$$
$$x_m = p_m \oplus k$$
$$y_m = SBox_m(x_m)$$

Higher-Order Threshold Implementation of the AES S-box

- グリッチの影響
  - グリッチ (入力信号の変化)があると、消費電力がマスクなしの生のデータに依存することから、リークが完全には防げない

古典的な、マスク付きAND回路
入力: $\tilde{x} = x \oplus a, \tilde{y} = y \oplus b$
出力: $\tilde{z} = (x \text{ and } y) \oplus c$

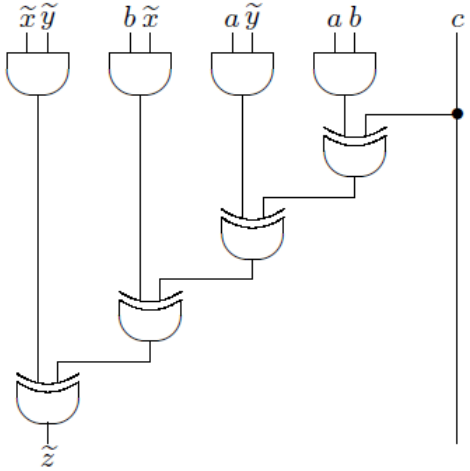


Fig. 1. Glitch propagation through a masked AND gate.

図は[NS06]より

b $\tilde{y}$	AND	XOR
0 0	0	0
0 1	1	1
1 0	1	2
1 1	2	2

表は[NS06]より

入力$\tilde{x}$にグリッチが起こった時に影響を受けるゲートの数

↓

スイッチングが起こる数

↓

消費電力の変動につながる

Higher-Order Threshold Implementation of the AES S-box

- 2nd-Order Attack
  - 同じマスクに関連づいた2個の中間値からのリークを利用する攻撃

2つの中間値 u, v が同じマスク m でマスキングされていると、

$$u_m = u \oplus m, v_m = v \oplus m$$

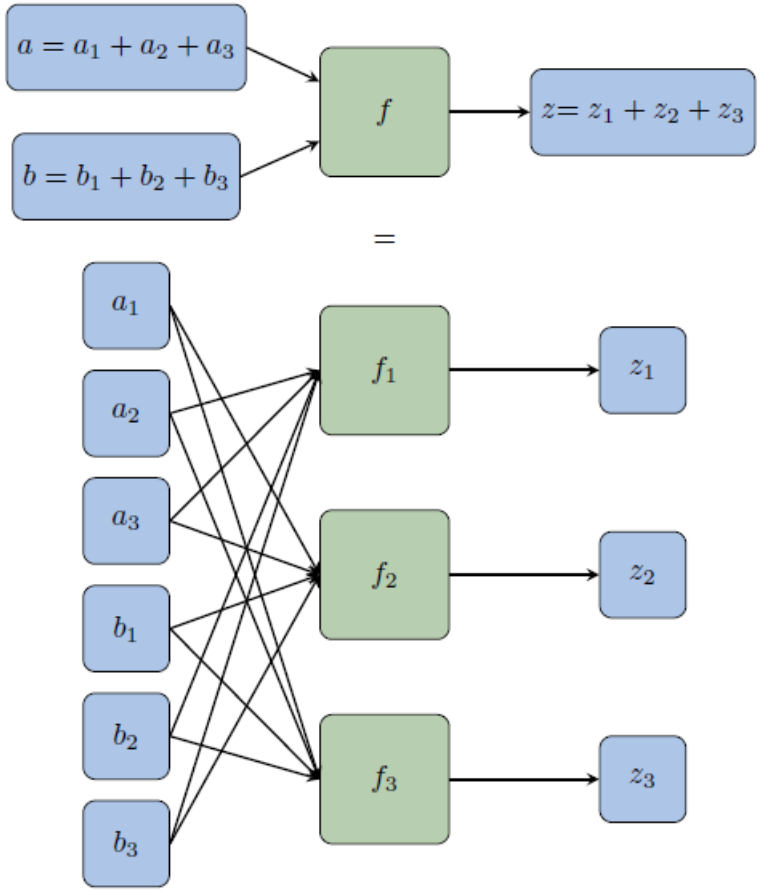
マスクされた中間値同士の排他的論理和を取ると、

$$u_m \oplus v_m = (u \oplus m) \oplus (v \oplus m) = u \oplus v$$

マスクがなくなる!

