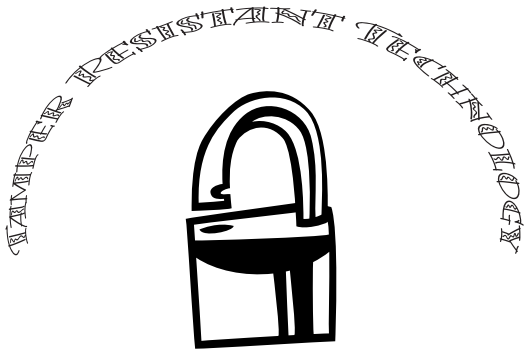
事例

ソフトウェアの耐タンパー化技術



PCおよびインターネットといったIT技術が普及するに伴い，
ソフトウェアの実装上の安全性が問題となりつつある．
たとえば，ソフトウェアDVDプレーヤが解析され，
DVDに記録されている映像データをPCにコピーする
ツールが作成された事件がある．
この事件に前後し，
ソフトウェアを解析や改造から保護する技術，
すなわちソフトウェアの耐タンパー化技術の必要性が
改めて認識されてきた．
本稿では同技術の概要を述べ，
業界の動向・事例を紹介する．

石間宏之（富士ゼロックス（株））
Hiroyuki.Ishima@fujixerox.co.jp
亀井光久（富士ゼロックス（株））
KAMEI.Mitsuhisa@fujixerox.co.jp
齊藤和雄（（株）アクセスチケットシステムズ）
k.saito@accessticket.com

◇ソフトウェアの耐タンパー化技術

耐タンパー化技術（tamper resistant technology）とは，もともとハードウェア分野の言葉であり，ハードウェアに対する物理的な不正工作に対して，耐性を持たせるための技術を指す．たとえば機密情報を含む電子回路は，基板上の接点に測定器を接続することで，その機密を窃取できるかもしれない．この回路に耐タンパー性を持たせるには，1つの方法として開封が困難なケースに電子回路を収め，ケースを無理やり開けようとした際，ケースが物理的に変形し回路上の情報が物理的に破壊されるよう設計しておくことができるであろう．このようなハードウェアに対する耐タンパー化技術に加え，ソフトウェアに対する耐タンパー化技術が近年注目されてきている．まずニーズの例を挙げて解説を行う．

ソフトウェアの改造

以前から一般化しているソフトウェアの販売方法として，ダウンロード販売がある．販売者側は，CD-ROMや紙のマニュアル，箱といった物理的コストや流通コストをかけずに済み，不良在庫のリスクがなくなる．最近ではADSLに代表される高速常時接続のインターネット環境が一般家庭にも普及し，数十メガバイトに及ぶソフトウェアでも，ダウンロード販売の障害にはならなくなりつつある．ダウンロード販売では，通常「体験版」と呼ばれる，機能や利用期間・利用回数などを限定したバージョンを顧客にダウンロードと試用をしてもらい，購入を希望した顧客に対して料金と引き換えに「シリアル番号」を通知する．シリアル番号を受け取った顧客は，ソフトにシリアル番号を入力することで，限定されていた機能や利用期間，利用回数を解除し，以後「正規版」としてそのソフトウェアを利用することができる．このようなソフトウェアは，シリアル番号の正当性をチェックするコードを内部に含んでおり，入力されたシリアル番号が正しければ正規版として動作するようプログラムされている．ここで，シリアル番号を入力せずに不正に「体験版」を「正規版」として利用しようと考えた不正行為者は，以下の手順を試みるであろう．

1. シリアル番号の正当性を判断している条件分岐命令を見つける．
2. シリアル番号を入力していない場合の分岐の流れを観察する．
3. 2で観察した分岐とは逆に分岐するように，条件分岐命令を絶対分岐命令で置き換える．

以上の改造が成功すると，シリアル番号を購入しなくても正規版として利用できるようになる．このような解析や改造に対して対策を講じていない場合，すなわち耐タンパー化されていない場合，特殊な知識と経験がある者にとって不正利用は，「朝飯前」のことであるかもしれない．

