社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

ランサムウェア検知ソフトCryptoDropの耐性評価

南　　翔†　西出　隆志††

† 筑波大学システム情報工学研究科　〒305–8573 茨城県つくば市天王台 1–1–1
†† 筑波大学システム情報系　〒305–8573 茨城県つくば市天王台 1–1–1
E-mail: †minami@cipher.risk.tsukuba.ac.jp, ††nishide@risk.tsukuba.ac.jp

あらまし　ランサムウェアはマルウェアの一種である．ランサムウェアに感染した場合，そのコンピュータ内のファイルは自動的に暗号化され，その後，金銭を支払うようそのコンピュータのユーザに要求が送られる．攻撃者が持つ秘密鍵で復号しない限り，暗号化されたファイルにユーザがアクセスすることは不可能である．ランサムウェアを検知するシステムに 2016 年にフロリダ大学の Nolen Scaife らが発表した CryptoDrop がある．これはユーザのデータを常時観察し，仮にユーザのデータが暗号化された場合，その原因となるアプリケーションを検知することによりランサムウェアを検知するシステムである．本研究ではこの CryptoDrop を構成するインジケータに着目して CryptoDrop の耐性を評価し，ランサムウェアに対してより安全なシステムの構築への貢献を目指す．
キーワード　CryptoDrop，ランサムウェア，エントロピー，ファイル形式，類似度

Sho MINAMI† and Takashi NISHIDE††

† Faculty of Systems and Information Engineering, University of Tsukuba　1–1–1 Tennoudai, Tsukuba, Ibaraki
†† Faculty of Engineering, Information, and Systems, University of Tsukuba　1–1–1 Tennoudai, Tsukuba, Ibaraki
E-mail: †minami@cipher.risk.tsukuba.ac.jp, ††nishide@risk.tsukuba.ac.jp

Abstract　A ransomware is a kind of malware. If infected with ransomwares, files in a computer are automatically encrypted, and then a request send to the user of the computer to pay money. There is a system that is able to detect ramsomwares. The system is named CryptoDrop which was developed by Nolen Scaife et al. of the University of Florida in 2016. This system always observes a user's data and detects ramsomwares if the data is encrypted. In this research, we focused on indicaters of CryptoDrop and evaluated resistence of CryptoDrop.
Key words　CryptoDrop, ransomware, entropy, file type, similarity

1. 序　　論

1.1 研 究 背 景

ランサムウェアを含むマルウェアを検知する既存の手法として，アンチウイルスソフトに用いられているシグネチャマッチングがある．シグネチャマッチングとは，プログラムのコードを調べ，マルウェアに特徴的なコードパターンを格納したデータベース（シグネチャという）と照合し，仮にシグネチャと合致するパターンが見つかった場合はそのプログラムをマルウェアと見なす方法である．しかしシグネチャマッチングの欠点として，検出できるのは既知のマルウェアに限られることが挙げられる．この欠点を解決する為にアンチウイルスソフトはシグネチャを定期的に更新するサービスを実施しているが，それでも全てのマルウェアの情報を網羅することは困難である．さらに，シグネチャは世に一般的に出回っているマルウェアを元に作成されている為，標的を一つに絞ってその標的を攻撃することのみに特化させたマルウェアを用いる標的型攻撃を検出することができない（偽陰性）[1].

標的型攻撃のようなシグネチャマッチングでは対処できない未知のマルウェアを用いた攻撃に対処する為に，アノマリ検知という手法も存在している．これはシグネチャマッチングとは逆に，正常なファイルの行動パターン等を元に正常なパターンを記憶し，記憶している正常パターンから逸脱する行動が発生した場合それを検出するという手法である．これならば標的型攻撃にも対処可能であるが，正常なファイルを誤検知することもあり（偽陽性），一長一短が存在するのが現状である．

ランサムウェアを検出しようとするこれらの既存の方法は，ランサムウェアのコードの中身やその動作に着目することによ

りランサムウェアを検知しようとするものである. 詳細は次節で述べるが, 以上の既存の方式とは反対に, ユーザのデータの中身に着目することによってランサムウェアを検知しようと Scaife らによって考案されたシステムが CryptoDrop である. これはランサムウェアに特化したシステムである為ランサムウェア以外のマルウェアを検出することは出来ないが, その分, ランサムウェアに対しては非常に低い偽陰性率を示している [2].

本研究は CryptoDrop の模擬ランサムウェアに対する耐性を評価することにより, ランサムウェアに対してより強固なセキュリティの実現を目指そうとするものである.

1.2 先行研究

ランサムウェアを早期検知する為のシステムとして CryptoDrop が考案された [2]. これは, ユーザのデータを暗号化するというランサムウェアの性質を逆手に取り, ユーザのデータを常時観察し, ユーザのデータに不審な変更が加えられた場合に, その原因となるアプリケーションをランサムウェアとして検出しようとするシステムである. シグネチャマッチングとアノマリ検知はマルウェアを観察する手法を用いていたのに対し, CryptoDrop はユーザのデータを観察する手法を用いているという点で, CryptoDrop は従来の検知手法とは大きく異なっている.

