Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

電子透かしの検出器用の難読化方式

大関和夫[†]　　魏　遠玉[††]

ソフトウェアを公開する際に難読化して，著作権を確保したり，ID，パスワード等の機密情報を隠蔽することがなされている．電子透かしの検出器を公開する際にも難読化して行うのが良いが，検出が必要なのは，著作権の争議のときであるため，常時は高速演算である必要は無い．本報告では，難読手法のうち，意味的な部分を保留し，計算量を正確に評価しながら，計算量的に仕様の設定可能な難読化手法を構築し，具体的な計算量の評価や見積例を示す．暗号化の手法を組み込み，確実な計算回数の増加を図る．プログラムの実行の途中結果の数値を素数化して復号鍵とし，以後の計算に必須な数値を暗号化して記述しておく．通常の実行も遅くなる点と，一回実行すると，復号鍵は全て入手できる点という問題が残るが，計算量負荷を確実に設定できる点が特徴である．

An Obfuscation Method for a Detector of Watermarking

KAZUO OHZEKI[†]　ENGYOKU GI[††]

Obfuscation is useful for protecting copyright of software and for hiding ID's and passwords, which are handled in software executing process. It is better to obfuscate a detector of watermarking. Actually, the detection is seldom carried out and can have the slowest performance, because the detection would be required to clarify authentication when arising disputing trouble. This report foucuses on computational amount. An essential number is encrypted by an encoding key derived from a result at the middle way in a program. This method provides a computational amount to carry out the detection software program.

1. はじめに

ソフトウェアを公開する際に難読化して，著作権を確保したり，ID，パスワード等の機密情報を隠蔽することがなされている．難読化の研究では，Barak の不可能論[1]があった. virtual Black-box を用い, 限定された関数という不自然な関数を用いた証明で，全てを対象にしたものではないことが，自己解説に述べられている[2]．そこで，この枠組みを外れた個別の難読化を行う事は依然として 意味があると考えられる．一方，難読化や暗号化処理は，途中の過程で機密情報の演算処理が観察できることから，公開領域での演算においても機密が保たれるよう White-Box 処理の必要性があり，White-Box における難読化方式の提案がある[3]．難読化処理はその前後でソフトウェアプログラムの動作結果が同じであることや，暗号手法のようにセキュリティ性を証明することが理想である．一方，コンパイラを延長したような複雑化(code obfuscation)なども使用されている．

電子透かしの検出器ソフトウェアの難読化は，実際に使用する必要性があるのは，著作権争議になり検出を確認する時であると見做せば，通常は使用されないままとなり，その動作速度が低速であっても不便とならない．本報告では，計算量が増加することを許容した形で，計算量のみに依存する難読化を確実に行う手法[4]を検討する．本報告では，難読手法のうち，まずリバースエンジニアリングにより容易にソースコードに戻るコンパイラの様なものでない意味的な難読化の部分を保留し，計算量を正確に評価しながら，計算量的に仕様の設定可能な難読化手法を構築し，具体的な計算量の評価や見積例を示す．暗号化の手法を組み込み，確実な計算回数の増加を図る．プログラムの実行の途中結果の数値を素数化して復号鍵とし，以後の計算に必須な数値を暗号化して記述しておく．通常の実行も遅くなる点と，一回実行すると，復号鍵は全て入手できるという問題が残るが，計算量負荷を確実に設定できる点が特徴である．

2. 途中結果を復号鍵とする方式

難読化では逆コンパイラのようなもので戻らない，理論的に検証されて意味的に難読化した上で，計算量を評価するのが理想である．ここでは，まず意味的な部分は保留したまま，計算量だけ評価することと，計算量を増加させることによる難読化を行うことを考える．難読化されたものを解読するのに一定の計算量がかかることを規定

[†] 芝浦工業大学 工学部　情報工学科
Dept. of Information Science & Engineering, Faculty of Engineering, Shibaura Institute of Technology
[††] 芝浦工業大学 大学院 工学研究科　電気電子情報工学専攻
Division of Elec. Eng. and Computer Science of Graduate School of Engineering, Shibaura Institute of Technology

Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

することは重要である．そのため，あるプログラムを実行する時に一定以上の手続きを要することが証明できれば，難読化の要素技術の一つとして，利用可能となる．計算途中の結果から復号鍵を決め，次の処理に必要な数値（必須数値）に対して対応する暗号鍵で暗号化を行うことにより，処理の流れを規制し，処理の並列化や処理の省略を行うことができなくなる．従って，上記途中結果を求めて復号鍵を求めた後，復号処理を行って先に進むことができ，処理量を確実に設定できる様になる．

図 1 に途中結果を復号鍵として，必須数字の暗号化を行う方式に変形する構成を示す. ある計算がなされ, 途中結果 y が得られたとする. 次に以下の計算に必要な定数, a=38 があり，y と a 等を使って次の B の計算結果を得る．これに対し，まず，途中結果 y=12 に関連する素数 13 を選定する．12 と 13 の差は Bias として，変形後のプログラムに書き加えておく．素数 13 を基に例えば p,=37,q=23 を選び，N=p*q=851 を求める. また, p-1 と q-1 の最小公倍数 n=396 から, 暗号鍵 Y=e=61 を得る. メッセージ a=38 を e=61 乗して，暗号結果 $E(a)=38^{61}$ (mod 851)=815 となる．変形後の動作では，途中結果 y=12 に Bias 値 1 を加え復号鍵 d=13 を得る．暗号化されたメッセージ 815 を 13 乗して a=38 を得る．

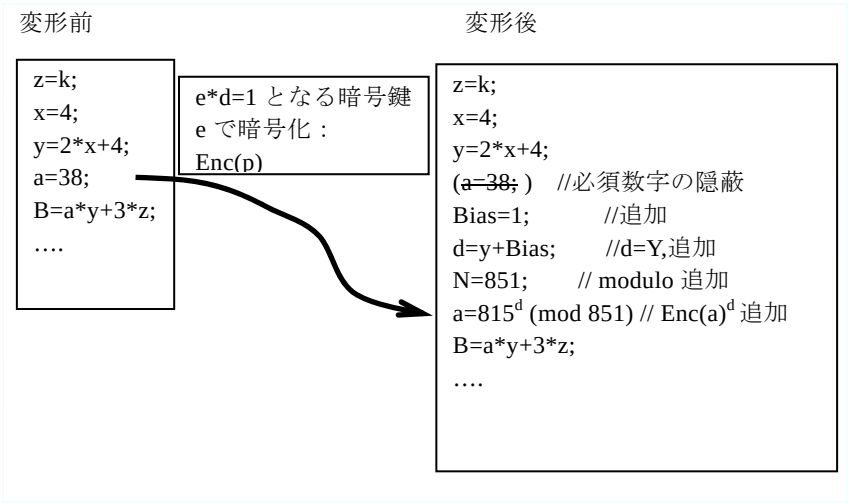

図 1　途中結果を使い，必須数字を暗号化する変形
Figure 1　The modification into encrypting essential number using an intermediate result.

以上，途中結果から復号鍵を作り，それに対応する暗号鍵を作って，必須数字を暗号化する手法（以下これを必須数字の暗号化手法：Encrypting Essential Number[EEN]と呼ぶ）により，検出プログラムを実行するときに必ず行わないとならない手続きを規定できる．具体的には，暗号結果 $E(a)^e$ を d 乗する計算する手間がそれにあたる．なお，このような冪乗演算は，モンゴメリー乗算法で一定の演算量の低減が図れる．

