US 20070118819A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2007/0118819 A1

Basin et al. (43) Pub. Date: May 24, 2007

(54) SYSTEMS AND METHODS FOR MANIPULATING AND MANAGING COMPUTER ARCHIVE FILES

(76) Inventors: **Yuri Basin**, Thiensville, WI (US); **Michael J. Beirne**, Chicago, IL (US); **James C. Peterson**, Menomonee Falls, WI (US); **Karen L. Peterson**, Menomonee Falls, WI (US)

Correspondence Address:
MCANDREWS HELD & MALLOY, LTD
500 WEST MADISON STREET
SUITE 3400
CHICAGO, IL 60661

(21) Appl. No.: **11/586,834**

(22) Filed: **Oct. 26, 2006**

Related U.S. Application Data

(63) Continuation of application No. 10/970,897, filed on Oct. 21, 2004, which is a continuation of application No. 09/803,355, filed on Mar. 9, 2001, now Pat. No. 6,879,988.

(60) Provisional application No. 60/187,969, filed on Mar. 9, 2000.

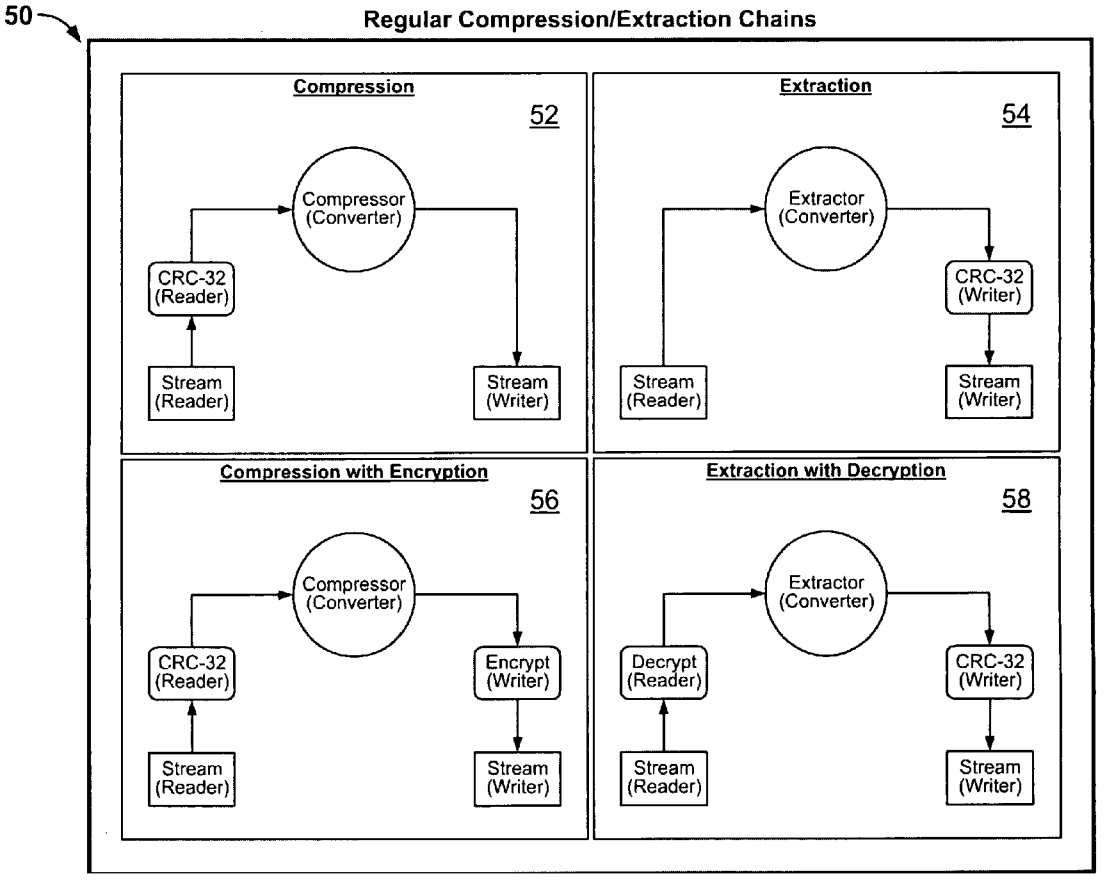
Publication Classification

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** **715/847**; 707/200

(57) **ABSTRACT**

A computer program for managing and manipulating archive zip files of a computer. The program includes a system and method for opening, creating, and modifying, and extracting zip archive files. The program is fully integrated into Microsoft Windows Explorer and is accessed via Explorer menus, toolbars, and/or drag and drop operations. An important feature of the program is the archive manager which may be used to open a zip file, create a new zip file, extract zip files, modify zip files, etc. The program is integrated into Microsoft Windows Explorer using the shell name space extension application program interface developed by Microsoft.