CryptoDrop はランサムウェアを検知する為に 3 つの主要インジケータと, それらの弱点を埋める為に用意された 2 つの補助インジケータを用いている. それらを以下に示す.

File Type Changes [2] : 主要インジケータの一つである. ユーザのデータのファイル形式を観察し, ファイル形式の変更を検知する. ファイル形式はそのファイルが存在する限り変化することは一般的に無いが, ファイル全体が暗号化された場合ファイル形式も変化する可能性が高い. その為, ファイル形式が変化する操作を行っているアプリケーションを発見した場合それをランサムウェアの可能性があると見なす.

Similarity Measurement [2] : 主要インジケータの一つである. ファイルを観察し, ファイル内部の変更を検知する. その際に similarity-perserving ハッシュ関数と呼ばれる, ファイル内の情報を幾分か保存する特殊なハッシュ関数を用いる [3] [4]. ファイル変更前後のハッシュ値を比較することによりファイル変更前後の類似度（0 − 100）を計算し, 類似度が 0 に近い値であれば, そのファイルはランサムウェアにより暗号化された可能性があると見なす.

Shannon Entropy [2] : 主要インジケータの一つである, ファイルを構成するバイト列のエントロピーを計算することにより, ファイルの暗号化を検知しようとするものである. 他のプログラムがファイルを読み込んでそのファイルに対し書き込み操作を行った場合, 読み込み時のエントロピーと書き込み時のエントロピーを比較し, ある閾値以上エントロピーが増加していた場合, そのファイルは暗号化された可能性があると見なす.

Deletion [2] : 補助インジケータの一つである. ランサムウェアの中にはファイルを暗号化する際, ファイルの内容を読み込んだ後新規ファイルを作成し, その新規ファイルに元のファイルの暗号化データを書き込んだ後, 元のファイルを削除する形式のものが存在する. ファイルの削除操作自体は正規のアプリケーションでも行っていることであるが, ランサムウェアによって削除操作が行われる場合, その操作の回数は正規のものに比べ膨大なものになる. このインジケータは, そのような大量削除操作を検知することにより, その操作を行っているアプリケーションをランサムウェアの可能性があると見なす.

File type funneling [2] : 補助インジケータの一つである. 異なるファイル形式の複数のファイルを入力とし, その結果出力するファイルのファイル形式の数は, 入力時より減少するような操作を行うアプリケーションを検知する. この操作自体は Pages や Microsoft Office Word 等といった正規のアプリケーションも行っているものである. しかしランサムウェアの場合, ファイルを暗号化するにつれて出力されるファイルの形式が次第に少なくなっていく傾向がある. このインジケータは出力されるファイルの形式の推移を監視することにより, ランサムウェアを検知しようとするものである.

以上のインジケータの特性を利用し実験を行うことによって, CryptoDrop の耐性を評価する.

1.3 本論文の構成

2 章では実験を行うにあたっての実験環境と前提知識及び実験方法を示し, その後, 実験に用いるアルゴリズムについて述べる. 3 章では実験結果を用いて CryptoDrop の耐性の検証を行い, 章の終わりに本研究に関する今後の課題を述べる. 4 章で本研究の結論を述べる.

2. 準　　備

CryptoDrop の対応 OS が 64bit の Windows7, 8, 10 のみである為, 本実験は 64bit の Windows 上で行う. CryptoDrop の動作環境が Windows である理由は, ランサムウェアの攻撃を頻繁に受ける OS が Windows であるからである [2]. 実験には, オープンソースの AES アルゴリズム [5] を元に作成した, ランサムウェアを模擬したプログラムを用いる. このプログラムはファイルの暗号化機能だけを有している. このプログラムを用いて 100 個の平文ファイルに対して暗号化を行い, この暗号化操作に対する CryptoDrop の振る舞いを観察することにより CryptoDrop の耐性検証を行う.

2.1 実験環境

まず VirtualBox を用いて, OS X El Capitan バージョン 10.11.6 上に Windows8.1 Pro 64bit（以降 Windows とする）をゲスト OS としてインストールした. 尚, この Windows の実装メモリ (RAM) は 2048MB, プロセッサは Intel(R) Core(TM) i5-5287U CPU @ 2.90GHz 2.90GHz, プロセッサ数は 2 である. 次に, CryptoDrop バージョン 1.0.257.1559 を Windows 上にインストールした. その後, 100 個の平文ファイルで構成されるファイルセットを用意した. このファイルセットは 25 のファイル形式で構成され, 各々のファイル形式に対して相異なる平文ファイルを 4 個ずつ用意した. ファイルセットの内訳は表 1 となる. 尚, 表 1 では各々のファイル形式を対応する拡張子で表現している. このファイルセットを Windows 上の Pictures フォルダ（C:¥Users¥［ユーザアカウント名］¥Pictures）上