秘密情報の窃取

昨今におけるコンピューティングパワーの増大に伴い，従来はハードウェアでしか実現できなかったさまざまな機能をソフトウェアで実現できるようになってきた．身近な例として動画再生機能が挙げられる．ソフトウェアで実現されたDVDプレーヤを使って，パソコン上でDVD映画をご覧になったことのある読者も多いのではないだろうか．市販のDVD映画はその映像データが暗号化されており，再生するソフトウェアはデータを復号処理している．復号には，ソフトウェア内部に秘密情報として埋め込まれた復号鍵を使用している．この復号鍵を何者かに取り出されると，映像データを自由に取り出すことができ，不正コピーが可能になる．そして，それは現実に起きた．1999年，ノルウェーの15歳の少年がDVDの複製防止システムであるContent Scrambling System（CSS）を解読した．某ソフトウェア会社が作成したソフトウェアDVDプレーヤが，解析に対し対処を行っていなかったため，少年はプレーヤの内部の解読に成功した．そしてその解読結果を利用したDeCSSという名のプログラムがインターネット上で公開され，これを使えば誰でもDVDビデオ映像をコピーできるようになった．これに対してDVDの著作権管理団体である，DVD Copy Control Association（DVD CCA）は訴訟を起こしたが，ノルウェーの裁判所は2003年初頭，上記少年に対して無罪判決を言い渡した．「合法的にDVD映画を入手した者は，たとえ製作者が想定しなかった方法であったとしても，それを再生する権利を持っている」ことを認めたのである．

解析からの保護の必要性

前述の例のように，十分な知識と経験を持つ者にはソフトウェアの内部を解析することは可能であり，ソフトウェアが改造されたり内部の秘密情報が窃取されることで，権利者の利益が損なわれる可能性がある．そのための防御技術として，ソフトウェアの耐タンパー化技術が必要となる．本稿で述べるソフトウェアの耐タンパー化技術以外に，ソフトウェアとドングル（dongle）と呼ばれるハードウェアを組み合わせて解析を防ぐ方法もある．秘密情報はあらかじめドングル内に収められており，ソフトウェアを使用する際にはドングルをPCに接続しておく，という使い方をする．しかし，ドングルとPC間の通信の解析が考えられ，いずれにしてもソフトウェア側を耐タンパー化することが重要となる．また，ドングルを使う方法はユーザの利便性を下げ，コストの面でも不利となる．

◇解析技術

耐タンパー化技術の詳細に触れる前に，まずソフトウェアの解析手法から見てみる．ソフトウェアの解析手法は，ソフトウェアを動作させずに行う静的解析と，実際に動作させながら行う動的解析とに分類できる．

ディスアセンブラを用いた解析（静的解析）

ソフトウェアの実体は，CPUに対する命令とデータを表したビット列の並びといえる．これらは，「マシンコード」あるいは「マシン語」などと呼ばれている．通常マシンコードは16進数の並びとして表現されるが，それぞれの数字がどのような命令を意味しているのかを判読することは困難な作業である．そのマシンコードを可読性のあるアセンブリ言語に変換するツールがディスアセンブラ（disassembler：逆アセンブラ）である．アセンブリ言語はマシンコードと1対1に対応し，16進数の羅列に比べてはるかに可読性が高い．また，呼び出しているAPI（Application Programming Interface）や，システムコールの名前や，静的に保持しているデータを表示させることもできる．これらのディスアセンブラの機能を利用し，ソフトウェアの内部構造を把握し命令列を詳細に調べることができる．