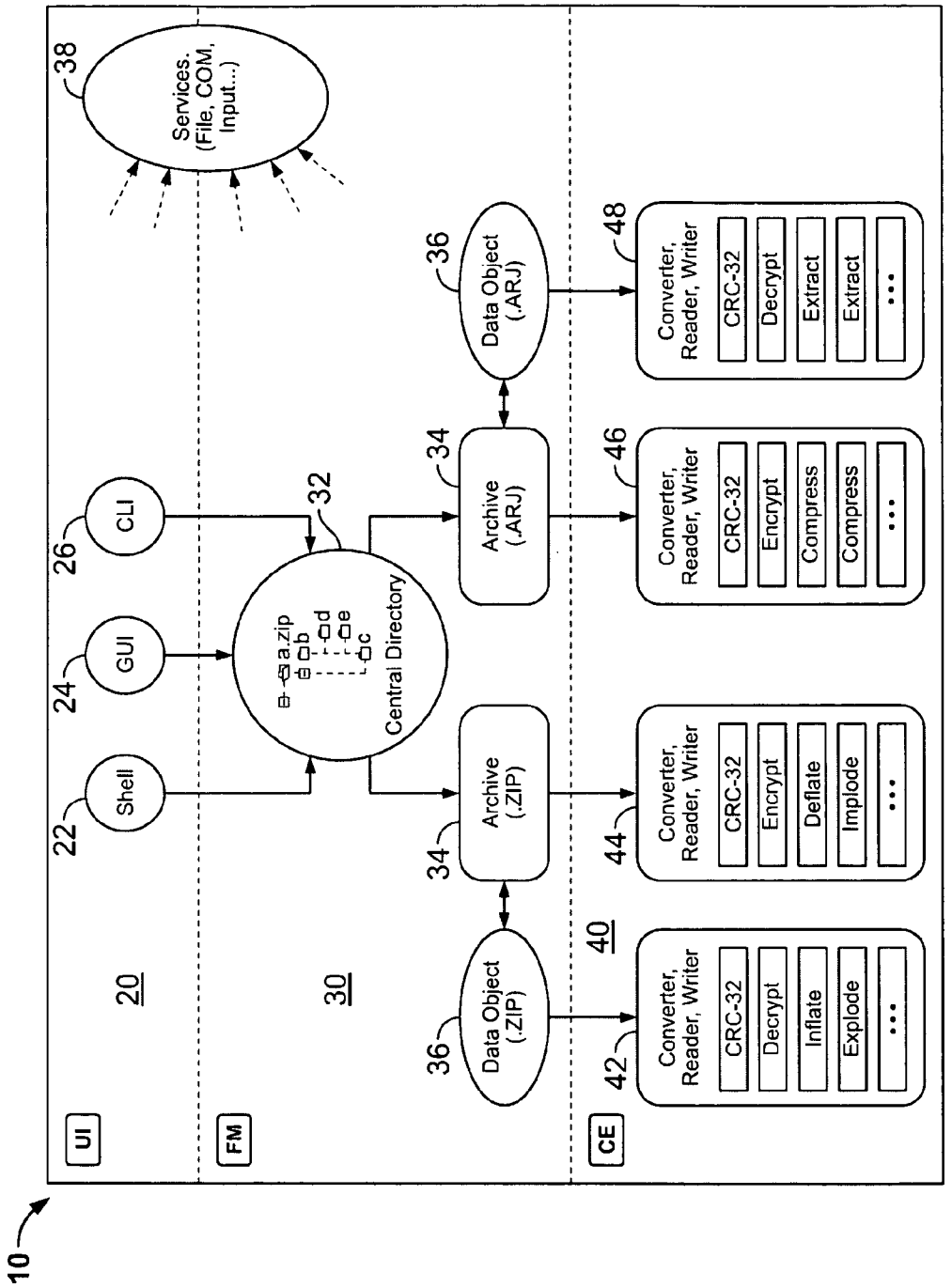

10
UI
20
FM
30
CE
40
Shell
22
GUI
24
CLI
26
Services, (File, COM, Input...)
38
Central Directory
a.zip
b
d
e
c
32
Archive (.ZIP)
34
Archive (.ARJ)
34
Data Object (.ZIP)
36
Data Object (.ARJ)
36
Converter, Reader, Writer
CRC-32
Decrypt
Inflate
Explode
...
42
Converter, Reader, Writer
CRC-32
Encrypt
Deflate
Implode
...
44
Converter, Reader, Writer
CRC-32
Encrypt
Compress
Compress
...
46
Converter, Reader, Writer
CRC-32
Decrypt
Extract
Extract
...
48

