マルウェアデータに対する重複除外の適用

大山 恵弘　　松宮 遼

電気通信大学
182-8585 東京都調布市調布ケ丘 1-5-1
oyama@inf.uec.ac.jp　　r.matsumiya@ol.inf.uec.ac.jp

あらまし 新たに発見されるマルウェアは現在非常に多く，解析のためのそれらのデータの保存に必要なストレージ容量も増大している．ストレージ使用量の削減に効果がある技術として，重複除外がある．重複除外は，データ内の重複部分を検出し，重複部分についてはそれらのうち一つだけを保存する技術である．マルウェアデータは重複部分を多数含むことが推測されるが，マルウェアデータに対する重複除外の効果を定量的に明らかにした研究は今まで存在しない．本論文では CCC DATAset 2013，2012，2011 に対する重複除外の効果を測定した実験の結果を報告する．

Application of Deduplication to Malware Data

Yoshihiro Oyama　　Ryo Matsumiya

The University of Electro-Communications.
1-5-1, Chofugaoka, Chofu-shi, Tokyo 182-8585, JAPAN
oyama@inf.uec.ac.jp　　r.matsumiya@ol.inf.uec.ac.jp

Abstract Numerous malware instances have been newly discovered today and an increasing amount of storage has been required to accumulate them for analysis. A technology effective for reducing storage consumption is deduplication, which detects duplicated data parts and saves only one data copy for them. Although malware data is expected to contain many duplicated parts, no work has quantitatively clarified the effectiveness of deduplication against malware data. In this paper, we report the result of experiments in which we measure the effect of deduplication on CCC DATAset 2013, 2012, and 2011.

1　はじめに

マルウェアの数は爆発的に増大している．カスペルスキー社のレポート [2] は，2013 年には 1 日あたり 315,000 個もの新種のマルウェアを検知したと報告している．マルウェア関連セキュリティ業務を行う組織は，収集したマルウェア検体を解析する上で，それらをストレージに保存する必要がある．マルウェアの数が爆発的に増えれば，扱うべきデータが増大し，必要なストレージ容量も爆発的に増える．大容量データを高速に処理するためには，データをキャッシュするための大きなメモリも必要になり，コストがかかる．

大容量データの保存に必要とされるストレージを削減するにはいくつかの代表的な方法がある．一つはデータを普段は圧縮して保存しておき，利用時に展開するという方法である．この方法はデータサイズ削減に有効ではあるが，圧縮と展開に計算コストがかかるという欠点がある．もう一つは**重複除外**をサポートするファイルシステムやストレージシステムを利用する方

法である．重複除外は，データ内の重複部分を検出し，重複部分については，それらのうち一つだけを保存することにより，ストレージやメモリの消費量を削減する技術である．重複除外は，圧縮と展開による方法と比べて，データをキャッシュする機構と組み合わせやすいという利点がある．重複除外は現在多くのファイルシステムやストレージで採用されている [1,5–7].

重複除外の有用性は広く認識されており，重複除外の効果を実験により評価した研究もこれまでに多数存在する．例えば，デスクトップ環境や高性能計算環境で用いるデータに対する重複除外の効果を評価した研究が存在する [3,4]. しかし，マルウェアデータに対する重複除外の効果を評価した研究は存在せず，その効果は明らかではなかった．たとえば，マルウェアデータの集合に対して重複除外を適用した場合に，既存研究 [3] における実験結果のように，20%を超えるデータを削減できるかどうかは明らかではなかった．実際には，マルウェアデータに対しては重複除外が効果的である可能性が高い．第一の理由は，多くのマルウェアは既存のマルウェアの亜種であり，亜種の関係にあるマルウェアデータ間には，共通のデータが多く存在することが予想されるからである．第二の理由は，共通のライブラリや共通のアンパックエンジンを利用するマルウェアが多く存在することが予想されるからである．

本研究では，重複除外を実際のマルウェアデータに適用する実験を行い，マルウェアデータに対する重複除外の効果を評価する．重複除外の方法として，ファイル単位，固定長ブロック単位，可変長ブロック単位の 3 つの方法を実装し，それらを比較する．また，重複除外によるデータ削減率を，zip および tar コマンドの圧縮によるデータ削減率と比較し，圧縮と比較した場合の重複除外の有用性も明らかにする．マルウェアの検体データとして用いるデータは，MWS 2013，2012，2011 Dataset [9–11] の一部として配布された CCC DATAset 2013，2012，2011 である．

