available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/diin

Digital Investigation

Data concealment and detection in Microsoft Office 2007 files

*Bora Park, Jungheum Park, Sangjin Lee**

Center for Information Security Technologies (CIST), Korea University, Anam-Dong, Seonbuk-Gu, Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 15 November 2007
Received in revised form
26 November 2008
Accepted 1 December 2008

Keywords:
Microsoft Office 2007 file
OOXML
Data concealment
Hidden data detection
Unknown part
Unknown relationship

ABSTRACT

As more offenders attempt to conceal incriminating data or stolen information, it is important for forensic examiners and computer security professionals to know where to look for concealed information. This paper demonstrates how data concealment in Microsoft Office 2007 files is possible. The Office Open XML (OOXML) format forms the basis of Microsoft Office 2007, and an individual can use OOXML to define customized parts, relationships, or both within a Microsoft Office 2007 file to store and conceal information. Fortunately for digital investigators, such concealed data can be detected by looking for the existence of unknown parts or relationships.

1. Introduction

The investigation of electronic documents is a critical aspect of computer forensics because such files frequently contain important information that can be used as a evidence of a crime. Although forensic examiners can often obtain significant information from electronic documents, data-concealment tools and techniques have emerged that can fully conceal or delete some data in these files, making it difficult to examine the true file contents. Terrorists can conceal communications, plans, and other data in electronic documents, and stolen data can be stored and transferred in hidden parts of an electronic document. Even applications designed to perform forensic examinations do not show hidden data, and therefore, forensic computer investigators need to use special techniques for detecting hidden data in electronic document files. Although this work is similar to "steganalysis," it is more accurately categorized as "document forensics" from the viewpoint of detecting and recovering hidden data in document files.

Microsoft (MS) Office files (*e.g.*, MS Word, PowerPoint, and Excel) are among the most popular electronic document types. Most new computer systems include MS Office products, particularly MS Word, PowerPoint, and Excel. For this reason, it is essential for forensic examiners to be able to detect concealed data in those files. From the viewpoint of data concealment, then, it is important not only to hide data completely but also to conceal the fact that the data have been hidden. MS Office files are a good choice as cover data for concealing information because the presence of such common files is unlikely to arouse an investigator's suspicion (Provos and Honeyman, 2003).

Several methods for hiding data in MS Office files have been suggested, but not specifically for Office 2007 files (Castiglione et al., 2007). The format of MS Office 2007 files is different from

* *Corresponding author.*
E-mail addresses: danver123@korea.ac.kr (B. Park), junghmi@korea.ac.kr (J. Park), sangjin@korea.ac.kr (S. Lee).
1742-2876/$ – see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2008.12.001

the format used by previous versions such as MS Office 1997–2003. MS Word, PowerPoint, and Excel 2007 files use the new file format referred to as Office Open XML (OOXML). On the basis of examination and testing of the MS Office 2007 file format, this paper demonstrates that it is in fact possible to hide data in an MS Office 2007 file. The key strategy for concealing data within an MS Office 2007 file is the creation of unnoticeable contents. These unnoticeable data can be created using the concept of "content relationship." If someone creates data which have no relationship with the main data contents of an MS Office file, these data are not shown on the MS Office screen display, as long as the data satisfy the minimal structural requirements of a normal MS Office file.

This paper presents in detail a method for concealing data in an MS Office 2007 file and provides an algorithm to detect hidden data in such a file. Once forensic examiners notice data that are not immediately visible within MS Office, they can reveal them using the approach outlined in this paper. Most of the examples for data concealment and detection in this paper focus on MS Word 2007 files, but the methods apply also to MS PowerPoint and Excel 2007 files.

The remainder of the paper is structured as follows. Section 2 introduces previous research and explains the need for this study. Section 3 explains the OOXML format. Section 4 focuses on a data concealment method based on the MS Office 2007 file format, OOXML. Section 5 describes the mechanism used by the "Detector" detection tool presented here to discover hidden data. Finally, Section 6 draws conclusions and describes work that remains to be done.

2. Related research on data concealment

Research studies of methods for concealing data in electronic documents using features of each electronic document's own native file format have included MS Office documents. For instance, MS Office files (*e.g.*, MS Word, PowerPoint, and Excel) from versions prior to Office 2007 (*i.e.*, 1997–2003) used the compound document file format defined by Microsoft (Rentz, 2007). Castiglione et al. describe a data concealment method using the compound document file format (Castiglione et al., 2007). Compound document files help to structure the contents of an electronic document. It is possible to divide the file data into several streams and to store these streams separately in the file. In this way, compound document files support a complete file system inside the file, where the streams are like files in a file system and the storage areas are like subdirectories (Rentz, 2007).

Compound document files consist of sectors and short sectors. A sector has a size of 512 bytes, and a short sector has a size of 64 bytes (Rentz, 2007). Because the files are saved as fixed sector and short-sector units, significant trash space (slack space in allocated sectors) and many empty sectors (unallocated sectors) are created (Castiglione et al., 2007). This trash space and empty sectors can be used to hide data in MS Office files. The concealment process involves four steps:

1. The size of a document's trash space is computed.
2. A secret message is compressed and encrypted with its header.
3. If necessary, empty sectors are added to the process.
4. Secret messages are concealed in trash space and empty sectors.

