

# **Shrinking the gap: carving NTFS-compressed files**

## *Recovering deleted NTFS-compressed files*

By Joachim Metz <forensics@hoffmannbv.nl>  
Hoffmann Investigations

## Summary

An important part of digital forensic investigation is the recovery of data, particularly files. The recovery of data and files highly depends on the recovery tooling used. This paper focusses on a part of file carving where most of the currently available data and file recovery tools do not provide support for, namely recovering NTFS-compressed data.

The paper also provides an overview of the consequences of NTFS compression on data and investigative techniques. In this paper a basic understanding of NTFS is assumed. For more information about NTFS check the references.

## Background

In 2006 the Digital Forensic Workshop (DFRWS) provided a challenge in which fragmented files were recovered from a randomized data image. Joachim Metz and Robert-Jan Mora developed a proof of concept tool called ReviveIt (revit, which was later renamed to revit06) [DFRWS06]. Revit uses a carving technique called Smart Carving, which is more of a conceptual technique that uses the combination of other techniques such as: file structure based carving.

The first implementation (currently known) was by Nick Mikus in foremost [MIKUS05]. This allowed foremost far better results than before when using header/footer and header/maximum (file) size carving techniques.

Revit06 provided interesting results. Far better than carving tools that used the header/footer and header/maximum (file) size carving methods like at that time. Note that today carvers like scalpel and EnCase still use this naive carving approach. This called for some research into the quality of file carvers. In late 2006 Bas Kloet was working on the subject of measuring and improving the quality of carving tools. His findings were published in his Master thesis 'Measuring and Improving the Quality of File Carving Methods' [KLOET07].

Early 2007 the DFRWS came with a new and improved file carving challenge [DFRWS2007]. This motivated Joachim Metz and Robert-Jan Mora to start working on revit07 in combination with Bas Kloet, while doing his research on the quality of file carvers. This version of revit was also sent in for the 2007 challenge.

In the meantime a lot of work has been done to improve carving tools, e.g. PhotoRec by Christophe Grennier is a very useful and qualitative carver with support for a vast amount of files and file systems.

However, in a recent investigation we needed to recover NTFS-compressed data and little of the currently available carving tools provided support for this. This paper discusses a technique to recover NTFS-compressed and its application in forensic carving tools.

## Document information

**Author(s):** Joachim Metz <forensics@hoffmannbv.nl>

**Abstract:** This document contains information about recovering NTFS-compressed files.

**Classification:** Public

**Keywords:** NTFS compression, carving, file recovery

## License

```
Copyright (c) 2009 Joachim Metz <forensics@hoffmannbv.nl>.
Permission is granted to copy, distribute and/or modify this document under
the terms of the GNU Free Documentation License, Version 1.1 or any later
version published by the Free Software Foundation; with no Invariant Sections,
with no Front-Cover Texts and with no Back-Cover Texts. A copy of the license
is included in the section entitled "GNU Free Documentation License".
```

## Version

Version	Author	Date	Comments
1.0	Joachim Metz	September 2, 2009	Initial version.

# Table of Contents

1. NTFS compression.....	1
1.1. Block-based storage.....	1
1.2. Block-based compression.....	1
1.3. NTFS compression algorithm.....	3
1.4. Implications for digital forensic investigation.....	5
2. Carving for NTFS-compressed data.....	5
2.1. Carving more than 64 KiB.....	6
2.2. Improving carving techniques.....	7
3. Conclusion.....	7
Appendix A. References.....	8
Appendix B. ReviveIT .....	9
Appendix C. GNU Free Documentation License.....	11

# 1. NTFS compression

Before discussing the details of recovering NTFS-compressed data, it is necessary to provide an overview of how NTFS compression works and what its effects are on binary data.

In the digital forensics field not much has been published about NTFS compression. [SANDERSON02] provides a discussion of NTFS compression and digital forensic investigation, but does not go into details of the compression algorithm used. Also [CARRIER05] does not provide any details of the NTFS compression algorithm.

Luckily for us, the Linux NTFS project has provided for elaborate documentation about NTFS including its compression algorithm [RUSSON05].