デコンパイラを用いた解析（静的解析）

デコンパイラ（decompiler）は，マシンコードやバイトコードなどをソースコードに復元するツールである．マシンコードをソースコードに復元することは困難な課題であり，実用レベルの効率的な解析に役立つレベルのものは見当たらないようである．なぜなら，一般ユーザにリリースされるソフトウェアは通常はシンボル情報が削除されており，関数名や変数名などの手がかりを得ることができない．また，通常コンパイル時の最適化処理により，より複雑なマシンコードに変換されており，デコンパイルできても複雑で読みにくいソースコードになってしまう．一方，昨今メジャーなソフトウェア開発言語の1つとなっているJavaに関しては，実用になるデコンパイラが存在している．Java言語で記述されたソースコードをコンパイルした結果はクラスファイルと呼ばれるファイルとして生成され，仮想マシン（VM：Virtual Machine）上で動作させる実行コード（バイトコード）と，シンボル情報を含んでいる．これらのバイトコードとシンボル情報から可読性のあるソースコードを復元することができる．

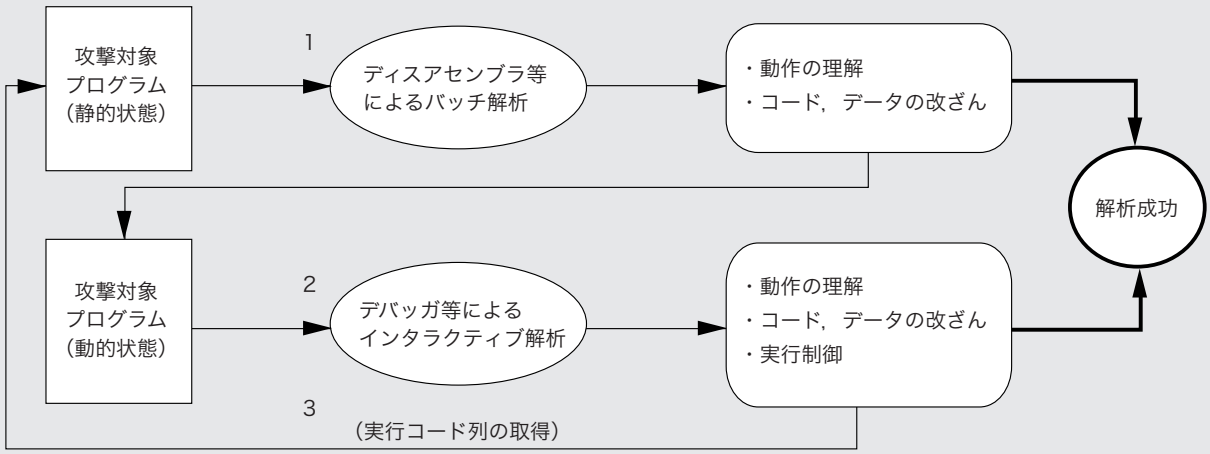


図-1　解析行為の流れ

デバッガを用いた解析（動的解析）

デバッガは，ソフトウェアの開発過程において実装上の不具合（バグ）を発見するためのツールである．開発者はソースコードを持っているので，ソースレベルで解析（デバッグ）を行うことができる．一方，解析を企てる者はソースコードを持っていないので，アセンブリ言語レベルの解析を行うことになる．ソースコードレベルに比べれば困難な作業になるが，解析対象となるソフトウェア内の注目する個所にブレークポイントを設定したりステップ実行させたりしながらメモリやレジスタの内容を調べたり，制御の流れを追跡したりすることができる．通常のデバッガはアプリケーションレベルで動作するが，インサーキットエミュレータ（ICE：In-Circuit Emulator）と呼ばれる解析ツールはCPUとロジックボードの間に入ってすべての命令やデータを観察できるツールである．OSやデバイスドライバなどハードウェアに密着したプログラムのデバッグや解析に用いられるが，OS上で動作する一般のアプリケーションを解析する用途には向かない．またICEは一般に高価であり，誰もが利用できるツールではない．そのため，アプリケーションの解析ツールとしての優位性は低い．

PCのハードウェアをソフトウェアでエミュレーションするいわゆるPCエミュレータというソフトウェアがある．エミュレータ自身は1つのアプリケーションであるため，デバッガ上でエミュレータを動作させることも原理上は可能であるが，解析対象のアプリケーションから見ると，ICEと同じレイヤーで解析されることになるため，やはり有効な解析手段とはいえない．