表 1　ファイル形式とサイズ

形式＼サイズ	0KB-10KB	10KB-100KB	100KB-1MB	1MB-16MB	16MB-128MB
.avi				4	
.eps				4	
.gif		1	1	2	
.jnt	1	3			
.jpg		3	1		
.key			4		
.mid	1	3			
.mp3			1	3	
.mp4				1	3
.numbers			4		
.docx	1		3		
.pages			3	1	
.pdf		1	2		1
.png		3	1		
.pptx		1	3		
.svg	2	1	1		
.tif			2	2	
.wav			3	1	
.xlsx		2	2		
.ico			4		
.exe		4			
.txt	3	1			
.bmp		2	2		
.rtf	2	2			
.zip		1	1	2	
合計	10	28	38	20	4

に設置し，このフォルダ内のファイルセットに対して暗号化操作を行うことで,CryptoDrop の振る舞いを観察する．

2.2　CryptoDrop の振る舞いに関する前提知識

ランサムウェアの疑いがあるプロセスを CryptoDrop が検知した場合，図 1 のような画面が表示され，そのプロセスは一時停止する．仮にユーザが"Ignore"ボタンを押し，そしてこのボタン操作に対する確認ウィンドウに対し肯定的な応答を行った場合は，一時停止したプロセスは再び動作を始める．この時，このプロセスは CryptoDrop のホワイトリストに登録される．対して"Lockdown"ボタンを押すか，或いはウィンドウが表示されてからユーザが何もしないまま一定時間が経過した場合は，図 2 に示されるウィンドウが表示される．このウィンドウが表示されている間はロックダウンモードになり，CryptoDrop の監視下に置かれているフォルダ内のファイルが read-only 状態となって，そのフォルダ内のファイルをそれ以上暗号化することが出来なくなる．ウィンドウの"Exit Lockdown"ボタンを押すことで read-only 状態が解除される．尚，"Lockdown"ボタンを押したことによりロックダウンモードに移行した場合，検知されたプログラムは CryptoDrop のブラックリストに追加される為，ユーザが手動でブラックリストから外さない限りそのプログラムは CryptoDrop 監視下のフォルダ内にアクセスすることは出来ないが，一方で，検知された後一定時間経過したことによってロックダウンモードに移行した場合は，検知されたプログラムはブラックリストに登録されないことになっている．

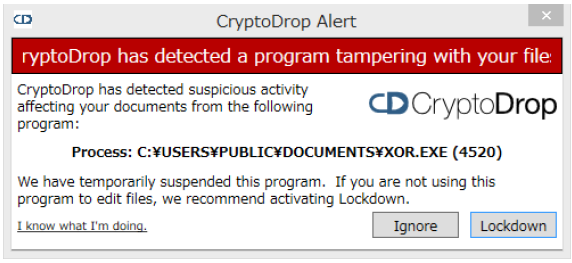

図 1　検知例（C:¥Users¥Public¥Documents¥XOR.exe を疑わしいプログラムとして検知）

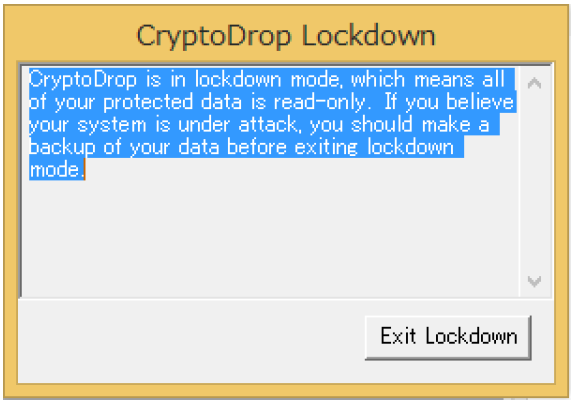

図 2　ロックダウンモード

2.3　実験方法

まず，鍵長が 128bit の AES アルゴリズム [6] を用いた暗号化プログラムによりファイルセットを 20 回暗号化し，CryptoDrop が検知するまでに完全に暗号化されたファイルの個数を各々の試行毎に調査する．この 20 回の試行は，同一のソースファイルをコンパイルすることによって得られた相異なる名前の 20 個の実行ファイルを一個ずつ実行することによって行われる．また，ファイルセットの各平文ファイル名は Flexible Renamer v8.4 のスクリプトによって全てランダム化されている．ファイ