2 重複除外

2.1 ファイル単位重複除外

ファイル単位重複除外は，内容が完全に同一にもかかわらず論理的には（ファイルパスの上では）異なる複数のファイルを，1 つのファイルにまとめて保存する方法である．多数のユーザのファイルを管理するデータセンタのためのストレージにおいては，異なるユーザが共通のファイルを利用することも多いため，この方法による重複除外は一定の効果を発揮する．

この方法には，実装が単純であるという利点がある．しかし，ほとんど内容が同じである複数のファイルがあっても，内容が 1 バイトでも異なれば重複除外が行えないという欠点がある．

2.2 固定長ブロック重複除外

固定長ブロック重複除外（Fixed-Size Chunking Deduplication, 以降では FSC と呼ぶ）は，ファイルを固定長のブロック（チャンク）に分割し，同一内容のチャンクを 1 つにまとめて保存する方法である．例えば，ファイルを先頭から 8 KB ごとにチャンクに分割し，内容が同一である複数のチャンクを重複して保存しないようにしながら，ファイルを構成するチャンクを保存していく．単一ファイル内にある同一内容の複数のチャンクの間でも，異なるファイルにある同一内容の複数のチャンクの間でも，重複除外を行う．

FSC には，ファイル単位重複除外と異なり，ファイルの一部のみが共通である場合でも重複除外が行えるという利点がある．しかし，FSC には，共通のデータが，異なるファイルオフセットに出現する場合に，それらに対して重複除外が行えないという欠点がある．例えば，ある 1 GB のファイル A の先頭に 1 バイト追加してファイル B を作成したとする．そしてこれらのファイルを先頭から 8 KB ごとにチャンクに分割したとする．両ファイルはほとんど内容が同じであるにもかかわらず，チャンクの境界が入る場所が 1 バイトずれることにより，重複除外が全く行われなくなる可能性がある．

2.3 可変長ブロック重複除外

可変長ブロック重複除外は，ファイルを可変長のブロック（チャンク）に分割し，同一内容のチャンクを 1 つにまとめて保存する方法である．その方法の中では，特に，データの内容に基づいてチャンクの境界の場所が決まるもの（Content-Defined Chunking Deduplication, 以降では CDC と呼ぶ）が広く用いられている．

CDC の実現方法の 1 つを以下に述べる．まず，ファイル内の長さ N のバイト列全てに対して，そのバイト列の fingerprint を求める．通常の実装では，バイト列の先頭を指すファイルポインタを，ファイルの先頭から 1 バイトずつ後ろにずらしていきながら，長さ N の各バイト列に対して fingerprint を求めていく．そして，fingerprint の値がある条件を満たしたら，そのバイト列の直後に，チャンクの境界を入れる．条件として，例えば「fingerprint の下位 12 ビットが 0xabc であること」などを用いる．Fingerprint のアルゴリズムとしては，Rabin fingerprint [8] がよく用いられる．条件を満たす確率により，平均チャンクサイズが定まる．たとえば，上述の条件を採用した場合には，平均チャンクサイズは 2^{12} バイトとなる（もし fingerprint の取る値が一様分布的であれば，下位 12 ビットが 0xabc になる確率はほぼ $1/2^{12}$ であるため）．チャンクの境界が入った場所でファイルを分割し，分割の結果できたデータの各断片をチャンクとして保存する．

CDC には，共通のデータが異なるファイルオフセットに出現する場合でも，それらに対して重複除外を行えるという利点があるが，実装が複雑であったり，fingerprint を計算するオーバヘッドが大きいという欠点がある．

2.4 チャンクの同一性とチャンクテーブル

FSC と CDC では，あるチャンクと別のチャンクの内容が同一であるかどうかを高速に判定する必要がある．そのためによく用いられるのはハッシュ値である．すなわち，チャンクの内容のハッシュ値をチャンクに関係づけておき，2 つのチャンクのハッシュ値が同じであれば，それらのチャンクの内容は同一であると判定する．そして，内容が同一である複数のチャンクは 1 つのチャンクにまとめる．一般に，ハッシュ値を用いた同一性判定には，ハッシュ値の衝突という問題が伴う．しかし，重複除外への応用においては，作られるチャンク数の規模に応じて適切な大きさのハッシュ値空間を設定し，これにより，ハッシュ値は衝突しないと想定する．

