命令のランダム性に基づくプログラム難読化の評価

二村 阿美　門田 暁人　玉田 春昭　神崎 雄一郎　中村 匡秀
松本 健一

本論文では，情報理論からのアプローチにより，プログラム解析の困難さを定量化する．基本アイデアは，各命令が全くランダムにプログラム中に出現する場合に，プログラム解析が最も困難な状態であると考える．命令がランダムに出現するとは，(1) 各命令の出現頻度が等しく，かつ，(2) その並びに規則性がない状態であると考える．前者はエントロピー，後者はコルモゴロフ複雑性の概念を用いて定量化する．ケーススタディを通して，本提案の妥当性や今後の課題を考察する．

This paper quantifies the difficulty of program analysis based on the information theory. The basic idea is to consider that a program is ultimately obfuscated if instructions appear at random; that is, (1) all instructions has an equal frequency of appearance, and (2) there is no pattern observed in the instruction sequence. We quantified (1) based on the entropy and (2) based on the Kolmogorov complexity. We evaluated the feasibility of our proposal through a case study.

1　はじめに

近年，不正行為を目的としたプログラム解析を防ぎたいという社会的要求が高まっており，数多くのプログラム難読化法が提案されてきた[3].

しかし，プログラム難読化の度合い (理解性) やその効果を評価することは容易ではない．プログラム理解の目的や手段が多種多様なためである．例えば，プログラムの複雑さメトリクスにより理解の困難さを評価する試みがある[2] が，複雑さメトリクスとプログラム理解に要する時間との間には必ずしも相関がないことも示されている[10]．また，人間にプログラム解析を行わせて評価する方法[6] やゴール木による評価[14] も行われているが，評価に多大な労力を要する点が問題となる．

本論文では，情報理論からのアプローチにより，プログラム解析の困難さ (理解性) を定量化することを試みる．基本アイデアは，プログラムを読んでもそれがノイズにしか見えない状態，つまり，各命令が全くランダムにプログラム中に出現する場合に，プログラムの理解性が最も低い状態であると考える．これは，暗号文の解析困難さの考え方に基づいている．暗号文に非ランダム要素があれば解析の手がかりとなるが，暗号文が全くランダムであれば解析は困難である．ここで，命令の出現がランダムであるとは，(1) 各命令の出現頻度が等しく，かつ，(2) その並びに規則性がない状態であると考える．本論文では，前者はエントロピーの概念を，後者はコルモゴロフ複雑性の概念を用いることで評価する．

Evaluation of Software Obfuscation Based on the Randomness of Instructions.

Ami Futamura, Akito Monden, 奈良先端科学技術大学院大学情報科学研究科, Graduate School of Information Science, Nara Institute of Science and Technology.

Haruaki Tamada, 京都産業大学コンピュータ理工学部, Faculty of Computer Science and Engineering, Kyoto Sangyo University.

Yuichiro Kanzaki, 熊本高等専門学校人間情報システム工学科, Dept. of Human-Oriented Information Systems Engineering, Kumamoto National College of Technology.

Masahide Nakamura, 神戸大学大学院工学研究科, Graduate School of System Informatics, Kobe University.

Kenichi Matsumoto, 奈良先端科学技術大学院大学情報科学研究科, Dept. of Graduate School Of Information Science Nara Institute of Science And Technology.

コンピュータソフトウェア, Vol.30, No.3 (2013), pp.18–24.

[研究論文 (レター)] 2013 年 1 月 14 日受付.