解析行為の流れ

ソフトウェアの解析行為は人為的な作業である．その作業の概略をモデル化して記すと以下のようになる．解析行為者は，たとえば最初に解析対象プログラムを静的状態，「非実行状態」で解析することを試みる（**図-1** 番号1）．主にディスアセンブラを用い，コード列を復元することを試みる．その結果，動作の流れを理解し，改変を施したり秘密情報の取り出しを試みるだろう．それができれば解析成功となる．静的解析で解析が十分にできないと判断した場合，解析者は次に動的解析，すなわち対象プログラムを実行させながら解析を試みる（**図-1** 番号2）．たとえばプログラムコードの一部分が暗号化されていて実行時に復号される，解析防御手法が施されている場合，暗号化部分はそのままではプログラムコードとしての意味をなさず，静的解析は不可能である．しかし，プログラムを実行させながら動的解析を行えば，暗号化が施されている部分を復号された状態で観測できる．また，実行中に計算されるデータの値やその流れを，静的解析に比べて効率よく観察することもできる．解析者はデバッガを用いてコードをトレースし，条件分岐命令を無視させたり，計算結果の値を記録したり，また値を書き換えて実行させたりするであろう．その結果，解析が成功するかもしれない．解析が不十分な場合であっても，解析者はそれまでのコードの実行履歴を記録できる．この情報を用いて，再び静的解析に戻り（**図-1** 番号3），解析を進めることも可能となる．このように解析者は，静的解析と動的解析を組み合わせ繰り返しながら解析を進めることになるだろう．

◇耐タンパー化技術

前述の解析行為に対抗する耐タンパー化を実現するための要素技術を紹介する．

なお，解析手法と耐タンパー化技術の関係を**表-1** に示しておく．

	解析ツール	解析手法	解析効率	耐タンパー化技術
静的解析	ディスアセンブラ	コードを読んで理解する	○	コードの難読化 コードの暗号化
動的解析	デバッガ	実行の流れの追跡	○	デバッガの検出 コード改変の検出
	ICE	CPU 上の実行コード列の取得	×	なし（検出は不可能）

表 -1　解析手法と耐タンパー化技術

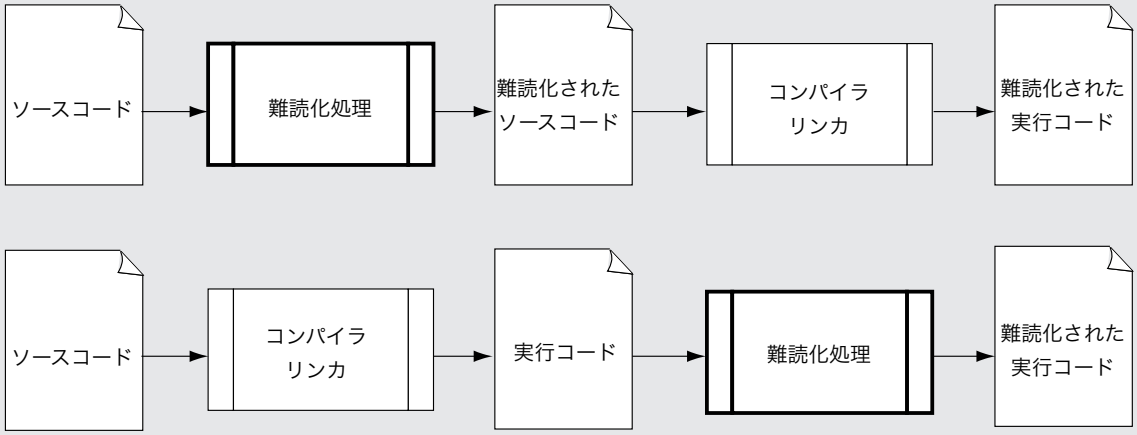


図 -2　難読化

難読化

「スパゲッティプログラム」という言葉がある．制御構造が入り組み，まるでスパゲッティのように絡まりあっているコードを指す言葉である．「難読化」とは，整然と書かれ読みやすい（はずの）プログラムを，読みにくいスパゲッティプログラムに変換し，解析にかかる時間を増大させ，耐タンパー化効果を狙う．変換の対象となるコードは，ソースコードの場合やバイナリコードの場合がある．Javaのように仮想マシン上で動作する言語の場合はソースコードを難読化の対象としていることが多く，C言語のようにネイティブ環境で動作するソフトウェアを開発するための言語の場合は，ソースを対象とした技術とバイナリを対象としたものがある．難読化処理を含んだソフトウェア開発プロセスを**図 -2** に示す．難読化の主な手法としては，以下のものがある．

