ソースコードに対するステガノグラフィの検討と実現

2002MT004 青木 一憲　　2002MT019 伊久美 茂雄　　2002MT039 河田 喬仁
指導教員　真野 芳久

1　はじめに

秘密通信を目的とし、画像や音楽などのメディア内に秘匿情報を隠し、他人に気付かれることなく送信したり保存する技術にステガノグラフィがある。ステガノグラフィは秘密通信を目的とし、大量の情報を埋め込めることが望まれる。

ステガノグラフィは効率良く情報を埋め込むことができる画像への適応が主流となっている。そのため攻撃者が画像を調べることが多い。そこで、画像だけではなく、ソースコードへの埋め込みを実現すると、攻撃者は画像以外にも目を通す必要が生じるため、攻撃が分散されると考えられる。

関連研究として、ドキュメントへのステガノグラフィがある。文字フォントを加工するなどのレイアウトの操作によるドキュメントを印字画像として扱う方法 [1] と不可視な文字を挿入するなどのドキュメントを文字列符号として扱う手法 [2] がある。

2　提案手法

本研究では、研究対象とする言語を Java とする。すべての手法において、埋め込みたい情報をビット列にして、埋め込むことができる場所を検討する。また、手法の分類としてプログラムの仕様を変更しない意味保存型、仕様を変更しても構わない意味非保存型、プログラムを構文規則にそって作成する生成型の 3 つの分類を提案する。また、検討した手法を組み合わせ実現することにより、埋め込み手法 n 個に対し $n!$ 通りの埋め込み順序が考えられ、解析を困難にする埋め込みが実現できると考えられる。

評価方法については、各手法において埋め込むことのできる情報量の比較を行なう。

3　意味保存型

意味保存型とは、情報を埋め込む前のプログラムと埋め込み後のプログラムの仕様が変わらないものとする。また、満たすべき条件としてコンパイルと実行が可能でなければならないとする。また、意味保存型での分類として、文字・字句レベル、体裁レベル、構文レベルに分け、手法を検討する。

3.1　文字・字句レベル

3.1.1　スペースによる埋め込み

プログラム中のスペースを利用して、スペースが 1 つの場合に 0、連続して 2 つの場合に 1 を埋め込む。

- 利点: プログラム中にはスペースが多く使われるため、多くの情報を埋め込むことができる。
- 欠点: 上下同じ様な事をしている場合や単語前をそろえている場合にズレが生じ、不自然になってしまう場合がある。

3.1.2　一行当たりの文字数

各行末に $2^n - 1$ 以下のスペースを入れることにより文字数を調節し、各行の文字数を 2^n で割った余りによって情報を埋め込む。この手法では 1 行当たり n bit 埋め込むことができ、本研究では $n = 2$ とする。

- 利点: 見た目で分かりにくい。
- 欠点: 行の終りを↓などで表示するエディタでは行末の空白が不自然になる。また、埋め込む量を多くしても不自然になってしまう場合がある。

3.1.3　変数宣言の並びによる埋め込み

表 1 のように辞書式順序と埋め込む情報との対応を決めて並び替えにより埋め込む。また、複数行にわたって宣言する型の違う変数もまとめて並び変えることにより、同時に並び変え可能な変数を増やす。この方法では n 個同時に宣言した場合、平均的に $\log_2 n!$ bit 埋め込むことができる。

- 利点: 同時に宣言する変数が多くなるほど埋め込み量が多くなる。
- 欠点: 複数行に渡り宣言するため、不自然になってしまう場合がある。

表 1　変数宣言の並びによる埋め込み例

変数宣言の並び	2 進情報	変数宣言の並び	2 進情報
a - b - c	000	b - c - a	011
a - c - b	001	c - a - b	10
b - a - c	010	c - b - a	11

3.2　体裁レベルへの埋め込み

3.2.1　改行による埋め込み

各行の改行の数によって情報を埋め込む。改行が 1 回の場合に 0、連続して 2 回の場合に 1 を埋め込む。

- 利点: 確実に 1 行に 1 bit 埋め込むことができる。
- 欠点: 処理毎のまとまり中に空行が入る場合、空行が欲しい所で空行がない場合には、不自然になってしまう場合がある。

3.2.2　字下げによる埋め込み

字下げの部分に Tab とスペースを使い、情報を埋め込む。Tab は一定の位置まで字下げをする機能を持つため、スペースを入れて Tab を入れた場合も同じ位置まで

