ランタイムデータ探索型耐タンパー性評価法*

赤井 健一郎† 三澤 学† 松本 勉†

耐タンパーソフトウェアの評価手法として，ランタイムデータ探索型耐タンパー性評価法を提案する．提案評価法は，耐タンパー化プログラムの実行過程でメモリ上に現れたデータの中から秘密データを見つけ出すことで，秘密データ守秘性の点からの評価を行うものである．また，提案評価法はすべての処理過程で人間の介在がなく，計算機上で実現可能であるため，その評価結果は客観的なものになると考えられる．提案評価法の有効性を確かめるために，提案評価法を実現する実験システムを用いて，オブフスケーション変換により耐タンパー性を付与された評価対象プログラムから秘密データを抽出する実験を行った．この実験の結果，評価対象プログラムから秘密データをごく短時間で見つけ出すことができた．この結果，提案評価法の有効性が確認できた．

Evaluating Tamper Resistance by Searching Runtime Data

Kenichiro Akai,† Manabu Misawa† and Tsutomu Matsumoto†

In this paper, we introduce an evaluation method for tamper resistant software. This method collects all data that appear on the memory during run time of a tamper resistant program, and searches for the secret data that is hidden in the program. We can implement all processes of this method on the computer, so its evaluation results are sufficiently objective. In order to confirm the effectiveness of this method, we implemented the experiment system which realized this method, and the system experimentally extracted secret data from the programs that have gained the tamper resistance by some obfuscation transformations. We observed that the system extracted the secret data from the programs correctly in this experiment. Consequently we confirmed the effectiveness of this method from the result.

1. はじめに

耐タンパー性とは，ソフトウェアまたはハードウェアのモジュール内部に存在する秘密情報の観測や，モジュールが実現する機能の改変が困難な性質のことである．耐タンパー性を構成する性質として，秘密情報の観測が困難な性質を秘密データ守秘性，機能を不当に改変することが困難な性質を機能改変困難性と呼ぶことにする．

耐タンパーソフトウェア技術として，オブフスケーション変換（*Obfuscation Transformation*）[6)~11)]がある☆．オブフスケーション変換は，解析者の不正な解析行為が困難になるように，プログラムの等価変換を行う．たとえば，ループ構造のようにプログラムの記述の中で頻繁に現れるような構造や，置き換え可能な命令群に着目して，プログラムを等価変換する．これにより，プログラムが実現する機能の理解を困難にする．オブフスケーション変換は，一般的なプログラム言語で記述されたプログラムに耐タンパー性を付与することができ，安価で利便性が高い．また，これまでの耐タンパーソフトウェア技術の研究では最もさかんに研究されてきている．そこで本研究においては，オブフスケーション変換のようにプログラムを等価変換することにより得られる耐タンパー性について論じることを目的とする．

このような耐タンパーソフトウェア技術の評価として，まず最初にあげられるのは人間の評価者を用いた評価である[7)]．しかし，プログラム言語を理解している評価者が多数必要であったり，評価者のレベルを的確に設定することが困難であったりするなどの問題があり，客観的な結果を得るためには多大な努力を払わ

† 横浜国立大学大学院環境情報学府/環境情報研究院
Graduate School of Environment and Information Sciences, Yokohama National University

* 本論文の内容は，研究発表論文[1)~5)]の内容を総括するものである．なお，方式の名称を，「全数探索型耐タンパー性評価法[1)~4)]」から「ランタイムデータ探索型耐タンパー性評価法[5)]」に変更した．

☆ 難読化技術[6)~9)]は，概念的にも技術的にもオブフスケーション変換と非常に似ていることから，本論文ではオブフスケーション変換に含めて扱う．

なければならない．

そこで，耐タンパー性の評価を行うためには，コストをかけずに，十分にその耐タンパー性を評価しうるものが求められる．つまり，計算機で行えるような評価手法が求められている．そのような評価の指標として，命令コードの分布に着目したもの[8]や，コンパイラの解析木に着目したもの[12]が報告されている．しかし，これらの評価指標と耐タンパー性との関係は必ずしも明確ではない．

耐タンパーソフトウェアの評価をより客観的なものにするために，共通鍵ブロック暗号の安全性評価を参考にすることができる．共通鍵ブロック暗号は，安全性という概念を可能な限り客観的に評価しようとし，その結果，信頼できる技術として実用化されている．共通鍵ブロック暗号の安全性評価では全数探索法以外に，線形攻撃法や差分攻撃法などの様々な攻撃法が提案されており，暗号方式の安全性を多角的に評価することができる．また，各攻撃法においては，攻撃者の能力や手法といったものが明確にモデル化されているため，その安全性を理解しやすいものにしている．つまり攻撃手法の多さと攻撃のモデル化が，評価結果を客観化するうえで重要な役割を果たしているといえる．

したがって，耐タンパーソフトウェアにおいても，解析手法が的確にモデル化されて，その効果を定量的に見積もれるような評価法を多くつくり出していくことで，評価結果を客観的にしていくことが可能となると考える．しかし，そのような評価法自体がまだ少ない．

そこで，本論文では，ランタイムデータ探索型耐タンパー性評価法を提案する．提案評価法では，耐タンパー化の対象となるプログラムとして暗号の復号方式を実装し，内部に秘密の復号鍵を内蔵するものを想定する．そして，秘密情報である復号鍵を隠蔽するように耐タンパー化したものを評価対象プログラムとする．提案評価法は，評価対象プログラムの実行過程を監視し，メモリ上に現れたデータの中から復号鍵を探索する．もし鍵を短時間で見つけることができたならば，評価対象プログラムは秘密データ守秘性の点で脆弱であると判断できる．提案評価法は，評価の全過程を計算機上で実現可能なため，その評価結果は十分に客観的であるといえる．

また，提案評価法の有効性を示すために，実験システムを実装して耐タンパー化プログラムの評価実験を行った．耐タンパー化対象プログラムとしては，RC5[13]またはDES[14]の2種類の暗号方式を個別に実装し，オブフスケーション変換を用いてそれぞれを耐タンパー化したものを用いた．実験の結果，ごく短時間で秘密情報である復号鍵を見つけ出すことに成功し，提案評価法の有効性を確かめることができた．

本論文では，2章で解析者のモデルと提案評価法について述べ，3章で共通鍵ブロック暗号ソフトウェアの場合の適用を説明し，4章で実証実験を述べ，5章で考察を行い，6章でまとめと今後の課題を述べる．