## 1.1. Block-based storage

It is important to know is that NTFS uses cluster blocks<sup>1</sup> to store file data. These cluster blocks are commonly 4096 bytes of size. Information about the usage of these cluster blocks is maintained in the allocation bitmap and in the Master File Table (MFT), in particular the data runs.

NTFS-compressed files can consist of both compressed, uncompressed and sparse cluster blocks. NTFS uses data runs to determine which cluster blocks make up the file data. These data runs also indicate which cluster blocks are compressed, uncompressed or sparse. Without these data runs there is no other information that can provide us with this information.

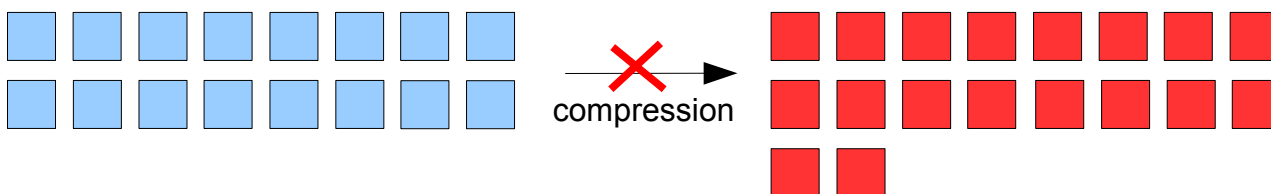
## 1.2. Block-based compression

The block-based storage technique is also used within NTFS compression. Data is compressed at 16 cluster blocks at a time.



*Illustration 1: Successful compression of 16 cluster blocks*

If 16 cluster blocks of the original data (above in blue) can be compressed to 11.5 cluster blocks (above in red) the data is stored in 12 cluster blocks. Note that the last block contains slack space (above in yellow), which will be referred to as NTFS compression slack space. The 16 compressed cluster blocks are also referred to as a compression unit.



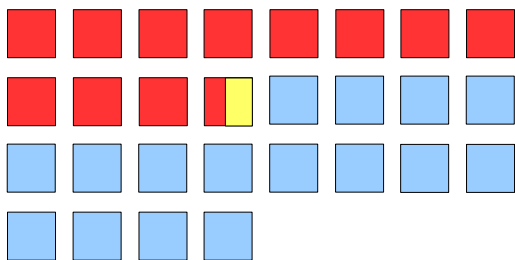
*Illustration 2: Unsuccessful compression of 16 cluster blocks*

If 16 cluster blocks of the original data (above in blue) cannot be compressed to fewer than 16 cluster blocks (above in red) the data is stored as the original 16 uncompressed blocks.

---

<sup>1</sup> Note that in other documentation the cluster block is sometimes referred to as just cluster.

If the 32 cluster blocks would be part of a single file, the file would be stored similar to the diagram below.

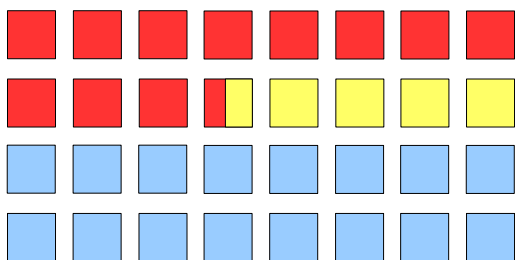


*Illustration 3: 2 compression units stored sequentially*

[SANDERSON02] states that files, that were initially stored uncompressed and then compressed, leave previously allocated cluster blocks in place:

On the face of it, it would seem logical for the compressed data from the second compression unit to be saved to disk immediately after the first compression unit. It is my belief that Microsoft has chosen this method for reasons of efficiency. By leaving the saved clusters within a compression unit initially empty means that if data is added to the first compression run then it just needs to be expanded into the free space that was left after compression.

This would mean that the compressed file would contain 4.5 cluster blocks of NTFS compression slack space.