Higher-Order Threshold Implementation of the AES S-box

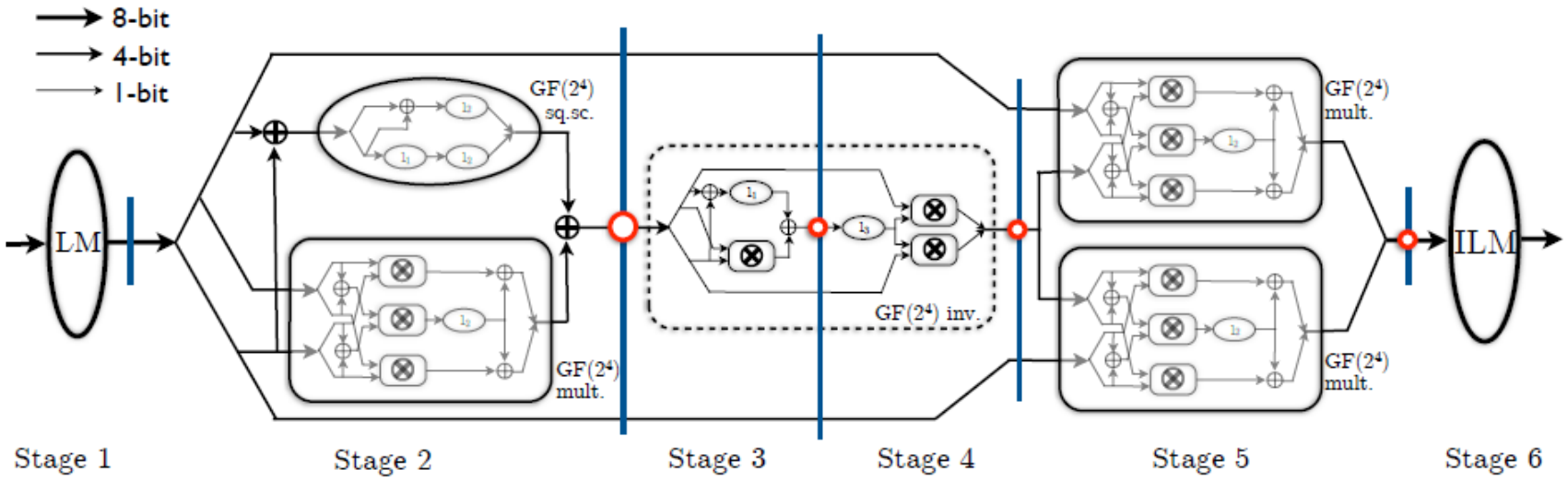
♦ Threshold Implementation



- 入力が一様
- d^{th} order non-completeness: fの要素関数f_iのd個以下のどの組み合わせも、少なくとも1個のinput share x_iと独立である
- グリッチが起こってもリークが起こらない

Higher-Order Threshold Implementation of the AES S-box

- AES S-boxをマスキングする



Higher-Order Threshold Implementation of the AES S-box

IPA

- 実装の結果

S-box	Area [GE]	Randomness [bit]	Clock Cyles	Security
[Moradi2011]	4244	48	5	1^{st}-order
[Bilgin2014]	3003	44	3	1^{st}-order
[Bilgin2015]	2224	32	3	1^{st}-order
This Paper	7849	126	6	2^{nd}-order

- 2nd-orderに対するセキュリティ

Java Card Virtual Machine Compromising from a Bytecode Verified Applet

Julien Lancia[1] and Guillaume Bouffard[2]

[1]THALES Communications and Security S.A.S
[2]Agence Nationale de la Sécurité des Systèmes d'Informations (ANSSI)

Java Card Virtual Machine Compromising from a Bytecode Verified Applet

IPA

- Java CardのByte Code Verifierに未チェック項目
- Byte Code Verifierを通るコードで、不正なネイティブメソッド呼び出しが可能になる

- 2015年9月に出されたOracleのパッチ (version 3.0.5u1)で対策された
- それより前のバージョンには脆弱性あり

Misuse of Frame Creation to Exploit Stack Underflow Attacks on Java Card

Benoit, Laugier and Tiana, Razafindralambo

Department of Electrical Engineering-ESAT/COSIC and iMinds, KU Leuven

https://eprint.iacr.org/2015/727

Misuse of Frame Creation to Exploit Stack Underflow Attacks on Java Card

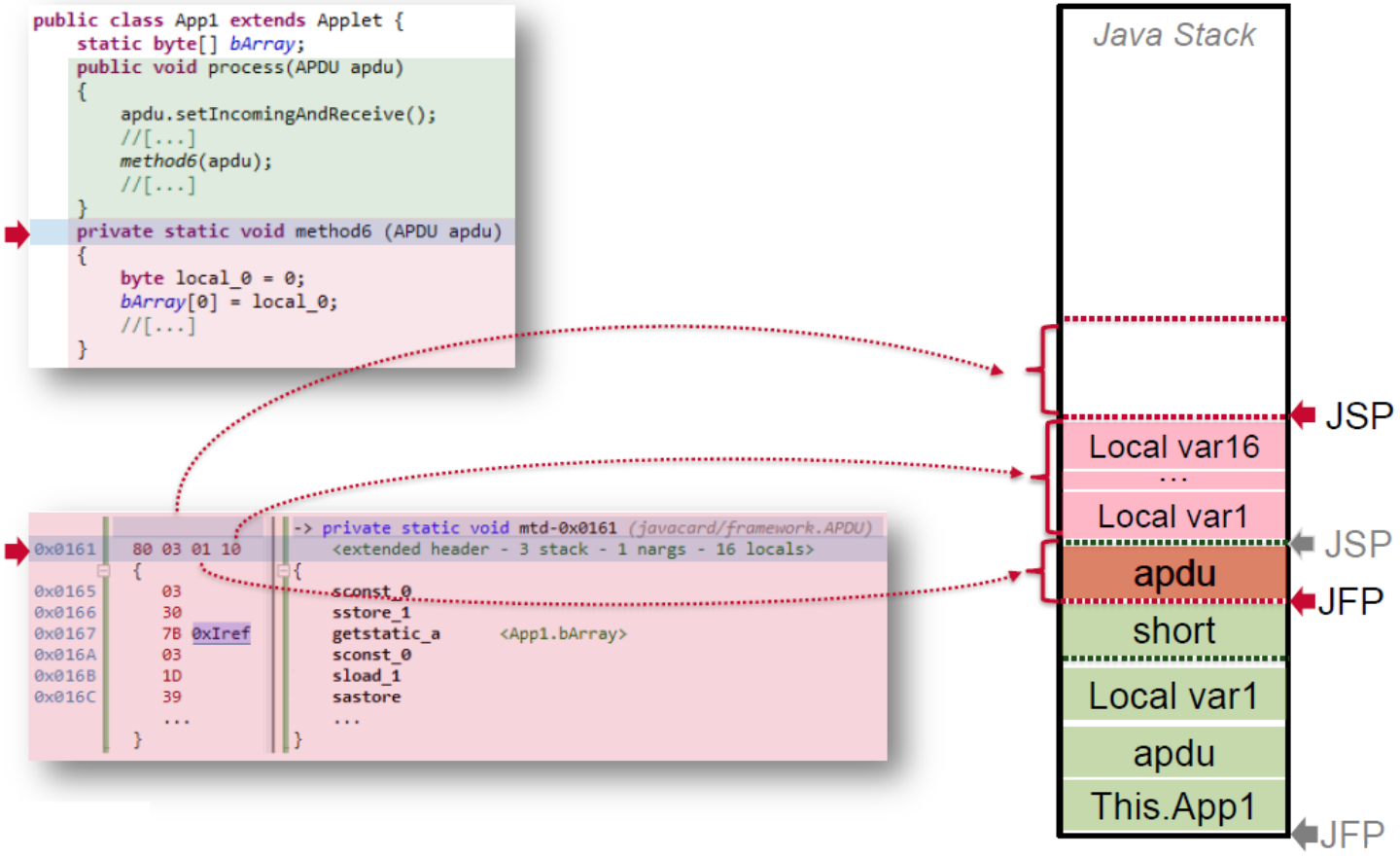
- フレームとは、データや途中経過、メソッドの返り値を格納するための領域である
- メソッドが呼び出されるたびに生成される
- フレームはJava VMのスタック上に確保される
- 各フレームはそれぞれローカル変数やオペランドスタックを保持している
- フレームサイズは、コンパイル時に決定される