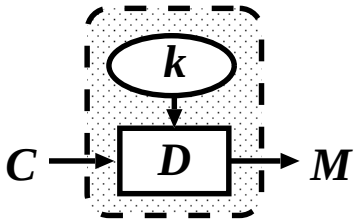
2. ランタイムデータ探索型耐タンパー性評価法の提案

2.1 評価のモデル

耐タンパー化対象プログラムは，2入力1出力の関数 D（暗号方式の復号関数）と秘密のデータ k（その暗号方式におけるある復号鍵）とにより定義される機能（入力 C と出力 M の関係）を有する（**図1**）[15]．そして，耐タンパー化対象プログラムに耐タンパー化手法を施し，これを評価対象プログラムとする．耐タンパー化対象プログラムが実現している暗号方式は公開されているものとする．この仮定により，耐タンパー化技術が厳しい条件のもとで評価される．

ここで耐タンパー化対象プログラムが，ソースコードであるか実行形式であるかは，耐タンパー化手法により適切に選択されるものとする．また，評価対象プログラムは，必ず実行形式であるとする．

解析者の目的は，プログラムに内蔵された秘密の復号鍵 k を導出することである．このようなタイプのプログラムから k を導出しようとする解析方法には，入出力の関係だけを用いる入出力解析と，入出力解析に加えてプログラム自体の静的・動的解析を含む内部解析の2種類がある．静的解析とはプログラムを読んで解析を行うことであり，動的解析とはデバッガなどを用いて，プログラムのデータフローなどを解析することである．いずれかの解析法のみを用いて秘



k: 秘密データ
D: 復号関数（方式公開）
C: 暗号文
M: 平文

図1 耐タンパー化対象プログラムのタイプ
Fig. 1 The target program.

密データ k を導出するためにかかる時間，費用などの手間の総量をコストとする．そして，入出力解析，内部解析で k を導出するためのコストを，それぞれ $Cost_{io}$，$Cost_{inside}$ とする．内部解析では，プログラムの入出力値に加えて，プログラムの記述や実行過程でのデータフローから得られる情報も利用できる．一方，入出力解析では，プログラムの入出力値だけが解析に利用可能である．したがって，それぞれの解析法では利用できる情報量に差があるため，$Cost_{io}$ の最小値 $Cost_{io}^{min}$ と $Cost_{inside}$ の最小値 $Cost_{inside}^{min}$ に関して，以下の不等式が成り立つ．

$$Cost_{inside}^{min} \leq Cost_{io}^{min}.$$

ある耐タンパー化プログラムにおいて，この不等式の等号が成立したすると，この耐タンパー化プログラムは十分な耐タンパー性を有していると考えられる．

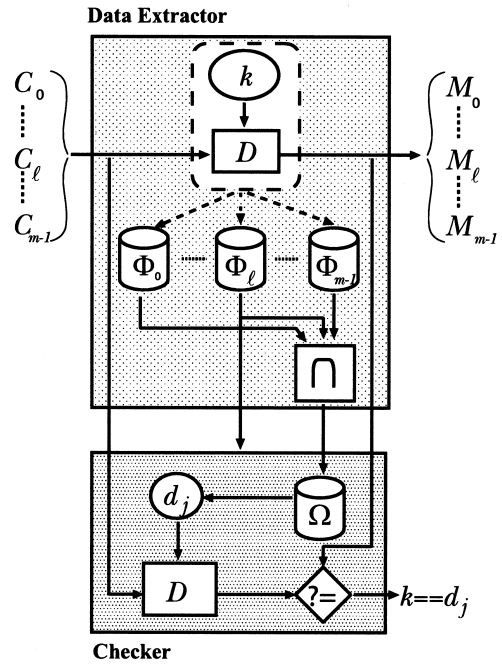
2.2 提案方式の詳細

プログラム内部に秘密データ k が巧妙に隠されていても，実行過程において 1 度はメモリ上に現れる可能性が高い．そこで，プログラムを実行し，実行形式上の 1 命令実行終了時にメモリ上に現れたデータを記録しておき，これを秘密データ候補の集合 Ω とする．そして，Ω の中に k が存在するかを確かめることができればよい．k が Ω 内に存在するかの確認には，評価対象プログラムが実装している暗号方式が公開であるため，その情報を使うことができる．

また復号鍵 k は，入力によって変化しない値と考えることができる．そこで，評価対象プログラムに入力を複数与えて，入力によって変化するデータと変化しないデータを分離することにより，効率的に秘密データを抽出することができると考えられる．

したがって，**図 2** のように評価手法を構成することで，耐タンパー化プログラムの評価が可能となる．評価手法は Data Extractor と Checker により構成される．

Data Extractor は評価対象プログラムの実行過程を監視し，秘密データの候補集合 Ω を構成する．m を評価対象プログラムに入力する暗号文の数とし，$C_0, C_1, \ldots, C_{m-1}$ を評価対象プログラムへの入力暗号文，対応する出力を $M_0, M_1, \ldots, M_{m-1}$ とする．$C_\ell (\ell = 0, 1, \ldots, m-1)$ を入力したとき，実行過程で現れたデータの集合を Φ_ℓ とする．Data Extractor は $\Phi_0, \Phi_1, \ldots, \Phi_{m-1}$ を求める．その後，$\Omega = \cap_{\ell=0}^{m-1}\Phi_\ell$ を求め，プログラムの入力によらない部分を求め，これを秘密データ候補の集合とする．$\Omega = \cap_{\ell=0}^{m-1}\Phi_\ell$ を求める際に，異なる集合に属する 2 つの元を比較する必要がある．この比較は，2 つの元（メモリ上に現れたデータ）の示す値とデータ型が一致する場合のみ，同じ元と判断することとする．秘密データの候補集合 Ω を構成することができたら，Data Extractor はある 1 つの $\ell (= 0, 1, \ldots, m-1)$ を選び，$\langle C_\ell, M_\ell, \Phi_\ell, \Omega \rangle$ を Checker に渡す．



Data Extractor: 実行過程の全データを抽出するデータ抽出器
C_ℓ : 評価対象プログラムへ入力される暗号文
M_ℓ : C_ℓ に対応する平文
Φ_ℓ : C_ℓ を入力とした時のメモリ上に現れたデータの集合
Ω : 秘密データの候補集合
d_j : j 番目の秘密データの候補
Checker : k が Ω にあるかの検査を行う検査器

図 2 ランタイムデータ探索型耐タンパー性評価法
Fig. 2 Evaluating tamper resistance by searching runtime data.