Castiglione et al. presented the first work on compound document files to be written from the viewpoint of computer forensics and steganography. The compound document file, similar to an MS Office document file, stores much more data than that which a user would save. Information derived from these data may be relevant to a forensic investigation, but proper tools and knowledge are required to retrieve and interpret this information.

All the space containing hidden information, along with the space wasted by MS Office structured storage may be useful for steganographic purposes. Castiglione et al. have introduced a tool, "StegOle," that takes advantage of this space to conceal messages in MS Office document files.

Data-hiding techniques involving XML have also been presented in past work. Inoue et al. suggested data concealment methods within XML files and noted that few researchers have looked into methods of hiding data in electronic document files (Inoue et al., 2001). In applications involving XML data exchange or XML web pages, secret data can be embedded into XML modules without changing the original contents. These methods can easily be translated to existing XML document files. Inoue et al. proposed five methods for data concealment in XML files: (1) representation of empty elements; (2) white spaces in tags; (3) use of the apparent order of the elements; (4) use of the apparent order of attributes; and (5) elements containing other elements. These methods involve mainly changing the order of XML attributes or creating empty space between the XML element name and the "/" as a mechanism for hiding data. These data-concealment algorithms may be somewhat simple, but their data-concealment capacity is very small because this algorithm is actually intended for text concealment. Because the data to be hidden in most contexts are not limited to text data, instead of using the concealment method suggested in (Inoue et al., 2001), the present work uses the XML schema for hiding data.

The XML schema defines what information may exist in the data flow, where it may appear, and how it should be used. In addition, the XML schema establishes a vocabulary by which the user and the application (or another user) can exchange information (Goldfarb and Prescod, 2001). In other words, an XML Schema reflects the demands of the application. Because the MS Office 2007 file format uses a specified XML format, MS Office 2007 files adhere to the XML schema, and therefore it is possible to hide data in MS Office 2007 files using a specified OOXML schema. To hide data in MS Office 2007 files, the present research tries to manipulate certain aspects of the OOXML schema while maintaining essential elements so that MS Office 2007 files will still open normally.

3. Microsoft Office 2007 file format

MS Office 2007 files consist of a series of compressed component parts that are stored in a container called a package, and each decompressed package conforms to the OOXML file

format. A package is an ordinary Zip archive, which contains that package content-type item, relationship items, and parts (ECMA International, 2006). For each package, there is a package-relationship Zip item that contains information about the relationship between a package and its parts. Similarly, there are part-relationship Zip items that contain information about relationships between the various parts of the document. In short, these packages define the overall structure of the document.

3.1. *OOXML format*

The formats of OOXML files are distinguishable from each other, except for the specific contents of the files, because of Open Packaging Conventions (OPCs). OPC is one of the method for storing packed contents in various forms (xml, image, metadata, etc.) and for expressing a document file completely. The most recommended format for OPC is a Zip archive.

As shown in Fig. 1, an OOXML file is based on the following:

- Package: ZIP archive.
- Part: files in ZIP archive.
- Relationship: the relationships between the parts and package or among parts.

3.1.1. *Packages*

A package is a Zip container that holds the components (also called parts) that comprise the document, as defined by OPC specifications. Many of these elements also exist in an MS Office 2007 file. Some of them are shared across all MS Office applications, for example, document properties, charts, style sheets, hyperlinks, diagrams, and drawings. Certain other elements are specific to each application, such as worksheets in Excel, slides in PowerPoint, or headers and footers in Word. When users store a document using an MS Office 2003 application or a previous version of the MS Office application, a single file is written to the disk and can be easily opened by the user. This metaphor is important in terms of understanding how documents are stored, managed, and shared in practice. Because the individual parts of an MS Office 2007 system file are wrapped in a Zip container, a document remains a single file instance. The use of a single package file to represent a single document entity enables users to have the same experience they did with previous versions of MS Office applications when storing and opening Office 2007 files—that is, they can continue to work with a single file (Rice, 2007).

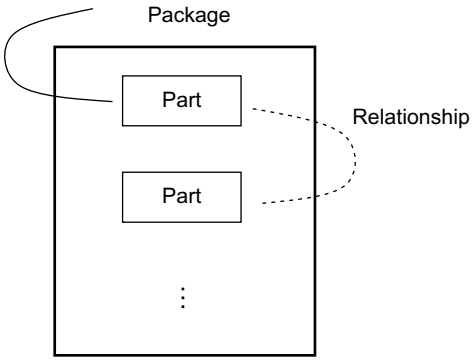

Fig. 1 – Office Open XML format.

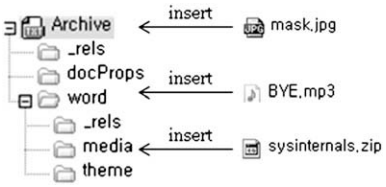

Fig. 2 – Example of data hiding.

3.1.2. *Parts*

The component parts of an MS Office document correspond to one file in a package. For example, if a user right clicks on an Excel 2007 file and chooses to extract it, he or she sees several files, such as a ''workbook.xml'' file and several ''sheetn.xml'' files. Each of those files is a part of the package (Rice, 2007).