Misuse of Frame Creation to Exploit Stack Underflow Attacks on Java Card

- メソッド呼び出し時にフレームが作成される時に、フレームサイズを改ざんすると、前のフレームとオーバーラップしたフレームが生成される
  - Fault attackでそれを引き起こす

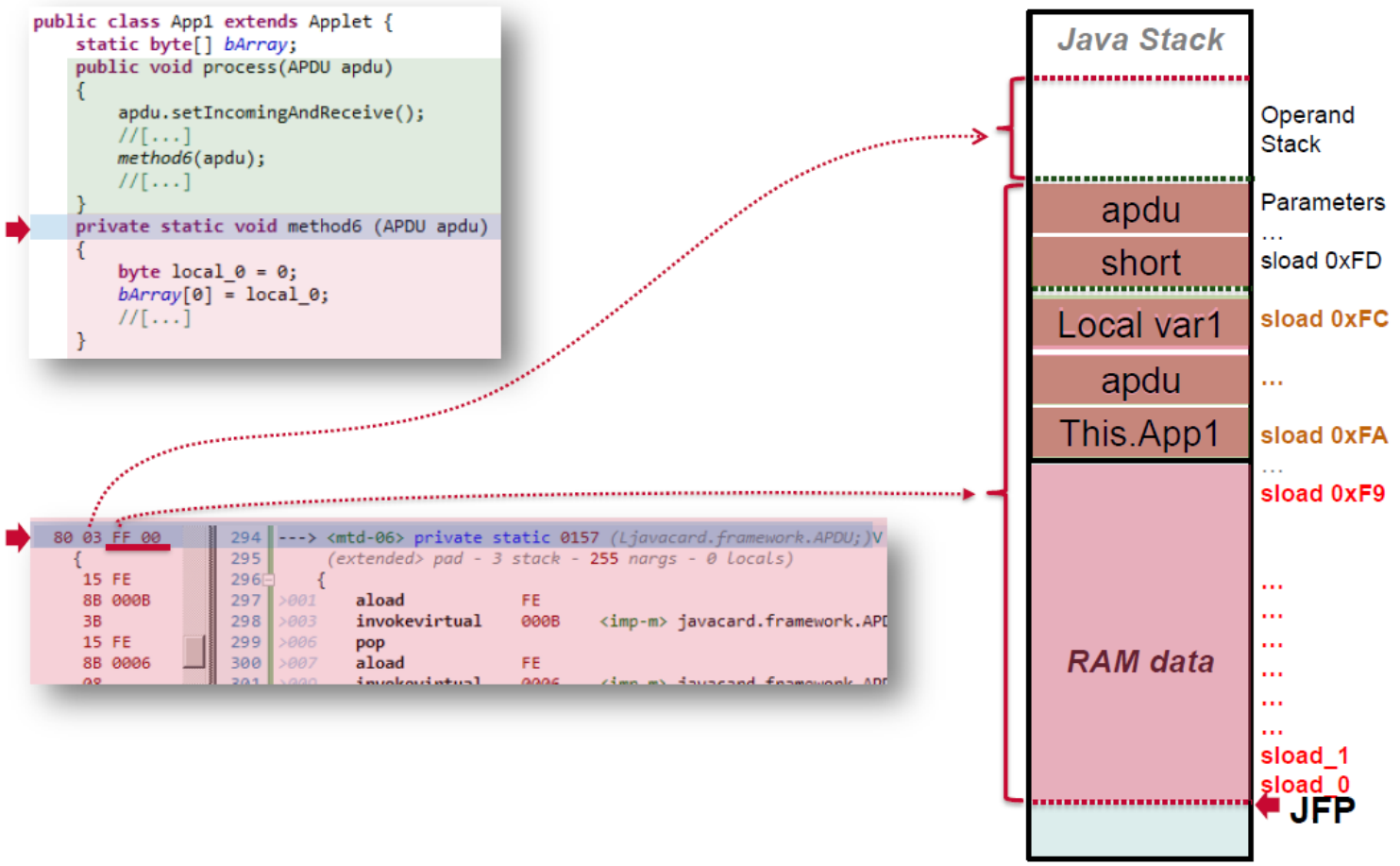
Misuse of Frame Creation to Exploit Stack Underflow Attacks on Java Card