FIG. 1

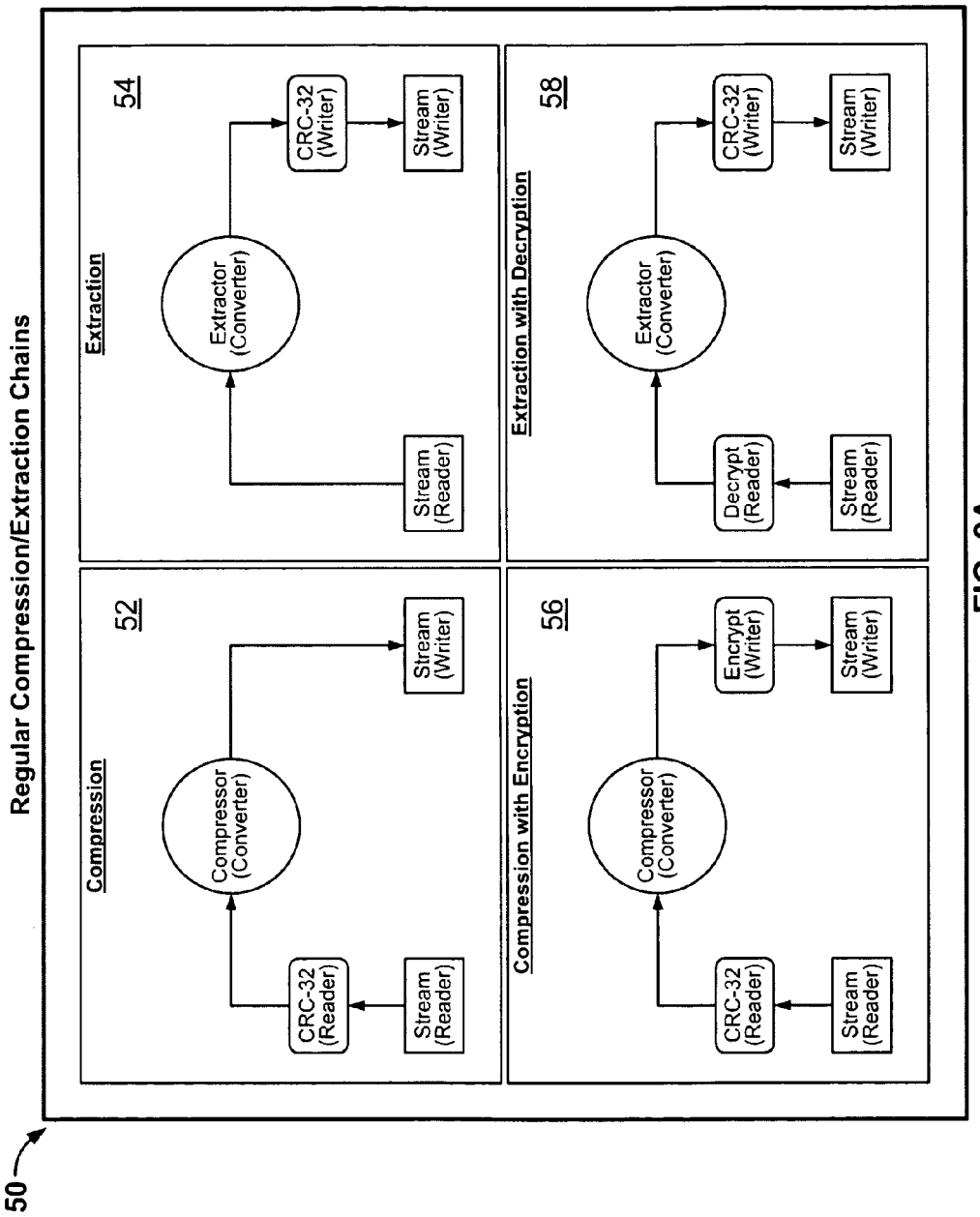

50
Regular Compression/Extraction Chains
Compression
52
Compressor (Converter)
CRC-32 (Reader)
Stream (Reader)
Stream (Writer)
Extraction
54
Extractor (Converter)
Stream (Reader)
CRC-32 (Writer)
Stream (Writer)
Compression with Encryption
56
Compressor (Converter)
CRC-32 (Reader)
Stream (Reader)
Encrypt (Writer)
Stream (Writer)
Extraction with Decryption
58
Extractor (Converter)
Decrypt (Reader)
Stream (Reader)
CRC-32 (Writer)
Stream (Writer)