Checker は，秘密データの候補集合 Ω に秘密データが存在するかを検査する．具体的には，Data Extactor から $\langle C_\ell, M_\ell, \Phi_\ell, \Omega \rangle$ を受け取り，$d_j \in \Omega$ に対して，

$$D(d_j, C_\ell) = M_\ell$$

が成立するかどうか検査する．もし成立するなら，d_j は秘密データ k であると判断し，これを結果として出力する．もし成立しないなら，d_{j+1} を用いて同じ検査を行う．Ω のすべての要素に対してこの検査を行っても，等号が成立するような d_j が見つからない場合は，秘密データが発見できなかったという結果を出力し，終了する．また，Φ_ℓ と Ω の差集合 $\Phi_\ell \setminus \Omega$ は，入

力によって変化するデータの集合であるので，これを $D(d_j, C_\ell)$ の計算を行う際に有力な情報として利用する．なお，$m = 1$ の場合は，$\Omega = \Phi_0$ として検査を行う．

Checker の実装は，評価対象プログラムが実装する暗号方式によって異なる．3 章では，共通鍵ブロック暗号が実装されている場合の，Checker の実装について述べる．

Data Extractor および Checker の処理は，すべて計算機上で実現可能であり，その処理にかかる計算量や実行時間などを評価対象プログラムの耐タンパー性の度合いとして示すことが可能である．また，評価の過程に人間が介在することがないので，個人差などにより評価結果が大きく左右されることもない．したがって $Cost_{inside}$ を定量的で客観的なものとして得られる．

提案評価法は，メモリ上に現れるすべてのデータを抽出して検査を行うので，実行中に秘密データが現れる限りは，それを見つけることができる．つまり，提案評価法により秘密データを取り出されてしまう耐タンパー化プログラムは，秘密データ守秘性のうえで脆弱さを持っていると判断できる．したがって，本評価法に対して十分な耐性を持つことが，秘密データ守秘性において最低限の条件であると考えられる．

3. 共通鍵ブロック暗号ソフトウェアへの適用

3.1 共通鍵ブロック暗号ソフトウェア

耐タンパー化対象プログラムを，内部に秘密鍵 k を持ち，r ラウンドの変換を行う共通鍵ブロック暗号とする．共通鍵ブロック暗号では，k がなくても r 段のラウンド関数と r 個のラウンド鍵の系列 $\{S_i\}$ があれば，暗号化・復号の処理が可能となる．つまり $\{S_i\}$ も秘密データであり，耐タンパー化による保護の対象となる．そこで，k および $\{S_i\}$ の導出が困難になるように耐タンパー化技術を施したものを評価対象プログラムとする（**図 3**）．

ラウンド関数 f_i とラウンド鍵系列 $\{S_i\}$ があれば，k によって定まる関数の機能と同等の機能を実現できる．よって，本論文では解析の目的を $\{S_i\}$ を見つけることとする．

3.2 提案手法でのラウンド鍵系列の探索

2.2 節で述べたように，Data Extractor は $\langle C_\ell, M_\ell, \Phi_\ell, \Omega \rangle$ を作成し，Checker に渡す．

共通鍵ブロック暗号は，ラウンド構造により構成されている．したがって，メモリ上に出現したデータの集合 Φ_ℓ の要素間でラウンド関数 f_i の入出力関係を

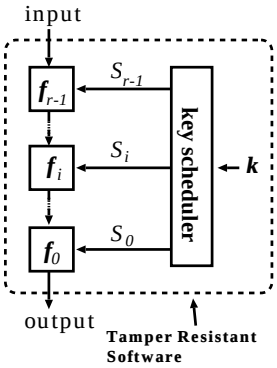
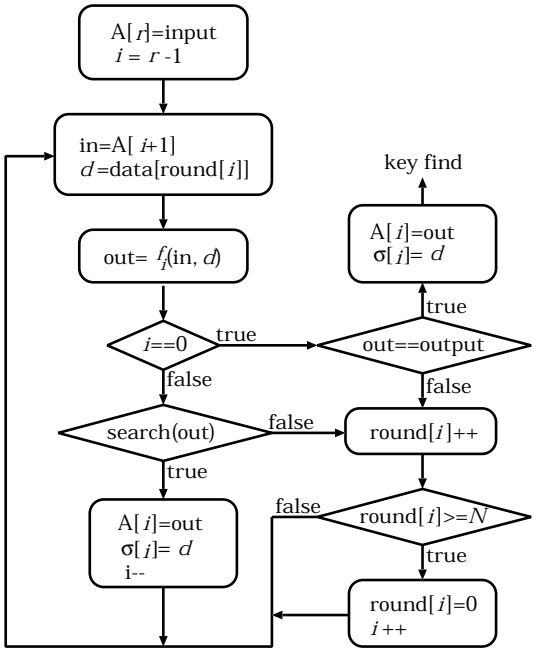


図 3 耐タンパー化共通鍵暗号プログラム（復号）
Fig. 3 A tamper resistant program of symmetric key block cipher (decipher).

見つけていき，それらをつなぎあわせて評価対象プログラムの入力 C_ℓ から出力 M_ℓ までを結ぶことができれば，$M_\ell = D(k, C_\ell)$ の関係を見つけたことと同等となる．つまり，処理の過程を Checker で再現することができればよいのである．