- フレーム生成



Misuse of Frame Creation to Exploit Stack Underflow Attacks on Java Card

- 壊れたフレームを生成して攻撃



A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations

Guilherme Perin and Łukasz Chmielewski

Riscure BV

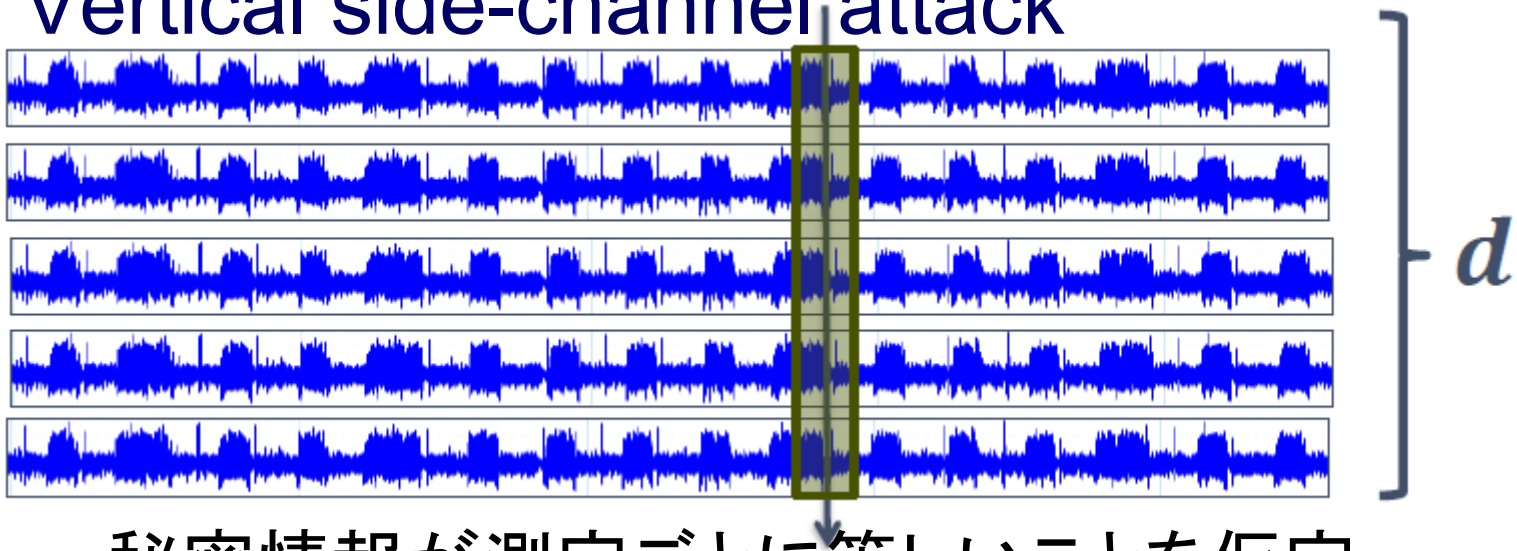
IPA

- 対策ありのRSAの実装に対するサイドチャネル攻撃のフレームワーク
  - Message Blinding
  - Exponent Blinding

A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations

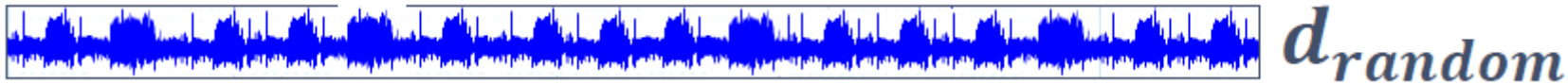
IPA

- Vertical side-channel attack



  - 秘密情報が測定ごとに等しいことを仮定

- Horizontal side-channel attack



  - 秘密情報が測定ごとに異なっていると仮定

A Semi-Parametric Approach for Side-Channel Attacks on Protected RSA Implementations

- Learning Phase
  - Unsupervised LearningによるLeakage Assessment
    - トレースのセットを入力として、Point of Interestを見つける
  - Horizontal Attack
    - Point of Interestとトレースのセットを入力として、指数の近似を求める
  - Point of Interestの選択を最適化
    - 指数の近似、トレースを入力として、
      - t-testによってPoint of Interestの精度を高める
      - 精度を高めたPoint of Interestを入力としてHorizontal Attackを繰り返して、指数の近似精度を高める
- Attacking Phase
  - 最終的な確率の計算
    - トレースのセットと精度を高めた指数を入力として、モジュロ演算における最終的な(それが2乗演算である)確率を求める
  - エラー検出と訂正
    - 1つのトレースに対して、精度を高めた指数と最終的な確率を入力として、正しい指数、あるいはエラーを返す

IPA

- 結論
  - 1個のトレースはからはかなりの情報漏れがある
  - 十分なデータを復元できる
  - 改善の余地は多くある

Precise Laser Fault injections into 90nm and 45nm SRAM-cells

Bodo Selmke[1], Stefan Brummer[1], Johann Heyszl[1], and Georg Sigl[2]

[1]Fraunhofer Institute for Applied and Integrated Security
[2]Technische Universität München, Department of Electrical and Computer Engineering

Precise Laser Fault injections into 90nm and 45nm SRAM-cells

IPA

- フォールトアタックは、正確なフォールトインジェクションを要求する
- 今日のフィーチャーサイズのチップに対して、どれくらい正確にフォールトを起こすことができるだろうか?