なお，本論文では，理解性の評価対象として，オペコードのみを扱う．オペランドの扱いについては今後の課題とする．また，ケーススタディでは，C 言語をコンパイルして得られる x86 アセンブリプログラムを対象とした実験を行う．以降では，提案方法，および，ケースタディについて述べる．

2 提案手法

2.1 エントロピーによる命令出現頻度のランダム性の評価

エントロピーとは，情報の乱雑さ (無秩序さ) を定量的に表す指標である [13]．シャノンの情報理論においては，1 つの事象 (情報源シンボル) が生起したことを知ったことにより得られる情報量の期待値 (平均情報量) として位置づけられる [8]．エントロピー $H(S)$ は，式 (1) で定義される．

$$H(S) = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i) \quad [\text{ビット/シンボル}] \tag{1}$$

ここで，s_i は情報源シンボル，$S = \{s_1, s_2, ..., s_n\}$ は情報源アルファベット，$P(s_i)$ は s_i の生起確率を表し，

$$\sum_{i=1}^{n} P(s_i) = 1$$

である．

本論文では，s_i を命令 (オペコード)，S を全命令の集合とし，$P(s_i)$ は与えられたプログラムにおける命令 s_i の出現頻度とする．

一般に，n が大きいほどエントロピーが大きくなるため，最大エントロピーで正規化した値として，(2) 式のとおりに相対エントロピー $h(S)$ が定義される．

$$h(S) = \frac{H(S)}{\log_2 n} \tag{2}$$

本論文では，難読化の評価に，この相対エントロピー $h(S)$ を用いる．全ての命令の出現頻度が等しいとき，つまり，最も難読化されている場合，$h(S) = 1$ となる．生起頻度の小さい命令がある場合は，$h(S)$ は小さな値を取る．

従来研究において，神崎ら [4] は，(一般的なプログラムと比較して) 出現頻度が著しく低い命令がプログラム中に含まれる場合，その命令がプログラム解析の手がかりとなり得ることを示している．本論文のアプローチでは，相対エントロピー $h(S)$ が 1 に近づくと，どの命令も出現頻度がほぼ等しくなるため，プログラム解析の手がかりが少なくなることになる．

なお，上記のエントロピーの定義では，記憶のない情報源が仮定されているが，実際のプログラムでは，各命令は互いに独立に出現するとはいえず，エルゴードマルコフ情報源を仮定する方が望ましい．ただし，その場合，各命令の出現パターンを長期間観測する必要があり，相当に長いプログラムでなければエントロピーを正しく算出できないという問題があるため，今後の課題とする．

2.2 コルモゴロフ複雑性による命令出現順序のランダム性の評価

2.2.1 コルモゴロフ複雑性の概念

ここでは，ADD と SUB からなる命令数 10 の次の 2 つのプログラムを考える．

プログラム 1: ADD SUB ADD SUB ADD SUB ADD SUB ADD SUB

プログラム 2: SUB ADD SUB SUB ADD ADD SUB ADD ADD SUB

いずれのプログラムも ADD と SUB の出現回数は同じであるためエントロピーは等しいが，前者は “ADD SUB” のパターンが 5 回繰り返されるのに対し，後者は特段のパターンが見られないため，後者の方が理解性が低いといえる．本論文では，この 2 者を区別する概念としてコルモゴロフ複雑性を導入する．

コルモゴロフ複雑性は，文字列の複雑さを表現する概念であり，与えられた文字列を出力するプログラムの最小の長さとして定義される．前述の 2 つの命令列の例では，前者は，例えば，

print “ADD SUB” × 5

のように短く書くことができるが，後者は，前者よりは短く書くことができない．従って，後者は前者よりも大きなコルモゴロフ複雑性を持つことになる．

より厳密には，コルモゴロフ複雑性 $K_u(x)$ は，有限長の文字列 x，および，万能計算機 (万能チューリ

ングマシン)u が与えられたときに，計算機 u のための x を出力するプログラム p の最小の長さとして定義される．なお，コルモゴロフ複雑性は，計算機 (プログラミング言語) に依存した関数であるが，言語の選択は本質ではない．言語の選択によってコルモゴロフ複雑性はたかだか定数分しか増減しないためである．