字下げされることを利用した方法である。Tab の下げ幅は使用環境によって 2、4、8 文字というように異なっているため、Tab の下げ幅によって入れることのできるスペースの数が変わってくる。そこで、すべての場合に対応させるためスペース 1 つと Tab を使って表 2 のように、Tab だけによる字下げの場合に 0、スペース 1 つを入れてからの Tab による字下げの場合に 1 を埋め込む。

- 利点: 見た目で分かりにくい。また、下げ幅が 8 文字分の場合はより多くの情報を埋め込む事ができる。(字下げ 1 つ分に 3 bit)
- 欠点: 使用環境によって、 Tab の下げ幅が違う。

表 2　字下げによる埋め込み例

スペース Tab	2 進情報
Tab	0
スペース +Tab	1

3.3　構文レベルへの埋め込み

3.3.1　ループ命令 (for-while) による埋め込み

ループ命令の for(;;)、 while()、 do while() は機械的な同値変換[*1]が可能であるので、表 3 のように命令の使い分け、条件の有無で埋め込む。

- 利点: 1 つのループで 3〜4 bit を埋め込むことができる。
- 欠点: 条件の位置、命令の使い方でプログラムの統一感がなくなり、不自然になってしまう場合がある。

表 3　ループ命令による埋め込み例

ループ命令	2 進情報	ループ命令	2 進情報
for(×; ×; ×)	1000	for(○; ○; ×)	1110
for(×; ×; ○)	1001	for(○; ○; ○)	1111
for(×; ○; ×)	1010	while(○)	000
for(×; ○; ○)	1011	while(×)	001
for(○; ×; ×)	1100	do while(○)	010
for(○; ×; ○)	1101	do while(×)	011

○、×は条件の有無を表す。(while の場合は×は True、○は他の条件)

3.3.2　不等号の向きによる埋め込み

条件式での不等号の向きを使い、情報を埋め込む。不等号の向きが {>、>=} の場合に 0、{<、<=} の場合に 1 を埋め込む。

- 利点: 条件式はプログラム中でよく使われるため多くの情報を埋め込むことができる。
- 欠点: 不等号の向きに統一感がなくなり、不自然になってしまう場合がある。

3.3.3　否定による埋め込み

条件に現れる否定演算子の有無を使い、情報を埋め込む。否定演算子がない場合に 0、ある場合に 1 を埋め込む。なお、!= は否定ではなく 1 つ関係演算子として扱う。

- 利点: 条件式はプログラム中によく使われるため多くの情報を埋め込むことができる
- 欠点: 肯定と否定を入れ替える事によって、不自然になってしまう場合がある。

3.4　評価と検証

Web 上より、ファイルサイズの異なるプログラムを 5 個用意し埋め込みを行ない、各レベル毎に情報を埋め込むことができる量の検証を行なった。表 4 に各ファイルサイズとプログラムの行数、各レベル毎の埋め込み量を示す。表 4 より、構文レベルより下のレベルでは多くの情報を埋め込むことができた。しかし、構文レベルではプログラムに大きな変化がみられるが、情報を多く埋め込むことができないことがわかった。また、平均埋め込み量は 1 行当たり 10.62 bit、1 Kbyte 当たり 303.75 bit であった。

表 4　意味保存型での埋め込み量

filesize	行数(行)	埋め込み量			
		文字・字句	体裁	構文	合計
9,007	282	1,910	1,240	87	3,237
17,586	532	3,159	1,996	137	5,262
23,993	740	4,275	2,667	171	7,113
55,279	1,419	8,270	4,654	105	13,029
111,293	2,726	17,965	11,774	519	30,258

(filesizeの単位はbyte、埋め込み量の単位はbit)

4　意味非保存型

意味非保存型とは、情報を埋め込む前のプログラムと埋め込み後のプログラムの仕様が変わっても構わないものとする。また満たすべき条件として、コンパイルが可能でなければならないとする。意味非保存型では、意味保存型における文字・字句レベル、体裁レベルの埋め込みも同時に使用することができるため、構文レベルでの検討のみを行なう。

4.1　コードの並びによる埋め込み

宣言・初期化などを最初に行ない、残りのプログラムのコード毎に、辞書式順序と埋め込む情報との対応を決めて並べることにより、情報を埋め込む。if-else、switch-case の場合は、かたまりとして扱うことにする。