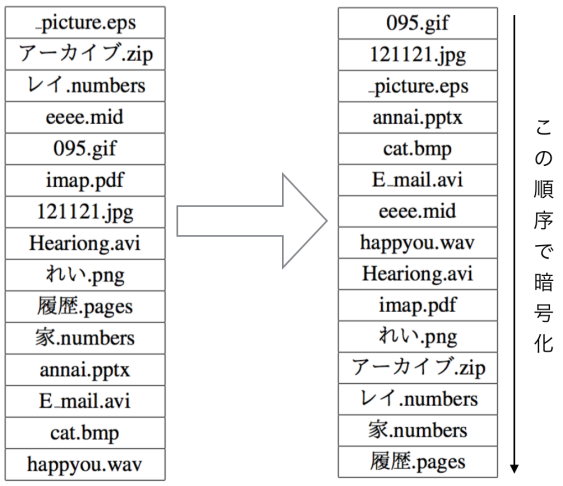

図 3　複数のファイルの暗号化の例（暗号化の順序は Shift_JIS コードの昇順に従う）

ルセットは事前にバックアップを取ってあり，AES アルゴリズムによって暗号化されたファイルセットは各試行の終了毎に削除し，そしてバックアップを Pictures フォルダに手動でコピーすることによって復元される．バックアップの平文ファイル名は試行終了毎に逐次ランダム化される為，対応する平文ファイルの名前は試行毎に全て異なる．本実験で使用する暗号化プログラムは全て，コマンドライン引数に［暗号化対象フォルダのフルパス］¥＊を渡すことで任意のフォルダ内の全ファイルを暗号化することが可能である．例えば Windows の Pictures フォルダ内の全ファイルを暗号化する場合は，コマンドライン引数として C:¥Users¥［ユーザアカウント名］¥Pictures＊を入力する． 本実験のように暗号化対象ファイルが複数存在する場合，暗号化は，Shift_JIS コードの上でファイル名に関して昇順になされる．この時，アルファベットの大文字と小文字は区別しない．暗号化対象ファイルが複数存在する場合の暗号化の順序の例を図 3 に示す．試行毎にファイルセットの各平文ファイル名はランダムに変更される為，暗号化プログラムのファイル暗号化の順序は試行毎に全く異なることになる．

次に，CryptoDrop のインジケータによる検知を回避すると考えられる新たな機能を前述の AES ベースのプログラムに加えることにより，新たな暗号化アルゴリズムを作成する．この際，新たな暗号化アルゴリズムに対応する復号アルゴリズムも同時に作成し，CryptoDrop の監視下に置かれていないフォルダ内に設置された表 1 と同一のファイルセットに対して暗号化・復号操作を行うことにより，新たな暗号化アルゴリズムによって暗号化されたファイルは復号可能であることを確認する．尚このファイルセットの各平文ファイル名は Flexible Renamer v8.4 によってランダム化されている．新たな暗号化アルゴリズムとそれに対応する復号アルゴリズムの概要は次節で示す．新たな暗号化アルゴリズムを作成した後はこのアルゴリズムを用いて Pictures フォルダ内のファイルセットに対し 20 回暗号化を行い，CryptoDrop が検知するまでに完全に暗号化されたファイルの個数を試行毎に調査する．暗号化されたファイルセットは各試行の終了毎に全て削除し，バックアップを Pictures フォルダにコピーすることによってファイルセットを復元する．尚，バックアップの各平文ファイル名は Flexible Renamer v8.4 のスクリプトによって試行終了毎に全てランダム化される為，暗号化の順序は試行毎に全く異なる．試行を 20 回行った後，単純な AES アルゴリズムと新たなアルゴリズム（模擬ランサムウェア）との実験結果を比較することにより，CryptoDrop の耐性評価を行う．

尚，CryptoDrop に検知された場合は，AES アルゴリズムの場合と新たに作成したアルゴリズムの場合共に，まず完全に暗号化されたファイル数を記録し，次にロックダウンモードに移行するまで待機した後，コマンドプロンプト上から手動で暗号化プロセスを強制終了し，最後に Pictures フォルダ内のファイルを全て削除するものとする．この操作を行う理由は，短時間に高頻度でロックダウンモードに移行した場合，暗号化されたファイルの一部が Pictures フォルダ内において削除も上書きも不可能な状態になることがあり，この状態になった場合は実験の継続が困難になるため，この状態になるのを回避する為である．