プログラム理解においては，プログラム p のコルモゴロフ複雑性 $K_u(p)$ は，プログラマが理解すべき最低限の情報量を表していると考えられる．プログラム p が大きくても，$K_u(p)$ が小さいならば，理解すべき情報量は少なくて済む．例えば，前述のプログラム 1 のように繰り返しを含む場合，そうでないプログラム 2 よりも理解すべき情報量は小さい．

2.2.2 コルモゴロフ複雑性の算出

現実には，コルモゴロフ複雑性は計算不可能である．そのため，近似値を求める必要がある．従来，コルモゴロフ複雑性の計算方法として，文字列 x を可逆圧縮により究極に圧縮したときのビット数 $C(x)$ として近似する方法が提案されている [1]．本研究ではこれを利用する．文字列 x に何らかの規則性や冗長性が含まれている場合，x は圧縮可能なため $C(x)$ は x のサイズ $l(x)$ より小さくなり，x に規則性や冗長性が含まれない場合は，圧縮できないため $c(x) \simeq l(x)$ となる．圧縮ツールとしては，なるべく圧縮効率の高いものが望ましいため，本研究では bzip2 を使用する．

ただし，実際の文字列 p を圧縮して $C(p)$ を求める場合，注意が必要である．命令のランダム性を正しく評価するためには，各命令をなるべく対等に扱う必要がある．例えば，ある命令の表現が別の命令の表現のサブセットになっている場合 (例えば mul と imul) は，サブセットになっていない場合 (add と sub) よりも圧縮できてしまい，命令のランダム性を正しく評価できなくなる．

各命令を対等に扱うために，本論文では，命令を文字列として扱うのではなく，あらかじめ各命令を固定長のビット列に符号化してから圧縮する．各命令を区別可能な最小のビット長を決め，ビット列を割り当ててプログラムを表現する．命令の種類数を r とすると，例えば $r = 32$ の場合，$\log_2 32$ より 5 ビットで全命令を区別して表現できる．ここで注意するべきことは，命令の種類数 r が 2 のべき乗でない場合，命令が割り当てられない (つまりプログラム中に存在しない) ビットパターンが生じることである．例えば $r = 33$ の場合，5 ビットでは全命令を表現できないため，1 命令あたり 6 ビットが必要になるが，冗長なビットパターンが生じてしまう．たとえ命令の出現順序が全くランダムであっても，冗長なビットパターンが存在すればプログラムを圧縮できてしまい，命令のランダム性を正しく評価できない．そこで，圧縮前のプログラムの長さ $l(p)$ を，冗長なビットがないと仮定した場合のプログラムの長さとして与えることとする．$l(p)$ は次式の通りである．

$$l(p) = n \log_2 r \tag{3}$$

n はプログラム p に含まれる命令数である．このように，本論文ではコルモゴロフ複雑性の近似値 $K(p)$ を，符号化されたプログラム p を圧縮したときのビット数 $C(x)$ として定義する．

次に，相対コルモゴロフ複雑性 $k(p)$ の算出について説明する．相対コルモゴロフ複雑性は，コルモゴロフ複雑性を圧縮前のプログラムの大きさで正規化したものと定義する．$k(p)$ は次式の通りである．

$$k(p) = \frac{K(p)}{l(p)} \tag{4}$$

命令順序に規則性が全くない場合，p を圧縮できたとしても圧縮後のサイズは $l(p)$ となり，$k(p) \simeq 1$ となる．完全に 1 にならないのは，コルモゴロフ複雑性が計算不可能であり，圧縮ツールを使用するなど，なんらかの方法で近似して求めるためである．

相対コルモゴロフ複雑性を用いることで，規模が異なるプログラムを比較できる．相対コルモゴロフ複雑性が最大値を取るためには，相対エントロピー $h(S) = 1$ を満たすことが必要条件である．$h(S) < 1$ の場合，即ち命令の出現頻度に偏りがある場合，出現頻度の高い命令をより短いビット列を符号化して保持し，それを復号するようなプログラムを書くことで，短い $k(p)$ を作成できるためである．

