解説

実装攻撃に対抗する耐タンパー技術の動向

通常はコピーできないビデオをコピーするようにプレイヤを改造することを困難にする，IC カード電子マネーを打出の小槌に変えさせないようにするといった耐タンパー技術は，システム実装に絡むセキュリティ技術であり，その内容が非公表であることが多く実態を掴みづらい．しかし，よりセキュアなシステムの構築を目指す立場からは耐タンパー技術に関して体系的な視点を持つことが重要である．本稿ではパソコン用のセキュリティチップ TPM や組込みシステムに対して公表された最近の攻撃事例や研究成果を手掛かりとして，耐タンパー技術の現状と課題を探る．

松本　勉　大石和臣　高橋芳夫
（横浜国立大学大学院環境情報研究院）

実装攻撃と耐タンパー性

本稿では，ハードウェアとソフトウェアからなるシステム（モジュール）を攻撃して何らかの利益を得ようとする者（攻撃者）の存在を仮定し，攻撃に対する耐性を期待するセキュリティ上の機能を考えるが，特に実装攻撃とそれに対抗するセキュリティ能力である耐タンパー性（大きく分けて機能改変困難性と秘密情報守秘性がある）に焦点を当てる（図 -1）．一般にシステムが期待通りの機能を発揮するかどうかを，システムを作る者の立場で，システムを導入または運用する者の立場で，あるいはシステムのエンドユーザとして使う者の立場で見る場合に，それぞれ次の 4 つの視点が必要であろう．

1. システムへの要求は何か
2. システムの仕様が要求を正しく反映しているか
3. システムの設計が仕様を満たしているか
4. システムの実装が設計どおりになされているか

実装攻撃（Implementation Attacks, Tampering）とは，実装されたシステム（モジュール）に対して，上記の 4 視点のあらゆる弱点を突いた攻撃である．

実は，耐タンパー性を有するモノは身近に溢れている．金庫のような強固な筐体や特殊な工具がないと回すことのできない耐タンパーネジなどはその代表例である．また，封印，シール，PET ボトルのキャップ，お菓子の袋，シュリンクラップ，はがきのシールは，攻撃の痕跡を残すタンパー証拠（Tamper Evidence）技術の例である．情報処理技術に関するシステムでは，計測器，電力メータ，水道メータ，ガスメータ，積算走行距離計（オドメータ），タクシーの料金メータ，携帯電話，POS 端末，自動販売機，ATM，電子財布，IC カード，電子署名生成装置，有料放送の復号装置，暗号化コンテンツ復号ソフトウェア，タイムスタンプ生成装置，構造計算ソフトウェアなどなど，限りなく例をあげることができ，これらには，攻撃をしにくく（狭義のタンパーレジスタンス（Tamper Resistance））作られ，攻撃を受けたら少なくともその証拠が残り（Tamper Evidence），できれば攻撃を水際で検出し（Tamper Detection）秘密情報をゼロ化（消去）するなどのタンパー応答性（Tamper Response）を有することが望まれる．タンパー応答性は，いわば，捕まりそうになったスパイが自白剤を打たれて秘密を漏らしそうになるときに服毒自殺をするなどといった対策を，システムに求めるものである．

「ある実用システムがクラックされた」といった報道がなされたとき，指摘された脆弱性がシステムを使う者にとってインパクトがある場合は騒ぎになることが多い．ところが，システムを作る側や導入する側ではリスク分析を行いその弱点を仕様の段階で織り込み済みであって，その弱点にはあえて目をつぶって製品化しているということがままある．そのこと自体はもちろん公表されない．しかし，仮に他の技術や仕組みによりその弱点はカバーできていて事実上セキュリティ上の問題がないとしても，上記の 4 つの事項について何も知らされていないエンドユーザからは看過できないとすれば，システムを作った側や導入・運用した側は，ビジネス上，あるいは社会的な責任を果たす上でも対処が必要となるはずである．

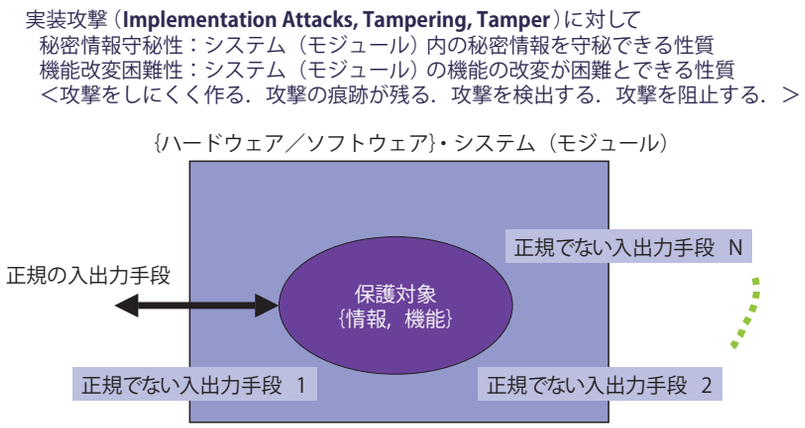

図 -1　ハードウェア／ソフトウェアの耐タンパー性とは

より深刻なのは，システムを作る者がシステムにセキュリティ上要求したいことを，システムを導入する者と緊密に協議をして定め設計・実装したにもかかわらず，それを反映したセキュアな実装ができていなかった場合である．攻撃者がどのような攻撃技術を駆使してくるかを設計や実装の時点で予見できない場合もあるから，単純にシステムを作る者の実力不足であったとはいいきれない難しい問題が横たわっている．実装攻撃に対する耐タンパー性の客観的セキュリティ評価・試験が可能な領域を広げていくような不断の努力が必要とされる．

実装攻撃の分類と事例（その 1）

システム（モジュール）は，それを使用するエンドユーザ自身が攻撃者になり得るという最も厳しい状況でのセキュリティをも考慮しておくことが望ましい．本稿では耐タンパー性が不足していると指摘された事例を数多く具体的に紹介するが，その指摘により，当該システムを作る側が求めていた耐タンパー性が崩れたのかどうかは，上記の理由で不明であり，指摘後に対策が講じられたとしてもそれが非公開であることが多い．

説明の便宜上，攻撃法を図 -2 のように分類してみた．

◆論理攻撃（攻撃タイプ 1）の事例

まず，鍵管理やアクセス制御，プロトコルの設計ミスなどを利用した論理的攻撃法がある．不揮発性メモリを内蔵した CPU 搭載型 IC カード内のプログラムやデータが容易に改ざんされないために設けられているアクセス制御の機構は攻撃対象になる．そのような攻撃の例として，暗号化コプロセッサ IBM4758 の例 [1] が有名である．IBM4758 の複雑な鍵管理用の API コマンドをうまく組み合わせると，Single DES key で暗号化した Triple DES key をエクスポートできるという脆弱性が指摘された．

◆ソフトウェアへの攻撃（攻撃タイプ 2, 3）の事例

実装されたソフトウェアの弱点を突いた攻撃法がある．Content Scramble System（CSS），Advanced Access Content System（AACS）は，それぞれ DVD，次世代 DVD（Blu-ray, HD-DVD）を暗号技術によって保護する方式である．これらの再生ソフトウェアに実装された暗号方式を解析して，本来ならば秘密に保たれるべき暗号鍵を盗み出し，保護を無効化した事例は有名である．