Each part can have a different content type. Parts used to describe data created by MS Office 2007 applications are stored as an XML file. These parts conform to the XML schema that defines the associated MS Office 2007 file features or objects. All XML schemas that represent default MS Office 2007 file parts are fully documented and made available as part of MS Office.

3.1.3. *Relationships*

A relationship is a method that specifies how a particular collection of parts comes together to form a document. This method specifies the connection between a source part and a target resource. Relationships are stored within XML parts in the document package (*e.g.*, _rels\rels) (Rice, 2007).

For example, through a relationship, a user can identify the connection between a slide and an image that appears on that slide. Relationships are stored within XML parts or ''relationship parts'' in the package. If a source part has multiple relationships, all subsequent relationships are listed in the same XML relationship part.

Relationship files play an important role in MS Office XML formats. Every document part is referred to by at least one relationship. The use of relationships makes it possible to discover how one part relates to another part without looking at the content of the parts. Within parts, all references to relationships are represented using a *Relationship ID*, which enables all connections between parts to remain free of content-specific schemata. Relationship files contain relationships based on the starting part of the document (Rice, 2007). Relationships are defined in the following format:

<Relationship Id="ID" Type="relaionshipType" Target="targetPart" (Targetmode ="Internal/External")>

where the *ID* value can be any string, as long as it is unique in the ''.rels'' file. The package requires a valid XML identifier. The *Type* of the relationship distinguishes relationships from one another and points to the schema that defines OOXML format types. In addition, the *Target* points to the path containing the target of the relationship. Target attribute values are dependent on the *TargetMode*'s attribute value. The

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="…">
<Override PartName="/word/footnotes.xml" ContentType="…"/>

<DefaultExtension="jpeg" ContentType="image/jpeg"/>
<DefaultExtension="rels" ContentType="application/vnd.openxmlformats-Package.relationship+xml"/>
<DefaultExtension="xml" ContentType="application/xml"/>

<Override PartName="/word/document.xml" ContentType="…"/>
                    ……
</Types>
```

Fig. 3 – [Content_Types].xml.

TargetMode indicates whether the target describes a resource inside the package or outside the package. Valid values are "internal" and "external." If *TargetMode* is "internal," the target attribute will be a relative reference that is interpreted relative to the "parent" part. For package relationships, the package implementer resolves relative references in the target attribute against the pack Uniform Resource Identifier (URI) that identifies the entire package resource. If *TargetMode* is "external," the target attribute may be either a relative reference or an URI. If the target attribute is a relative reference, then that reference is interpreted relative to the location of the package (ECMA International, 2006).

A relationship in OOXML format is either explicit or implicit. In the case of an explicit relationship, a resource is referenced from a source part's XML using the ID attribute of the *Relationship* tag. For example, a document part can have a relationship to a hyperlink only if the ID attribute value of that hyperlink relationship element is referenced explicitly by the document part's XML. Because this mechanism is used generically across multiple tag types, explicit relationships can be extracted from an OOXML document without prior knowledge of tag semantics. Certain relationships are defined as explicit relationships; all other relationships are implicit relationships (ECMA International, 2006).

To illustrate, an example relationship in OOXML is provided here:

Suppose that a particular image file is saved as an MS Word 2007 file. If the image is subsequently renamed and placed in the Media folder in a "document.xml" file, the following XML code is generated:

```
<w:pict>
   <v:image data r:Id= "rId4"/>
</w:pict>
```

This XML code indicates that this document file contains an image whose ID is "rId4." When an application comes across this code, it attempts to find the target whose ID is "rId4." In a relationship file, ID information is listed for each target file. In this case, the following XML code appears in the relationship file:

```
<Relationships…>
   <Relationship Id= "rId4" Type = "/relationships/image"
   Target= "media/image1.jpeg"/>
</Relationships>
```

With this code, the application can find the target "rId4" and insert the image in the document file.

This is a simple example of an explicit relationship. More complex connections occur in the case of implicit relationships.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types xmlns="…">
<Override PartName="/word/footnotes.xml" ContentType="…"/>

<DefaultExtension="jpeg" ContentType="image/jpeg"/>
<DefaultExtension="rels" ContentType="application/vnd.openxmlformats-Package.relationship+xml"/>
<DefaultExtension="xml" ContentType="application/xml"/>
<DefaultExtension="zip" ContentType="application/zip"/>
<DefaultExtension="mp3" ContentType="application/mp3"/>
<DefaultExtension="jpg" ContentType="application/jpg"/>

<Override PartName="/word/document.xml" ContentType="…"/>
                    ……
</Types>
```