FSC と CDC を利用した場合には，各ファイルに対して，そのファイルを構成するチャンクの情報を管理するデータ構造（チャンクテーブル）が関係づけられる．FSC におけるチャンクテーブルは，論理的には，各チャンクのハッシュ値を，ファイル先頭から順に並べたものである．CDC におけるチャンクテーブルは，論理的には，各チャンクのファイルオフセット範囲とそのチャンクのハッシュ値の組を，ファイル先頭から順に並べたものである．チャンクテーブルを具体的にどう表現するかは実装に依存する．

チャンクテーブル自身もストレージやメモリを消費することに注意が必要である．重複除外が適用されるデータに共通部分がない場合には，ファイルをチャンクに分割しても，データの総サイズは変化しない．しかし，その場合でもチャンクテーブルは必要であるため，ストレージやメモリの消費量は，チャンクテーブルのサイズの分だけ逆に増える．また，内容が同一であるチャンクがより多く作られるようにするためには，平均チャンクサイズを小さくすることが効果的である．しかし，平均チャンクサイズを小さくすると，1 つのファイルを構成するチャンクの数が増加するため，チャンクテーブルのサイズが増加する．よって，平均チャンクサイズは小さいほど良いというものではなく，適切な値に設定する必要がある．

3 実験

3.1 方法

重複除外の適用によるマルウェアデータセット CCC DATAset のストレージ消費量の変化を測定した．CCC DATAset に含まれる検体の数と検体の平均ファイルサイズを表 1 に示す．

表 1: CCC DATAset に含まれる検体の数と平均サイズ (KB)

	2013	2012	2011	2010	2009	2008
数	7,028	10,538	50	50	10	1
サイズ	104	87	115	220	169	199

2011 年から 2008 年のデータセットについては，検体数が少ないため，検体数が少ない場合の結果を見るために 2011 年のデータセットを使う以外，実験から除外した．すなわち，2013 年から 2011 年のデータセットに対して実験を行った．異なる年のデータセットをまたがっての重複除外は行わず，各年のデータセットの中での重複除外を行った．

なお，2013 年から 2008 年のデータセットに含まれる検体ファイルの形式を Linux 上の file コマンドで調べたところ，2012 年のデータセットの 2 検体と 2011 年のデータセットの 1 検体を除く全てのファイルが PE（Portable Executable）フォーマットだった．すなわち，これら 3 検体以外の全ファイルは，Windows 向けのマルウェアバイナリであると推定できる．

マルウェアデータが持つ特徴を明らかにするために，マルウェアではない通常のプログラムファイルも利用した．具体的には，Windows 7 の `c:\Windows\System32` フォルダにある，`exe`，`EXE`，`dll`，`DLL` という拡張子を持つシステムプログラムファイル（2415 個）を用いた．

2 章では，重複除外の方法として 3 つを提示した．そのうち，ファイル単位重複除外は，今回用いたデータセットに対して全く効果が無い．それは，各年のデータセットに含まれる検体は一意であり，内容が全く同じ検体は存在しないためである．よって，本実験では FSC と CDC による重複除外を比較した．

FSC の適用においては，チャンクサイズとして 2 KB，4 KB，8 KB の 3 通りを用いた．CDC の適用においても，平均チャンクサイズとして 2 KB，4 KB，8 KB の 3 通りを用いた．CDC では fingerprint として Rabin fingerprint を用いた．チャンクの境界をファイルに入れる条件としては，「平均チャンクサイズが 2^x であるとき，fingerprint の値の下位 x ビットが，0xaaaaaaaa の下位 x ビットと等しい」を用いた．例えば，平均チャンクサイズを 4 KB に設定したときには，fingerprint の下位 12 ビットが 0xaaa である場所に，チャンクの境界が入ることになる．Fingerprint を求めるバイト列のサイズ N は 48 とした．さらに，最大チャンクサイズを設定し，16 KB の区間の間ずっとチャンクの境界が入らないときには，fingerprint の値にかかわらず無条件に境界を入れて，チャンクサイズは最大でも 16 KB になるようにした．