CSS は，ある DVD 再生ソフトウェアのプログラム中に暗号の鍵が暗号化されずに埋め込まれていたため，それを手がかりにして CSS の非公開の暗号アルゴリズムの仕様等が解析され，CSS の暗号によるアクセス制御を回避するためのプログラム DeCSS が 1999 年 10 月に開発された（**攻撃タイプ 2**）．AACS においては，ある次世代 DVD 再生ソフトウェアが再生動作を行う際に，鍵がメモリ上に記録され消去される過程が観察され 2007 年 2 月頃に鍵（Media Key）が突き止められた（**攻撃タイプ 3**）．ただし，AACS は鍵が暴露された場合に備えて鍵を失効する仕組みを有しており，暴露以後に発売される次世代 DVD メディアは失効された鍵を用いては再生されず，被害を限定可能だとされる．これらの事例は，当該暗号方式のソフトウェア実装の秘密情報守秘性が十分ではなかったことを示している．

2008 年 3 月には，Blu-ray ディスクが採用する BD ＋という保護機能の無効化が報じられた．

◆ソフトウェアのリバースエンジニアリング

このような攻撃ができたのは，ソフトウェアをコードとして入手でき，それを解析して機能を分析して攻撃に有用なデータを調べたり攻撃に都合のよいプログラムの

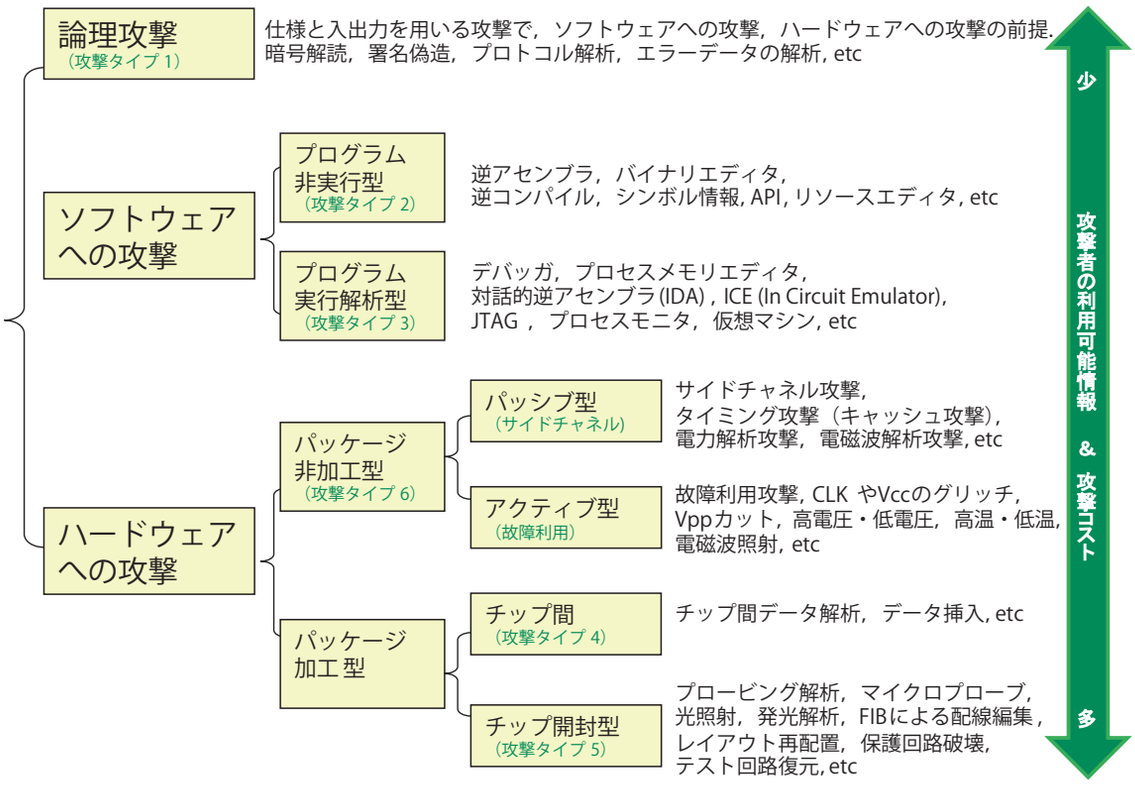

図 -2　システム（モジュール）に対する攻撃法の分類

```
#include <stdio.h>
#define PIN 6397

int main() {
  int entered_pin;

  scanf("%d", &entered_pin);

  if(entered_pin==PIN) {
    printf("OK\n");
    return 0;
  }
  printf("NG\n");
  return -1;
}
```

(a) C ソースプログラム

```
アドレス： マシン語                    オペコード  オペランド
-------------------------------------------------------------------
 401088: e8 f3 00 00 00           call   401180 <_scanf>
 40108d: 81 7d fc fd 18 00 00     cmpl   $0x18fd,0xfffffffc(%ebp)
 401094: 75 15                    jne    4010ab <_main+0x5b>
 401096: c7 04 24 03 20 40 00     movl   $0x402003,(%esp)
 40109d: e8 ce 00 00 00           call   401170 <_printf>
 4010a2: c7 45 f8 00 00 00 00     movl   $0x0,0xfffffff8(%ebp)
 4010a9: eb 13                    jmp    4010be <_main+0x6e>
 4010ab: c7 04 24 07 20 40 00     movl   $0x402007,(%esp)
 4010b2: e8 b9 00 00 00           call   401170 <_printf>
 4010b7: c7 45 f8 ff ff ff ff     movl   $0xffffffff,0xfffffff8(%ebp)
 4010be: 8b 45 f8                 mov    0xfffffff8(%ebp),%eax
 4010c1: c9                       leave
 4010c2: c3                       ret
 4010c3: 90                       nop
```

(b) 実行可能ファイルを逆アセンブルした結果（一部を抜粋）

図 -3　暗証番号チェックプログラム

要所を見つけたりできるからにほかならない．

図 -3 に，入力値があらかじめ登録された暗証番号と一致するか否かを判定するプログラムの一例を示す．同図（a）の C ソースプログラムから作成された実行可能ファイルを逆アセンブラ（GNU objdump）で処理した結果の一部が同図（b）である．ここでは Intel x86 系 CPU のアセンブリを AT&T 表記で表している．入力された値（entered_pin）が PIN（10 進数の 6397 ＝ 16 進数の 0x18fd）と等しいか否かがアドレス 40108d の cmpl で比較され，等しくないときはアドレス 401094 の条件ジャンプ jne によってアドレス 4010ab にジャンプし，等しいときは次のアドレス 401096 に制御が移る．

CPU の仕様書を参照すれば，アセンブリプログラムのインストラクション（オペコードとオペランド）に対応するマシン語が分かり，インストラクションと実行可能ファイルの対応をとることができる．