3. 難読化の計算量評価

難読化の評価は変異能力を用いるもの[5]や，JAVA コードに対する，メソッドとフィールドの呼び出し回数に基づく評価尺度を用いる手法[6]がある．また，具体的攻撃法であるスライシングに対応した評価法に基づく設計などが試みられている．これらは，code obfuscation に該当する手法と考えられるが，更に DES 暗号化処理プログラムを対応表に置き換えセキュリティ性を高めた方式[3]もあり，これらは意味的な難読化がなされていると考えられる．計算量は全ての手法に関わる項目である．難読化はその適用前後で，性質が不変であるのが理想で，プログラムの量や動作結果と処理速度が変わらないのが本来の難読化である．しかし，これを必須とすると，拘束条件が厳しくなり，不可能になる場合も生じる[1]．そこで難読化前後で性質は有限範囲内まで変更可能という拡張を行えば，攻撃者の対応も多様化するという負荷が増え，実用域も拡張されると考えられる．

計算量の評価には，多項式時間内かどうかを検討することがなされているが，次数に対する相対的な基準になり，絶対量を定めていない．実用的には，ある時点で計算量を絶対量で規定し，それに対する計算機能力をその時点で対照すれば，解読に要する計算時間を見積もることができる．ここでは，電子透かしの検出器の難読化が，高速で常時認証する必要がないものと見なし，むしろ計算量が多い方が望ましいものとして，確実に計算量を増加させることを第一目標とする．

結果を得るまでに必ず手間がかかるという計算負荷も難読化の拡張と見なした場合，その計算量の規定は多項式時間等のパラメータがある相対的な物でなく，そのプログラムやシステムに固有の絶対時間で示されていることが，実用上は望ましい．難読化の計算量評価基準として，

「処理結果を得るために必須の，加算・乗算・制御などの処理量」　(1)

を決め，以下の実験で，その評価量について検討していく．

図 1 に示した方式は，途中結果を復号鍵とする暗号化を以下の計算に必須の定数値に施す物である．これは，スライシングの様な並列分解解析が不可能な構成になっている．必須数字が暗号化されているので，復号鍵(d)を求め，暗号化されたデータを d 乗する計算量が主たる計算量となる．この計算量は，d やメッセージ a の語長に依存するため，実際に使用される途中結果は長い語調であることが望ましい．Bias 値は途

中結果 y が素数でないとき，使用したい素数との差であり，これは，検出プログラム中に公開しておく．この例では，途中結果より大きい最初の素数が採用されたが，最初の素数である必要はなく，単に差だけが示されるので，この値からは，復号鍵 d を推定できないものとする．

表 1 に別の事例をあげる．この例では，途中結果がある実数値になった場合で，その有効数字を整数化することにより上記の枠組みに入れる事ができ，大きい素数を復

表 1　実数の途中結果を復号鍵とする場合
Table 1　Encrypting fractional essential number .

途中結果	指数乗数	指数乗数後	復号鍵（素数）	Bias	index
3.19584	5 桁	319584	319733	149	i
	5.1 桁	351542	355753	4211	ii
	5.5 桁	479376	479371	-5	iii
	6 桁	3195840	3195869	29	iv

号鍵に設定する．小数点以下 5 桁の実数が途中結果の場合，十進で 5 桁のシフトアップを行い，6 桁の整数を得ている．この整数を起点に素数を探索し，素数 319733 を選択したとすると，Bias 値は 149 となる．指数乗数は整数でない物も可能で，例えば 5.1 桁と設定すれば，5 桁シフトアップしたものを 1.1 倍し，351542.4 となったものを，四捨五入して 351542 とすることが出来る．また，Bias 値は負の数も可能である．Bias 値は任意に設定可能で，bias 値や大小から復号鍵を推定できない．指数乗数がある場合のプログラム変形例を図 2 に示す．