チャンクの同一性判定のためのハッシュ値を計算するアルゴリズムとしては SHA-1 を用いた．SHA-1 のハッシュ値空間（160 bit）に比べ，作られるチャンクの数ははるかに少なく，ハッシュ値の衝突が起こる可能性は極めて低い．実際，調査によると，本実験ではハッシュ値の衝突は一度も発生していない．

データサイズを求める際には，各ファイルに対して作成されるチャンクテーブルのサイズも計算に入れている．チャンクテーブルのサイズは実装に依存するが，本実験では次の値を用いた．FSC では，ファイルを構成するチャンク数 ×160 bit のサイズのチャンクテーブルが作られるとした．この 160 bit は SHA-1 のハッシュ値を格納するための領域である．CDC では，ファイルを構成するチャンク数 × (64 bit + 64 bit + 160 bit) のサイズのチャンクテーブルが作られるとした．この 2 つの 64 bit はファイル中のチャンクの開始オフセットと終了オフセットを格納するための領域であり，160 bit は SHA-1 のハッシュ値を格納するための領域である．

3.2 結果

3.2.1 圧縮コマンドによる圧縮の効果

重複除外の効果を評価する基準を得るために，zip コマンドと tar コマンド（gzip 圧縮付き）によるマルウェアデータセットの圧縮効果を測定した．重複除外によるデータ削減率を，両コマンドの圧縮によるデータ削減率と比較することにより，重複除外の有効性をより正確に評価することを狙っている．2 つのコマンドを用いた理由は，zip コマンドは複数ファイルの情報を

表 2: 圧縮コマンドによる圧縮の効果

	2013	2012	2011	システムプログラム
総データサイズ	732,408,195	924,758,408	5,791,067	1,407,397,944
zip ファイルのサイズ	591,250,851	830,249,316	4,478,634	554,639,556
zip によるデータサイズの減少率	19.27	10.21	22.66	60.59
tgz ファイルのサイズ	592,403,549	832,160,041	4,484,112	553,707,040
tar によるデータサイズの減少率	19.11	10.01	22.56	60.65

利用した圧縮を行わず，tar コマンドは行うためである．

用いたデータセットの総データサイズと，それらのファイル群を圧縮したファイルのサイズを表 2 に示す．データ削減率はマルウェアデータセットでは最大で 22.66%だった一方，システムプログラムでは最大で 60.65%だった．圧縮コマンドによる圧縮効果は，システムプログラム群に対しては著しく大きい（または，マルウェアデータに対しては比較的小さいものだけしか期待できない）ことがわかった．すなわち，マルウェアとシステムプログラムとの間で，圧縮効果に関する傾向が大きく異なることがわかった．この原因の一つは，一部のマルウェアはパッカーでパックされており，既に圧縮済みであることだと推測している．

3.2.2 マルウェアデータセットに対する重複除外の効果

CCC DATAset 2013, 2012, 2011 に対する重複除外の効果を測定した結果をそれぞれ表 3, 4, 5 に示す．各表中で CDC や FSC の下の括弧に書かれたサイズは（平均）チャンクサイズである．総データサイズの単位はバイトである．重複除外の効果を端的に示す最も重要な数値は，総データサイズの減少率である．

測定結果からわかることの 1 つ目は，多くの検体を含む 2013 年と 2012 年のデータセットに対しては，圧縮コマンドによる圧縮よりも重複除外の方がデータサイズの削減率が大きいことである．重複除外により，2013 年のデータセットに対しては最大約 28.7%（圧縮コマンドでは最大 19.27%），2012 年のデータセットに対しては最大約 21.69%（圧縮コマンドでは 10.21%），データサイズを削減することができた．この結果は，マルウェアデータによるストレージやメモリの消費量を削減する目的に対する重複除外の有用性を端的に示すものである．