Precise Laser Fault injections into 90nm and 45nm SRAM-cells

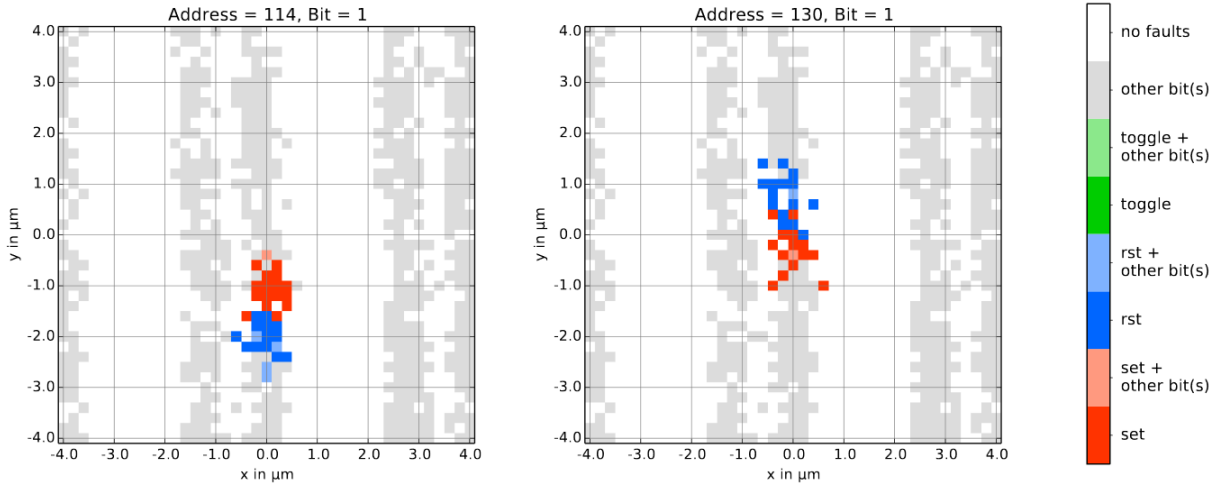
IPA

- 裏面からのレーザー攻撃
- FPGAのBlock RAM (BRAM)を攻撃対象にする
- 以下のチップをテスト
  - 90nm: Xilinx Spartan-3A
  - 45nm: Xilinx Spartan-6

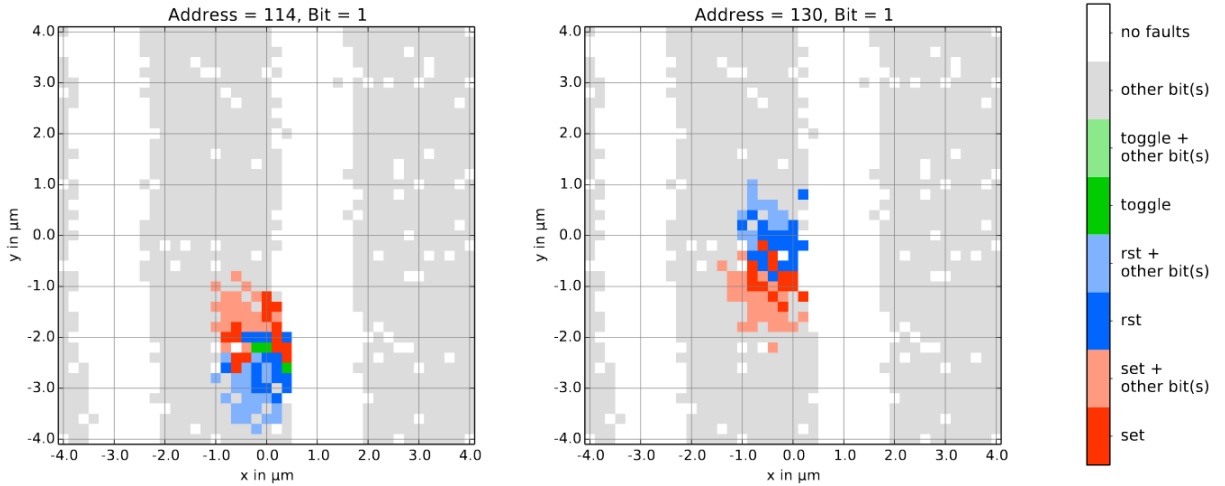
Precise Laser Fault injections into 90nm and 45nm SRAM-cells