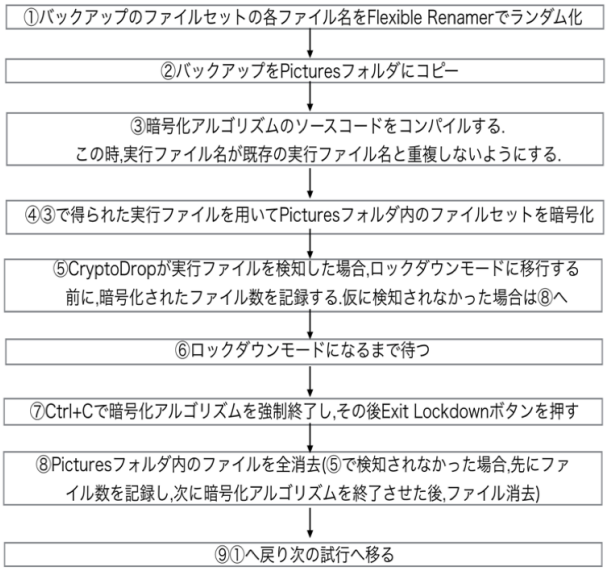

図 4　各試行の手順

表 2　排他的論理和に用いられる XOR 集合（2 進数表記）

00000000	00000001	00000010	00000100	00001000	00010000	00100000	01000000
10000000	00000011	00000101	00000110	00001001	00001010	00001100	00010001
00010010	00010100	00011000	00100001	00100010	00100100	00101000	00110000
01000001	01000010	01000100	01001000	01010000	01100000	10000001	10000010
10000100	10001000	10010000	10100000	11000000			

この節で述べた各暗号化プログラムの実験の各試行の手順を図 4 に示す．

2.4　耐性評価用模擬ランサムウェア

2.4.1　使用する暗号化アルゴリズム

模擬ランサムウェアの中で使用するために作成したアルゴリズムを以後 new_algorithm と表現することとする．new_algorithm による 1 ファイルの暗号化の概略を，図 5 と後述の Algorithm1 に示す．

このアルゴリズムはまず平文ファイルのバイナリデータを読み込む．その後，読み込んだバイナリデータを複製して全く同じバイナリデータを用意した後，それぞれを異なる 2 つのメモリ領域に格納し，一方のメモリ領域に格納されたバイナリデータの先頭 160B 以外に対して AES による暗号化を行う．先頭 160B を暗号化しない理由は，ファイルの暗号化前後でファイル形式が変化しないようにする為である．ファイル形式を記述する数 B～数十 B 程度のフォーマット識別子は通常そのファイルのバイナリデータの先頭部分に記述されている為，バイナリデータの先頭部分を敢えて暗号化しないことにより File Type Changes インジケータを回避することが可能と考えられるからである．これ以後，バイナリデータの先頭 160B に対し暗号化を行わない場合は，全てこのような理由によるものとする．

次に，AES による暗号化によって得られた暗号データに対して，16B 間隔で，0 のみから成る 32B のデータブロックを挿入する．この挿入操作によって得られた暗号データを A と

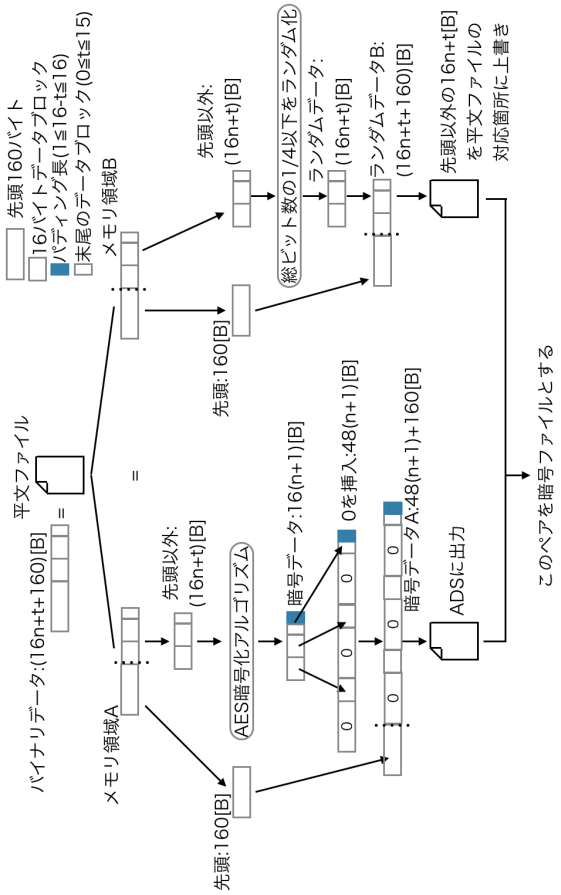

図 5　new_algorithm の概略（n=2 の場合）

する．この挿入操作を行う理由は，低エントロピーのデータブロックを暗号データに一定間隔で挿入することによって暗号化後のデータ全体のエントロピーの増加を小さく抑えることにより，Shannon Entropy インジケータを回避することが可能と考えられるからである．