FIG. 2A

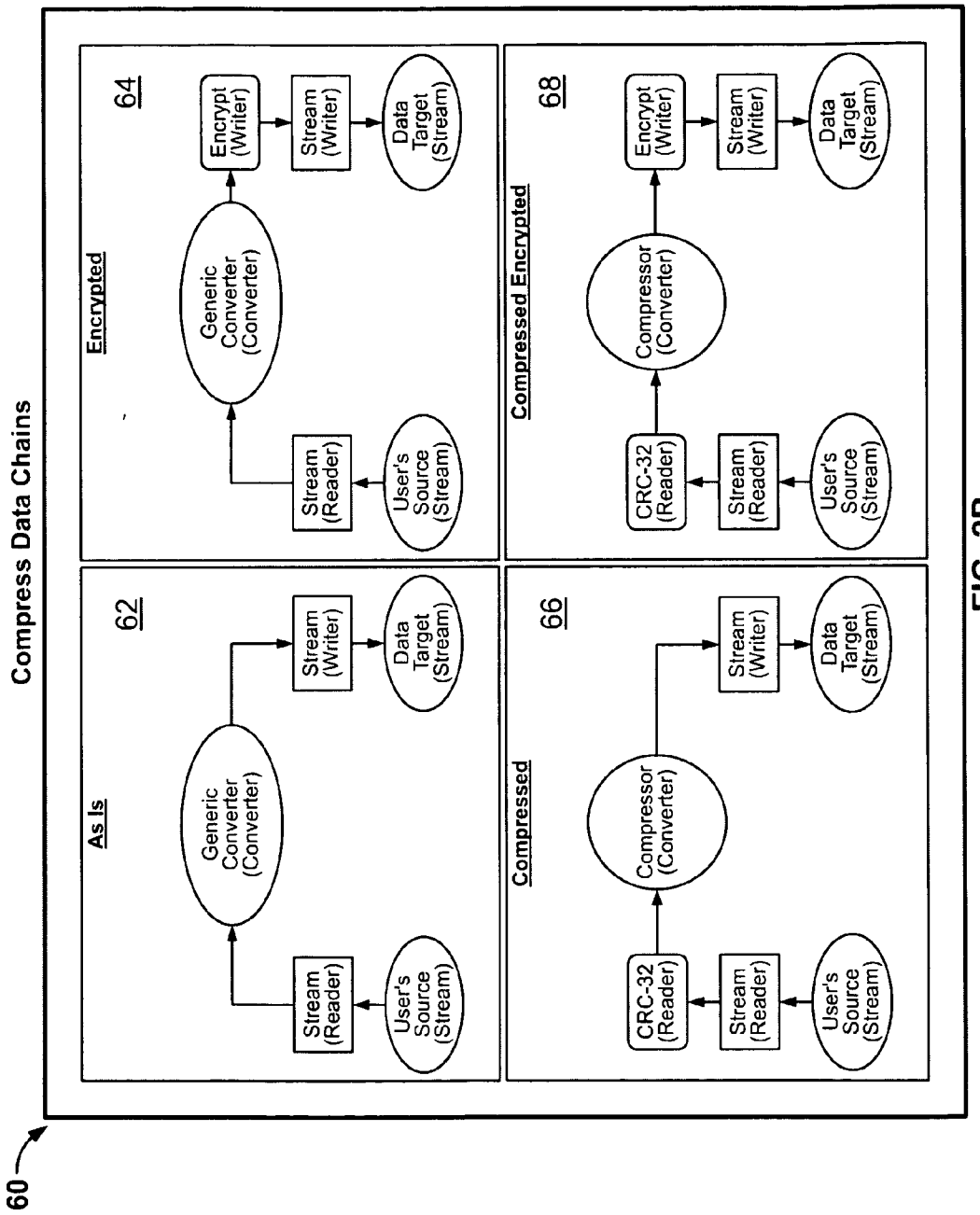

60
Compress Data Chains
As Is
62
User's Source (Stream)
Stream (Reader)
Generic Converter (Converter)
Stream (Writer)
Data Target (Stream)
Encrypted
64
User's Source (Stream)
Stream (Reader)
Generic Converter (Converter)
Encrypt (Writer)
Stream (Writer)
Data Target (Stream)
Compressed
66
User's Source (Stream)
Stream (Reader)
CRC-32 (Reader)
Compressor (Converter)
Stream (Writer)
Data Target (Stream)
Compressed Encrypted
68
User's Source (Stream)
Stream (Reader)
CRC-32 (Reader)
Compressor (Converter)
Encrypt (Writer)
Stream (Writer)
Data Target (Stream)