- Spartan-6 (90nm)での結果

Pulse energy: 1nJ



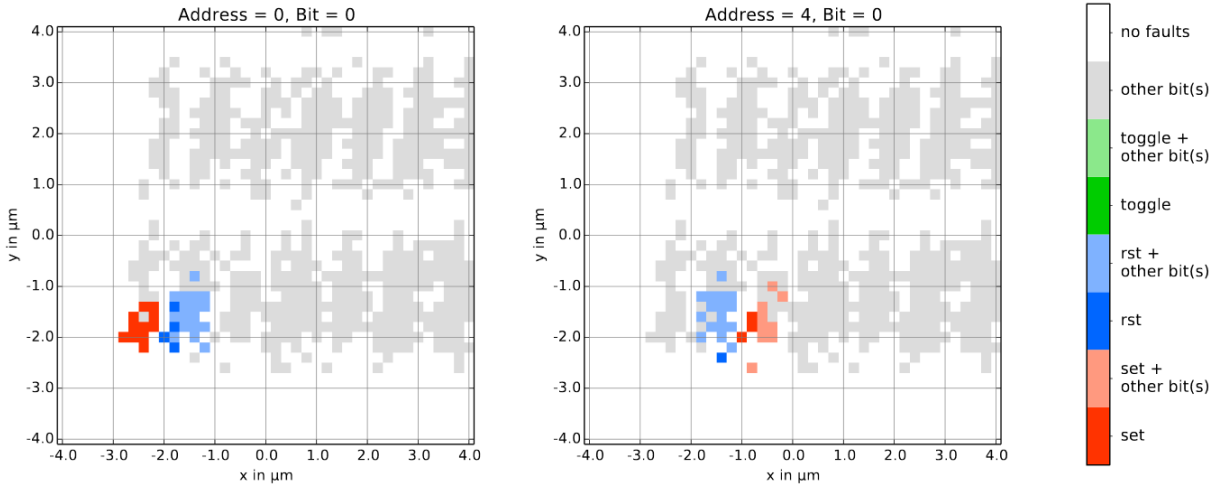
Pulse energy: 1.5nJ



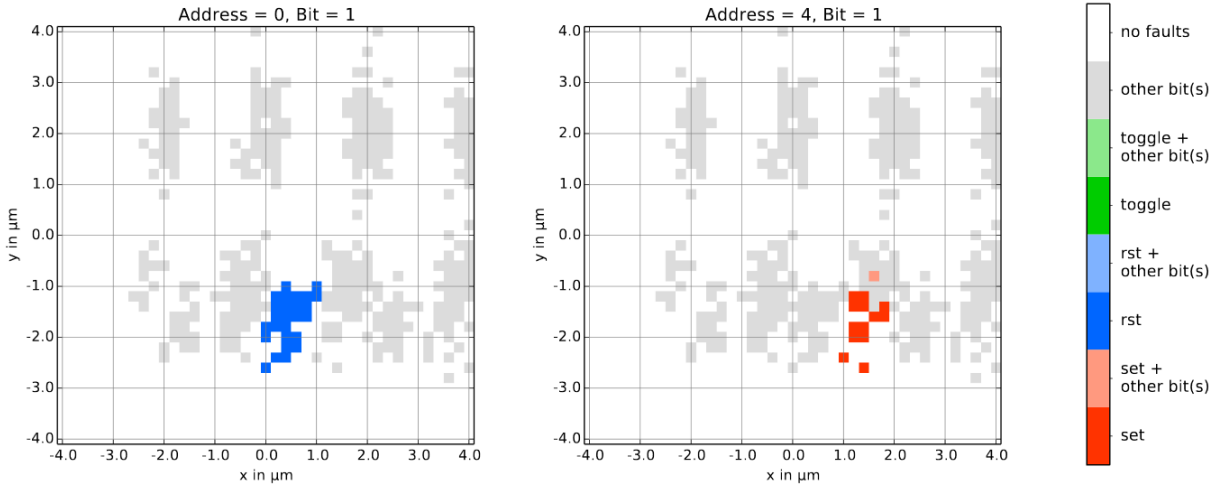
Precise Laser Fault injections into 90nm and 45nm SRAM-cells

- Spartan-6 (45nm)での結果

Pulse energy: 1nJ



Pulse energy: 1nJ
データ依存



Precise Laser Fault injections into 90nm and 45nm SRAM-cells

IPA

- 結論

- Spartan-3A (90nm)
  - 90nmサイズに対して、単一ビットフォールトは十分可能
  - 特定のビットを特定の値にセットすることが可能
    → 最も制限的なフォールトモデルが適用可能
- Spartan-6 (45nm)
  - 単一ビットのセットはまだ可能であるが成功率は落ちる
  - 隣接したビットに影響が及ぶことが多い
  - 単一ビットフォールトが起こせるかどうかはデータ依存

From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

Louis Dureuil[1,2,3], Marie-Laure Potet[1,3], Philippe de Choudens[1,2], Cécile Dumas[1,2], and Jessy Clédière[1,2]

[1]Univ. Grenoble Alpes
[2]CEA, LETI, MINATEC Campus
[3]CNRS, VERIMAG

From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

- Common Criteria (ISO/IEC 15408)における、スマートカードと類似デバイスの脆弱性評価

値の範囲	次の攻撃能力を持つ攻撃者に対するTOEの抵抗力
0～15	レート付けなし
16～20	基本
21～24	基本強化
25～30	中
31以上	高

[CCD13] より
TOE: Target Of Evaluation

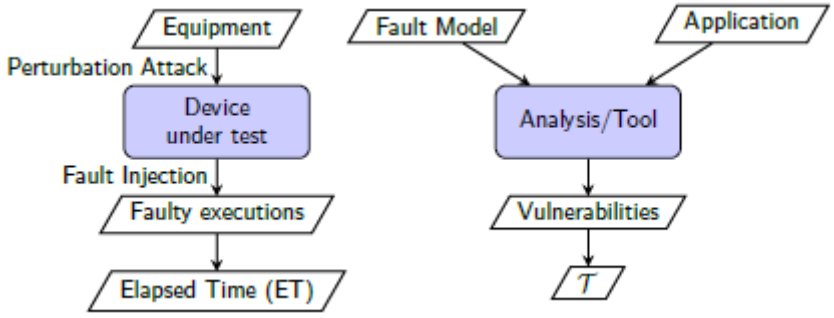
要素	識別	悪用
所要時間		
1時間未満	0	0
1日未満	1	3
1週間未満	2	4
1カ月未満	3	6
1カ月を超える	5	8
非現実的	*	*
専門知識		
素人	0	0
熟練者	2	2
エキスパート	5	4
複数のエキスパート	7	6
TOEの知識		
公開	0	0
制限	2	2
秘密	4	3
危機的	6	5
非常に重要なハードウェア設計	9	該当なし
TOEへのアクセス		
10サンプル未満	0	0
30サンプル未満	1	2
100サンプル未満	2	4
100サンプルを超える	3	6
非現実的	*	*
機器		
なし	0	0
標準	1	2
特殊	3	4
特別注文	5	6
複数の特別注文	7	8
オープンサンプル		
公開	0	該当なし
制限	2	該当なし
秘密	4	該当なし
危機的	6	該当なし