このアルゴリズムの 1 例をあげる（**図 4**）．共通鍵ブロック暗号における 1 つのラウンド関数 f_i を，ラウンド鍵系列 $\{S_i\}$ を見つけるための検査の単位として用いる．これを check-round と呼ぶ．そして，Ω の元の中で，check-round への入力として想定できるデータ型の元を，最大のビット長のデータ型に変換して配列 data[ ] に格納する．そして check-round f_i に対して，入力 in が与えられていたとしたとき，d_j を配列 data[ ] から選び，$\mathrm{out} = f_i(d_j, \mathrm{in})$ を計算する．そして，out とデータ型も含めて等しい元が $\Phi_\ell \setminus \Omega$ に存在するかを確認する．もし存在するならば，d_j を第 i ラウンドのラウンド鍵であると仮定し，ラウンド鍵と仮定されるデータを格納する配列の i 番目の要素 σ_i に記憶する．そして，out を次のラウンドの入力 in とし，次のラウンドの検査を行う．存在しないのであれば，d_j とは異なる $d_{j+1} \in \Omega$ を選び，同様の計算を行う．もし，Ω のすべての要素に対して検査を行っても $\mathrm{out} \in \Phi_\ell \setminus \Omega$ を満たす d_j を見つけることができなかったならば，前ラウンドの仮定が間違っているので，第 $i-1$ check-round に戻り検査をやり直す．このとき，これまで誤ってラウンド鍵と仮定していた Ω の要素を排除したうえで，検査を行う．この一連の手続きを評価対象プログラムの入力である C_ℓ から行っていき，最終 check-round の出力 out が M_ℓ と一致するかの判定を行う．もし一致するならば，ラウンド鍵と仮定してきた系列 $\{\sigma_i\}$ が評価対象プログラムに内蔵されているラウンド鍵系列 $\{S_i\}$ である．



r： ラウンド数
input： 評価対象プログラムへの入力
output： 評価対象プログラムからの出力
in： ラウンド鍵以外の check-round の入力
out： check-round の出力
A[ ]： 各ラウンドの出力と仮定したデータを格納する配列
d： 検査対象データ
N： Ω の要素数
data[ ]： Ω の要素を格納した配列
round[ ]： 各 check-round i で data[ ] で現在の検査位置を示すインデックスを格納する配列
σ[ ]: ラウンド鍵 S_i と仮定されるデータを格納する配列
f_i： i ラウンドにおける check-round
search()： $\Phi_\ell \setminus \Omega$ の要素に out が存在するか確認する関数
i： 検査中のラウンドを示すインデックス

図 4 ラウンド鍵探索のフローチャート
Fig. 4 Flow-chart of searching for round key sequence.

評価対象プログラムの復号処理の過程において，ラウンド鍵系列 $\{S_i\}$ は暗号文 C_ℓ によらない値であり，ラウンド関数 f_i への入出力の値は，C_ℓ に依存して変化する．したがって，$\{S_i\}$ は秘密データの候補集合 Ω から探索し，check-round の出力を $\Phi_\ell \setminus \Omega$ から探索することで，評価対象プログラムの行う復号変換過程の再現が可能となるのである．

また $m = 1$ の場合は，$\Phi_0 \setminus \Omega = \phi$ である．そこで d_j を Φ_0 から選択し，check-round の出力 out も Φ_0 に存在するかを判定することで，同様にラウンド鍵系列 $\{S_i\}$ を探索することができる．

ここで示した探索法を用いて，正しいラウンド鍵系列 $\{S_i\}$ を探し出せる条件は，ラウンド i における f_i のラウンド鍵 S_i を含めた入力値と出力値が Φ_ℓ に含まれていることである．本探索法では，メモリ上に現れたデータと暗号方式から系列 $\{S_i\}$ を構成するので，この条件を満たすのであれば，特に評価対象プログラムの記述を解析する必要はない．

4. 実　　験

オブフスケーション変換により耐タンパー化された評価対象プログラムから，秘密データであるラウンド鍵系列 $\{S_i\}$ を正しい順序で取り出すことを試みる実験を行った．

本実験では，RC5-32/12/16[13] または DES[14] を実装した耐タンパー化対象プログラムに，オブフスケーション変換を施し，これを評価対象プログラムとした．また，評価対象プログラムは Java の実行コードとした．そのため，Java 実行コード専用の実験システムを構築し，実験を行った．

実験に際して，システム実装および実験に用いた環境は以下のとおりである．

- Pentium 600 MHz CPU
- 256 MB Memory
- Windows NT 4.0
- Java 1.3 SDK

4.1 実験システム

Data Extractor

実験システムの Data Extractor は，Java 実行コードの評価対象プログラムを Jasmin[16] 形式に逆アセンブルを行う D-Java[17] と，Java 仮想マシン（Java 実行環境）を模倣し Jasmin コードを解釈実行する Jasmin Code Interpreter（JCI）で構成した．

Java 仮想マシンは，スタックマシンである[18]．その計算は，主に Java 仮想マシン内のオペランドスタックと呼ばれる領域で行われる．したがって，Data Extractor の機能を実現するためには，オペランドスタックの状態を読み出す必要がある．この機能は，実行形式におけるインストラクションごとにオペランドスタックの状態を読み出すため，通常のデバッガなどでは実現が困難である．そこで，Java 実行コードと 1 対 1 に対応したニーモニックコードを解釈実行するエミュレータを作成し，その動作を監視することで Data Extractor の機能を実現した．Java 実行コードの変換を行うのが D-Java であり，その変換された実行コードの形式が Jasmin 形式であり，Jasmin 形式の実行コード（以下 Jasmin コード）を解釈実行するのが JCI

である．

評価対象プログラムの Jasmin コードを解釈実行する JCI のオペランドスタックに対する操作は，Java 仮想マシンのものと比較して異なる部分がある．Java 仮想マシンでは，32 ビットを 1 ワードとして扱う．したがって，64 ビット整数型 long や double は 2 ワードとして扱われる．一方 JCI では，データが示す値をそのままオペランドスタックに 1 ワードとして積んで，演算を行う．また，JCI の扱えるデータ型は 64 ビット整数型 long，32 ビット整数型 int と long 型配列，int 型配列のみ扱うことができる．しかし，これらの制約があったとしても暗号方式を実現する耐タンパー化対象プログラム P は作成可能である．本実験で作成した P はこの制約のもとに作られている．

Checker

Checker は耐タンパー化対象プログラムが実装する暗号方式によって異なる実装を行う必要がある．実験に用いた暗号方式は RC5-32/12/16 と DES である．RC5 に関してはラウンド関数のハーフラウンドを 1 つの check-round とし，DES に関してはラウンド関数を check-round とした．また，秘密データの候補集合 Ω から配列 data[ ] に格納される元は，RC5 では int 型の値を選び，DES では int 型の値を変換して long 型として扱うこととして，int 型および long 型の値を選択した．