このように，特別な工夫なしに作られたソフトウェアは，その仕様やプログラミング言語が分かりツールが入手できれば，簡単にリバースエンジニアリングが行える．

どのような仕組みであるかが分かれば，秘密情報取得や機能改変を行うための手掛かりが得られるはずである．

◆機能改変に関する実装攻撃の事例

たとえば図 -3 の暗証番号チェックプログラムにおいては，バイナリエディタで実行可能ファイルを開き，jne のインストラクションに対応する部分を，何もしないインストラクション nop のマシン語に書き換えて保存すれば，暗証番号として入力された値が何であっても次のアドレスに処理が進むようにプログラムを改変できる．

同様に，たとえば携帯電話の制御プログラムを解析すれば，たとえばその SIM ロックを無効化する手掛かりが得られる可能性がある．

やや複雑なシステムに対する機能改変の公表例として，ARM アーキテクチャの CPU が搭載された家庭用 ADSL / CATV 対応ブロードバンドルータを対象とした文献 2）の攻撃事例がある：

1. 攻撃対象を分解し，CPU の型番を読み取り，JTAG エミュレータ（プロセッサに内蔵された JTAG ポートを介して通信するエミュレータ）やパソコンで動作するデバッガ等の機材を入手し，攻撃対象にパソコンを物理的に接続する．
2. デバッガで攻撃対象のバッファオーバフロー脆弱性を特定する．
3. CPU の型番から，それが周辺装置制御機能を有することを突き止める．仕様書等から，その制御機能のメモリマップ，レジスタ，プログラマブル I/O 等の情報を入手し，攻撃対象の制御プログラムの API やシステムコールは用いずハードウェアを直接制御するプログラム“shellcode”を作成する．
4. この shellcode を遠隔地から通信で同型のルータに送り込み，バッファオーバフロー脆弱性を利用し攻撃対象上で実行させる．LED の点滅や ICMP パケットの送信を行うことに成功した．

すなわち，攻撃者作成のコードが実行されないように作られているはずのルータの制御プログラムの機能改変がなされた．この攻撃方法は，図 -2 の分類における攻撃タイプ 1 から 4 までを用いたものといえる．組込み機器の内部の仕様が公開されていなかったとしても，このように攻撃を受ける可能性がある．

システムや装置を構成するソフトウェア（組込み機器の場合にはファームウェアと呼ばれる）がバッファオーバフローを起こさないようにするには，脆弱性を持たないライブラリ（Libsafe など）を使う，バッファへの入力のサイズ確認処理をソースコードに記述する，バッファオーバフローを起こすコードを検出する機構を組み込んだコンパイラ（FailSafe C など）でコンパイルするといった方法がある．また，システムの装置的な対策としては，CPU の名称や型番等を外見から判別困難にする，JTAG や入出力ポートを基板に残さない，といった古典的な方法も一定の効果はある．

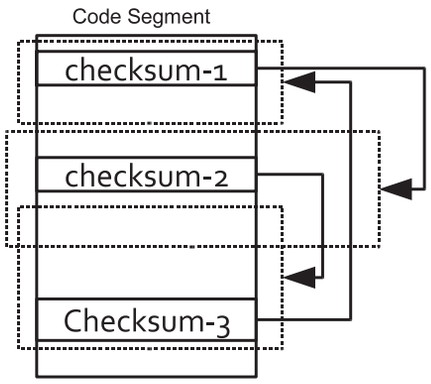

図 -4　自己インテグリティ検証

ソフトウェアの耐タンパー技術

ソフトウェアの耐タンパー性を強化する方法は，逆アセンブラやデバッガを用いた解析に対する対抗策としてソースプログラムやマシン語プログラムの難読化や暗号化，解析ツールの検出等が知られており，文献 3）および文献 4）に解説がある．

本稿では，実行可能ファイルの耐タンパー性を強化する技術として研究の進展が著しい自己インテグリティ検証，自己書換え，テーブルネットワーク型実装についてこれらの解説を補完する．

◆自己インテグリティ検証

これは Self Integrity Verification, Self Checking, Self Hashing, Self Check-Summing などと呼ばれ，プログラムが改ざんされているか否かをプログラム自身が実行時に確認する手法である（図 -4）．改ざんを検出したときは，後続の処理に確率的なエラーを起こす，遅延する，中断し終了するなどの対応をとる．

検証部 checksum-1 等（論文によっては testers, guards と名づけられている）を内部に持つ．これはプログラムのある部分（図 -4 中の破線で囲まれた部分）を対象にしてチェックサム（あるいは暗号的ハッシュ値）を計算し，あらかじめ計算してプログラム中に埋め込んである参照値と比較して，改ざんの有無を確認するプログラ

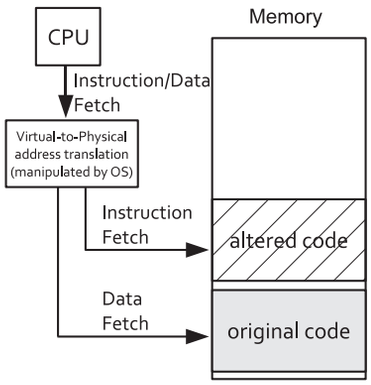

図 -5　ページ複製攻撃の概念図

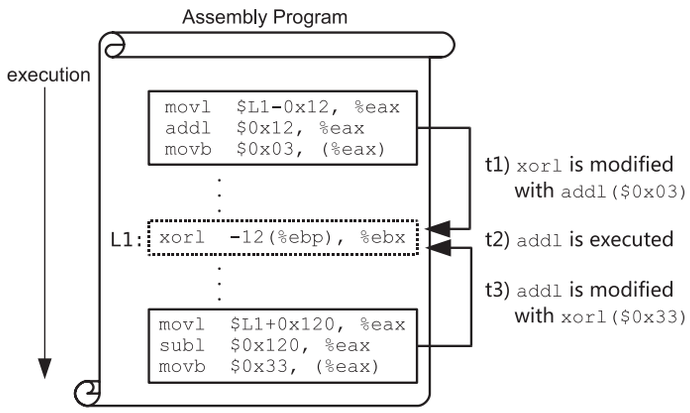

図 -6　偽装プログラム

ム断片である．ある検証部を対象とする別の検証部を内部に持ち，対象と検証部が相互に複雑に依存するように構成される．攻撃者が改変を成功させるには依存関係のつじつまが合うように変更しなければならない．

ただし，適切な相互依存関係を持つように設計する体系的な方法は公表されていない模様である．

なお，自己インテグリティ検証に対しては，攻撃対象のプログラムを本来のメモリアーキテクチャとは限らない別の実行環境で動作させられる場合に汎用的に適用できる攻撃法（ページ複製攻撃，Page Replication Attack）がある(図 -5)．通常のコンピュータでは CPU が参照する仮想アドレスは物理アドレスに変換され，メモリ上のその物理アドレスに記録されているビット列がフェッチされる．同一の仮想アドレスへの参照が異なる物理アドレスへのアクセスとなるように改造を施した OS のもとで，改変されたプログラム部分（図 -5 の altered code）とオリジナルのプログラム部分（図 -5 の original code）をメモリ上に配置し，対象プログラムを実行する．インストラクションフェッチのときは改変されたプログラム部分にアクセスし，インテグリティ検証対象部分の計算を行うためのデータフェッチのときはオリジナルのプログラム部分にアクセスを行うという攻撃である．