From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

IPA

- Common Criteriaにおける脆弱性評価
  - 侵入テスト
    - 攻撃を成功させようと試みる
    - 攻撃成功への所要時間(ET): 所要時間は評価の尺度のひとつ
  - コード解析
    - あるフォールトモデルを用いてアタックパスを探求
    - 攻撃の成功率($\mathcal{T}$=$\mathcal{F}s/\mathcal{F}$)をはじき出す

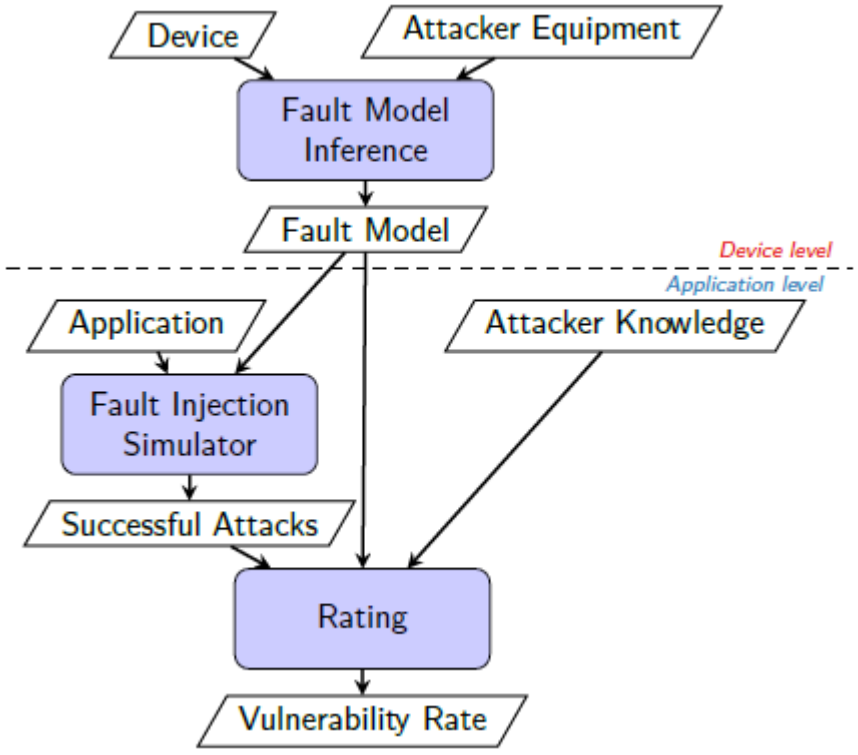
要素
所要時間
熟練度
TOEの知識
TOEへのアクセス
装置
オープンサンプル



From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

- Device Level
  - 評価対象と攻撃の装置に紐づいたフォールトモデルの特性を示す
    - 攻撃のパラメタに基づいて可能なフォールトを記述
    - それぞれの種類のフォールト発生の確率を関連付ける

    → Probablistic Fault Model (PFM)
- Application Level
  - フォールトインジェクションシミュレータをPFMと結びつける
    - 入力としてのPFM: アプリケーションへの攻撃の精密なテスト
    - 出力としてのPFM: アプリケーションの堅牢さの定量化



From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

- 実験結果
  - $\mathcal{V}$: Vulnerability rate
  - $\mathcal{T} = \frac{|F_S{}'|}{|F'|}$: Traditional vulnerability rate
  - φ : physical success rate from experiments on cards

Card	Command	$\mathcal{V}$	$\mathcal{T}$	φ	$\lvert \mathcal{P}' \rvert$
A	VerifyPIN	2.35×10^{-5}	3.2×10^{-2}	3.40×10^{-5}	5883
A	SecureVerifyPIN	2.08×10^{-6}	8.5×10^{-5}	0	5000
A	GetChallenge	2.01×10^{-5}	1.75×10^{-3}	2.94×10^{-5}	6800
A	SecureGetChallenge	7.1×10^{-7}	2.74×10^{-6}	0	3000
B	GetChallenge	1.1×10^{-3}	1.2×10^{-3}	1.4×10^{-3}	231
B	SecureGetChallenge	0	2.14×10^{-4}	0	833

Rating criteria for several implementations on various cards

From Code Review to Fault Injection Attacks: Filling the Gap using Fault Model Inference

- 実験結果
  - Attack PotentialにおけるElapsed Time(所要時間)の要素: $(s \times \mathcal{V})^{-1}$
  - Card A: s = 1.27 attacks· s^{-1}
  - Card B: s = 3.30 attacks· s^{-1}

Card	Command	$(s \times \mathcal{V})^{-1}$ (ET)	$(s \times \mathcal{T})^{-1}$ (ET)	$(s \times \varphi)^{-1}$ (ET)
A	VerifyPIN	8h (3)	24s (0)	6h (3)
A	SecureVerifyPIN	1w (4)	2.5h (3)	$> 3d$ (≥ 4)
A	GetChallenge	10h (3)	7min (0)	7.4h (3)
A	SecureGetChallenge	2w (6)	3.5d (4)	$> 3d$ (≥ 4)
B	GetChallenge	5min (0)	5min (0)	5min (0)
B	SecureGetChallenge	not practical (_)	20min (3)	$> 3d$ (≥ 4)

Comparison of (predicted) elapsed times

所要時間 (Elapsed Time)	悪用 (Exploitation Rating)
< 1時間	0
< 1日	3
< 1週間	4
< 1か月	6
> 1か月	8
Not Practical	-

- この手法の、Application of Attack Potentialの脆弱性評価の各要素への適用可能性

要素	$\mathcal{V}$
所要時間	✓
熟練度	部分的 (*Pr(p)*: 攻撃パラメタ*p*を選択する確率)
TOEの知識	部分的 (*Pr(p)*: 攻撃パラメタ*p*を選択する確率)
TOEへのアクセス	–
装置	部分的 ($\mathcal{M}_{d,e}$: フォールトモデル)
オープンサンプル	–