ただし，命令の種類数が少ない場合，規則性が現れやすくなるため，相対コルモゴロフ複雑性は小さくな

る傾向にある．そこで，相対コルモゴロフ複雑性を命令の種類数で正規化した正規化コルモゴロフ複雑性によっても評価を行う．正規化コルモゴロフ複雑性を次式のとおり定義する．

$$k'(p) = \frac{k(p)}{r} \tag{5}$$

正規化コルモゴロフ複雑性を用いることで，命令の種類数が異なるプログラムを比較できる．他の尺度と同様に，命令順序に規則性が少ない場合，正規化コルモゴロフ複雑性は大きくなる．

3 ケーススタディ

本ケーススタディの目的は，難読化前のプログラムと難読化後のプログラムで，相対エントロピー $h(S)$，コルモゴロフ複雑性 $K(p)$，相対コルモゴロフ複雑性 $k(p)$，正規化コルモゴロフ複雑性 $k'(p)$ をそれぞれ比較し，これらの尺度が難読化の評価に利用できるかを検討することである．

3.1 実験概要

評価対象は，AES(Advanced Encryption Standard) [11] による暗号化・復号を行う C プログラムをコンパイルして得られる x86 アセンブリプログラムである．コンパイルには，cygwin 環境の gcc 4.5.3 を用いた．この AES プログラムに対し，次の 5 つの難読化手法を適用した．

- O_1　ループ難読化 [9]
- O_2　関数の分割・マージ [3]
- O_{3a}　偽装コードの追加 (ランダム) [15]
- O_{3b}　偽装コードの追加 (パターンを消す) [15]
- O_4　高レベル命令を低レベル命令に書き換える [15]

難読化対象はそれぞれ，O_1,O_2 は C プログラム，O_{3a},O_{3b},O_4 は C プログラムをアセンブリしたアセンブリプログラムである．実験の手順を説明する．まず，対象の C プログラムに難読化 O_1,O_2 をそれぞれ適用する．本実験の対象はアセンブリプログラムなので，難読化後の C プログラムをコンパイルして，アセンブリプログラムを作成する．評価対象のプログラムをそれぞれ $O_1(p)$,$O_2(p)$ とする．また，対象の C プログラムをコンパイルし，得られたアセンブリプログラムを p とする．p に難読化 O_{3a},O_{3b},O_4 を適用する．評価対象のプログラムをそれぞれ $O_{3a}(p)$,$O_{3b}(p)$,$O_4(p)$ とする．p,$O_1(p)$,$O_2(p)$,$O_{3a}(p)$,$O_{3b}(p)$,$O_4(p)$ の相対エントロピー $h(S)$，コルモゴロフ複雑性 $K(p)$，相対コルモゴロフ複雑性 $k(p)$，正規化コルモゴロフ複雑性 $k'(p)$ を計測する．

表 1 アセンブリプログラムの命令数および命令の種類数

プログラム	命令数 n	命令の種類数 r
p (難読化前)	1355	31
$O_1(p)$	1411	32
$O_2(p)$	1388	32
$O_{3a}(p)$	1384	31
$O_{3b}(p)$	1717	33
$O_4(p)$	1429	27

アセンブリプログラムの命令数および命令の種類数を表 1 に示す．表 1 に示されるように，命令数は難読化によって増加している．命令の種類数についても，変わらないかあるいは増加しているが，$O_4(p)$ のみ減少している．$O_4(p)$ は高レベル命令を低レベル命令に置き換える難読化であり，置き換えによってプログラム中から完全に存在しなくなった高レベル命令があるため，難読化前と比較して命令の種類数が減少した．

3.2 結果

難読化の評価を行った結果を表 2 に示す．

難読化によって，相対エントロピー $h(S)$ はほとんど増加していない．解析の手がかりとなりうる出現頻度の低い命令が，プログラム中にまだまだ存在することがわかる．手がかりの隠ぺいという観点では，