◆自己書換え

自己書換え（Self Modifying）は古くからあるプログラミング技法であり，主としてプログラムをコンパクトに書くために広く使われていた．自己書換えプログラムはその作成もデバッグも難しく，最近のパソコンの OS では通常は自己書換えプログラムを実行することはできないこともあり一般的ではなくなったが，プログラムのコードセグメントを書換え可能に設定することにより最近のパソコンの OS でも実行することはできる．

機能改変困難性のためにプログラムの自己書換えを用いる方法の 1 つが文献 5）で提案されている．プログラムの中の命令を本来の命令とは異なる命令に偽装しておく．実行時に自己書換えが行われ本来の命令に書き戻され，本来の命令が実行され，その後に再び自己書換えが行われ偽装された命令に書き戻される(図 -6)．プログラムの実行時のある期間にだけ本来の命令が出現するので，プログラムを動作させずに局所的に読むだけでは本来の動作を正しく理解することが困難となる．逆アセンブラを用いる場合には，自己書換えを行うルーチンを探し出して解析する必要があるため，広い範囲の解析を強いられる．

ただし，デバッガ等を用いた動的な解析への耐性評価がはっきりしていない点は課題であると考えられる．

自己インテグリティ検証プログラムにおけるページ複製攻撃の検出のために，自己書換えプログラムが実行されたことを確認する処理を自己インテグリティ検証プログラムに追加し，自己書換えが反映されていないときはページ複製攻撃を受けていると判断する方法がある [6]．

プログラムの開発や解析に制御フローグラフ（Control Flow Graph）が一般に有用であるが，プログラムの状態を示す情報を追加するなどの拡張を行って自己書換えプログラムを表現できるようにした State Enhanced Control Flow Graph が提案されている [7]．

◆テーブルネットワーク型実装

共通鍵ブロック暗号の鍵を固定した実装が対象であり，プログラムファイルにも実行時のメモリにも鍵が現れないようにしてソフトウェアに対するあらゆる静的・動的解析による鍵導出の困難化を目指す．すなわち，対象とする暗号の暗号化／復号アルゴリズムを変形し，すべての演算や変換を，テーブル（入出力の対応表）を接続したネットワーク構造で表現する．これは，アルゴリズムを組合せ論理回路に展開することができることから必ず行

えるが，サイズの爆発を抑えることが重要である．さらに秘密の全単射（とその逆変換）を用いて，テーブル間に現れるデータを符号化し，同時にテーブル自体も整合性がとれるように変換する．この方法を Table-Network Implementation または White-box Implementation という．任意の平文／暗号文を入力したときの途中の計算結果から鍵を推定することで総当りよりも少ない計算量で内部の鍵を突き止める攻撃や，その対策などが研究されている[8]．

実装攻撃の分類と事例（その 2）

図 -2 のハードウェアへの攻撃について詳しく見てみる．

◆チップ間攻撃（攻撃タイプ 4）の事例

複数の IC チップを組み合わせて使用する場合には，パッケージを加工したり筐体カバーを開けたりして，チップ間のインタフェースのモニタや，コマンド・レスポンスに介入するタイプの攻撃法がある．チップ間のバス配線をモニタする攻撃例としては，家庭用ゲーム機 X-BOX の例[9]がよく知られている．

◆チップ開封型攻撃（攻撃タイプ 5）の事例

ハードウェア，特に集積回路（IC チップ）は樹脂やセラミクスのパッケージに入っていて内部の解析は難しそうに見える．IC チップの微細な構造の観察には専用装置が必要なので，IC チップの解析の準備に必要な機材や費用はソフトウェアの場合に比べて高いと考えられる．

しかし IC チップの解析は開発過程での故障解析や製品化後の特許侵害の調査等に必須であるため，解析技術が開発されていて，チップ解析を専門とする会社もある．そのような技術を利用すれば，パッケージを開封して IC チップを取り出し，さらにチップをスライスして配線を調べることができる．つまり手間はかかったとしても，IC チップの回路図を復元することは一般に可能である．FIB（集束イオンビーム）加工装置を使用して配線を変更して機能改変を行ったり，配線上にパッドを作成して動的にチップ内の信号をモニタして秘密情報を取得したりすることもできる（図 -7）．

1997 年に発表された IC カード H8/3101 の例[10]では，Mondex 電子マネーの実証実験で使用されていたこの IC カードを開封し，プローブ（先の細い針）でヒューズをショートするとテストモードになって，暗号鍵も含めた全メモリがダンプできたという．

2007 年 12 月には，CRYPTO-1（MIFARE Classic カードのストリーム暗号）がリバースエンジニアリングによりアルゴリズムが特定され，その脆弱性が明らかにされた．CRYPTO-1 はアルゴリズム非公開であったが，回路が特定され，分析された結果，2008 年 4 月までに，1 個の IV（初期ベクトル）と暗号文があれば 200 秒で，4 個の IV と暗号文のペアがあれば 12 秒で秘密鍵を計算できると発表されている．いったん脆弱であることが分かると，プロトコル解析のみによるアルゴリズムの特定も行われ，CRYPTO-1 のアルゴリズムはハードウェアでは秘匿できていないことが明らかになった．

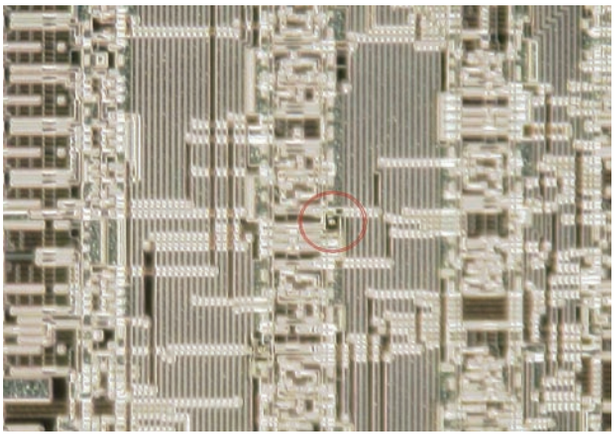

図 -7　チップ表面の様子：比較的低倍率でも回路の配線やバスが見える．図中の赤丸で囲んだ部分が，FIB で形成した信号モニタ用パットである．ここにプロービングステーションを使用してニードルをあてて信号を取り出せる（松本研究室撮影）．

◆パッケージ非加工型攻撃（攻撃タイプ 6）の事例

6 番目の類型として最近注目されているパッケージを開封せずに行う攻撃法がある．IC カードが早くから普及した欧州では，1990 年代前半までに VPP カットや CLK グリッチなどによって，メモリや CPU の動作にエラーを生じさせ，所定のアクセス制御を回避するなどの故障利用攻撃が知られており，そのための器具やダミーカードが販売されている．

1990 年代後半からは，文献 11）などで提案されたサイドチャネル攻撃とその対策技術が盛んに研究されている．サイドチャネル攻撃は，IC チップ動作中の消費電力などを観察し，チップ内部の処理内容を特定して秘密情報を推定する攻撃法である．特に IC カードのようなモジュールには効果的である．