2 つ目は，FSC と CDC による総データサイズの減少率としては近い値が出ており，FSC が良いか CDC が良いかはデータセットや（平均）チャンクサイズによって決まるということである．2013 年と 2011 年のデータセットでは，どの（平均）チャンクサイズ間で比較しても，FSC の削減効果の方が大きくなっている．2012 年のデータセットでは，CDC の削減効果の方が大きい場合もある．一般には，共通のデータ部分が様々なファイルオフセットに出現する場合に，CDC が有利になる．本実験で CDC が FSC に対して大きく優位ではない理由は，今回用いたデータセットでは共通のデータ部分が共通のオフセットに出現することが多かったためであると推測している．現時点での結論は，マルウェアデータセットに対する重複除外において，FSC と CDC のどちらがより多くのデータを削減できるかについては一概に言えず，さらなる調査が必要であるというものである．

3 つ目は，年によって重複除外の効果が大きく異なることである．重複除外を適用するデータの種類がマルウェアであることを条件として固定しても，そこにどんなマルウェアが含まれるかによって，重複除外の効果は大きく変化することがわかる．なお，2011 年のデータセットでは，検体が極めて少ないため，重複除外の効果が小さいのは自然である．一方，ともに多くの検体を含む 2013 年と 2012 年の結果を比較すると，検体が少ない 2012 年の方が総データサイズの減少率が大きくなっており，それは自然ではない．その原因としてまず考えられることは，2013 年のデータセットは，2012 年のデータセットに比べて，亜種を含む割合が高かった

表 3: CCC DATAset 2013 に対する重複除外の効果

	CDC (2KB)	CDC (4KB)	CDC (8KB)	FSC (2KB)	FSC (4KB)	FSC (8KB)
チャンク数（重複除外前）	358,815	185,397	104,660	359,316	181,047	92,204
チャンク数（重複除外後）	256,229	133,648	78,194	252,892	128,213	66,302
チャンク数の減少率	28.59	27.91	25.28	29.61	29.18	28.09
総データサイズ（重複除外前）	732,408,195	732,408,195	732,408,195	732,408,195	732,408,195	732,408,195
総データサイズ（重複除外後）	534,666,822	533,627,244	533,690,755	522,202,948	522,756,207	529,053,535
総データサイズの減少率	26.99	27.14	27.13	28.70	28.62	27.76

表 4: CCC DATAset 2012 に対する重複除外の効果

	CDC (2KB)	CDC (4KB)	CDC (8KB)	FSC (2KB)	FSC (4KB)	FSC (8KB)
チャンク数（重複除外前）	470,273	242,686	139,136	454,793	230,080	117,006
チャンク数（重複除外後）	354,089	184,231	108,665	351,029	178,949	92,693
チャンク数の減少率	24.70	24.08	21.90	22.81	22.22	20.77
総データサイズ（重複除外前）	924,758,408	924,758,408	924,758,408	924,758,408	924,758,408	924,758,408
総データサイズ（重複除外後）	734,204,899	735,559,798	747,205,771	724,117,728	728,368,721	742,247,465
総データサイズの減少率	20.60	20.45	19.20	21.69	21.23	19.73

表 5: CCC DATAset 2011 に対する重複除外の効果

	CDC (2KB)	CDC (4KB)	CDC (8KB)	FSC (2KB)	FSC (4KB)	FSC (8KB)
チャンク数（重複除外前）	2,641	1,346	772	2,838	1,426	725
チャンク数（重複除外後）	2,448	1,268	730	2,555	1,322	688
チャンク数の減少率	7.30	5.79	5.44	9.97	7.29	5.10
総データサイズ（重複除外前）	5,791,067	5,791,067	5,791,067	5,791,067	5,791,067	5,791,067
総データサイズ（重複除外後）	5,520,528	5,520,490	5,560,184	5,268,243	5,393,603	5,510,655
総データサイズの減少率	4.67	4.67	3.98	9.02	6.86	4.84

表 6: Windows 7 のシステムプログラムファイルに対する重複除外の効果

	CDC (2KB)	CDC (4KB)	CDC (8KB)	FSC (2KB)	FSC (4KB)	FSC (8KB)
チャンク数（重複除外前）	696,025	326,410	189,941	688,128	344,637	172,922
チャンク数（重複除外後）	538,779	284,069	167,100	625,506	316,887	160,032
チャンク数の減少率	22.59	12.97	12.02	9.10	8.05	7.45
総データサイズ（重複除外前）	1,407 M	1,407 M	1,407 M	1,407 M	1,407 M	1,407 M
総データサイズ（重複除外後）	1,242 M	1,241 M	1,246 M	1,293 M	1,301 M	1,306 M
総データサイズの減少率	11.70	11.81	11.39	8.10	7.53	7.19