表 2 評価結果

	$h(S)$	$K(p)$	$k(p)$	$k'(p)$
p (難読化前)	0.74	507	0.076	0.0024
$O_1(p)$	0.74	553	0.078	0.0025
$O_2(p)$	0.72	531	0.077	0.0024
$O_{3a}(p)$	0.74	529	0.077	0.0025
$O_{3b}(p)$	0.76	1019	0.118	0.0036
$O_4(p)$	0.69	484	0.071	0.0026

実験で適用した難読化は不十分である．もし，手がかりとなりうる出現頻度の低い命令を隠ぺいしたいのならば，他の難読化を適用するべきであることが分かった．

$O_{3b}(p)$ は，難読化前の p と比較してコルモゴロフ複雑性の値が約 2 倍になっている．意図的にパターンを消去することの重要性が示された．

$O_4(p)$ は，コルモゴロフ複雑性，相対コルモゴロフ複雑性の値が減少している．減少の原因は，同じ命令置換方法を繰り返し適用したためであり，難読化法の実装に問題があることを発見できた．

3.3 追加実験

実験の結果より，O_4 の難読化の実装方法に問題があることが発見できた．そこで，O_4 と O_{3b} を併用した難読化を新たに O_5 として評価を行う．$O_5(p)$ の命令数と命令の種類数を表 3 に示す．

表 3 より，$O_5(p)$ は命令数が著しく増大しているが，O_4 を併用したことによって命令の種類数は減少していることがわかる．

結果を表 4 に示す．難読化前のプログラム p あるいは，難読化を単体で適用した O_{3b} や O_4 と比較して，コルモゴロフ複雑性，正規化コルモゴロフ複雑性が増大している．命令の種類数が減ったにもかかわらず，コルモゴロフ複雑性が増大しており，従って正規化コルモゴロフ複雑性が増大している．これらの結果より，複数の難読化を併用した効果を読み取ることができた．

表 3 アセンブリプログラムの命令数および命令の種類数

プログラム	命令数 n	命令の種類数 r
p (難読化前)	1355	31
$O_{3b}(p)$	1717	33
$O_4(p)$	1429	27
$O_5(p)$	1994	29

表 4 評価結果

	$h(S)$	$K(p)$	$k(p)$	$k'(p)$
p (難読化前)	0.74	507	0.076	0.0024
$O_{3b}(p)$	0.76	1019	0.118	0.0036
$O_4(p)$	0.69	484	0.071	0.0026
$O_5(p)$	0.72	1114	0.115	0.0040

3.4 考察

エントロピーの計測により，命令の出現頻度の均一さの観点からプログラム解析の困難さを評価した結果，既存の難読化手法はこの点では効果がないことが明らかとなった．また，コルモゴロフ複雑性により，より短く書けるような繰り返し部分があるか (命令の並びに規則性があるか) という観点から，解析の困難さを評価した結果，難読化 O_4 の実装上の問題点を見つけることができた．このことから，提案尺度は，難読化の評価の一助となると期待される．

一方で，既存の難読化法によってエントロピーやコルモゴロフ複雑性は必ずしも増大しないことが分かった．既存の難読化法はそれぞれプログラム解析を妨げる効果があるはずであるから，提案尺度のみを用いて難読化の効果を測ることはできないと言える．従って，提案尺度は，あくまでも命令の出現頻度の均一さと規則性という観点からの評価であることに注意する必要がある．