- 実行に影響を与えない不要なコードを追加する．
- 命令コードを別の等価なコードに置き換える．
- 制御構造を複雑化させる．
- 1 つの計算を複数の計算に分割する．
- 1 つの機能を持ったモジュール（関数，サブルーチン）を，複数のモジュールに分割する．
- 同一の処理を行うモジュールを複数用意して冗長化する．
- ソースコードの関数名や変数名を無意味な文字列に変換する．

なお，耐タンパー化を目的としていても，プログラマ自らが故意に（あるいは故意でないとしても）読みにくいコードを書くことは生産性や保守性の観点から避けるべきであろう．

コードの暗号化

コードの暗号化とは，コードをあらかじめ暗号化しておき実行時に復号する技術である．暗号化されたコードは，コードとしては無意味なビット列にすぎず，ディスアセンブルしても暗号化部分の解析はできない．解析するためには，復号されたタイミングで実行を中断させ，メモリ上のコードを読む必要がある．あるいは復号ルーチンを解析して復号鍵をとらえ，暗号化されたコードを復号するツールを作るという解析方法も考えられる．それらの解析を困難にさせることを狙って，実行された直後に再度暗号化し，復号されたコードがメモリ上に存在する時間を極小化させる方法がある．コードの暗号化処理を含んだ開発プロセスを**図 -3** に示す．コードの暗号化を実現するためには，実行時にメモリ上のコードを暗号化状態から復号状態に書き換える必要がある．現在パーソナルコンピュータ上で主流となっているOSでは，コード領域は通常は書き込みができない属性となっている．そのためその属性を書き込み可能に変更するか，あ

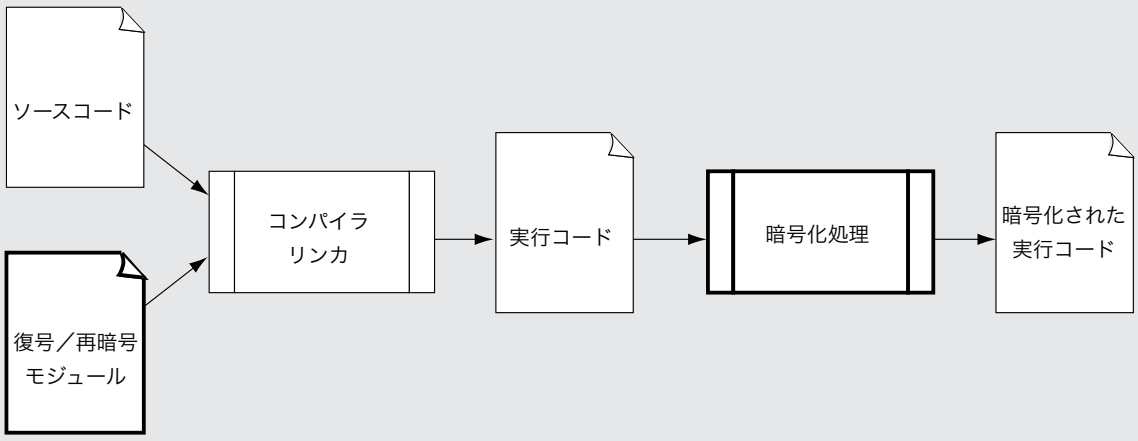


図-3　コードの暗号化

るいは書き込み可能な領域に復号コードを展開して実行させるなどの工夫が必要である．

解析ツールの検出

解析行為者は，解析の際には何らかの解析ツールを利用する．保護したいソフトウェアの実行中に解析ツールの存在を検出することによって，解析行為への対処が可能になる．たとえばデバッガやメモリダンプツールの存在を検出したら実行を中断するように，あらかじめプログラミングしておく方法がある．