というものであるが，それについては今後確認が必要である．

4つ目は,(平均) チャンクサイズが小さいほど総データサイズの削減効果が大きい傾向があることである．この傾向は全てのデータセットで観測できる．なお，本実験では，チャンクテーブルはチャンクに比べて，はるかに小さいストレージしか消費していない．チャンクテーブルのデータサイズを計算に入れることによるデータ削減率の変化は，どの場合でも 2%以下である．すなわち，チャンクテーブルのサイズは常に 0 であると仮定したとしても，総データサイズの減少率は 2%以下しか変動しない．

次に，多くのファイルに共通に出現するチャンクはどのようなものであるかを調べた．2013 年のデータセットに対して CDC（平均チャンクサイズ 4 KB）を行った結果作られたチャンク群を取り上げる．出現回数順に並べると，まず，4080 回出現するチャンクが 1 種類あり，839 回出現するチャンクが 1 種類ある．4080 回出現するチャンクは，ファイルの先頭部分に出現している．PE フォーマットのファイルの先頭にはヘッダが置かれるが，このヘッダ部分が共通のデータとして重複除外されたことがわかる．839 回出現するのは，全バイトが 0 であるチャンクであり，そのサイズは最大チャンクサイズ 16 KB である．以降，214 回出現が 112 種類，74 回出現が 9 種類，73 回出現が 5 種類と続く．

3.2.3 システムプログラムファイルに対する重複除外の効果

Windows 7 のシステムプログラムファイルに対する重複除外の効果を測定した結果を表 6 に示す．重複除外による総データサイズの削減率は最大 11.81%であり，CCC DATAset 2013, 2012 を用いた場合と比べて削減の度合いが小さい．また, 圧縮によるデータ削減率（最大60.65%）と比較してもだいぶ小さい．重複除外の効果が小さいことは，2 KB，4 KB，8 KB という大きな単位での共通データがシステムプログラムファイル群の中に少ないことを示唆している．このような結果が出た原因として第一に考えられることは，単純に，CCC DATAset 2013, 2012 と比べてファイル数が少ないことである．第二に考えられることは，マルウェアデータセット中には多くの亜種が含まれるが，互いに亜種のような関係にあるシステムプログラムファイルの組は少ないという可能性である．第三に考えられることは，ライブラリが静的リンクされている割合が検体とシステムプログラムで大きく異なる可能性である．プログラムの共通部分を動的ライブラリにまとめることは一種の重複除外とみなせる．システムプログラムでは共通部分を DLL として共有することにより，重複除外の効果を既にある程度実現しており，さらなる効果が出にくくなっている可能性がある．

4 関連研究

実用的なデータに対する重複除外の効果を測定した研究が存在する．Meister らによる研究 [3] は，高性能計算のためのデータセンタに保存されたデータに対して重複除外を適用し，大半のデータセットに対して 20%から 30%のデータを削減可能であることを示している．本研究は，高性能計算向けのデータではなく，マルウェアデータセットに対する重複除外の効果を明らかにしている．

村上らの研究 [12] では，気象予測アプリケーションが生成する大規模データに対して CDC を適用し，データサイズを 30%以上削減できたことを示している．この研究は科学技術計算に用いられるデータの冗長度を明らかにするものであり，マルウェアファイルに含まれるデータの冗長度は明らかにしていない．我々の研究の結果，少なくとも今回用いたマルウェアデータセットに対しては，重複除外によるサイズ削減率は 30%に届かないことがわかった．

Meyer らによる研究 [4] では，マイクロソフト社の 857 台のデスクトップコンピュータの 4 週間に渡るファイルシステムデータに対して重複除外を適用した結果が報告されている．彼らの研究では，ファイル単位，固定長ブロック単位，可変長ブロック単位の 3 つの重複除外を実験により比較している．彼らの研究では，可変長ブロック単位での重複除外が最大のデータ削