Checker からの出力は，ラウンド鍵系列 $\{S_i\}$ の発見の有無にかかわらず，Data Extractor の実行時間 T_1，Φ_ℓ の要素数 N_{Φ_ℓ}，Ω の要素数 N_Ω を出力する．ラウンド鍵系列を発見できた場合には，さらに，Checker の実行時間 T_2 と check-round の使用回数 $Trials$，$\{S_i\}$ を使用される順番で出力する．$Trials$ は check-round を使用してある $d_j \in \Omega$ がラウンド鍵であるかを判定する回数であるので，$Trials$ に比例して T_2 は増加する．しかし，T_2 は計算機の能力に依存して変化するので，計算機に依存しない値として $Trials$ を計測した．また秘密データの候補集合 Ω のすべての要素を探索しても $\{S_i\}$ が発見できなかった場合は，その結果を通知する．

実装面から，評価速度の向上のために check-round の出力 out を確かめる際に，$\Phi_\ell \setminus \Omega$ の要素を 2 分木に格納した．これにより，1 回の out の確認を対数のオーダで行うことができる．また，配列 data[ ] の格納するデータとして 0 からラウンド数までの値と -1（すべてのビットが 1）を配列の後方に配置し，後から検査されるようにした．この理由は，このような値はプログラム中でよく利用される値であるが，ラウンド鍵の値として用いられる可能性は低いと考えられるためである．

4.2 評価対象プログラム

実験で用いた耐タンパー化対象プログラムが実現する復号アルゴリズムには，共通鍵暗号 RC5-32/12/16 と DES を用いた．耐タンパー化対象プログラムは，RC5 は 32 ビット int 型でラウンド関数の入出力を構成し，DES は 64 ビット long 型で構成して実装した．

暗号アルゴリズム復号処理を実装したプログラムを耐タンパー化対象プログラム P とする．P に対して耐タンパー化を施したプログラムを評価対象プログラム P_u とする．

耐タンパーソフトウェア技術としてオブフスケーション変換を用いることにより，P に耐タンパー性を付与する．オブフスケーション変換手法としては，データオブフスケーション，制御構造の複雑化，ダミーコードの挿入を用いて実験を行った．各オブフスケーション技術を関数として OBF_n で表す．

4.2.1 データオブフスケーション

データオブフスケーションは，定数データに対してある変換を施し，それが持つ意味を曖昧化し，解析行為を困難にする．用いる変換としては，1 つの定数データを複数の定数データに分割して管理したり，複数の定数データを 1 つの定数データとして管理したりするものがある．しかし，データを使用する直前に逆変換を施し，もとのデータに戻してから使用するのものが多い．

実験では，ソースコードにラウンド鍵と `0xaaaaaaaaa` の排他的論理和をとった値を記述し，ラウンド鍵を使用する直前に再び `0xaaaaaaaaa` と排他的論理和をとるというデータオブフスケーションを施した．このオブフスケーション変換を OBF_1 とする．

4.2.2 制御構造の複雑化

制御構造の複雑化は，プログラム中でよく用いられるループ構造や if 文に着目し，制御構造を複雑化する．

実験では，データランダム化部に相当する for 文に着目し，**図 5** のように制御構造を複雑化した．このオブフスケーション変換を OBF_2 とする．

4.2.3 ダミーコード

ダミーコードの挿入は，プログラムのコードに冗長な命令を加えることで，解析行為を難しくする．

実験ではアルゴリズムに影響を与えないように，新たに変数を置き，それぞれに各ラウンドで `0xbbbbbbbbb` と排他的論理和をとるダミーコードを**図 6** のように挿入した．これを OBF_3 とする．

```
for( int i = 12 ; i >= 1 ; i-- ){
   B = rightRotate(B-S[2*i+1],A)^A;
   A = rightRotate(A-S[2*i],B)^B;
}
```

↓

```
int i = 12;
for(;;){
  if( i < 1 ) break;
  B = rightRotate(B-S[2*i+1],A)^A;
  A = rightRotate(A-S[2*i],B)^B;
  i--;
}
```

図 5 制御構造の複雑化 OBF_2
Fig. 5 Control transformation OBF_2.

```
for( int i = 12 ; i >= 1 ; i-- ){
   B = rightRotate(B-S[2*i+1],A)^A;
   A = rightRotate(A-S[2*i],B)^B;
}
```

↓

```
int so = 0;
int se = 0;
for( int i = 12 ; i >= 1 ; i-- ){
   B = rightRotate(B-S[2*i+1],A)^A;
   so ^= 0xbbbbbbbb;
   A = rightRotate(A-S[2*i],B)^B;
   se ^= 0xbbbbbbbb;
}
```

図 6 ダミーコードの挿入 OBF_3
Fig. 6 Dummy code insertion OBF_3.

表 1 実験で使用した秘密鍵 k
Table 1 Secret key k values used in this experiment.

	RC5
k_0	0x102030405060708010203040506070 80
k_1	0xf0e0d0c0b0a090807060504030201000
k_2	0x7a7bba4d79111d1e797bba4d78111d1e
k_3	0x8285e7c1b5bc7402fc586f92f7080934
	DES
k_0	0x4dfd1f5c4d190f62
k_1	0xe51f1c1b1691e474
k_2	0x1f080c71de69a9ac
k_3	0xc9322638f8a50f9c

表 2 入力暗号文 C の値
Table 2 Ciphertext C values used in this experiment.

	RC5
C_α	0x294ddb46b3278d60
C_β	0xdcfe098577eca5ff
C_γ	0x9646fb77638f9ca8
C_δ	0xb2b3209db6594da4
	DES
C_α	0xb45facfc4bf159b7
C_β	0xe0204bc255938fe9
C_γ	0x46155ea53cb67f85
C_δ	0x874aeb68cf5fc555

4.3 評価対象プログラムのバリエーション

耐タンパー化対象プログラム P に内蔵する秘密の復号鍵は，暗号方式ごとに 4 つの異なる鍵 k_0，k_1，k_2，k_3 で作成した（**表 1**）．つまり，RC5 と DES でそれぞれ 4 つの耐タンパー化対象プログラム P が存在する．

各々の P に対して，上述したオブフスケーション変換 OBF_1，OBF_2，OBF_3 を用いて，耐タンパー化を施した．オブフスケーション変換を施さない場合も含め，次の 8 通りの評価対象プログラム P_u を作成した．なお，$OBF_{n_1}(OBF_{n_2}(P))$ は多重に異なるオブフスケーション変換を適用したことを示す．