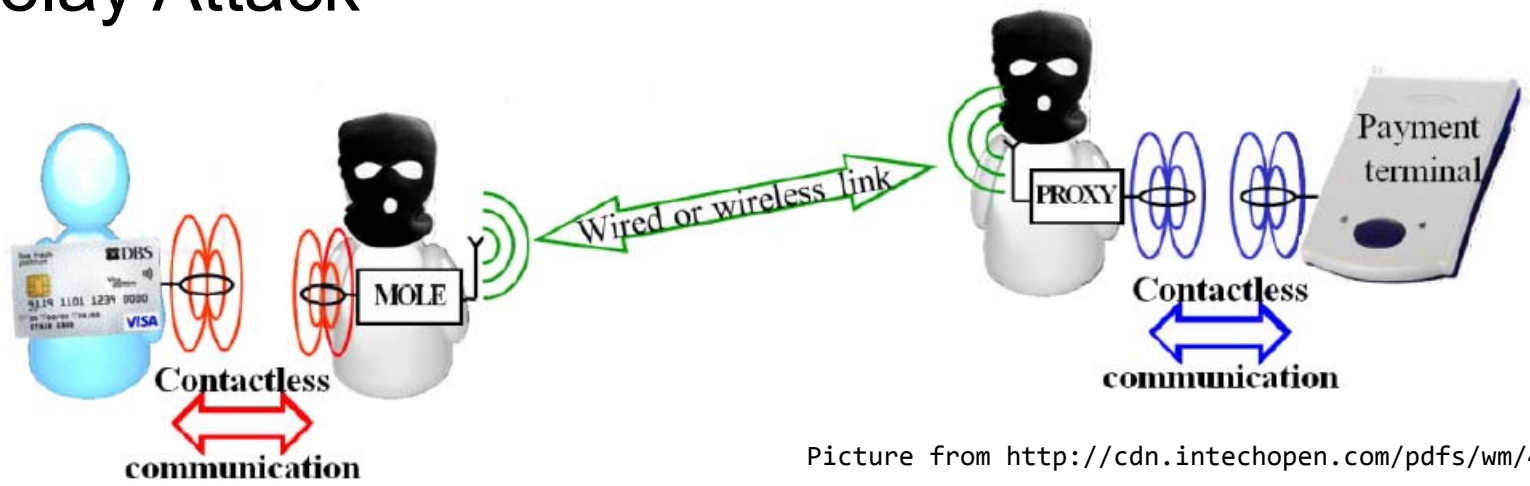
The not-so-distant future: Distance-bounding protocols on smartphones

Sébastien Gambs[1], Carlos Eduardo Rosar Kós Lassance[2], and Cristina Onete[3]

[1]Université de Rennes 1 - Inria
[2]Université de Rennes 1 / Télécom Bretagne
[3]Inria / INSA Rennes

The not-so-distant future: Distance-bounding protocols on smartphones

Relay Attack



Picture from http://cdn.intechopen.com/pdfs/wm/44973.pdf

Distance-Bounding Protocol

- VerifierとProverとの物理的な距離が近い場合のみ認証を許可する
- 信号遅延時間で判断する

The not-so-distant future: Distance-bounding protocols on smartphones

IPA

- Distance-bounding protocolの、スマートフォンでRFIDタグエミュレーションモードを動かしている場合での実装可能性を探求
- ハードウェア(SIMカード、携帯電話のプロセッサ)の変更なしのAndroid端末を対象とする
- 実装してみた結果: relay attackで1.5ms以上の遅延が引き起こされるのであれば、relay attackを十分検出できる

まとめ

まとめ

- Common Criteria流評価
  - Fault Injection攻撃耐性の評価を、Application of Attack Potential to Smartcardsのレーティング算出につなげたことは注目に値する
- Threshold Implementationの研究が盛んである
  - 2nd order attackに対するセキュリティを主張した実装例
- Java Cardへの攻撃
  - 攻撃方法についていろいろ研究されている
- 対策ありのRSAの実装への攻撃
  - Message Blinding/Exponent Blindingの対策ありのRSA実装に対しての攻撃が研究されている
  - いろいろな乗算方式に対する攻撃についても将来的な研究動向に注目
- 45nmプロセスのチップへの攻撃
  - レーザー攻撃による単一ビットのセットは、45nmプロセスのチップに対しても手が届く範囲である

参考文献

- [FIPS197] National Institute of Standards and Technology (NIST): Advanced Encryption Standard (AES) FIPS Publication 197. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf , Nov 2001
  AESの定義
- [CCD13] Application of Attack Potential to Smartcards, Joint Interoretation Library, http://www.commoncriteriaportal.org/files/supdocs/CCDB-2013-05-002.pdf
  日本語訳 (参考): スマートカードへの攻撃能力の適用, https://www.ipa.go.jp/security/jisec/hardware/documents/CCDB-2013-05-002_J.pdf
  Common Criteriaにおけるハードウェア評価についてのサポート文書
- [CD05] Canright, D.: A Very Compact S-Box for AES. http://www.iacr.org/archive/ches2005/032.pdf
  Canrightの非常にコンパクトなAESのS-Box実装
- [NS06] Nikova, S., Rechberger, C., Rijmen, V.: Threshold Implementations Against Side-Channel Attacks and Glitches. https://securewww.esat.kuleuven.be/cosic/publications/article-847.pdf
  AESのThreshold Implementation について

- CARDIS 2015 Program
  https://wiki.crypto.rub.de/cardis15/program.html
  プレゼンテーションスライドがダウンロード可能