減率を与えるものの，ファイル単位での重複除外が極めて有効であったことが報告されている．また，拡張子が dll であるファイルで最も多くの重複が検出されたことが報告されている．彼らの研究では多様な種類のファイルに対して重複除外を適用しているが，本研究では対象をマルウェアプログラムのファイルに限定し，マルウェアファイルが持つ一般的な性質と重複除外の関係についての知見を得ることを目指している．

5 まとめと今後の課題

本論文ではマルウェアのデータセットである CCC DATAset 2013，2012，2011 に重複除外を適用し，データサイズがどの程度削減されるかを明らかにした．検体が多い 2013 年と 2012 年のデータセットに対しては，重複除外によって，圧縮コマンドでの圧縮よりも，総データサイズを小さくできることがわかった．また，Windows のシステムプログラムに対してよりも，マルウェアデータセットに対しての方が，重複除外を適用した場合のデータ削減率が大きいことがわかった．これは，マルウェアデータが重複除外に適したパターンを多く含むことを示唆している．大量のマルウェアデータを保存する際には，重複除外の導入により，ストレージやメモリの消費量を効果的に減らせる可能性が高いことを本研究の実験結果は示している．

今後の課題を以下に示す．第一に，マルウェアのどのようなコードやデータ間に共通部分が存在したかを明らかにする必要がある．例えば，パッカーの復号エンジンが共通部分として検出されたかどうかや，亜種マルウェア間と異種マルウェア間とで重複除外の効果はどの程度異なるか，などを調査していきたい．第二に，マルウェア検体のみならず，通信データなどについても重複除外の効果を測定したい．今回は MWS Dataset のうち，CCC DATAset のみを実験に用いたが，D3M Dataset を利用して，攻撃を含む通信データに存在する冗長性について知見を得ることは興味深いテーマであると考えている．

謝辞 電気通信大学の高橋一志氏から本研究に有益なコメントをいただいた．本研究の一部は科学技術振興機構戦略的創造研究推進事業「ポストペタスケールデータインテンシブサイエンスのためのシステムソフトウェア」による支援を受けている．

参考文献

[1] Jeff Bonwick. ZFS Deduplication. http://blogs.oracle.com/bonwick/entry/zfs_dedup, 2009.

[2] Kaspersky Lab. 数字で振り返る 2013 年：1 日に 315,000 個もの新種のマルウェアが発生. http://www.kaspersky.co.jp/info?id=207585920, 2013.

[3] Dirk Meister, Jürgen Kaiser, Andre Brinkmann, Toni Cortes, Michael Kuhn, and Julian Kunkel. A Study on Data Deduplication in HPC Storage Systems. In *Proceedings of SC12*, 2012.

[4] Dutch T. Meyer and William J. Bolosky. A Study of Practical Deduplication. In *Proceedings of the 9th USENIX Conference on File and Storage Technologies*, pages 1–13, 2011.

[5] NetApp. Data ONTAP 8 Operating System. http://www.netapp.com/us/products/platform-os/data-ontap-8/.

[6] Opendedup. http://opendedup.org/.

[7] João Paulo and José Pereira. A Survey and Classification of Storage Deduplication Systems. *ACM Computing Surveys*, 47(1), 2014.

[8] Michael O. Rabin. Fingerprinting by Random Polynomials. Technical Report TR-15-81, Center for Research in Computing Technology, Harvard University, 1981.

[9] 畑田充弘, 中津留勇, 秋山満昭. マルウェア対策のための研究用データセット 〜 MWS 2011 Datasets 〜. マルウェア対策研究人材育成ワークショップ 2011 (MWS2011), 2011.

[10] マルウェア対策研究人材育成ワークショップ 2012 (MWS2012). http://www.iwsec.org/mws/2012/, 2012.

[11] 神薗雅紀, 畑田充弘, 寺田真敏, 秋山満昭, 笠間貴弘, 村上純一. マルウェア対策のための研究用データセット 〜 MWS Datasets 2013 〜. マルウェア対策研究人材育成ワークショップ 2013 (MWS2013), 2013.

[12] 村上じゅん, 石黒駿, 大山恵弘. Content-Defined Chunking を用いた重複除外キャッシュ機構の実装と評価. 情報処理学会研究報告, volume 2012-HPC-137, 2012.