- 利点: プログラムには多くのコードがあるので、大量の情報を埋め込むことができる。

[*1] 変数の追加は許す。意味保存の意味での同値。例えば、元のプログラムで for(;;){S} であった時に、embed=1;do{S}while(embed==1); とすることで、{010} が埋め込まれる。

- 欠点: 処理の順序が明らかに不自然になってしまう場合がある。

4.2 等号・不等号の種類による埋め込み

条件式での等号・不等号の種類を変えることにより、情報を埋め込む。但し、意味保存型のように比較対象のものを入れ替える必要はない。また、{==、!=} と {>、>=、<、<=} の変換は型によってはコンパイル時にエラーが起きる可能性があるため、表 5 のように分けて考えることにする。

- 利点: プログラム中には等号・不等号が多く使われるので、多くの情報を埋め込むことができる。
- 欠点: 等号・不等号を変更することにより、不自然になってしまう場合がある。

表 5　等号・不等号の種類による埋め込み例

不等号の種類	2 進情報	等号の種類	2 進情報
>	00	==	0
>=	01	! =	1
<	10		
<=	11		

4.3 否定による埋め込み

条件に現れる否定演算子を使い、情報を埋め込む。否定演算子がない場合に 0、ある場合に 1 を埋め込む。但し、意味保存型のように条件を変更する必要はない。

- 利点: 条件式はプログラム中によく使われるため多くの情報を埋め込むことができる。
- 欠点: 肯定と否定を入れ替える事により、不自然になってしまう場合がある。

4.4 ループ (for-while) による埋め込み

ループ命令の for(;;)、while()、do while() の使いわけ、条件の有無で埋め込む。但し、意味保存型とは違い、終了条件以外の条件を残す必要はない。なお、情報の埋め込み方は意味保存型の表 3 と同様である。

- 利点: 1 つのループで 3〜4 bit を埋め込むことができる。
- 欠点: 条件の位置、命令の使い方でプログラムで統一感がなくなり、不自然になってしまう場合がある。

4.5 数値による埋め込み

プログラム中の初期値やプログラム中の数値を追加・変更することにより、情報を埋め込む。

プログラム中に数値が現れた時、または変数が宣言された時、埋め込みたい情報を 10 bit 毎に分け 10 進数にし、表 6 のように 10 進数 ∗3−953 などの特定の式を用い、数値を変更して情報を埋め込む。数値の場合と宣言の場合に特定の式を変え、さらに宣言は変数の型 (int、long など) によって式を変えることにより、埋め込まれている情報を特定しにくくする。byte、char、boolean 変数は使用しないものとする。

- 利点: 変数、又は数値 1 つにつき、10 bit の情報を埋め込むことができる。
- 欠点: 扱われる数が不自然になってしまう場合がある。

表 6　数値による埋め込み例

型	式
short	10 **進情報**∗3−953
int	10 **進情報**∗7−2181
long	10 **進情報**∗9−3212
float	10 **進情報**∗13+55
double	10 **進情報**∗17+103
数値：式	10 **進情報**∗3+121

4.6 演算子の種類による埋め込み

プログラム中の計算式の四則演算を、表 7 のように 3 つのグループに分類し、グループ内で演算子を入れ替えることにより、情報を埋め込む。

- 利点: プログラム中には多くの演算子が使われるため、多くの情報を埋め込むことができる。
- 欠点: コンパイルエラーは出ないが、0 演算をする可能性がある。また、ループ条件などで不自然になってしまう場合がある。

表 7　演算子の種類による埋め込み例

演算子	情報	演算子	情報	演算子	情報
+	00	+=	00	++	0
−	01	−=	01	−−	1
∗	10	∗=	10		
/	11	/=	11		

4.7 論理演算子による埋め込み

比較条件を複合させる時の論理演算子を使い、情報を埋め込む。論理演算子が||の時に 0、&&の時に 1 を埋め込む。

- 利点: プログラム中には、論理演算子が多く使われるため、多くの情報を埋め込むことができる。
- 欠点: 1 つの条件の中に多くの論理演算子が使われるとき、不自然になってしまう場合がある。

4.8 評価と検証

意味保存型で検証したファイルを用いて埋め込みを行ない、情報を埋め込むことができる量の検証を行なった。意味非保存型は構文レベルへの検討を中心に行なったため、意味保存型での文字・字句レベル、体裁レベルも用い、情報を埋め込んだ。表 8 より、意味保存型での埋め込み手法と併用することにより、より多くの情報を埋め込むことが可能になった。意味保存型を併用するこ