*Illustration 4: 2 compression units stored sequentially with additional slack space*

In our initial review of carving NTFS-compressed files we mainly found compressed units with less than 1 block NTFS compression slack space (such as in illustration 3). This could be dependent on how the compressed file was created or by the operating system and/or file system version used. Our tests were done on an forensic copy of a Microsoft Windows XP SP2 system containing NTFS version 5.1. It is unknown which operating system and version of NTFS were used in [SANDERSON02], probably Microsoft Windows 2000 with NTFS version 5.0. However more research is needed to get more conclusive answers about the allocation behavior of NTFS-compressed files.

An important aspect of carving is that an empty cluster block (a cluster block entirely consisting of zero byte values) can be stored as a sparse block. The only reference to these sparse blocks is in the NTFS data runs.

Another important aspect of for carving is that although a cluster block of data is stored within a compression unit, it still can consist entirely of uncompressed data. The compressed unit starts with a block header that contains the size of the compressed data within the block and a flag to indicate if

the data within the block is compressed or not. In the illustration below the compression unit contains 2 uncompressed blocks.

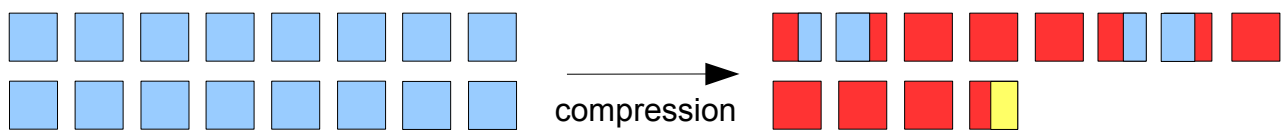


Illustration 5: a compression unit containing 2 uncompressed blocks

### 1.3. NTFS compression algorithm

NTFS uses a compression algorithm that is based on LZ77, this algorithm is referred to as LZNT1. The data is compressed a cluster block at a time. A 'compressed' cluster block starts with a 16-bit block header. This block header consists of the following values:

Bits	Description
0-11 (LSB)	Compressed data size minus 1
12-14	Unknown flags
15 (MSB)	Data is compressed

If the first block starts at offset 0, the next block starts at offset of the first block + the size of the block header + the size of the compressed data + 1, e.g.  $0 + 2 + 3043 + 1 = 3046$ .

The block header is directly followed by data. The MSB in the block header indicates if the data is compressed or not. If the data is uncompressed, the data is stored unaltered. If the data is compressed, the data is stored in groups of 8 values preceded by a tag byte.



Illustration 6: A tagged compression group

Every bit in the tag byte indicates if a value is compressed or not. If the bit is set (1) the corresponding value is compressed if not set (0) the value is uncompressed. Uncompressed values are stored as bytes, compressed values are stored as a 16-bit compression tuple<sup>1</sup>. This compression tuple is a combination of offset and size values. The actual amount of bits used by both the offset and size values within the tuple is dependent of the amount of data already uncompressed by the algorithm. The less the amount of data already uncompressed, the smaller the offset and larger the size. The greater the amount of data already uncompressed, the larger the offset and smaller the size.

The offset value is relative from the end of the uncompressed data. The size contains a value that indicates how much bytes from the uncompressed data should be back referenced.

As long as the size is non-zero, the decompression algorithm will look for the byte at the offset in the uncompressed data and copy it to the end of the uncompressed data and reduce the size by one. Note that the offset does not change, but because the end of the uncompressed data has grown by one, the offset refers to the next byte in the uncompressed data.

To provide you with an idea of how the NTFS decompression algorithm works, consider the following compressed data:

<sup>1</sup> Tuple meaning combination of two values.

00000000	<b>00</b>	23	69	6e	63	6c	75	64	65	<b>00</b>	20	3c	6e	74	66	73	. #include. <ntfs
00000010	2e	68	<b>04</b>	3e	0a	<b>07</b>	<b>88</b>	3c	73	74	64	69	<b>01</b>	<b>01</b>	<b>80</b>		.h.>...stdio...

In the example above the tag bytes have been made bold and the compression tuples bold and red. Below the same data is represented somewhat different.