その後，AES による暗号化を施さなかったもう一方のメモリ領域に格納されているバイナリデータに対して操作を行う．まずこのバイナリデータの先頭 160B 以外の各バイト p_i に対して，表 2 で表される要素数 37 の集合（以後，XOR 集合とする）から一様ランダムに一つの要素 x_i を選び，そしてバイナリデータの各バイト p_i とそれに対応する x_i とを排他的論理和することによって A とは別のランダムバイナリデータを生成する．XOR 集合を用いて排他的論理和操作を行う理由は，ランダム化の際に反転させるバイナリデータの bit 数をバイナリデータ全体の bit 数の $\frac{1}{4}$ 以下に抑えることにより，Similarity Measurement インジケータを回避することが可能と考えられるからである．以後，この排他的論理和操作によって生成されたランダムデータを B とする．B を生成する際に使用された鍵である各 $x_i \in XOR$ は Algorithm2 で示される方法で生成されており，AES アルゴリズムを用いることによって各 x_i は暗号論的擬似乱数の条件を満たしている．各 x_i は B の各バイトを生成すると同時に破棄される為，B から平文を復元することは攻撃者にも実質的に不可能である．なぜなら B は元の平文バイナリデータの中のランダムに選んだ 25% 程度の部分を乱数に置き換えたデータとなっているからである．詳細は後述するが，B は真の暗号文である A をユーザから秘匿し，かつ，平文バイナリデータを CryptoDrop に検知されない範囲で破壊する

Algorithm 1 New Algorithm for Encrypting a file

Require: $file$

Ensure: $encrypted\_file\ and\ its\ ADS$

1: $file\_size \leftarrow get\_file\_size(file)$
2: $data\_block \leftarrow 16[Bytes]$
3: $padding\_length$
$\leftarrow data\_block - (file\_size \bmod data\_block)$
4: $encrypted\_file\_size$
$\leftarrow 3 * (file\_size + padding\_length) - 20 * data\_block$
5: $*data \leftarrow get\_file\_data(file)$
6: $define\ *p\ and\ copy\ data\ to\ p$
7: $define\ *encrypted\_data\ and\ copy\ the\ first\ 160\ bytes$
$of\ data\ to\ that\ of\ encrypted\_data$
8: $make\ ADS\ of\ file$
9: $AES\ encrypts\ data\ except\ the\ first\ 160\ bytes$
10: **for** $i = 0; i < \{(file\_size + padding\_length) - 10 * data\_block\}; i \leftarrow i + data\_block$ **do**
11: $cat\ 32\ Bytes\ data\ block\ that\ consists\ of\ only\ 0$
$to\ encrypted\_data$
12: $memcpy(encrypted\_data + 3 * i + 12 * data\_block,$
$data + i + 10 * data\_block, data\_block)$
13: **end for**
14: **for** $i = 160; i < file\_size; i \leftarrow i + 1$ **do**
15: $define\ x_i\ that\ is\ selected\ uniformly\ at\ random$
$from\ the\ set\ XOR$
16: $*(p + i) \oplus x_i$
17: **end for**
18: **if** $1000 \leq encrypted\_file\_size$ **then**
19: $write\_block \leftarrow \lfloor encrypted\_file\_size/1000 \rfloor [Byte]$
20: **else**
21: $write\_block \leftarrow 1[Byte]$
22: **end if**
23: **for** $i = 10 * data\_block; i < encrypted\_file\_size;$
$i \leftarrow i + write\_block$ **do**
24: $fopen\ both\ (file, "rb+")\ and\ (ADS, "ab")$
$and\ then\ fseek(file, i, SEEK\_SET)$
25: **if** $i + write\_block < file\_size$ **then**
26: $fwrite(p + i, 1, write\_block, file)$
27: **else**
28: $fwrite(p + i, 1, file\_size, file)$
29: **end if**
30: **if** $i + write\_block < encrypted\_file\_size$ **then**
31: $fwrite(encrypted\_data + i, 1, write\_block, ADS)$
32: **else**
33: $fwrite(encrypted\_data + i, 1,$
$encrypted\_file\_size, ADS)$
34: **end if**
35: $fclose\ both\ (file, "rb+")\ and\ (ADS, "ab")$
36: **end for**
37: $output\ file\ as\ encrypted\_file\ and\ also\ output\ its\ ADS$
$as\ part\ of\ encrypted\_file$

為に生成される．

最後に，B と A をそれぞれ平文ファイル（暗号対象ファイル）本体とその代替データストリーム（Alternative Data Stream. 以後 ADS とする）[7] に出力する．この際ループ文を使用し，A，B 共にループ毎に $\lfloor A$ のバイト数/1000$\rfloor$B 単位でランダムデータの出力を行い，さらに B は平文ファイルバイナリデータの先頭 160B 以外の箇所に対してその箇所に対応するランダムデータで上書きを行う形式で出力する．また，ループ開始毎に暗号対象ファイルと ADS を共にオープンし，ループ終了毎に暗号対象ファイルと ADS を共にクローズする．ループにおけるこの一連の操作を行う理由は，ランダムデータの一回当たりの出力量を小さく抑えることにより，一回の出力操作当たりのファイル内のエントロピーの変化量及びファイルの類似度の変化量を小さく抑えることで Shannon Entropy インジケータ及び Similarity Measurement インジケータを回避することが可能であると考えられるからである．また，A と B をそれぞれ ADS と暗号対象ファイルに書き込み，A を ADS によって秘匿しその代わりに B をユーザに公開することによって，真の暗号文 A と真の暗号化アルゴリズムをユーザに対して秘匿する効果が期待される．以上の手法を取ることによって B が上書きされたファイルと A が書き込まれた ADS が，本アルゴリズムによって出力される暗号ファイルである．