4 関連研究

Kirk ら [5] は，本論文と同様，コルモゴロフ複雑性をソフトウェアメトリクスとして提案し，デザインメトリクスの代替となり得るかを検討している．ただし，Kirk らは，Java クラスファイル全体を圧縮して得られるビット数をコルモゴロフ複雑性として用いており，クラスファイル中の多様な要素 (クラスヘッダ，コンスタントプール，フィールド情報など) のいずれがコルモゴロフ複雑性に寄与しているかは不明確であった．また，相対コルモゴロフ複雑性に該当するメトリクスは提案しておらず，命令のランダム性を評価していない．本論文は，プログラム中からオペコードの系列だけを取り出して，各オペコードが対等になるように符号化してから (相対) コルモゴロフ複雑性を求めることで，命令のランダム性を評価している点が異なる．

Buse ら [16] は，コードの可読性についての論文で

ある．Buse らは，プログラム中の変数名の長さやインデント，記号など複数の要因に着目して可読性を評価している．我々は，プログラムのオペコードに絞って評価しているという点が異なる．

5 終わりに

本論文では，エントロピーとコルモゴロフ複雑性に基づいてプログラム解析の困難さを定量化する方法を提案した．ケーススタディにおけるエントロピー計測の結果，既存の難読化手法は，命令の出現頻度を均一に近づけて手がかりを消すという効果が弱いことが明らかとなった．また，コルモゴロフ複雑性の計測の結果，命令の並びに規則性の観点から，難読化法の実装の問題点を見つけることができた．

本論文の制約は次の通りである．まず，本論文では，オペコードを対象とし，その系列の理解性のみを評価している．オペランドも含めた理解性評価や，名前 (関数名や変数名など)，コメント文などの理解性評価は対象外である．また，本論文では，他の理解性尺度との比較を行っていない．その理由は，本論文ではオペコードの理解性のみを評価しているため，他の尺度との直接的な比較が困難なためである．今後，オペコード以外の要素についても評価方法を検討し，他の尺度との比較を行っていく予定である．また，本論文では，一部の難読化法のみを用いた評価となっており，他の難読化法を用いた評価も今後必要となる．

参 考 文 献

[1] Cilibrasi, R. and Vitanyi, P.: Similarity of objects and meaning of the words, in *Proceedings of Third International Conference on Theory and Applications of Models of Computation*, Lecture Notes in Computer Science, Vol. 3959, Springer, 2006, pp. 21–45.

[2] Colberg, C., Thomborson, C. and Low, D: A taxonomy of obfuscating transformations, Department of Computer Science, Technical report 148, Department of Computer Science, University of Auckland, 1997.

[3] Collberg, C. and Nagra, J.: Surreptitious software, Addison-Wesley Professional, 2009.

[4] 神崎雄一郎, 門田暁人：ソフトウェア保護機構を構成するコードの特徴評価の試み，コンピュータセキュリティシンポジウム 2011(CSS2011) 予稿集, 2011, pp. 827–832.

[5] Kirk, S. R. and Jenkins, S.: Information theory-based software metrics and obfuscation, *Journal of Systems and Software*, Vol. 72, Issue 2 (2004), pp. 179–186.

[6] 松岡賢, 赤井健一郎, 松本勉：鍵内蔵型暗号ソフトウェアの人手による耐タンパー性評価, in *Proceedings of Symposium on Cryptography and Information Security*, No. 6C-2, 2002, pp. 359–364.

[7] Miltersen, P: Course notes for Data Compression —2 Kolmogorov complexity Fall 2005, University of Aarhus, 2005, http://www.daimi.au.dk/~bromille/DC05/Kolmogorov.pdf

[8] 宮川洋：情報理論, コロナ社, 1979.

[9] 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 D-I, Vol. J80-D-I, No. 7 (1997), pp. 644–652.

[10] Nakamura, M., Monden, A., Itoh, T., Matsumoto, K., Kanzaki, Y. and Satoh, H.: Queue-based cost evaluation of mental simulation process in program comprehension, in *Proceedings of 9th IEEE International Software Metrics Symposium* (*METRICS2003*), 2003, pp. 351–360.