00000000b	'#'	'i'	'n'	'c'	'l'	'u'	'd'	'e'
00000000b	' '	'<'	'n'	't'	'f'	's'	'.'	'h'
00000100b	'>'	'\n'	<b>0x07</b>	<b>0x88</b>	's'	't'	'd'	'i'
00000001b	<b>0x01</b>	<b>0x80</b>						

The tag byte of the first compression group is 0, indicating that none of the values is compressed, so is the second. Therefore the values of the first and second compression group can directly be copied to the uncompressed data.

```
#include <ntfs.h>
```

However, the third tag byte contains a value of 0x04, which indicated the third value is a compression tuple. This compression tuple consists of 0x8807.

For this compression tuple the amount of bits used for the offset is 5 (which results in an offset shift of 11 bits) and the amount of bits used for the size is 11 (which results in a size mask of 0x07ff). For more information about the actual calculation of the amount of bits used for the offset and size, check [RUSSON05].

The tuple consists of:

offset:	0x8807 >> 11	=>	-1 * ( 17 + 1 )	=>	-18
size:	0x8807 & 0x07ff	=>	7 + 3	=>	10

The tuple refers to an offset in the uncompressed data. The uncompressed values in a compression group can be added directly to the uncompressed data. Therefore, the uncompressed data we have up to now consists of the following 18 bytes:

```
#include <ntfs.h>
```

Note that the last byte is the end of line character represented above by the empty line.

Therefore the tuple (-18,10) refers to the following part of the uncompressed data.

```
#include <
```

This results in the following uncompressed data

```
#include <ntfs.h>
#include <
```

The remaining uncompressed values of the third compression group are added directly to the uncompressed data, which results in the following uncompressed data:

```
#include <ntfs.h>
#include <stdio
```



The fourth tag byte refers to another compression tuple that consists of:

```
offset: 0x8001 >> 11      => -1 * ( 16 + 1 ) => -17
size:   0x8001 & 0x07ff => 1 + 3           => 4
```

As you might have guessed the tuple (-17,4) refers to the following part of the uncompressed data:

```
.h>
```

Eventually the uncompressed data is:

```
#include <ntfs.h>
#include <stdio.h>
```

## 1.4. Implications for digital forensic investigation

NTFS compression has several implications for digital forensic investigation. Strings in unallocated compressed data are not stored continuously. A string and an index-based search can only find strings of 8 bytes or less and only when the string is not separated by a tag byte.

NTFS-compressed data can be tricky to carve. Most of the currently available carvers do not handle NTFS-compressed data correctly. NTFS compression breaks most of the file structure based carving techniques like header/footer, header/embedded length and file structure based carving, but also other carving methods that assume the data is 'plain' representation of file data.

## 2. Carving for NTFS-compressed data

As explained in the previous chapter NTFS compression breaks several of the most commonly used carving techniques. So how to carve NTFS-compressed data? The solution is very simple: compensating for the NTFS compression.

Recently in a case we had to recover NTFS-compressed Microsoft Outlook message (.msg) files. A .msg file basically is a Message Application Programming Interface (MAPI) message stored in an OLE Compound File (aka Compound Binary File) [MICROSOFT09].

As you might know, the first 8 bytes of an OLE Compound File contain the following signature.

```
d0 cf 11 e0 a1 b1 1a e1
```

For uncompressed files these 8 bytes would be the first 8 bytes of the first cluster block of the file. However if the OLE Compound File is NTFS-compressed the signature is similar to the following:

```
09 80 00 d0 cf 11 e0 a1 b1 1a e1
```

The first 2 bytes are the compression block header, the third is the tag byte and the remaining 8 bytes are the actual signature. Because the NTFS compression algorithm uses the compression tuples to refer previously uncompressed data in the same compression block and the OLE Compound file signature consists of 8 different bytes, the 'uncompressed' signature can be found at the fourth byte in the first compressed cluster block of the file. Also note that the tag byte is always

0 for the OLE Compound File signature value.