尚 new_algorithm は上記の他にも，暗号化の際にファイルを削除する操作を行わないことにより Delete インジケータを回避することが可能であると考えられ，また，暗号化前後間におけるファイル形式の変更を避けることにより File Type Funneling インジケータを回避することが可能であると考えられる．すなわち，new_algorithm は CryptoDrop の全てのインジケータを回避することが可能であると期待される．

2.4.2 復号アルゴリズム

new_algorithm の復号の概略を Algorithm3 に示す． 復号は CryptodDrop の検知を回避する機能を搭載する必要性が無い為，復号アルゴリズムは暗号化アルゴリズムに比べ簡潔である．

まず暗号ファイルの ADS から，暗号データ A と A のサイズを取得する．次に A の先頭 160B 以外に対してループ文を用いることにより，0 のみで構成される 32B のデータブロックが一定間隔で挿入されている A から，AES によって暗号化されている平文ファイルバイナリデータのみを抽出する．そしてそのバイナリデータに対して AES の復号アルゴリズムを用いることにより，平文ファイルのバイナリデータを得る．その後，ファイルに上書きする形式で暗号ファイル本体に平文ファイルのバイナリデータを出力し，最後に ADS を削除することによってファイルの復号が完了する．

Algorithm 2 Algorithm of Random Number Generation

Require: $current\_time$
Ensure: $x_i \in XOR$
1: declare an int type array named $rand\_block$[4], the total size of $rand\_block$ is 16[$Byte$]
2: $srand((unsigned)time(NULL))$
3: **while** $TRUE$ **do**
4: **for** $i = 0\ ;\ i < 4\ ;\ i \leftarrow i + 1$ **do**
5: $rand\_block[i] \leftarrow rand()$;
6: **end for**
7: AES encrypts $rand\_block$
8: $x_i \leftarrow (unsigned\ char) rand\_block[0]$
9: **if** $x_i \notin XOR$ **then**
10: back to the start of while loop
11: **else**
12: break out of the while loop
13: **end if**
14: **end while**
15: output x_i

Algorithm 3 New Algorithm for Decrypting a file

Require: $encrypted\_file$ and its ADS
Ensure: $decrypted\_file$
1: $encrypted\_file\_size \leftarrow get\_file\_size(ADS)$
2: $data\_block \leftarrow 16[Byte]$
3: $decrypted\_file\_size = (encrypted\_file\_size + 20 * data\_block)/3$
4: $*encrypted\_data \leftarrow get\_file\_data(ADS)$
5: $define * data$ and copy the $first$ 160 bytes of $encrypted\_data$ to that of $data$
6: **for** $i = 0\ ;\ i < (decrypted\_file\_size - 10 * data\_block)\ ;\ i \leftarrow i + data\_block$ **do**
7: $memcpy(data + i + 10 * data\_block, encrypted\_data + 3 * i + 12 * data\_block, data\_block)$
8: **end for**
9: AES decrypts data except the $first$ 160 bytes
10: $fopen(encrypted\_file, "rb+")$ and $fseek(encrypted\_file, 0, SEEK\_SET)$
11: $fwrite(data, 1, decrypted\_file\_size, encrypted\_file)$
12: $fclose(encrypted\_file)$ and $remove(ADS)$
13: output $encrypted\_file$ as $decrypted\_file$

3. 実　　験

3.1 実験結果

本稿 2.3 節の手法に則って実験を行った結果，CryptoDrop が暗号化プロセスを検知するまでに完全に暗号化された平文ファイルの個数は表 3 になる．尚，表では，本研究で作成した new_algorithm を実装したプログラムを NEW と表記している．表 3 より，AES の 20 回の試行における暗号化ファイル数の平均値は 9.40，中央値は 7.50 であった．また，NEW（new_algorithm）の 20 回の試行における暗号化ファイル数の平均値は 29.1，中央値は 26.5 であった．

3.2 考　　察

表 3 から分かるように，暗号化と復号以外の機能を搭載していない単純な AES アルゴリズムの場合，CryptoDrop に検知されるまでに完全に暗号化した平文ファイル数の最大値は 25 個であった．これに対し，new_algorithm が CryptoDrop に検知されるまでに完全に暗号化した平文ファイル数の最大

表 3　各試行における完全に暗号化されたファイル数

アルゴリズム＼試行	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
AES	7	3	4	5	4	8	7	25	15	23	10	16	3	9	7	10	16	6	1	9
NEW	18	56	31	39	23	30	6	46	40	11	2	16	52	17	55	9	40	78	6	6