Fig. 4 – Modified [Content_types].xml.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="…">
<Relationship Id="rId3" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties"
Target="docProps/app.xml"/>
<Relationship Id="rId2" Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties"
Target="docProps/core.xml"/>
<Relationship Id="rId1" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
Target="word/document.xml"/>
</Relationships>
```

Fig. 5 – Relationship file.

4. Data concealment

4.1. *Data concealment using OOXML format*

Although the OPC specification is designed for representation of OOXML documents, it can also support a much broader range of applications. The use of some OPC features is restricted within OOXML documents (ECMA International, 2006). In particular, there are unknown parts and unknown relationships that make it possible to hide data.

4.1.1. *Unknown parts*

With the exception of relationship parts, all other parts of an OOXML document that are not the target of a valid relationship are treated as unknown parts (ECMA International, 2006). Unknown parts are ignored when a document is read and can, but need not, be discarded once created. In other words, these parts normally exist in any given file, but cannot be identified by MS Office applications.

4.1.2. *Unknown relationships*

Any relationship not defined within the ECMA376 Standard is considered to be an unknown relationship. Unknown relationships are valid within an OOXML document, provided that they conform to relationship markup guidelines as defined by the OPC specifications (ECMA International, 2006). Files containing unknown relationships open normally, and the unknown relationships never disappear, even if a user saves these files under new names.

4.1.3. *Data concealment using unknown parts and unknown relationships*

This section provides specific examples of data concealment. Inserting data into the carrier archive file is the first step, as shown in Fig. 2. After decompression of an MS Word ".docx" file, three files were inserted into the extracted folders: "mask.jpg" into the main folder, "BYE.mp3" into the word folder, and "sysinternals.zip" into the media folder.

Fig. 3 shows the "[Content_Types].xml" file associated with the extracted MS Word document, with all the file extensions in a normal package highlighted in bold. Because each extension of every part in the package must be stated in "[Content_Types].xml" with its related path, an individual who wants to conceal data whose extension is not already present in "[Content_Types].xml" must add some code. In this example, the extensions of the concealed data are "jpg," "mp3," and "zip," which are not included in "[Content_Types].xml." Therefore, it is necessary to insert code that defines these extensions and their associated paths, as shown in Fig. 4 with the added code highlighted in bold.

After "[Content_Types].xml" has been modified, the "BYE.mp3," "mask.jpg," and "sysinternals.zip" files become unknown parts of the MS Word document. A document file containing these hidden files (in this example, an MS Word 2007 file) opens normally. Thus these three files are completely hidden. At this point, if relationships between an MS Word 2007 file and hidden data are defined, discovering the hidden data becomes even more difficult. In other words, even when the user stores the data in a new file, the hidden data still remain in the file. One approach to eliminate the hidden data is

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships xmlns="…">
<Relationship Id="rId3" Type="…" Target="docProps/app.xml"/>
<Relationship Id="rId2" Type="…" Target="docProps/core.xml"/>
<Relationship Id="rId1" Type="…" Target="word/document.xml"/>

<Relationship Id="rId100" Type="http://schemas.openxmlformats.org/officeDocument/2006/Relationships/a"
Target="word/media/sysinternals.zip"/>
<Relationship Id="rId101" Type="http://schemas.openxmlformats.org/officeDocument/2006/Relationships/b"
Target="mask.jpg"/>
<Relationship Id="rId102" Type="http://schemas.openxmlformats.org/officeDocument/2006/Relationships/c"
Target="word/BYE.mp3"/>

</Relationship>
```

Fig. 6 – Modified relationship file.

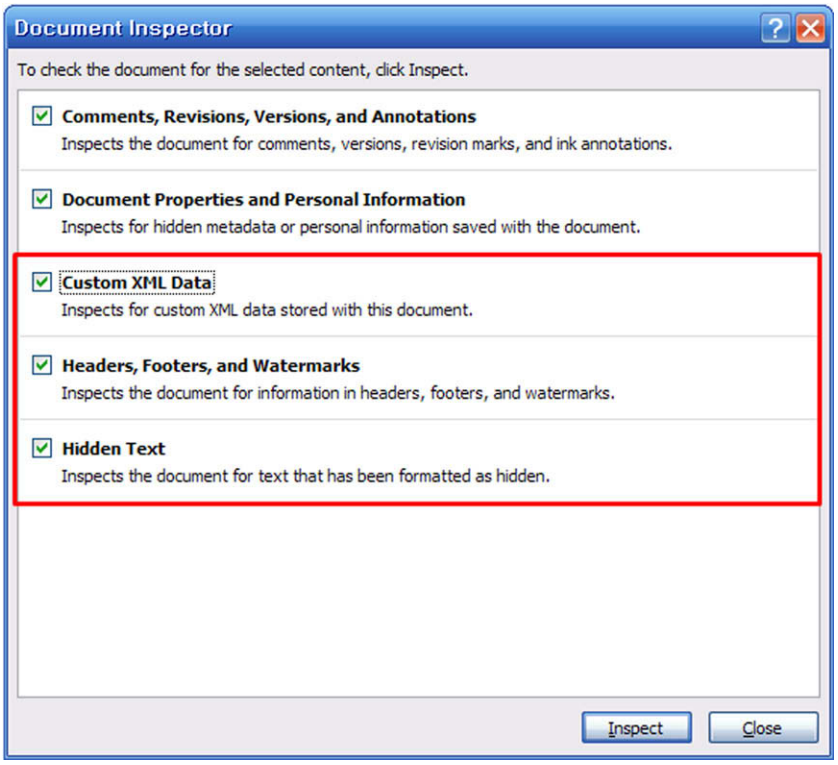

Fig. 7 – Document inspector.

to remove the "jpg," "mp3," and "zip" extensions from "[Content_Types].xml" and to save the document as a new file.