以上の様な方式を想定し，小さい数の復号鍵として，素数を選択した後，対応する RSA 公開鍵を暗号鍵として求めた例を参考のため，表 2 に示す．暗号鍵 e と復号鍵 d は，

$$e \bullet d = 1 \pmod{n} \qquad (2)$$

なる関係があるので，復号鍵から設計しても，従来の暗号鍵から設計する設計法と同様な手法になる．途中結果と実際の素数値の関係は無くても良いので，復号鍵となる素数は自由に設定できる．設計パラメータである p,q は想定する復号鍵の長さに応じた値を設定すれば良い．また，Bias 値の正負の符号も任意に選択できる．計算量は復号鍵の長さとモジュロ値の大きさによるため，目標とする計算量に合わせ，これらの長さ（桁数）を調整すれば良い．図 2 に示すように暗号鍵は公開鍵ではあるが，設計者しか使用しないため，公開されない．そこで，復号鍵を試行により探索することもできない．素数表などから類推する解析を想定した場合は，素数表の最大長付近の素

```
z=k;
x=3;
y=x+0.19584;
(a=38; )   //必須数字の隠蔽
pw=5;              //指数乗数
Bias=149;          //追加
d=y*10^pw+Bias;    //追加
N=modulo;      // modulo 追加
a= Enc(a)^d (mod N) // Enc(a)^d 追加
B=a*y+3*z;
….
```

(a) index (i)の場合

```
z=k;
x=3;
y=x+0.19584;
(a=38; )            //必須数字の隠蔽
pw=5.1;             //指数乗数
Bias=4211;          //追加
d=y*10^5*1.1+Bias;  //追加
N=modulo;      // modulo 追加
a= Enc(a)^d (mod N) // Enc(a)^d 追加
B=a*y+3*z;
….
```

(b) index (ii)の場合

図 2　実数の途中結果を復号鍵とする場合
Figure2 Encrypting fractional essential number.

数を d,p,q として選ぶ様にしていけば，N=p*q は当分素因数分解できない大きい数になってくる．

表 2　復号鍵と暗号鍵の設計例
Table 2　Examples of designing decoding key and encoding key.

d	e	p	q	N	lcm
13	61	23	37	851	396
17	233	23	37	851	396
19	667	23	37	851	396
29	437	23	37	851	396
31	511	23	37	851	396
41	425	23	37	851	396
43	571	23	37	851	396
47	455	23	37	851	396

Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

計算量的に負荷を目標仕様に応じて増加させた難読化ソフトウェアの変換の流れを図３に示す．上記説明した難読化変形処理は，途中結果が得られる度に１ユニット生成できる．途中結果の出現回数の範囲内で，設定したい計算量に合わせこのユニットを所定回数分繰り返す．初めのプログラムに陽に表示されていた必須数字は暗号化して隠蔽する．