Using revit07 we devised a carving configuration that searches for a compressed OLE Compound File signature with a tag byte at the third byte of a compressed cluster block (see appendix B). When the signature was found, revit07 was instructed to use the structure of the NTFS compression blocks to carve at least the first 16 compression blocks (compression unit). To understand the configuration note that the last compression block is always followed by a zero block header. The configuration must allow for less than 16 compression blocks for small files, which require only a smaller amount of blocks.

We decompressed these blocks by using an implementation of the NTFS decompression algorithm according to the information in [RUSSON05]. This accounted for a maximum of  $16 \times 4096 = 65536$  bytes (64 KiB) of uncompressed data. For some of the .msg files this amount was sufficient to successfully carve the file.

Using this technique, we were able to carve near a 1000 fragments of the first 64 KiB of OLE Compound Files from the NTFS volume we were investigating.

In most cases, the first 64 KiB of a .msg file contains the RTF compressed<sup>1</sup> message body and the transport headers. The RTF compressed message body can be extracted uncompressed from fragments of the carved .msg files in a similar way as carving for the NTFS-compressed data.

## 2.1. Carving more than 64 KiB

However, 64 KiB was not sufficient for a certain .msg file. We had been able to recover the message body and the transport headers (SMTP headers in this case) but we also needed the attachments. From the transport headers it was clear that the attachments were JPEG files. When using the NTFS compression algorithm, JPEG files remain fairly uncompressed. Sometimes the first part of the JPEG file is NTFS-compressed.

Using a hex editor we were able to tell that most of the remainder of the OLE Compound File and the JPEG data it contained were stored directly after the first compression unit. This provided us with a rough end offset of the .msg file.

We devised the following approach to carve the remainder of the file.

```
While the end offset has not been reached:
```

```
    Try to decompress a compression unit (16 cluster blocks)
```

```
    Consider the decompression a success if the decompression was
    successful and ( if the end offset has not been reached and the
    uncompressed data is 16 x 4096 bytes of size or the if the end
    offset has been reached )
```

```
    If the decompression was successful copy the uncompressed data to
    the output file, otherwise copy the raw data to the output file
```

```
    If the decompression was successful, continue at the cluster block
    offset after the compression unit. Otherwise continue with the next
    cluster block
```

---

<sup>1</sup> For details on the RTF compression (LZFu) see the Personal Folder File format of the libpff project

This provided us with large amount of the .msg file. Alas it was not complete. Because OLE Compound Files can have large parts of 0 byte data, this was probably caused by missing sparse blocks.

However, carving the uncompressed output file we were able to retrieve 2 of 4 the JPEG files directly from the file. Adroit Photo Forensics 2009 was even able to recover 3 of 4 JPEG files completely and 1 partially from the uncompressed output file . We finally obtained the evidence we needed.

## **2.2. Improving carving techniques**

Although the method we used to recover most of the NTFS-compressed .msg file is a very crude one it worked sufficiently. A brute force approach to recovering NTFS-compressed data should work fairly well.

However, to improve file structure based carving techniques for handling NTFS they will need to operate on multiple levels, which fits neatly in the idea of Smart Carving, namely:

- at the lowest level the carver should handle file system characteristics e.g. like the cluster block size and NTFS compression. Handling the NTFS compression could be achieved by testing for both a NTFS-compressed as normal file signatures as provided by the next level, or decompressing cluster blocks if the higher levels indicate the file structure no longer matches the specifications.
- at the intermediate levels the carver should handle file and content characteristics, in this case the characteristics of an OLE Compound File and/or the JPEG files.
- at a highest level the carver should handle fragmentation and compensate for sparse blocks.

To create an implementation of this conceptual approach further research is needed.

## **3. Conclusion**

NTFS compression has a high impact on how to proceed in a digital forensic investigation. However, little information is available about NTFS compression. Investigative techniques like string and index-based search are fairly limited when searching through NTFS-compressed data. Also, most of the carving tools fail to detect and recover NTFS-compressed files.

In this paper we saw that a brute force approach to decompress NTFS-compressed data works fairly well. But processing large hard disks in this manner can be time-consuming. In short, handling NTFS compression is definitely something that would be very useful if added to carving tools.