$P_0 = P$
$P_1 = OBF_1(P)$
$P_2 = OBF_2(P)$
$P_3 = OBF_3(P)$
$P_4 = OBF_1(OBF_2(P))$
$P_5 = OBF_1(OBF_3(P))$
$P_6 = OBF_2(OBF_3(P))$
$P_7 = OBF_1(OBF_2(OBF_3(P)))$

4.4 実 験 結 果

実験システムに評価対象プログラム P_u の Jasmin コードと，P_u への入力暗号文を入力し，実験を行った．1 つの評価対象プログラム P_u に入力する値は，**表 2** に示す暗号方式ごとに異なる 4 つの値を用いた．そして，Ω を構成するために用いた $C_\ell(0 \le \ell \le m-1)$ の数 $m = 1$，2，3 の場合で，実験を行った．

実験の結果，すべての評価対象プログラム P_u から秘密データであるラウンド鍵系列 $\{S_i\}$ を取り出すことができた．各 P_u で計測した値の平均値を**表 3**，**表 4** に示す．$\overline{T_1}$，$\overline{T_2}$，$\overline{Trials}$，$\overline{N_{\Phi_\ell}}$，$\overline{N_\Omega}$ は T_1，T_2，$Trials$，N_{Φ_ℓ}，N_Ω の平均値を表している．

Data Extractor，Checker の実行時間 T_1，T_2 は，各 P_u と入力した値の組合せごとに 100 回行い，その平均を結果 $\overline{T_1}$，$\overline{T_2}$ とした．また，時間計測は Java 標準ライブラリを用いて計測した．

5. 考 察

提案評価法を用いて，実験で用いたすべての評価対象プログラム P_u から，秘密データであるラウンド鍵系列 $\{S_i\}$ を取り出すことができた．

この結果をふまえて，オブフスケーション変換と秘密データ守秘性および，提案評価法の効率，データ守秘性を考慮した耐タンパー化手法について考察を行う．

5.1 オブフスケーション変換と秘密データ守秘性

すべての P_u から $\{S_i\}$ を抽出できたという事実から，メモリ上に秘密データが現れるようにプログラミングされている場合には，実験で用いたようなオブ

表 3 実験結果 1：RC5
Table 3 The result of this experiment: RC5.

Programs	m	$\overline{T_1}$[ms]	$\overline{T_2}$[ms]	$\overline{Trials}$	$\overline{N_{\Phi_\ell}}$	$\overline{N_\Omega}$
P_0	1	18	67	24062	178	178
	2	33	5	386	178	60
	3	50	5	375	178	59
P_1	1	19	79	27909	205	205
	2	35	8	725	205	87
	3	54	8	714	205	86
P_2	1	17	66	24062	178	178
	2	33	5	386	178	60
	3	51	5	375	178	59
P_3	1	19	66	24971	179	179
	2	35	5	412	179	61
	3	54	5	401	179	60
P_4	1	19	78	27909	205	205
	2	35	8	725	205	87
	3	54	8	714	205	86
P_5	1	21	90	31759	232	232
	2	41	10	1067	232	114
	3	61	11	1056	232	113
P_6	1	19	66	24217	179	179
	2	36	5	412	179	61
	3	55	5	401	179	60
P_7	1	21	90	31759	232	232
	2	41	10	1067	232	114
	3	62	11	1056	232	113

表 4 実験結果 2：DES
Table 4 The result of this experiment: DES.

Programs	m	$\overline{T_1}$[ms]	$\overline{T_2}$[ms]	$\overline{Trials}$	$\overline{N_{\Phi_\ell}}$	$\overline{N_\Omega}$
P_0	1	555	523	16446	1859	1859
	2	1122	106	2506	1859	308
	3	1701	85	1752	1859	218
P_1	1	555	533	16609	1876	1876
	2	1127	112	2669	1876	325
	3	1876	92	1915	1876	235
P_2	1	558	518	16446	1859	1859
	2	1130	105	2506	1859	308
	3	1709	86	1752	1859	218
P_3	1	554	526	16456	1860	1860
	2	1129	108	2516	1860	309
	3	1714	87	1762	1860	219
P_4	1	559	523	16609	1876	1876
	2	1137	111	2669	1876	325
	3	1731	91	1915	1876	235
P_5	1	557	538	16764	1893	1893
	2	1137	117	2823	1893	342
	3	1702	97	2069	1893	252
P_6	1	558	519	16456	1860	1860
	2	1139	106	2516	1860	309
	3	1742	87	1762	1860	219
P_7	1	563	529	16764	1893	1893
	2	1148	114	2823	1893	342
	3	1730	97	2069	1893	252

フスケーション変換手法を用いて耐タンパー化しても，ごく短時間で秘密データを取り出されてしまう危険性があることが分かる．つまり，表層的に理解が難しくなっただけでは，プログラムの耐タンパー性としては，不十分であるといえる．したがって，秘密データの守秘性を考えた耐タンパー化プログラムを考えた場合，提案評価法に対して十分な耐性を持つことが最低限の条件となるといえる．

5.2 提案手法の効率

入力暗号文数 $m = 1$ のときは入力の値に依存するデータの集合 $\Phi_\ell \setminus \Omega$ とそうでない集合 Ω を分けていない場合，$m = 2$，3 は分けている場合であると考えられる．実験結果のグラフ（**図 7**，**図 8**，**図 9**，**図 10**）を見ると，データの試行回数 $Trials$ は $m = 1$ よりも $m = 2$，3 のときの方が少なくなっていることが分かる．したがって，Φ_ℓ から Ω を分割した方が，秘密データを探索するのに効率的であったといえる．このことから，プログラムの解析を行う場合には，入力値を様々に変えて実行を行うことで，解析を効果的に行うことが可能であるといえる．

提案評価法では，秘密データの候補集合 Ω に属し，check-round の入力として想定する元を配列 data[ ] に格納し，探索を行っていく．したがって，配列 data[ ] に格納されなかった Ω の元の中に秘密データが存在

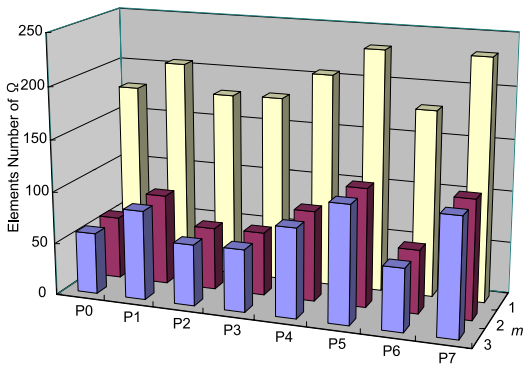


図 7 P_u と Ω の関係：RC5
Fig. 7 Relation between P_u and Ω: RC5.