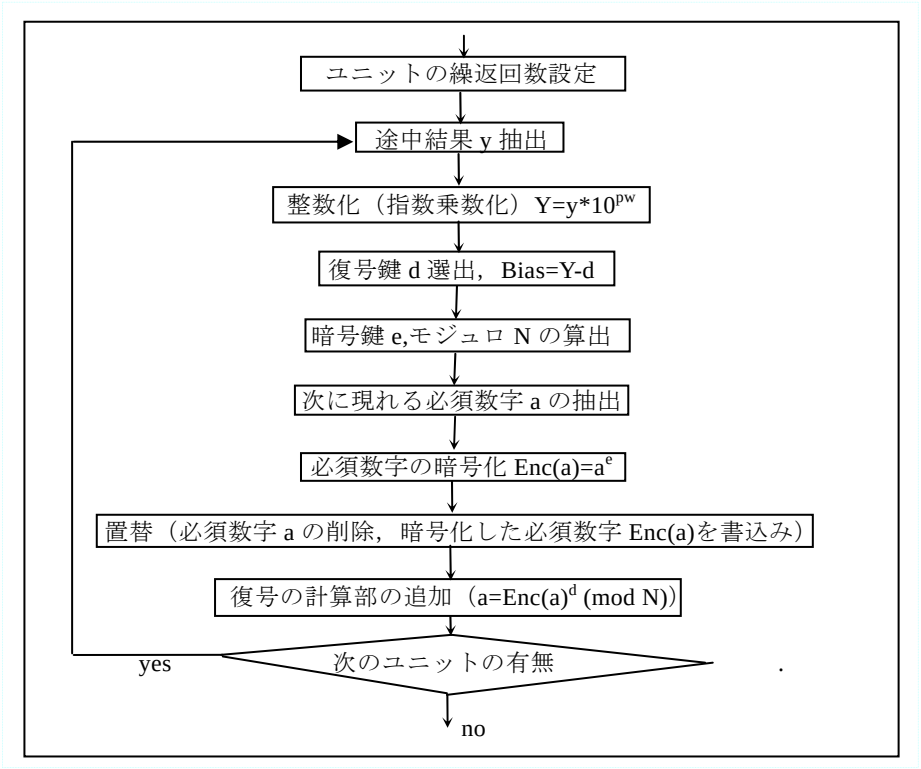

図　３　難読化ソフトウェアの変換の流れ
Figure 3 Flowchart of software obfuscation.

4.　電子透かしの検出器の公開

図４に検出器公開型の電子透かしシステムの構成図を示す．画像の電子透かしは種々の攻撃があり，認証性を主張するのが難しい．この方式では，検出器を公開することにより，著作権所有者が内密に検証を行うことに比べて，公開した分だけ，信憑性の向上がなされるはずであるとの考えで提案されている[7]．電子透かしの検出器はソフトウェアプログラムであるが，実行モジュールを難読化してある．実行をして動作解析する攻撃に対応して，計算量的に負荷を増加し，実行速度を遅くして，動作検証をしにくくする．また，意味的な難読化としては，計算方式の複雑化と ROM 関数による表に置き換える方式が検討されている[7]．これは，Chow らの DES 計算の行列によるアフィン変換で表現できるものをテーブル化するもの[3]に比べ，非線形の関数も含まれることと，用途において，使用する入力値と使用しない入力値があることを利用して，使用しない入力値の関数出力を偽の値にしたり，離散的な計算を行っているところでは，結果を量子化することになるため，微小な誤差が含まれても最終的には正しい値になることなどの工夫を盛り込むことができる物である．この意味的な難読化を行う方式についての計算量や意味が難しくなっていることの詳細は別途報告をすることとし，本報告では，計算量を増加する点に絞った形で行う．

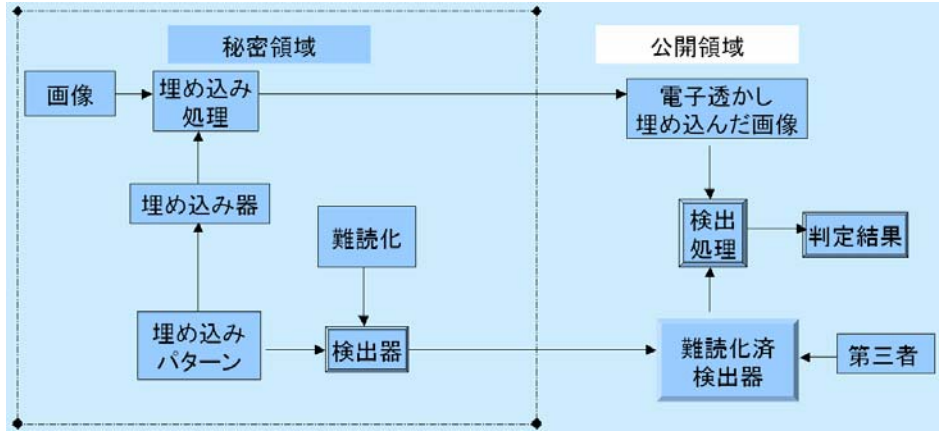

図　４　検出器公開型の電子透かしの構成
Figure 4 Watermarking system with disclosing a detector.

Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

5. 実験と考察

計算量を規定する難読化における計算回数の算出と，実行時間例から見積もりを行い，目標とする動作時間に合わせた復号鍵の長さの設計データを作成していく．図５は，設計する復号鍵の長さに対する暗号鍵を探索した場合の設計時間である．

素数の長さとしては初歩的なものであり，素数表で検索すれば，計算時間は不要だが，計算で求めるような長さの場合の見積もりデータとするため，計算により素数探索を行っている．なお，暗号鍵や p,q の値は，復号鍵に準じた類似の長さのものを用いることとしたが，小さい数から探索する設計手順にしたため，必ずしも統一されていない．

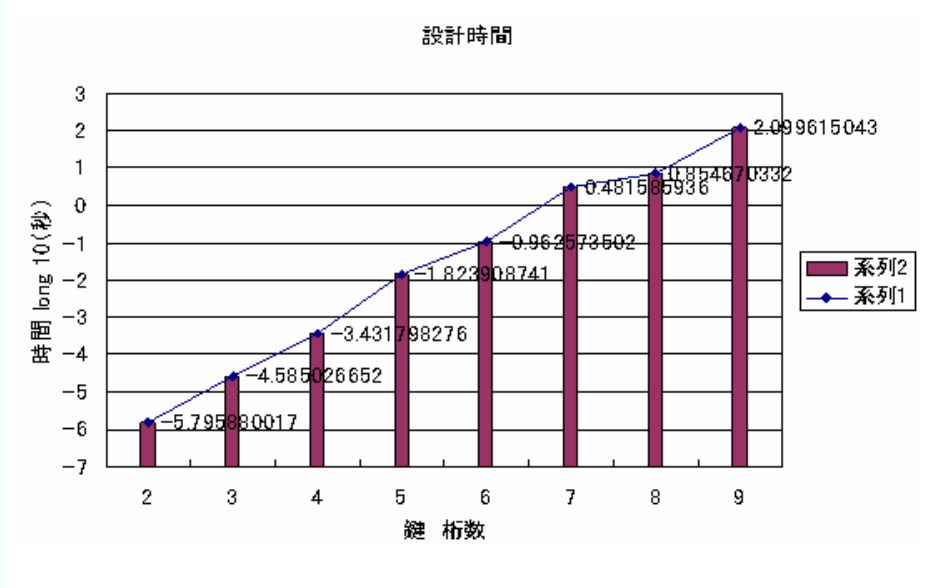

図 ５　復号鍵の桁数と設計に要する時間の関係
Figure 5 Time for design vs. digit number of decoding key.

図６に復号鍵の長さに対する復号時間を示す．復号時間は，他にメッセージ長も影響するが，メッセージ長は復号鍵と同じ長さを用いた．メッセージは必須数字であるため，その長さは，難読化前の原プログラム次第である．そこで，途中結果 y に対し，指数乗と Bias 値により，希望する大きさの素数に調整することができたが，それと同様な補正を加えることが可能である．即ち，必須数字を指数乗で，整数化し，更に Bias 値で小さい補正を行えばメッセージ長を希望の長さに調整できる．これら調整用の指数乗の数と補正項を変更後の難読化プログラムに記述し，復号後に逆補正を行えば良い．図５にやや段差があるのは，復号鍵，暗号鍵の探索で，見つかったものを使用し，桁の中間位置又は最大位置などと正確に求めてはいないためと考えられる．

復号時間に関しては，十進８桁で約１秒程度である．使用した計算機仕様は，Pentium4，3.2GHz．C 言語での乗算演算を行った場合で，各乗算ごとにモジュロをとり，又モンゴメリー乗算等の高速化処理は行っていない．縦軸は時間の対数であり，ほぼ直線と見做せば，更に長い桁数の場合の結果をグラフを延長して推定することができる．