Fig. 5 shows a normal relationship file, which has a ".rels" extension, as extracted from an MS Office document. Within this relationship part, attributes such as ID, Type, and Target ID must be unique, and the ID connects the document and the target files. The document part's XML explicitly references the ID attribute value (ECMA International, 2006).

Fig. 6 shows the relationships involving hidden data. For example, the relationship of "BYE.mp3" is as follows: the ID of "BYE.mp3" (in the "word" folder) is "rId102," and the *Type* of this ID is "http://schemas.OpenXMLformats.org/officeDocument/2006/relationships/c." At this point, if a user changes the *Type* to one that is stated in an OOXML specification, a warning appears when the document file is opened. Therefore, when setting a *Type*, a user needs to use a value that does not exist in the OOXML specifications (*e.g.*, "a," "b," "c," as shown in Fig. 6).

After the ".rels" file and the "[Content_Types].xml" file have been modified, the document can be opened normally by a user without any warnings. Furthermore, if the user modifies the document file with an MS Word 2007 application and saves it again, the hidden data remain in the file. Because MS Office 2007 applications do not check unknown parts and unknown relationships, it is possible to use them to hide data.

This data-hiding method is the natural result of an explicit relationship in OOXML. The main source part of a document relies on the unique ID value of each component part and associated information in the relationship part to locate content such as embedded files. The key point of this hiding process is that assigning a new ID to a new target results in the target being overlooked by the MS Office application. Because the new ID is not referenced in the relationship part, the main source part is not aware of the new content, and the hidden data are not shown on-screen. However, the hidden data will not be eliminated by MS Office because these hidden data have an ID. The MS Office 2007 application does have a feature to "detect unknown.xml," as shown in Fig. 7. However, if someone hides data using the method described in this paper, this function does not detect the hidden data.

4.1.4. *Data concealment using comments*

It is also possible to hide data within an MS Office document using XML comments. XML parts in normal files created by MS Office 2007 applications do not have XML comments. This method is very simple, but this concealment method works for hiding messages as well.

5. Detection of hidden data

This section provides the algorithm and pseudocode for detecting data that have been hidden using techniques detailed in this paper.

As described above, using unknown parts and unknown relationships in an MS Office 2007 file, it is possible to hide data in the file. The properties of unknown parts and unknown relationships can also be used for detecting hidden