Trusted Platform Module（TPM）

重要性を増しているセキュリティチップの一種に Trusted Platform Module（TPM）がある．その仕様[12]は，2003 年に結成された民間の標準化団体 Trusted Computing Group（TCG）が定めており，2006 年頃からノート PC への搭載が始まり，最近のパソコンには標

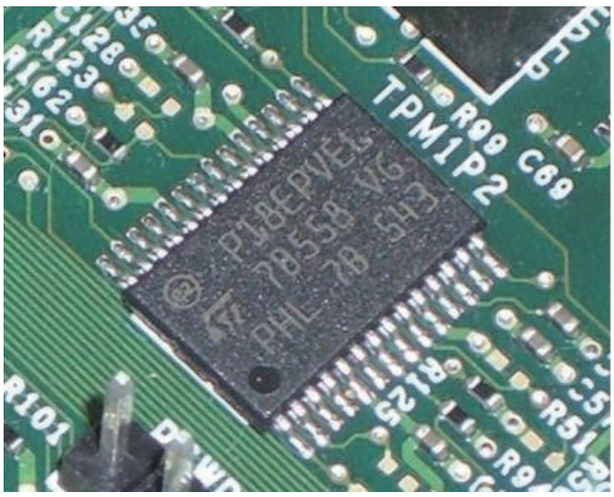

図 -8　マザーボード上の TPM チップ（松本研究室撮影）

準的に搭載されている．

TPM は物理的には TSSOP-28 ピンのパッケージで実装されることが多く，マザーボード上では LPC バスに接続されている（図 -8）．内部に CPU，不揮発メモリ，公開鍵暗号 RSA の鍵生成とメッセージ変換機能，ハッシュ関数 SHA-1 の機能，乱数生成器などを持つ．

TPM の用途には，ファイルの暗号化や署名生成（および鍵管理）と，Trusted Boot などの仕組みを利用した機器認証（アテステーション）などがあげられている．

TPM とその利用システムの解説には，文献 13）および文献 14）がある．

TPM を使ったファイル暗号化の例を図 -9 に示す．TPM は内部に公開鍵暗号 RSA の公開鍵と秘密鍵のペア Storage Root Key（SRK）を記憶している．この鍵は TPM 内部で生成され，秘密鍵は平文のままで TPM 外部に出力されることはない．ここでファイルの暗号化は共通鍵暗号 AES を用いて行うとする．使用する AES の鍵（図 -9 の R）は，Storage Key（SK）で暗号化され，SK は SRK で暗号化される．機器認証で使用する RSA 鍵ペア Attestation Identity Key（AIK）も SRK で暗号化されて TPM の外部（の HDD など）に保存される．外部に保存される RSA 鍵は，公開鍵の法 N は平文のままで，N の秘密の素因数である P は SRK で RSA-OAEP 方式により暗号化されたものである．N（= PQ）と P があれば Q は N/P で求められる．公開鍵のベキ指数 E はシステム共通の値 65537（$= 2^{16} + 1$）に固定である．

TPM には，Platform Configuration Register（PCR）と呼ばれる特殊な役割を持ったレジスタが複数ある．このレジスタは 160bit 長で，チップのリセット時に 0（ゼロ）クリアされる．その値は，TPM 内で PCR の現在の値とコマンドのパラメータで受信したデータを連結してハッシュした値で更新できるだけであり，直接的に値を設定することはできない．この PCR はその名のとおり機器認証を実現するために使われる．BIOS からブートセクタ，OS のバイナリコードまで起動時に実行されるコードを順番に PCR に反映させ，最後に PCR の値に TPM 内の AIK の秘密鍵で署名を付与して，サーバなどに送信することで実行環境が改ざんされていないかを調

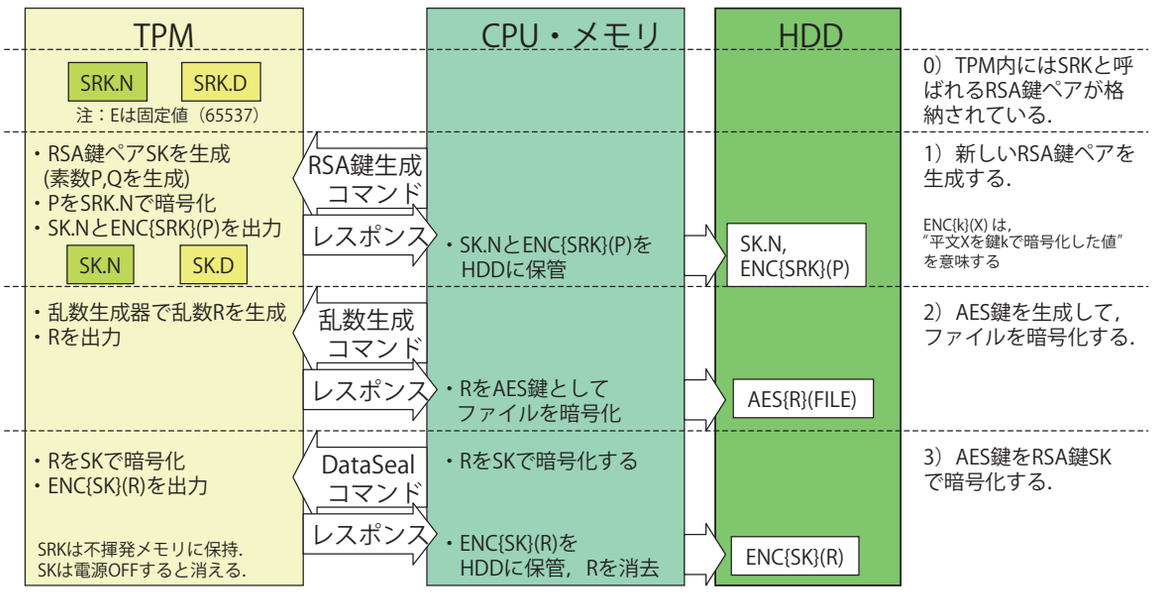

図 -9　ファイル暗号化の仕組み

べる．

TPM 利用システムがセキュアであるためには，TPM 内部に記憶している秘密鍵が取り出されないこと，あるいは PCR が都合よく改ざんされないことが必要である．つまり，TCG の TPM 仕様書では TPM への耐タンパー性についての明示的な要求がなされていないものの，TPM には耐タンパー性が必要であると考えられる．

ここで TPM 利用システムに対する攻撃例を紹介する．

◆攻撃例 a）論理攻撃（2005 年 5 月）

TPM のプロトコル仕様上，リプレイ攻撃が可能であるという指摘がなされた [15]．TPM の認証付きプロトコル OIAP には毎回値が更新される nonce データがあり，典型的な中間者攻撃はできないが，別のセッションには引き継がれないため，コマンドの実行順番を入れ替えることができる．その他，BIOS の改ざん，HDD に格納されている BOOT LOADER のすり替えなど，Trusted Boot を無効化する攻撃法が複数提案されている [16]．