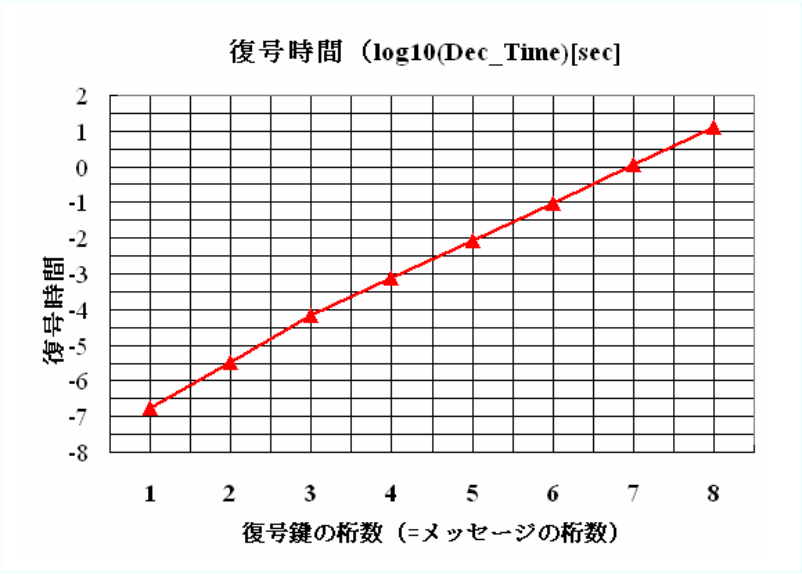

図 ６　復号鍵の桁数と復号に要する時間の関係
Figure 6 Time for decoding vs. digit number of decoding key.

Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

図６の計算において，横軸の桁数が 1-3 については，具体的な復号鍵 d，暗号鍵 e，p,q,lcm,N,メッセージを表３のように全て求めて計算した．桁数が４以上のものについては，p,q,復号鍵 d を各桁数内でほぼ最大の素数を使い，暗号化されたメッセージの桁数を各桁内で，最大桁が１である適当な数字を用いている．演算時間は主に d の値で示される回数の乗算とその都度行うモジュロ除算の合計回数であるが，2 の冪乗のみ先に求めて組み合わせる高速計算手法がある．最下段にその回数を参考にあげてある．各２の冪乗のあと行うモジュロ除算と，各２の冪乗結果を組み合わせて行く乗算とモジュロ除算で最大 4 倍になりうる．

計算量は桁数に依存した指数乗とモジュロの除算であるため，規則的で，実測値も規則的な線上にある．現在の結果は計算式の冪乗剰余算をそのまま実行した場合を表示してあるが，高速化の手法を決めれば，その差分も規定可能になる．

電子透かしの検出器を公開することにより，非公開で機密処理をする場合に比べ，信憑性の向上に寄与することができる．検出器を公開するに当たり，難読化を行う手法として，計算量を増加させることのみに着目し，その量の見積もりを行った．計算量の拠り所として，暗号化データの復号鍵をプログラム中の途中結果によって与えることにより，プログラムを分割して並列解析することができない構成になっている．

本構成では，冪乗と剰余算の計算量として，正確な見積量を与えることができる．実験に使用した計算機では，高速計算を使用しない場合は，８桁で 10 秒程度であり，また１桁約 10 倍秒であることから，10-11 桁で１日，12 桁で１カ月，13-14 桁で約１年になる．

本構成で，必須数字に関し，例えば，フーリエ変換領域での埋込みをするような方式が多く使用されているが，その係数などを使用可能である．しかし，定義どおりのフーリエ変換では，変換の係数は既知である．その対策として，拡張された変換，モディファイドフーリエ変換（MFT）[7][8]を使用するようにすれば良い．これにより，同一の埋込みソフトを使用しても，MFT の部分は，埋込みを行う画像ごとに変更できるため，異なる埋込みがなされ，検出器も画像ごとに異なるものが生成される．

必須数字を暗号化した部分以外は全て共通の埋込みソフトとなるが，この部分についても各画像ごとに改変するというダイナミックな構成をとることもできる．検出器の秘密情報は，上記のような変換面での埋込みでは，その変換係数の値と埋込み位置等が本質的なものである．これに加えて，一般的な従来からの手法である code obfuscation を組み込むこともできる．その例は，if 文の様な分岐文を追加していく手法[4][7][8]，加算や乗算などの基本演算を複雑化変換を施して同値の結果を出力する式にする手法[4][7][8]等がある．

表３ 検出器内の１個の EEN ユニットの復号時間の計算用データ(太枠内は実際に存在するデータで，それ以外は桁数に応じた値)
Table 3 Calculating data for obtaining decoding time of an EEN unit in a detector. (The values in a bald frame really exist. The others are provisional with lengths of digit numbers.).