FIG. 2B

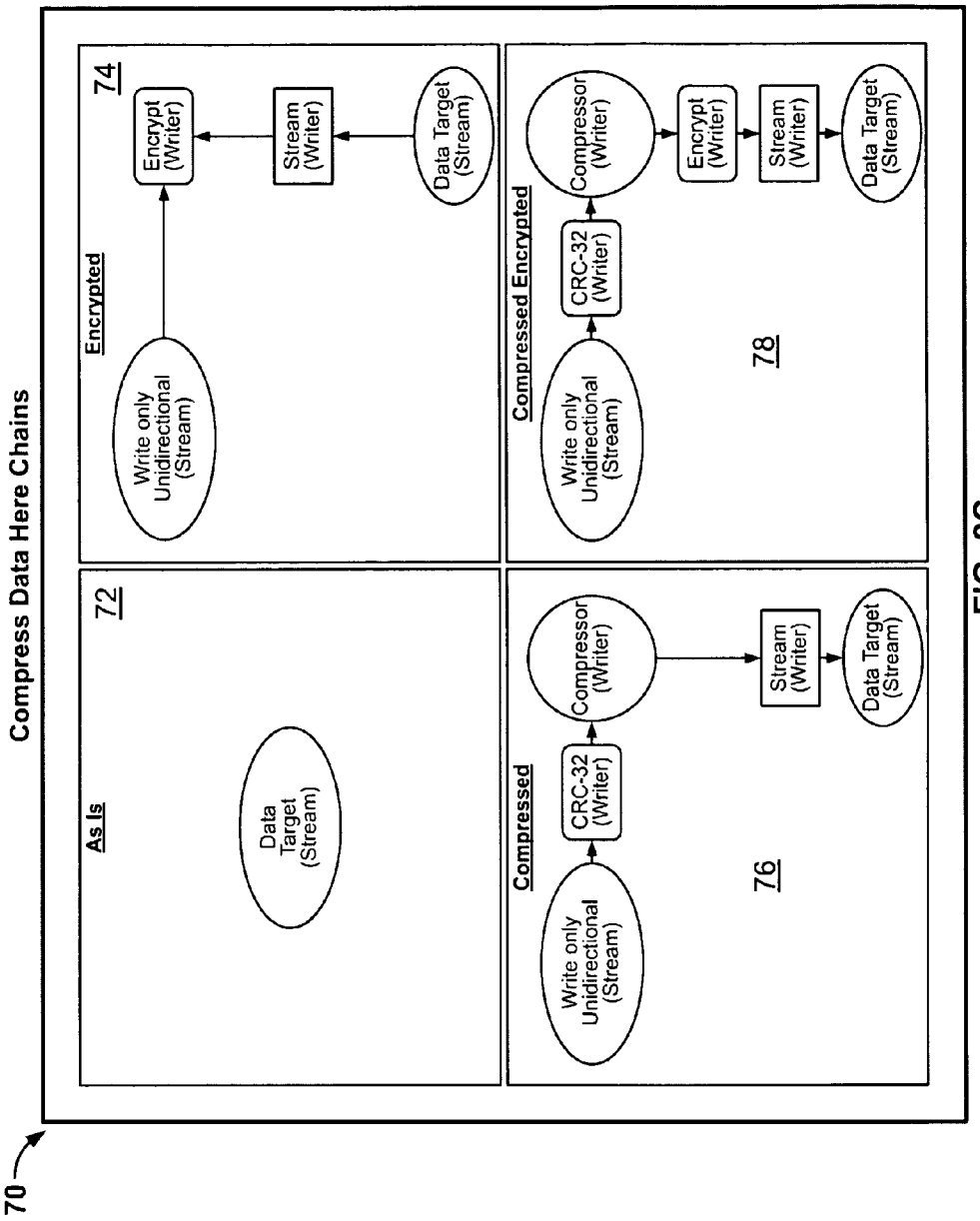

70
Compress Data Here Chains
As Is
72
Data Target (Stream)
Encrypted
74
Write only Unidirectional (Stream)
Encrypt (Writer)
Stream (Writer)
Data Target (Stream)
Compressed
76
Write only Unidirectional (Stream)
CRC-32 (Writer)
Compressor (Writer)
Stream (Writer)
Data Target (Stream)
Compressed Encrypted
78
Write only Unidirectional (Stream)
CRC-32 (Writer)
Compressor (Writer)
Encrypt (Writer)
Stream (Writer)
Data Target (Stream)