値は 78 個であり，これは単純な AES アルゴリズムのそれを大きく上回る．また，Nolen Scaife らが CryptoDrop の開発に際して実施した 14 ファミリに渡る 492 個の実際のランサムウェアを用いた実験では，CryptoDrop に検知されるまでにランサムウェアが完全に暗号化した平文ファイル数の最大値は 33 個であり，new_algorithm が完全に暗号化した平文ファイル数の最大値はこれを大きく上回った．更に今回の実験において，new_algorithm が完全に暗号化した平文ファイル数が 33 個を上回った試行は，20 回中 8 回である．以上のことにより new_algorithm は，従来のランサムウェアに適用されている暗号化アルゴリズムと比べると，CryptoDrop の各種インジケータの監視を回避しやすいアルゴリズムであると考えられる．

一方で，new_algorithm が CryptoDrop に検知されるまでに完全に暗号化した平文ファイル数が 10 個未満の試行が，20 回中 5 回存在することも表 3 から分かる．この原因としては，暗号対象ファイルのサイズが小さくなるにつれ，平文ファイルの各バイトと x_i との排他的論理和前後における類似度及びエントロピーの変化量が大きくなり易いことが挙げられる．その為，サイズが小さく且つエントロピーも低い平文ファイルが new_algorithm の暗号化の序盤に出現した場合，new_algorithm は比較的早期に CryptoDrop に検知される可能性があると考えられる．

しかし，上述のように早期に検知されることがあるとはいえ，今回の実験によって new_algorithm が CryptoDrop に対して従来のランサムウェアを上回る回避能力を備えていることが確認された為，new_algorithm のような暗号化アルゴリズムを検知する新たなインジケータ或いは機能を，早急に CryptoDrop に搭載する必要がある．単純な AES アルゴリズムに対する早期検知能力と Nolen Scaife らの先行研究から分かるように，従来のランサムウェアに採用されている暗号化アルゴリズムに対して CryptoDrop は確かに高い耐性を備えているが，new_algorithm のようなアルゴリズムに対しては比較的耐性が低く，その為 new_algorithm のようなアルゴリズムを用いたランサムウェアを攻撃者が考案した場合，より多くのユーザの重要ファイルが暗号化される危険性がある．

3.3　今後の課題

本実験は 2.1 節で述べた環境で行った．しかし異なる環境で同様の実験を行った場合，CryptoDrop がどのような振る舞いをするのかは未検証である．従って CryptoDrop に対してより精密な耐性評価を行う為には，本実験環境の他でも同様の実験を行う必要がある．また，ファイルセットを構成するファイル形式の種類及び比率をより実環境に即したものにすることにより，より実際的な耐性評価が可能となることが期待される．以上のことを今後の研究課題とする．

4. 結　　論

ランサムウェアからユーザのデータを防衛する為のシステムとして Nolen Scaife らにより CryptoDrop が開発された．先行研究と本研究により CryptoDrop は従来のランサムウェアの暗号化アルゴリズムに対して高い耐性を持つことが判明したが，一方で，本研究によって CryptoDrop の各種インジケータを回避するような機能を持った暗号化アルゴリズムに対しては比較的弱い耐性を持つこともまた判明した．ランサムウェアに対してより安全なシステムを構築する為には，本研究で述べたような新たな暗号化アルゴリズムに対して早期検知することを可能にする新たなインジケータ或いは機能を開発することも重要な課題と考えられる．

謝辞　本研究の一部は JSPS 科研費 17K00178 の助成を受けたものです．

文　　献

[1] 菊池浩明，上原哲太郎，IT Text ネットワークセキュリティ，情報処理学会 (編)，(社) オーム社，東京，2017
[2] Nolen Scaife, Henry Carter, Patrick Tranor, and R.B. Bulter, “CryptoLock(and drop it):stopping ransomware attacks on user data,” 2016 IEEE 36th International Conference on Distributed Computing Systems, 2016.DOI:10.1109/ICDCS.2016.46
[3] Jesse Kornblum, “Identigying almost identical files using contect triggered piecewise hashing,” Digital Investigation, vol.3,Supplement, pp.91-97, Sep.2006.
[4] Vassil Roussev, “Data fingerprinting with similarity digests,” DigitalForensics 2010:Advances in Digital Forensics VI, pp.207-226, 2010.
[5] 有限会社 ビイジェイ データ復旧 事業部，“AES 暗号，“有限会社 ビイジェイ，http://free.pjc.co.jp/AES/speed.html，参照 Jan.15,2018.
[6] National Institute of Standards and Technology, “Announcing the advanced encryption standard(aes),” Federal Information Processing Standards Publications 197, Nov.26,2001.
[7] ASCII. jp,“インターネットからダウンロードしたファイルは Zone.Identifier でセキュリティ管理をする，“ http://ascii.jp/elem/000/001/550/1550399/，参照 Jan.25,2018.