する場合には，提案手法では秘密データを見つけられないことになる．この場合，check-round の入力となる元の想定を変更して，配列 data[ ] を再構成して探索を行うことで，秘密データを見つけ出せる可能性はある．しかし，その可能性は配列 data[ ] の構成の仕方に依存する．配列 data[ ] を再構成して探索を行うというプロセスを繰り返すことで，秘密データを見つけ出せる可能性は高くなるが，提案評価法の効率は下がっていくことになる．

5.3 データ守秘性を考慮した耐タンパー化手法

実験でラウンド鍵系列 $\{S_i\}$ を見つけられた理由と

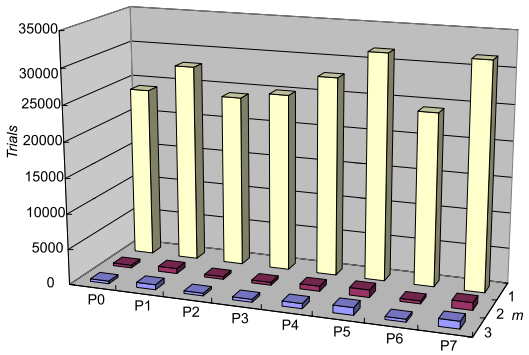


図 8 P_u と $Trials$ の関係：RC5
Fig. 8 Relation between P_u and $Trials$: RC5.

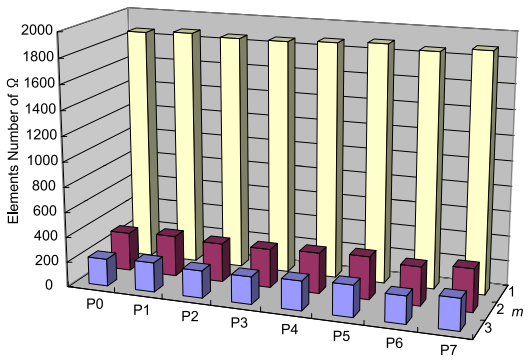


図 9 P_u と Ω の関係：DES
Fig. 9 Relation between P_u and Ω: DES.

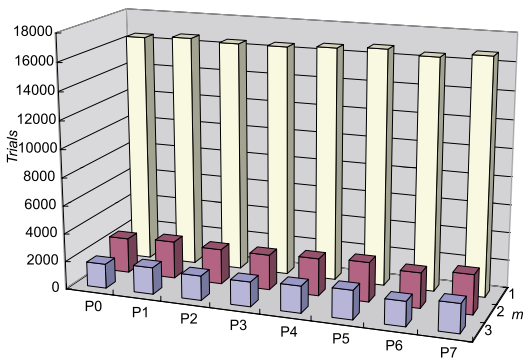


図 10 P_u と $Trials$ の関係：DES
Fig. 10 Relation between P_u and $Trials$: DES.

して，すべての S_i がメモリ上に現れるような実装であったことと，ラウンド関数 f_i の出力がメモリ上に現れていたことの 2 点が考えられる．提案評価法に対して耐性を持つような耐タンパー化プログラムを構成するためには，2 つの方針があると考えられる．1 つは秘密データがメモリ上に現れないように等価変換を行うこと，もう 1 つはラウンド関数 f_i を変換し，出力値を f_i のものとは異なるものとすることである．

6. ま と め

耐タンパーソフトウェアのデータ守秘性に関する評価法として，ランタイムデータ探索型耐タンパー性評価法を提案した．そして，実験を行い，その有効性を確かめた．

提案評価法は，メモリ上に現れるすべてのデータを抽出して検査を行うので，実行中に秘密データが現れる限りは，それを見つけることができる．つまり，提案評価法により秘密データを取り出されてしまう耐タンパー化プログラムは，秘密データ守秘性のうえで脆弱さを持っていると判断できる．したがって，提案評価法に対して十分な耐性を持つことが，秘密データ守秘性において最低限の条件であると考えられる．

提案評価法においては，その耐タンパー化対象プログラムが実現する関数として，2 入力 1 出力の暗号の復号関数としているが，もちろん暗号化関数でもよい．

また，Java などの高級言語では，実行環境が公開されている場合が多いので，解析者は自分の都合がよいように実行環境をエミュレートすることができる．つまり解析者は，解析しやすいように Data Extractor を作ることも十分に考えられる．したがって本提案方式は，厳しく耐タンパー化プログラムの耐タンパー性を評価しているといえる．

今後の課題は，Data Extractor に対する制限を取り除くこと，RC5 や DES 以外の暗号方式に関する Checker を構成していくこと，コンパイラの最適化機能やプロファイラを用いた最適化が提案評価法に与える影響を考察することがあげられる．

謝辞 本研究の一部は，文部科学省平成 13 年度科学技術振興調整費の支援を受けて行われた．

参 考 文 献

1) 坂巻辰哉，赤井健一郎，松本 勉：耐タンパーソフトウェア技術の評価法の一提案，2000 年暗号と情報セキュリティシンポジウム予稿集，SCIS2000-D09 (2000).
2) 赤井健一郎，松本 勉：耐タンパーソフトウェアの全数探索型強度評価法，信学技報，Vol.99, No.699, IT99-81 (2000).
3) 赤井健一郎，松本 勉：共通鍵暗号ソフトウェアの全数探索型耐タンパー性評価法，コンピュータセキュリティシンポジウム 2000 論文集，pp.205–210 (2000).
4) 赤井健一郎，松本 勉：共通鍵暗号ソフトウェアの全数探索型耐タンパー性評価法（その 2），2001 年暗号と情報セキュリティシンポジウム