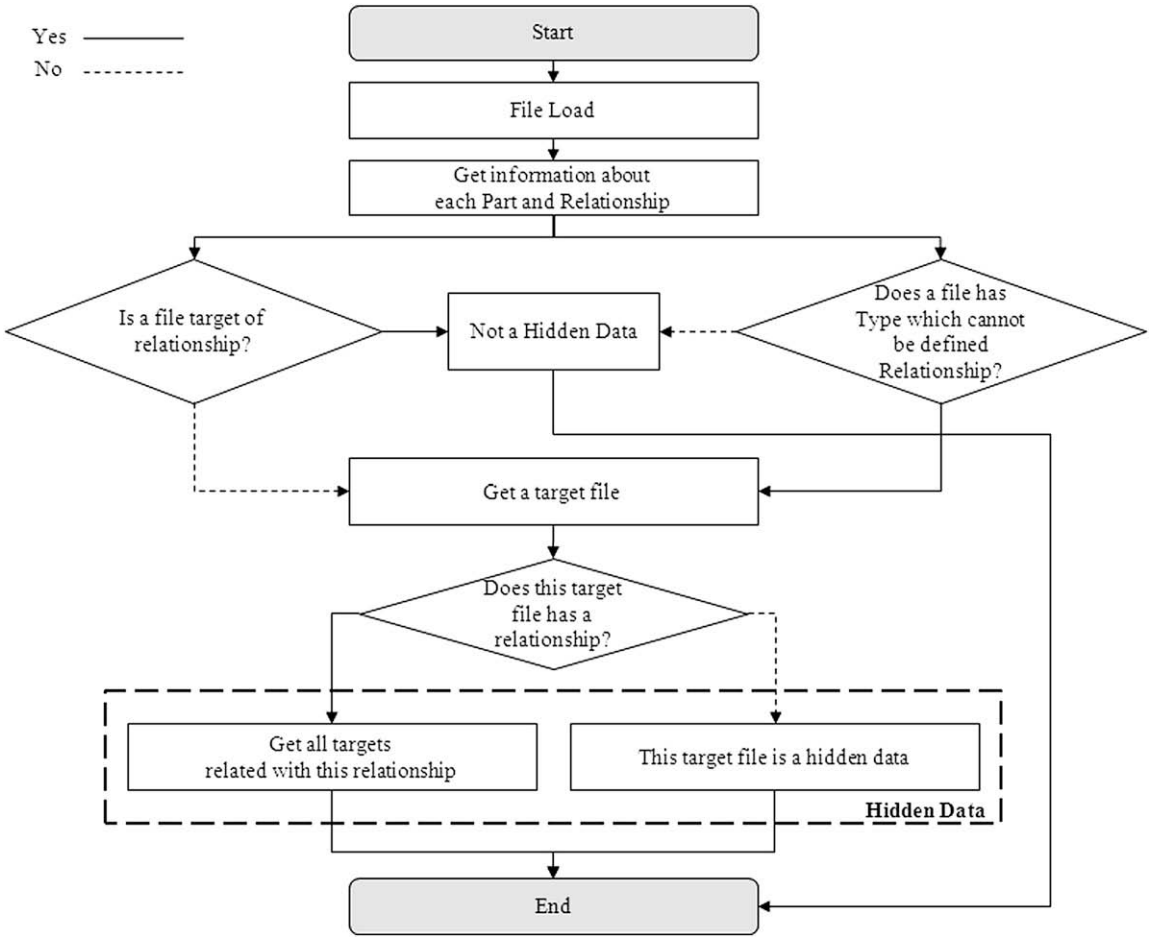

Fig. 8 – Hidden-data detection algorithm.

data. Fig. 8 shows a summary of the detection algorithm. A detailed explanation of this algorithm is provided in the next paragraph, with pseudocode in Figs. 10–12.

After a file containing hidden data is loaded, the ''Detector'' detection program (developed using C#) collects information about each part and each relationship in the file. Information about parts is stored in the ''SetFileInfoList'' structure:

```
struct FileInfo
{
        public bool IsUnknownPart;
        public bool IsUnknownRel;
        public bool HasComment;
        public string DirPath;
        public string FileName;
        public string comment;
}
```

''IsUnknownPart'' is a variable which indicates whether or not a particular part is unknown. In other words, this variable indicates whether or not each part is defined (ECMA International, 2006). A part which does not have an unknown relationship can be detected simply by the value of the ''IsUnknownPart'' variable. ''IsUnknownRel'' is a variable which indicates whether or not a particular part is an unknown relationship. If the input part has an undefined type, then this detection algorithm determines that this part has an unknown relationship. ''HasComment'' is ''TRUE'' when the input part has a comment and ''FALSE'' otherwise. ''DirPath'' is the absolute path of the input part, and ''FileName'' is the name of the part. A comment, if detected, is stored in the ''comment'' variable. Similarly, relationship information is collected and processed.

```
Detect
{
    Detect.FileLoad
    Detect.ExtractZip

    Detect.SetFileInfoList
    Detect.SetRelationshipList

    Detect.FindUnknownParts
    Detect.FindUnknownRel_from_Root("_rels\\.rels")
    Detect.FindUnknownRel_from_UnknownPart
    Detect.FindXMLComments
}
```

Fig. 9 – Pseudocode 1.

```
Detect.FindUnknownParts
{
    foreach (File in Package)
    {
        Get FileInfo of this file in FileInfoList
        Assume that this file is Unknown Part

        foreach (Relationships in RelationshipList)
        {
            foreach (Relationship in Relationships)
            {
                if (path of this file == path of Relationship.target )
                {
                    This file is not Unknown Part
                    break
                }
            }
        }

        if (This file is Unknown Part)
        {
            FileInfo.IsUnknowsPart = TRUE
        }
    }
}
```

Fig. 10 – Pseudocode 2.

After this data-collection work is complete, the following two steps are performed:

1. Check to see whether each file is the target of a specific relationship stated in the relationship file. If not, it consists of hidden data. If these hidden data have a relationship file, all targets in this relationship file consist of hidden data.
2. Check to see whether each file has an abnormal content type that is not described in the OOXML specifications. If so, this file contains hidden data. If this file has a relationship file, all targets in this relationship file also contain hidden data.

The pseudocode for this algorithm will now be described and the detection algorithm is presented in detail.

Fig. 9 shows the order of the detection process. After loading the package file (*e.g.*, an MS Word 2007 file), the detection program extracts the package and collects information about each part and relationship part.

Figs. 10–12 show how unknown parts and unknown relationships are detected in a document file.

The detection process is performed in three steps:

First, the program checks each part to determine whether or not it is a target of a relationship part. If not, then the part is an unknown part—that is, it is hidden data. Furthermore, if the part has a relationship part in its subfolder (where the name of the subfolder is '_rels'), the relationship part and all targets of the relationship part are hidden data. This process continues recursively.

```
Detect.FindUnknownRel_from_Root (rels_path)
{
    foreach (Relatioinship file in Package)
    {
        Get Relationships of this relationship file in RelationshipList

        if (Relationships.rels_path != rels_path)  break

        foreach (Relationship in Relationships)
        {
            if (Type and Target of Relationship are not in Known_Relationships_in_MS_Word_2007)
            {
                This relationship is Unknown Relationship

                Get FileInfo of Relationship.target in FileInfoList
                FileInfo.IsUnknowsRel = TRUE

                if (Relationship.target has sub relationship file)
                {
                    FindUnknownRel_from_Root (sub_rels_path)
                }
            }
            else
            {
                This relationship is not Unknown Relationship
            }
        }
    }
}
```

Fig. 11 – Pseudocode 3.

```
Detect.FindUnknownRel_from_UnknownPart
{
    foreach (FileInfo in FileInfoList)
    {
        if (FileInfo.IsUnknownPart == TRUE)
        {
            if (relationship file of this file is exist)
            {
                Relationships in this relationship file are Unknown Relationship
                FindUnknownRel_from_Root (path of this relationship file)
            }
        }
    }
}
```

Fig. 12 – Pseudocode 4.