桁数	1	2	3	4	5	6	7	8
p	7	97	997	9973	99991	999983	9999991	99999989
q	5	89	991	9967	99989	999979	9999973	99999989
N	35	8633	988027	99400891	9998000099	9998000099	99999640000243	9999996000000320
n	24	8448	986040	16563492				
d	5	79	907	9931				
e	29	10159	97843					
a	4	78	900					
E(a)	9	6157	306326	12345678	1234567890	123456789012	12345678901234	1234567890123456
$log_2(E(a))$*4	2.81	6.51	9.95	96	124	148	176	204

Vol.2010-DPS-142 No.26
Vol.2010-CSEC-48 No.26
2010/3/5

また，作成した暗号鍵自体は公開しておらず，暗号化結果 E(a)とモジュロ値を与えただけなので，試行により復号鍵 d を探索することができない．ユニットの多重配置は目標とする演算量に合わせ，復号鍵の長さと個数を調整するときに自由に設定可能である．

検出器公開型の電子透かしの検出器の実行は通常なされず，従って実行速度が遅いことも許容される．また，検出器を作った著作者は，必須数字を知っているため，復号処理を飛び越し実行することができる．これを今後有効に切り換えることができれば，実行速度の柔軟な調整も可能となる．

6. まとめ

電子透かしの検出器を公開する方式における検出器の難読化ついて検討し，演算量のみを確実に増加させる手法について検討した．計算量の見積もりを行い，現状の計算機(PC)で，冪乗と剰余算の計算量を実測した．計算の高速化にはいくつかの既知の手法があるが，いずれも方式が確定しているので，一定の対応で，見積もりを補正できる．

今後は，計算量を固定したり，低減した状態で，プログラムを非線形にランダム化する手法と組み合わせて，一方向性を有する形の変換を構成し，本文で述べた意味的な難読化を構築し、通常のソフトウェアの難読化へも適用可能な形にすることを目指す．

参考文献

1) Barak, B., Goldreich, O., Impagliazzo, R., Rudich, S., Sahai, A., Vadhan, S. and Yang, K.: On the (Im)possibility of Obfuscating Programs, pp. 1-18, CRYPTO (2001).

2) Barak, B. : http://www.cs.princeton.edu/~boaz/Papers/obf_informal.html

3) Chow, S., Eisen, P., Johnson, H., Van Oorschot, P.C.,: A White-Box DES Implementation for DRM Applications, Proceedings of ACM CCS-9 Workshop DRM pp.1-15, Nov., (2002)

4) 魏遠玉，大関和夫,: 検出器公開型の電子透かしの難読化方式比較と計算量, ITE 冬季大会 9-6,2009

5) Kiyomoto, S., and Tanaka, T.,: Evaluation of Mutational Capability and Real-Time Applicability of Obfuscation Techniques, EICE transactions on fundamentals of electronics, communications and computer sciences E89-A(1) pp.222-226 (2006).

6) Fukushima, T. Tabata, and K. Sakurai.: Proposal and Evaluation of Obfuscation Scheme for Java Source Codes by Partial Destruction of Encapsulation, Proc. of International Symposium on Information Science and Electrical Engineering (ISEE2003), pages 389–392, Fukuoka, Japan, November 14–15.

7) 大関和夫，叢力,: 計算量的難読化を仮定した，不特定第三者の認証に依存する電子透かし方式，情報処理学会，研究報告, CSEC-32, pp.61-66, 3月(2006).

8) Ohzeki, K., Cong, L.,: Consideration on Variable Embedding Framework for Image Watermark against Collusion Attacks, Wavilla Challenge (WaCha) 2005, Proceedings of the WAVILA Workshop on Watermarking Fundaentals, ECRYPT, D.WVL.2-1.0.pdf, pp.54-62., June 8-9, http://www.ecrypt.eu.org/ecrypt1/documents/D.WVL.2-1.0.pdf (2005).