改変の検出

仮に解析者によってソフトウェアが改変された場合，その改変を検出することによって対処する方法がある．保護対象ソフトウェアが起動時に自分自身が改変されていないかどうかをチェックし，改変されていたら実行を中止するというようにプログラミングしておくことが1つの方法である．あるいは，保護対象ソフトウェアの実行時にそのソフトを監視するプロセスを常駐させておき，対象ソフトウェアに対する解析者の干渉を検出するという方法もある．

各技術の組合せ

それぞれの耐タンパー化技術は，単独で用いてもその耐タンパー化効果は限定的である．それぞれの技術を相互補完的に組み合わせることによって，耐タンパー性が向上することが期待できる．たとえば，コードの暗号化を実現するには，暗号化されたコードを実行時に復号する必要がある．その復号ルーチンを難読化しておくことで，復号ルーチンの解析を困難にさせることが可能である．また，その復号ルーチンの内部に解析ツールの検出を組み込んでおけば，解析行為中に実行が妨げられて解析効率を低下させることも期待できる．このように組み合わせておくことによって，たとえば以下のように「保護の連鎖」を実現することができる．

- 暗号化されたコードは，復号しなければ解析できない．
- 復号するためには復号ルーチンを解析し，復号鍵を窃取しなければならない．
- 難読化された復号ルーチンは通常のコードよりも読みにくいため，解析に時間がかかる．
- 復号ルーチンの動作をデバッガで解析しようとすると，デバッガ検出機能の働きでデバッガ上での解析が妨げられる．

◇技術動向

ソフトウェアの耐タンパー化技術の重要性が高まってくるに伴い，研究事例や製品を目にするようになってきている．以下そのいくつかを紹介する．

研究事例

Aucsmithらによる研究[1]は，ソフトウェアの完全性を検証するモジュール（IVK：Integrity Verification Kernel）が保護対象ソフトウェアの実行時に改変を監視することにより，対象ソフトウェアの解析を防ぐ．またIVKの内部は実行時に復号と再暗号を繰り返しており，IVK自体の解析を防いでいる．Nickersonらによる研究[2]は，ソースコードの難読化を実現している．コードを複数のピースに分け，それぞれのピースに含まれる計算式を数学的な変換によってより複雑な計算式に分割している．日本においては，村山ら[3]がバイナリコードの難読化を，門田ら[4]がソースコードの難読化を取り上げている．筆者らによる研究[5]は，コードの暗号化とデバッガ検出を基本とした技術である．WindowsおよびMacintoshのソフトウェアに適用できるツールとして実現されており，実際のソフトウェア製品に適用された実

績を持つ．

製品

ソフトウェアの耐タンパー化を実現している製品をいくつか紹介する．Macrovision社（米国）はビデオのコピー防止技術で有名な企業であるが，SafeWrapという名称で耐タンパー化ツールも提供している（http://www.macrovision.com/solutions/software/safewrap.php3）．WindowsのEXEおよびDLLに対応しており，コンパイル済みのEXEやDLLに対して後から保護を適用することができる．またより強力な保護を施すために，SafeWrapが提供するAPIを保護対象となるソースコード（C/C++）に埋め込むこともできる．難読化・暗号化・デバッガ検出を組み合わせた保護技術を採用しているようである．cloakware/transcoderはカナダを拠点とするcloakware社の製品である（http://www.cloakware.com/products/transcoder/index.html）．Cのソースコードを複雑に変換する難読化技術を持っている．暗号化やデバッガの検出などは行わない．バイナリでなくCのソースコードを変換するのみなので，WindowsやMacintosh，Linuxなどの広範囲なプラットフォームをサポートしている．富士ゼロックス（株）と（株）アクセスチケットシステムズが開発したTRCS（Tamper Resistant Coding System）は，コードの暗号化とデバッガ検出技術を組み合わせた耐タンパー化ソリューションである（http://www.fujixerox.co.jp/randd/13/Ishima/absj.html）．WindowsのEXEとDLL，さらにMacintoshのアプリケーションと共有ライブラリに対応している．