◆攻撃例 b）LPC バスのモニタ（2005 年 9 月）

TPM は，マザーボード上の単独のチップとして実装されていることが多い．図 -8 のようにマザーボード上にハンダ付けされている TPM を見つけることは容易で，チップのピンアサインはデータシートなどで，信号の定義は LPC バスの仕様書などで調べられる．そして図 -9 のように AES 鍵 R は TPM から平文で出力されるため，LPC バスをモニタすることで取得できる．実際に LPC バスを観察した例 [17] では LPC バスを効率的にモニタするために，TPM へのパケットだけを取り出すフィルタ装置を FPGA で作成して，データを収集している．その結果，ブート時の TPM へのアクセス状況を把握でき，ファイル暗号化用の鍵の特定に成功している．

◆攻撃例 c）TPM リセット攻撃（2007 年 6 月）

チップ間のインタフェースに介入する攻撃として，LPC バスのリセット信号（LRESET）を操作して，TPM チップをリセットする攻撃 [18] がある．この攻撃のターゲットは，TPM の PCR である．TPM はリセット後，PCR を 0（ゼロ）クリアするため，その後，攻撃者は偽装したいマシンの BIOS や OS のコードを TPM に送ることで PCR を操作できる．この攻撃方法は実験手順などを解説する動画ファイルがインターネットで公開されている．

◆攻撃例 d）冷却ブート攻撃（2008 年 2 月）

メモリは電源オフしてもすぐには値を失わない．この現象を利用して，ファイル暗号化に使用した秘密鍵（図 -9 の R に相当）がメモリ上にある間に電源を遮断し，メモリを冷却して取り外してデータを復旧させることで秘密鍵を得る攻撃がある [19]．これによれば，DRAM 表面をマイナス 50 度まで冷却すると，電源オフ後 10 分経過しても 99％のビットが残存しメモリ上の特徴的なデータパターンを検索して秘密鍵を特定できたという．

◆攻撃例 e）サイドチャネル攻撃（2008 年 3 月）

我々横浜国立大学松本研究室のハードウェアセキュリティ研究チームは，チップの動作を反映する消費電力分布などの漏えい情報を活用するサイドチャネル攻撃の考え方で，ある特定の TPM を解析し，そのチップの RSA 鍵生成処理中の消費電力を測定して得たデータからチップが生成した秘密鍵を求められることを実証した．

RSA 鍵生成には素数を 2 個求める処理があり，候補の整数に対して次々と素数判定をして素数が見つかったら終了する方法が普通である．素数判定はミラーラビン法のように，ベキ乗剰余演算を繰り返すものが主流である．

図 -10（a）～（c）が鍵生成時の電源電圧の時間変動を測定したものである．この波形から，素数生成のためにベキ乗剰余演算を繰り返し実行していて，さらにベキ乗剰余演算を詳細に観察すると，凹凸の様子からベキ乗剰余演算をバイナリ法（SQR-MUL 法）で実行していることが推定された．これを検証するために，ベキ乗剰余演算の波形からベキ指数の推定を試みたところ，その値は図 -10（d）のように時間とともに単調増加していた．つまり，候補の整数をインクリメントしながら素数が見つかるまで素数判定を繰り返していることが分かった．

そこで上記のベキ指数の推定値の 1 つをインクリメントしていき最初に素数になった値を P としたところ，P は RSA 公開鍵 N の素因数であることが確認できた．つまり N に対応する RSA 秘密鍵が特定できた．

このように，この TPM が生成する RSA 鍵はすべて解読できるという意味でこのチップの耐タンパー性が十分でないことが分かった．これは我々がターゲットとしたチップについていえることであって他のチップについては何らの主張をするものではない．

ハードウェアにおける耐タンパー技術

耐タンパー技術の実際はその詳細がメーカから明らかにされることは少ない．耐タンパー性が低いために解読されて，明らかになるほかは，カタログやデータシート等の資料でキーワードが示される程度である．稀には攻撃に対するセキュリティを有することをアピールするために対策内容が公表されることもある．ここでは対処すべき攻撃タイプごとに耐タンパー技術を紹介する．

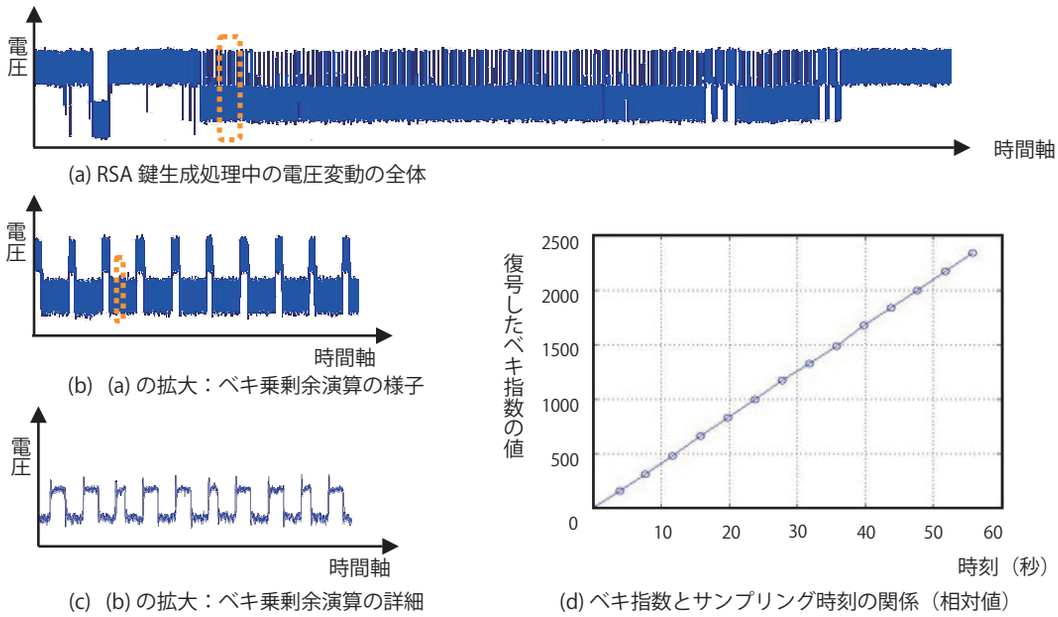

図 -10　RSA 鍵生成処理中の電圧変動波形（松本研究室測定）

◆技術 1）設計ミスやソフトバグによる脆弱性への対処

攻撃例 a で見たようなプロトコルレベルのセキュリティを確保するには，設計時に形式的検証等の手法によりセキュリティを確認することが役立つ．セキュリティ評価の国際標準 ISO/IEC 15408 の枠組みでは，評価保証レベル EAL5 以上では準形式的設計が，また EAL7 では形式的設計が要求される．