To be able to handle NTFS-compressed data within carving tools would require a multiple level (layer) approach. The lower level handling file system characteristics, intermediate levels handling the file format and the highest level handling fragmentation and compensating for sparse blocks. The implementation of this conceptual approach is a topic for another paper.

## Appendix A. References

### [SANDERSON02]

Title: NTFS Compression – a forensic view  
Author(s): Paul Sanderson  
Date: October 2002  
URL: <http://www.sandersonforensics.co.uk/Files/NTFS%20compression%20white%20paper.pdf>

### [CARRIER05]

Title: File System Forensic Analysis  
Author(s): Brian Carrier  
Date: 2005  
ISBN-10: 0-321-26817-2

### [MIKUS05]

Title: An analysis of disc carving techniques  
Author(s): Nicholas A. Mikus  
Date: march 2005  
URL: <http://handle.dtic.mil/100.2/ADA432468>

### [RUSSON05]

Title: NTFS Documentation  
Author(s): Richard Russon, Yuval Fiedel  
Date: 2005  
URL: <http://linux-ntfs.org/>

### [DFRWS06]

Title: File Carving Challenge 2006  
URL: <http://www.dfrws.org/2006/challenge/>

### [DFRWS07]

Title: File Carving Challenge 2007  
URL: <http://www.dfrws.org/2007/challenge/>

### [KLOET07]

Title: Measuring and Improving the Quality of File Carving Methods  
Author(s): S.J.J. Kloet  
Date: August 2007  
URL: <http://sourceforge.net/projects/revit/files/Documentation/Master%20Thesis%20-%20Advanced%20File%20Carving/thesis.pdf/download>

### [Libpff08]

Title: The libpff project  
URL: <http://sourceforge.net/projects/libpff>

### [Microsoft09]

Title: [MS-OXMSG]: .MSG File Format Specification  
URL: <http://msdn.microsoft.com/en-us/library/cc463912.aspx>

### [ForensicWiki09]

Title: File Carving Taxonomy  
URL: <http://www.forensicswiki.org/wikwi/Carving>

## Appendix B. ReviveIT

```
#
-----
#
# ReviveIt 2007 (revit07) configuration file
#
# Creation date:      August 9, 2009
# Modification date: August 9, 2009
#
-----
#
#
-----
#
# Copyright (c) 2009, Joachim Metz <forensics@hoffmannbv.nl>,
# Hoffmann Investigations. All rights reserved.
#
#
# This software is free software: you can redistribute it and/or modify
# it under the terms of the GNU Lesser General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# (at your option) any later version.
#
# This software is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU Lesser General Public License
# along with this software. If not, see <http://www.gnu.org/licenses/>.
#
-----
#
#
-----
#
# Identifier:          NTFS-compressed OLE Compound File
# Category:            binary data
# Modification date:   August 9, 2009
# References:          ...
#
-----
#
file      : ntfs_compressed_olecf
{
    extension : inflated

    element   : compressed_olecf_signature
    {
        pattern : "\x00\xd0\xcf\x11\xe0\xa1\xb1\x1a"
    }

    layout
    {
        variable : size
        {
```

```

        value      : 2 little_endian
        bitmask    : 0x0fff
        add        : 1
    }
    compressed_olecf_signature
while size not_equals 0
{
    alignment : size
    variable   : size
    {
        value      : 2 little_endian
        bitmask    : 0x0fff
        add        : 1
    }
}
}

```

# Appendix C. GNU Free Documentation License

Version 1.1, March 2000

Copyright (C) 2000 Free Software Foundation, Inc. 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
---

## 0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other written document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

## 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you".

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not

"Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

## **2. VERBATIM COPYING**

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## **3. COPYING IN QUANTITY**

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## **4. MODIFICATIONS**



You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.
- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words

as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## **5. COMBINING DOCUMENTS**

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

## **6. COLLECTIONS OF DOCUMENTS**

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## **7. AGGREGATION WITH INDEPENDENT WORKS**

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

## **8. TRANSLATION**

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special

permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

## **9. TERMINATION**

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## **10. FUTURE REVISIONS OF THIS LICENSE**

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.