[11] National Institute of Standards and Technology (NIST): *Advanced Encryption Standards* (*AES*), Federal Information Processing Standards Publication 197, 2001.

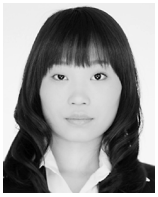
[12] Ogiso, T., Sakabe, Y., Soshi, M. and Miyaji, A: Software obfuscation on a theoretical basis and its implementation, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science, Vol. E86-A, No. 1 (2003), pp. 176–186.

[13] 山下洋史：エントロピー・モデルにおけるエントロピーの役割, 明大商学論叢, Vol. 84, No. 2 (2002), pp. 71–88.

[14] Yamauchi, H., Monden, A., Nakamura, M., Tamada, H., Kanzaki, Y. and Matsumoto, K: A goal-oriented approach to software obfuscation, *International Journal of Computer Science and Network Security*, Vol. 8, No. 9 (2008), pp. 59–71.

[15] 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, in *Proceedings of Symposium on Cryptography and Information Security*, No. 8D, 1996, pp. 1–6.

[16] Buse, R. and Weimer, W.: Learning a metric for code readability, *IEEE Transactions on Software Engineering*, Vol. 36, Issue 4 (2010), pp. 546–558.

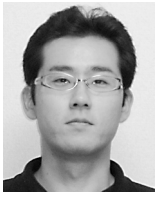


二村阿美

2011 年名古屋大学工学部情報工学科卒業．奈良先端科学技術大学院大学情報科学研究科博士前期課程在学中．ソフトウェア工学の研究に従事．

門田暁人

1994 年名古屋大学工学部電気学科卒業．1998 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年同大学同研究科助手．2004 年同大学助教授．2007 年同大学准教授．2003〜2004 年 Auckland 大学客員研究員．博士 (工学)．ソフトウェアメトリクス，ソフトウェアプロテクション，ヒューマンファクタ等の研究に従事．日本ソフトウェア科学会，情報処理学会，電子情報通信学会，IEEE, ACM 各会員．



玉田春昭

1999 年京都産業大学工学部情報通信工学科卒．2001 年同大学大学院博士前期課程了．2006 年奈良先端科学技術大学院大学情報科学研究科博士後期課程了．同年同大学産学官連携研究員．2007 年同大学情報科学研究科特任助教．2008 年京都産業大学コンピュータ理工学部助教．博士 (工学)．ソフトウェアプロテクション，エンピリカルソフトウェア工学の研究に従事．電子情報通信学会, IEEE 各会員．

神崎雄一郎

2001 年神戸大学工学部情報知能工学科卒業．2006 年奈良先端科学技術大学院大学情報科学研究科博士後期課程修了．同年熊本電波工業高等専門学校情報工学科助手．2010 年熊本高等専門学校人間情報システム工学科准教授．博士 (工学)．ソフトウェアプロテクション等の研究に従事．電子情報通信学会，情報処理学会各会員．

中村匡秀

1994 年大阪大学基礎工学部情報科学科卒．1999 年同大学大学院博士後期課程了．2000 年同大学サイバーメディアセンター助手．2002 年奈良先端科学技術大学院大学情報科学研究科助手．2007 年神戸大学大学院工学研究科准教授．2010 年同大学院システム情報学研究科准教授．博士 (工学)．サービス・クラウドコンピューティング，ライフログ，ホームネットワーク，ソフトウェア工学の研究に従事．IEEE, ACM，情報処理学会各会員．

松本健一

1985 年大阪大学基礎工学部情報工学科卒業．1989 年同大学大学院博士課程中退．同年同大学基礎工学部情報工学科助手．1993 年奈良先端科学技術大学院大学助教授．2001 年同大学教授．工学博士．エンピリカルソフトウェア工学，特に，プロジェクトデータ収集／利用支援の研究に従事．情報処理学会，日本ソフトウェア科学会，ACM 各会員，IEEE Senior Member．