◆技術 2）チップ間攻撃への対策

攻撃例 b への対策としては，筐体カバーの施錠やカバーのオープンをセンサスイッチで検出して動作停止するなどがある．IBM のパソコン ThinkCentre S50 では筐体にそのための器具が備えられている．IBM4758 では基板を電磁遮蔽とタンパー検出用メッシュで覆い，その外側を封止材で埋めているという [20]．封止材の中に細いワイヤからなるセンサを入れ，開封して回路を動的に解析することを困難にする仕組みである．TPM の場合には，LPC バスの盗聴に対するソフトウェア的な対策として暗号化がある．しかし攻撃例 c のようにリセット制御線を操作する攻撃はそれだけでは防げない．対策としてブートプログラム OSLO（Open Secure Loader）は，AMD の sk_init 命令を使って PCR のリセットとローダの PCR 値を求めることで，BOOT ローダのすり替え攻撃に対処できるという．

そのほかに攻撃例 b や c への対策として，TPM を他のデバイス（SuperIO や CPU）に統合してワンチップ化するなどのアーキテクチャレベルの提案がある．たとえば，AEGIS などのセキュリティプロセッサの提案がある．

なお，攻撃例 d のようにメインメモリ（DRAM）に残留する電荷の読み取りが可能であるため，タンパー応答のためには，積極的にゼロ化することが必要である．残留メモリの現象は古くから知られており，暗号モジュールのセキュリティ要件を定めた FIPS140-2 では，その付属文書にてタンパー応答のために鍵のゼロ化を行う際には電源 OFF によって揮発メモリの値が消えることだけでは不十分であるとしている．タンパー応答性を備えていてもモジュールの電源をオフするとともにドリルで筐体に穴を開けてゼロ化回路を破壊し，急激に冷凍すれば，RAM の電荷を測定できて中味が読み出せるのではないかと先にあげた Anderson は考察している．

◆技術 3）チップ開封型攻撃への対策

IC チップの回路表面をセンサメッシュで覆い，下の配線層を目視できなくする対策がある．Infineon 社の TPM にはこのような対策 "Active Sheild" が施されていて，異常を検出したらアラームをあげると説明されている [21]．図 -11 にパッケージの一部を開封したある TPM を，図 -12 にそのチップ表面の様子の顕微鏡写真を示す．一方，IC カードのチップであっても回路配線が顕微鏡で見える例がある（図 -13）．

◆技術 4）パッケージ非加工型攻撃への対策

パッケージ非加工型の攻撃法には，故障利用攻撃とサイドチャネル攻撃がある．故障利用攻撃への対策には，低クロック動作解析などの防止や低電圧やグリッチなど

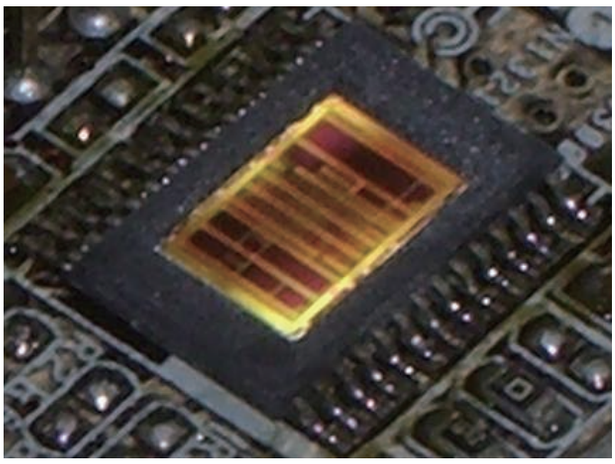

図 -11　TPM チップを一部開封した様子（松本研究室撮影）

による動作不良を回避するための各種センサの導入などがある．

特に中国剰余定理（CRT）を用いた RSA 高速実装方式を使う場合には故障利用攻撃への対策が重要で，TPM 仕様書では CRT 採用時には故障利用攻撃への対処を必須としている．

この対策法としては，単純には，同じ演算を 2 回行い，結果を比較する，あるいは RSA の復号（d 乗）の場合には暗号化（e 乗）してもとの暗号文と比較するという方法がある．ほかには，たとえば文献 22）で示された方式がある．32bit 程度の乱数 R を使って，$Sp' = m^d \bmod (P \cdot R)$ と $Sq' = m^d \bmod (Q \cdot R)$ を求める．そして $Sp' \equiv Sq' \pmod R$ が成立することを確認して，$S = crt(Sp', Sq') \bmod N$ とする．ここに crt で中国剰余定理を用いた法の合成式を表す．Sp', Sq' の計算時にエラーが発生した場合には高い確率で $Sp' \equiv Sq' \pmod R$ が成立しない点がポイントである．加えて，d でなく d mod P を使う方式やサイドチャネル攻撃対策との合体などの改良が複数提案されている．

一方，TPM 実製品では中国剰余定理を用いず RSA を実装するという単純な対策を採用したチップも複数ある．

攻撃例 e はサイドチャネル攻撃としては単純電力解析攻撃に分類される． サイドチャネル攻撃に対しては，ゲートレベルの対策技術とアルゴリズムレベルの対策技術とがある．保護対象によらず適用可能であることからゲートレベルの対策が根本的であるという意見もある．ただし，攻撃例 e のような場合については，アルゴリズムの変形による対策で効率のよいものがある可能性がある．

サイドチャネル攻撃対策の議論は近年我が国で盛んに行われ，日本規格協会情報技術標準化研究センターの耐タンパー性標準化調査研究委員会の報告書[23]やCRYPTREC 暗号モジュール委員会の報告書[24]等に有益な情報がある．

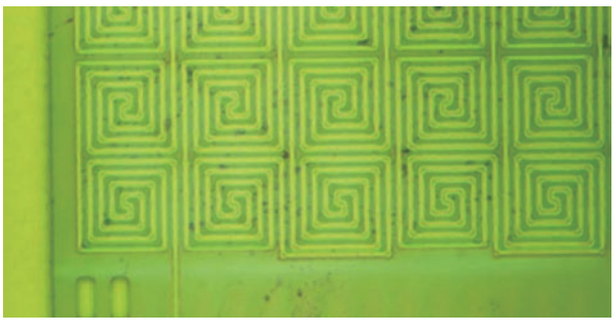

図 -12　TPM の例：チップ表面が一筆書きのラインで覆われていて，メタル層が目視できない（松本研究室撮影）．

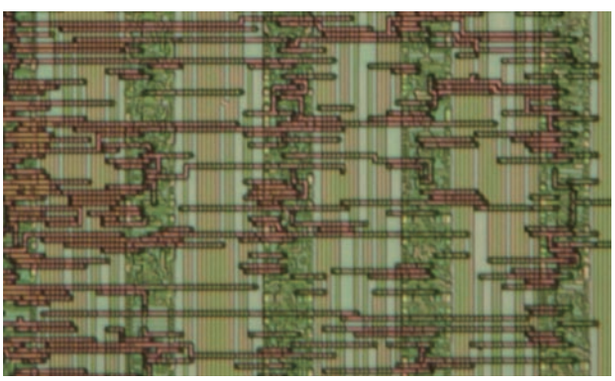

図 -13　IC カードの例：ナイフ 1 本でチップを削り出すことができ，回路配線を目視できた（松本研究室撮影）．

耐タンパー性の試験・評価の充実に向けて

仕組みを公表しても十分な耐タンパー性を持つと示せるシステムを作るための低コスト技術の開発が究極の目標であろう．仕組みを公表というブレークスルーは突破できなかったとしても，少なくとも実装攻撃に関する技術の体系化を図り，既存の個々の実装攻撃方法のどこまでは対策を打てるか／打つかをシステムの開発者／利用者が議論できるようにしていくことが必要であろう．