とにより、解析を困難にできると考えられる。また、平均埋め込み量は 1 行当たり 13.25 bit、1 Kbyte 当たり 378.52 bit であった。

表 8　意味非保存型での埋め込み量

filesize	行数(行)	埋め込み量			
		文字・字句	体裁	構文	合計
9,007	282	1,835	1,146	1,150	4,131
17,586	532	2,947	1,826	1,236	6,009
23,993	740	4,154	2,570	2,716	9,440
55,279	1,419	8,157	4,556	6,404	14,443
111,293	2,726	17,709	11,258	9,287	38,254

(filesizeの単位はbyte、埋め込み量の単位はbit)

5　生成型

生成型は、意味保存型、意味非保存型とは違い、元のプログラムを必要とせず、情報が埋め込まれているプログラム部品を用い、構文規則に従って生成する。本研究では満たすべき条件として、生成後のプログラムがコンパイル可能でなければならないとする。

5.1　埋め込み方法

プログラムを生成するにあたり、クラスや変数の宣言、初期値を予め設定したプログラム基盤を用意し、埋め込みたい情報にあったプログラム部品を構文規則 [3] に従い、カタログから選び出し、プログラムを生成する。本研究では、表 9 のように構文規則の Statement の if、for、while、do-while を扱うことにする。クラス名はツール利用者が自由につけられるものとする。また、変数名はカタログリストとして使用するため、あらかじめ宣言を行い、初期値は全て 0 とおいている。表中の StatementExpression と Expression は演算子、変数、数値などの情報によって構成されている。

表 9　使用する構文規則の一部

構文規則(Statementからの導出規則)	2進情報

if(Expression){
　Statement
[else Statement]
}
	100[0]

for(StatementExpression;Expression;
　　StatementExpression){
　Statement
}
	101

while(Expression){
　Statement
}
	110

do{ 　Statement }while(Expression)	111
StatementExpression	0

- 利点: Statement 以外の構文規則での埋め込みを行なうと、同じ情報を埋め込む場合にも、複数の選択肢からランダムに選ぶことによって情報を埋め込み、解析を困難にすることも可能である。
- 欠点: 現段階では Statement の部分しか行なっていないため、クラスを 1 つしかもたない長いプログラムになり、不自然である。

5.2　評価と検証

埋め込む文字数を 1,000 字、2,000 字、3,000 字とし、プログラムを生成して、それぞれ検証した結果を、表 10 に示す。生成後の平均埋め込み量は 1 行当たり 14.59 bit、1 Kbyte 当たりは 500.28 bit であった。

表 10　生成後の埋め込み量の比較

埋め込み量(bit)	filesize(byte)	行数(行)
8,000	15,917	543
16,000	34,739	1,065
24,000	44,638	1,711

6　おわりに

本研究では、埋め込み量を重点において考えた。そのため、実際書かれているプログラムと比べると自然さは見劣りする手法もある。しかし、提案した手法を組み合わせる順番によって、情報の解析を困難にすることができ、ステガノグラフィの本来の目的は達成できる。

意味保存型、意味非保存型では、構文レベルより大きなレベル (クラスレベルなど) を利用した手法も考えられるため、検討していくことが必要である。

生成型は、現時点では構文規則の Statement の一部しか使用していない。構文規則にそって多くの手法を用いることで、より複雑なプログラムの生成を行なうことができる。

本研究では、プログラムの自然さについてあまり考慮していない。今後、プログラムの不自然さをなくすことによって攻撃者に疑われないような埋め込みを考えていかなければならない。

参考文献

[1] 林克明, 上田芳弘, 梅原丈宏, 山本渉, 岩田雅士, 安部武彦, 木村春彦: “文字フォントの加工による情報付加手法の開発”, http:// www. irii.go.jp theme h15/pdf guidance02_3.pdf

[2] 中川裕志，滝沢修，井上信吾: “ドキュメントへのインフォメーションハイディング”, 情報処理 44 巻 3 号,pp.248-253 (2003.3).

[3] B.Joy，J.Gosling，G.Bracha，G.Steel 共著，村上雅章 訳:“Java 言語仕様 第 2 版”, ピアソン・エデュケーション (Dec. 2000).