Second, the program must check the type of each relationship part to determine whether or not it is defined in the OOXML specification. If the type of a relationship part is not defined in the OOXML specification, the relationship part is an unknown relationship, that is, it is a hidden data. For example, in the case of ".docx" files (MS Word 2007 files), the types of a relationship part consist of:

"http://schemas.OpenXMLformats.org/officeDocument/2006/relationships/officeDocument"
"http://schemas.OpenXMLformats.org/officeDocument/2006/relationships/extended-properties"
"http://schemas.OpenXMLformats.org/package/2006/relationships/metadata/core-properties"
"http://schemas.OpenXMLformats.org/package/2006/relationships/digital-signature/origin"
"http://schemas.OpenXMLformats.org/package/2006/relationships/metadata/thumbnail"

and so on. These types are in the OOXML specification for the relationship parts.

Figs. 13–15 show the detection results for hidden data. In the example used earlier in this paper, the hidden data are "mask.jpg," "BYE.mp3," and "sysinternals.zip," all represented in Fig. 5. The hidden data are represented by icons that differ from those used for normal data. The program has verified that "mask.jpg" is hidden in the root folder, "BYE.mp3" is hidden in the "word" folder, and "sysinternals.zip" is hidden in the "word\media" folder.

Finally, the program must check each part in the XML format to determine whether or not it has an XML comment. All parts in XML format that are normally created by MS Office 2007 applications do not have XML comments. Therefore, if an XML comment exists in an MS Office 2007 file, it can be considered to be a hidden data inserted by a user. Fig. 16 shows the results of this inquiry, including the XML comment detection algorithm.

6. Conclusions and discussion

Except for the occurrence of data loss when converting an MS Office 2007 document from the OpenXML format to the compound format (used by MS Office versions 2003 and prior), the data-concealment method outlined here is very powerful, especially given the fact that the hidden data cannot be detected by any of the functions supported by MS Office 2007 applications. The Microsoft Office Isolated Conversion Environment (MOICE) (Microsoft Corporation, 2007) uses "document.xml" and parts and relationships related to "document.xml" for reading data from an MS Office application from 2003 or prior versions. For this reason, when opening an MS Office 2007 file with MOICE in an older application, hidden data that are not related to "document.xml," such as data generated by the method presented here (which does not modify "document.xml"), disappear after compound reformatting or viewing in the 2007 version of the application. In other words, MOICE

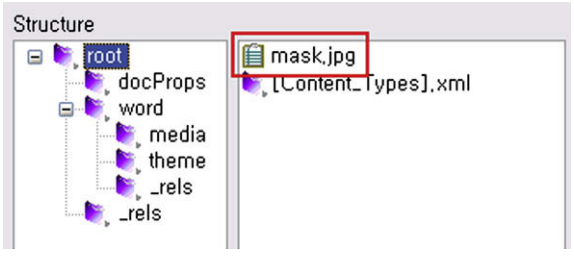

Fig. 13 – Results of detection 1.

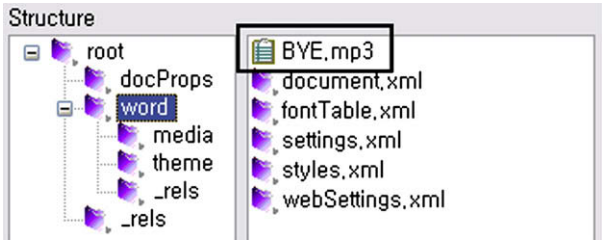

Fig. 14 – Results of detection 2.

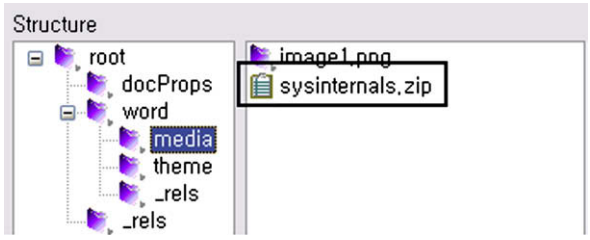

Fig. 15 – Results of detection 3.

cannot examine unused data such as unknown parts or unknown relationships and cannot do any validation testing. Instead, it can only convert data related to "document.xml." It can, therefore, be concluded that there is no way to detect data concealment as presented here except by the detection algorithm described in this paper.

Considering that steganography deals with hiding messages in cover data or hiding the fact that certain data are being transferred, if it is possible to insert data into a file without causing notification when the file is opened, this method of hiding data is critical to steganography. The results from this study lead to the conclusion that data concealment in MS Office 2007 files is possible, and it has just been shown how such hidden data can be detected. This paper describes the creation of tools that can detect data potentially hidden using OOXML format.