そのためにはシステム（モジュール）を導入する側が耐タンパー性に関する要求をメーカに的確に出せるようにすることが肝要である．併せて対象システムが所望の耐タンパー性を有するかどうかの試験・評価を行うための技術を充実させていくことも重要である．

幸い，暗号を実装したソフトウェアまたはハードウェアである暗号モジュールについては，「JIS X 19790 セキュリティ技術ー暗号モジュールのセキュリティ要求事項」（FIPS 140-2 をベースとして策定された ISO/IEC 19790 に対応する“国際一致規格”）で要求事項が定められ，これに基づく暗号モジュール試験認証制度 JCMVP が 2007 年より本格運用を行っている．一般の IT 関連

製品に対しては情報技術セキュリティの評価基準（いわゆるコモンクライテリア CC）に基づく評価認証制度である「IT セキュリティ評価及び認証制度」が整備されているが，IC チップについては，CC を補足する JIL（Joint Interpretation Library）文書に基づく評価認証が欧州を中心になされている．TPM 製品の多くはこの枠組みで CC 認証を受けている．また，ペイメント用カードの読み取り端末についてはセキュリティの業界基準がありそれに基づく認証制度が設けられている．

このように制度は整備されつつあるので，最新の研究成果に基づき，システムの応用分野やソフトウェアやハードウェアの技術分野に応じた具体的で詳細なセキュリティ基準を充実し，耐タンパー技術の試験・評価技術を継続的に改善していくことが望まれる．このことはより優れた耐タンパー技術を生む素地ともなるはずである．

参考文献

1）Bond, M.: Attacks on Cryptoprocessor Transaction Sets, Cryptographic Hardware and Embedded Systems - CHES 2001, LNCS, Vol.2162, pp.220-234, Springer (2001).
2）Ukai, Y. : JTAG エミュレータを利用したリアルタイム OS ベースの組み込みシステム Exploit，PacSec.JP/core05 Conference (Nov. 2005).
3）石間宏之，亀井光久，齋藤和雄：ソフトウェアの耐タンパー化技術，情報処理 , Vol.44, No.6, pp.622-627 (June 2003).
4）門田暁人，Thomborson, C.：ソフトウェアプロテクションの技術動向（前編，後編），情報処理，Vol.46, No.4, pp.431-437（前編），Vol.46, No.5, pp.558-563（後編）(Apr. /May 2005).
5）神崎雄一郎，門田暁人，中村匡秀，松本健一：命令のカムフラージュによるソフトウェア保護方法，電子情報通信学会論文誌，Vol.J87-A, No.6, pp.755-767 (June 2004).
6）Giffin, J. T., Christodorescu, M. and Kruger, L. : Strengthening Software Self-Checksumming via Self-Modifying Code, ACSAC '05 : Proceedings of the 21st Annual Computer Security Applications Conference, pp.23-32, IEEE Computer Society (2005).
7）Anckaert, B., Madou, M. and Bosschere, K. D. : A Model for Self-Modifying Code, Information Hiding : 8th International Workshop, IH 2006, LNCS, Vol.4437, pp.232-248, Springer (2007).
8）Matsunaga, A. and Matsumoto, T. : Security Evaluation of a Type of Table-Network Implementation of Block Ciphers, Advances in Computer Science - ASIAN 2006, LNCS, Vol.4435, pp.1-12, Springer (2008).
9）Huang, A. : Keeping Secrets in Hardware: the Microsoft X-BOX Case Study, Cryptographic Hardware and Embedded Systems - CHES 2002, LNCS, Vol.2523, pp.355-430, Springer (2003).
10）Bovelander, E. : Smart Card Security - How can you be so sure?, Advances in Cryptology - EUROCRYPT'97 (12 May 1997).
11）Kocher, P., Jaffe, J. and Jun, B.: Differential Power Analysis, Advances in Cryptology - CRYPTO'99, LNCS, Vol.1666, pp.388-397, Springer (1999).
12）TPM Main Part 1 Design Principles, Specification Version 1.2 Level 2 Revision 103 (9 July 2007).
13）中村智久 , 東川淳紀 : PC 搭載セキュリティチップ (TPM) の概要と最新動向，情報処理，Vol.47, No.5 pp.473-378 (May 2006).
14）上杉忠興，坏　毅，宗藤誠治，吉濱佐知子 : Trusted Network Connect - TPM の利用管理技術の動向，情報処理，Vol.48, No.11, pp.1232-1241 (Nov. 2007).
15）Bruschi, D., Cavallaro, L., Lanzi, A. and Monga, M. : Attacking a Trusted Computing Platform, Technical Report RT 05-05, Università degli studi di Milano, Italia (2005).
16）Kauer, B. : OSLO : Improving the Security of Trusted Computing, 16th USENIX Security Symposium (6-10 Aug. 2007). http://www.usenix.org/events/sec07/tech/kauer.html
17）Knursawe, K., Schellekens, D. and Preneel, B. : Analyzing Trusted Platform Communication, CRyptographic Advances in Secure Hardware - CRASH2005 Workshop (7 Sep. 2005).
18）Sparks, E. R. : A Security Assessment of Trusted Platform Modules, Technical Report TR2007-597, Dartmouth College Computer Science (June 2007).
19）Haldermany, J. A., et al. : Lest We Remember : Cold Boot Attacks on Encryption Keys (21 Feb. 2008). http://citp.princeton.edu/pub/coldboot.pdf
20）Anderson, R. : Security Engineering - A Guide to Building Dependable Distributed Systems, John Wiley & Sons Inc (22 Jan. 2001).
21）Brandl, H. : Deep Insides the TPM, TRUST 2008 Educational Event (10-13 Mar. 2008).
22) Shamir, A. : How to Check Modular Exponentiation, EUROCRYPT' 97 Rump Session (11-15 May 1997).
23）http://www.jsa.or.jp/stdz/instac/committe/tamper-resistance/TSRC.pdf
24）http://www.cryptrec.go.jp/

（平成 20 年 5 月 28 日受付）

松本　勉（正会員）
tsutomu@ynu.ac.jp

横浜国立大学大学院環境情報研究院教授．産業技術総合研究所情報セキュリティ研究センター研究顧問．1986 年東京大学大学院工学系研究科博士課程修了．国際暗号学会理事．情報・物理セキュリティ分野を楽しむ．

大石和臣
oishi@mlab.jks.ynu.ac.jp

横浜国立大学大学院環境情報学府博士課程後期在籍．カリフォルニア大学サンタバーバラ校大学院修了（M.S.）．暗号と情報セキュリティ，特に耐タンパーソフトウェアの研究開発に興味を持つ．

高橋芳夫
takahasi@mlab.jks.ynu.ac.jp

横浜国立大学大学院環境情報学府博士課程後期在籍．暗号実装のセキュリティ評価，サイドチャネル攻撃に関する研究に従事．CRYPTREC 暗号モジュール委員会委員．電子情報通信学会正会員．