オープンソースと耐タンパー化

昨今のIT業界では「オープンソース」というキーワードをよく目にする．Linuxに代表されるオープンソースソフトウェアの価値が一般にも認識され始めてきており，ある種の流行となっている．オープンソースソフトウェアはその名のとおりソースコードが公開されており，誰でもその内容を確認でき，また改良できる．腕に自慢のプログラマたちはインターネットで緩やかに結ばれた共同体を形成し，彼らの探究心とプライドにかけて日々改良を続けている．一方「耐タンパー化」はソフトウェアの中身をいかに見せないか，という技術であり，オープンソースと対立する概念である．本稿の前半で触れたように，認証機能や暗号技術を応用した機能を持つソフトウェアの場合，改変に対する耐性や秘密情報の安全な保持は欠かせない要素である．オープンソースと耐タンパー化という対立する2つの概念の存在は，「安全なソフトウェア」を論じる際に2つの側面があることを示唆している．オープンソースは「開発の場」において多数の目にさらされることによってその品質を高めることができ，バグが少ない・脆弱性が低いという意味において「ユーザにとっての安全性」を高める．一方，耐タンパー化はソフトウェアの「使用の場」において，不正な使用に対する耐性を付与することによって「サービス提供者にとっての安全性」を高めるのである．

今後の課題

ソフトウェアの耐タンパー化技術における課題を挙げてみよう．一般にセキュリティ技術の適用の局面においては，保護の対象物の価値に応じて，必要かつ十分な安全性のレベルで適用することが望まれる．ソフトウェアの耐タンパー化技術においても同様である．すなわち，安全性の定量化が望まれる．しかし，ソフトウェアの耐タンパー化技術の分野では，難読化技術において静的解析の困難さがNP-Hard（Nondeterministic Polynomial Hard）になる手法を用いて安全性の根拠とする例[6]があるものの，まだまだ安全性を定量化できるレベルには至っていない．この分野でそれを難しくしている理由は，人為的な行為である動的解析に対する安全性を定量化することの困難さ，および，主に手法の秘密性に安全性の根拠を置いているという2つの大きな要因があるためだろう．このように，耐タンパー化を実現する技術だけでなく，安全性の定量化というテーマも重要な研究課題となっている．マルチプラットフォームへの対応も重要である．汎用的なパーソナルコンピュータだけでなく，組み込みデバイス上にもさまざまなソフトウェアが実装されつつある．それぞれのデバイス用のソフトウェア開発環境で利用できる耐タンパー化ツールが望まれる．

参考文献

1）Aucsmith, D. and Fraunke, G.: Tamper Resistant Software: An Implementation, Proceedings of the 1996 Intel Software Developers' Conference.
2）Nickerson, J.R., Chow, S.T. and Johnson, H.J.: Tamper Resistant Software-Extending Trust into a Hostile Environment, ACM Multimedia Workshop, Ottawa (Oct. 2001).
3）村山隆徳，満保雅浩，岡本栄司，植松友彦：ソフトウェアの難読化について，電子情報通信学会技術研究技報（情報セキュリティ）ISEC95-25, pp.9-14 (1995).
4）門田暁人，高田義広，鳥居宏次：ループを含むプログラムを難読化する方法の提案，電子情報通信学会論文誌，Vol.J80-D-I, No.7, pp.644-652 (July 1997).
5）石間宏之，齊藤和雄，亀井光久，申　吉浩：ソフトウェアの耐タンパー化技術，富士ゼロックス テクニカルレポートNo.13, pp.20-28 (2000).
6）小木曽俊夫，刑部裕介，双紙正和，宮地充子：手続き間呼出しの関係に着目した難読化手法の提案とその評価，2002 Symposium on Cryptography and Information Security, SCIS2002-6C1 (Jan.-Feb. 2002).

（平成15年4月30日受付）