From the viewpoint of computer forensics, it is very important to confirm the existence of any data that are not examined by specific applications. In computer forensic investigations, investigators may assume that all data in electronic document files can be examined by their applications. However, it is not enough to investigate electronic document files only through their associated applications, because data that cannot be examined or detected by most applications could nonetheless exist in the file.

In this paper, most of the descriptions focus on MS Word 2007 files. However, in the case of MS Office PowerPoint and Excel 2007 files, if they contain any hidden data, these data can also be detected using the same algorithm.

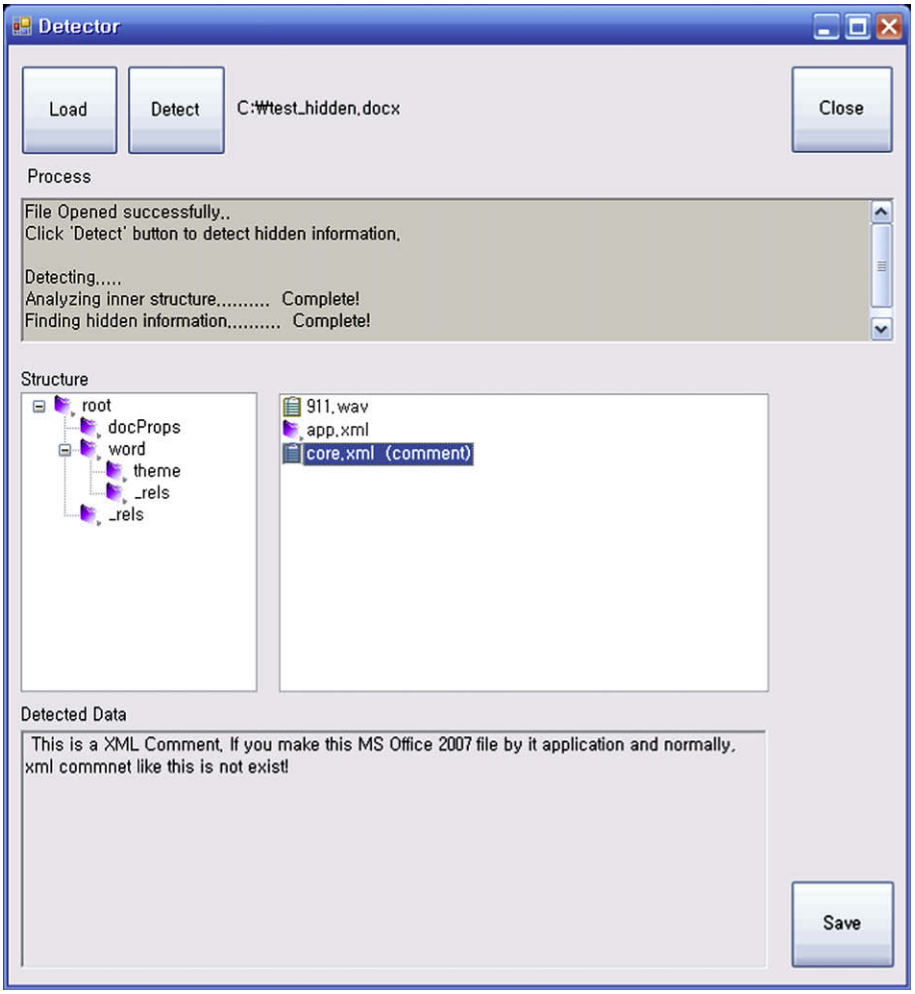

Fig. 16 – Final detection results using "Detector."

In the future, the authors intend to find methods for hiding data in files other than those studied here and to create tools for detecting hidden data within them.

Acknowledgments

This work was supported by the IT R&D program of MKE/IITA [2007-S019-02, Development of Digital Forensic System for Information Transparency].

REFERENCES

Provos N, Honeyman P. Hide and seek: an introduction to steganography. IEEE Security and Privacy 2003. May–June 2003.

Castiglione A, De Santis A, Soriente C. Taking advantage of a disadvantage: digital forensics and steganography using document metadata. Journal of Systems and Software 2007; 80(5):750–64.

Rentz D. OpenOffice.org's documentation of the microsoft compound document. URL: http://sc.openoffice.org/compdocfileformat.pdf; 2007. The Spreadsheet Project, OpenOffice.org, June 2007.

Inoue S, Makino K, Murase I, Takizawa O, Matsumoto T, Nakagawa H. A proposal on information hiding methods using XML. 1st workshop on NLP and XML, Nov. 2001; 2001. Available online at: <http://takizawa.ne.jp/nlp_xml.pdf>.

Goldfarb C, Prescod P. The XML handbook. 3rd ed. Prentice-Hall; 2001. Professional Technical Reference Series, p. 499.

ECMA International. Office open XML file formats, Part 1. ECMA-376. 1st ed.; 2006. December 2006, p. 11.

Rice F. Open XML file formats. Microsoft Corporation. URL: http://msdn2.microsoft.com/ko-kr/library/aa338205.aspx; 2007. May 2006.

Microsoft Corporation. Description of the microsoft office isolated conversion environment update for the compatibility pack for word, excel, and powerpoint 2007 file formats. URL: http://support.microsoft.com/kb/935865; 2007. October 2007.