（SCIS2001）予稿集，Vol.2, 9C-4, pp.507–512 (2001).
5) 三澤　学，赤井健一郎，松本　勉：ランタイムデータ探索型耐タンパー性評価法の効率化，信学技法，Vol.101, No.311, ISEC2001-58, pp.37–44 (2001).
6) 門田暁人，高田義広，鳥居宏次：プログラムの難読化法の提案，情報処理学会第 51 回全国大会講演論文集，5G-7, pp.4-263–4-264 (1995).
7) 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 (1997).
8) 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報告，ISEC95-25 (1995).
9) 村山隆徳，満保雅浩，岡本栄司，植松友彦：プログラムコードの難読化について，1996 年暗号と情報セキュリティシンポジウム講演論文集，SCIS1996-8D (1996).
10) Collberg, C., Thomborson, C. and Low, D.: A taxonomy of obfuscating transformations, Technical Report 148, Department of Computer Science, University of Auckland (1997).
11) Collberg, C., Thomborson, C. and Low, D.: Manufacturing cheap, resilient, and stealthy opaque constructs, *Proc. 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages '98*, pp.184–196, ACM (1998).
12) 松村憲二郎，後藤英昭，満保雅浩，静谷啓樹：耐タンパー性ソフトウェアの強度評価法について，2000 年暗号と情報セキュリティシンポジウム予稿集，SCIS2000-D08 (2000).
13) Rivest, R.L.: The RC5 Encryption Algorithm, *Proc. 1994 Leuven Workshop on Fast Software Encryption*, pp.86–96 (1995).
14) National Bureau of Standards (U.S.): Data Encryption Standard, Federal Information Processing Standards Publication 46 (1977).
15) 井上信吾，赤井健一郎，桑原　潤，坂巻辰哉，松本　勉：情報利用管理と耐タンパーソフトウェア，1999 年暗号と情報セキュリティシンポジウム予稿集 T1-2.1 (1999).
16) jasmin. http://mrl.nyu.edu/meyer/jvm/jasmin.html, (Last visited June. 2002).
17) D-Java. http://www.cat.nyu.edu/ meyer/jvm/djava/, (Last visited June. 2002).
18) Jon, M. and Troy, D. ：Java バーチャルマシン，オライリー・ジャパン (1997).

付　　録

実験で用いた RC5 を実装した耐タンパー化対象プログラム P_0（**図 11**）と，評価対象プログラム P_7（**図 12**）を示す．鍵の値は表 1 の鍵 k_3 である．メソッド rightRotate() は右巡回シフトを示す．

```
static long decrypt( long input ){
   long output = 0;

   int[] S = {
        0x13c14f3eL,0xe52bd825L,0xe9490b53L,
        0x39ffa2a6L,0x1bb37783L,0x06f70e48L,
        0xd4cf83e0L,0xaee722c5L,0xcefc65d7L,
        0xfe607c34L,0xa5dc8312L,0xb6215a1dL,
        0xaa45d162L,0x98a33ed3L,0x2e871858L,
        0xa02062ebL,0x04c31e93L,0x8e4ceb0aL,
        0xf8b21e95L,0xc237815bL,0x8088c012L,
        0x2f8fd445L,0x94e83bdfL,0x85fe2ac0L,
        0x45fbc614L,0xc675c4fdL
        }; \\Round Key
   int A = (int)(input>>>32);
   int B = (int)(input&0x00000000ffffffffL);

   for( int i = 12 ; i >= 1 ; i-- )
   {
      B = rightRotate(B-S[2*i+1],A)^A;
      A = rightRotate(A-S[2*i],B)^B;
   }
   B -= S[1];
   A -= S[0];
   output = (output+A)<<32;
   output += B;
   return output;
}
```

図 11　耐タンパー化対象プログラム P_0: RC5
Fig. 11　A target program P_0: RC5.

```
static long decrypt( long input )
{
    long output = 0;
    int[] S = {
       0x13c14f3eL,0xe52bd825L,0xe9490b53L,
       0x39ffa2a6L,0x1bb37783L,0x06f70e48L,
       0xd4cf83e0L,0xaee722c5L,0xcecf65d7L,
       0xfe607c34L,0xa5dc8312L,0xb6215a1dL,
       0xaa41d612L,0x98a33ed3L,0x2e871858L,
       0xaa45d162L,0x04c31e9eL,0x8e4ceb0aL,
       0xf8b21e95L,0xc237815bL,0x8088c012L,
       0x2f8fd445L,0x94e82bdfL,0x85fe2zc0L,
       0x45fbc614L,0xc675cf4dL
       }; \\Round Key
    int A = (int)(input>>>32);
    int B = (int)(input&0x00000000ffffffffL);

    int i = 12;
    int so = 0;
    int se = 0;
    for(;;)
    {
       if( i<1 ) break;
       so = S[2*i+1];
       so ^= 0xaaaaaaaa;
       B = rightRotate(B-so,A)^A;
       so ^= 0xbbbbbbbb;
       se = S[2*i];
       se ^= 0xaaaaaaaa;
       A = rightRotate(A-se,B)^B;
       se ^= 0xbbbbbbbb;
       i--;
    }
    so = S[1];
    so ^= 0xaaaaaaaa;
    B -= so;
    so ^= 0xbbbbbbbb;
    se = S[0];
    se ^= 0xaaaaaaaa;
    A -= se;
    se ^= 0xbbbbbbbb;
    output = (output+A)<<32;
    output += B;
    return output;
}
```

図 12　評価対象プログラム P_7: RC5
Fig. 12　An evaluated program P_7: RC5.

(平成 13 年 12 月 3 日受付)
(平成 14 年 6 月 4 日採録)

赤井健一郎

1975 年生．2001 年横浜国立大学大学院人工環境システム学専攻博士課程前期修了．同年同大学院環境情報学府情報メディア環境学専攻博士課程後期に進学し，現在に至る．情報セキュリティの研究に従事．

三澤　学

1977 年生．2000 年横浜国立大学工学部電子情報工学科卒業．2002 年同大学大学院工学研究科人工環境システム学専攻博士課程前期修了．同年三菱電機株式会社入社．

松本　勉（正会員）

1958 年生．1986 年東京大学大学院博士課程（電子工学）修了，工学博士．同年横浜国立大学工学部専任講師．現在，同大学大学院環境情報研究院教授．1981 年より主として暗号や情報セキュリティの研究・教育に従事.「明るい暗号研究会」を数人の仲間とともに創り研究を始めた．ASIACRYPT '96 プログラム委員長．ASIACRYPT 2000（国際暗号学会主催）実行委員長．電子情報通信学会より「情報セキュリティの基礎理論」への貢献に関して業績賞を受賞．