FIG. 2C

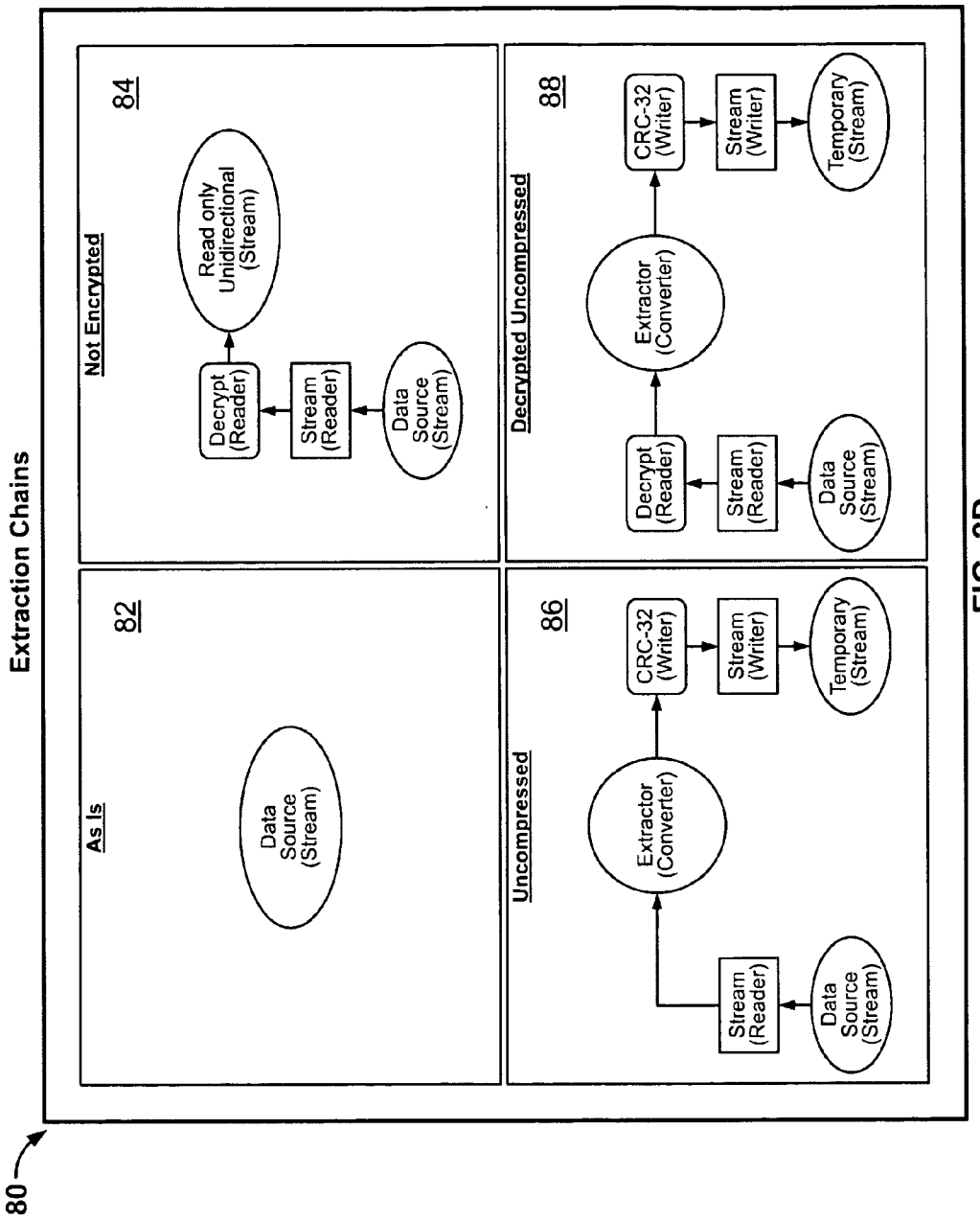

80
Extraction Chains
As Is
82
Data Source (Stream)
Not Encrypted
84
Data Source (Stream)
Stream (Reader)
Decrypt (Reader)
Read only Unidirectional (Stream)
Uncompressed
86
Data Source (Stream)
Stream (Reader)
Extractor (Converter)
CRC-32 (Writer)
Stream (Writer)
Temporary (Stream)
Decrypted Uncompressed
88
Data Source (Stream)
Stream (Reader)
Decrypt (Reader)
Extractor (Converter)
CRC-32 (Writer)
Stream (Writer)
Temporary (Stream)

FIG. 2D

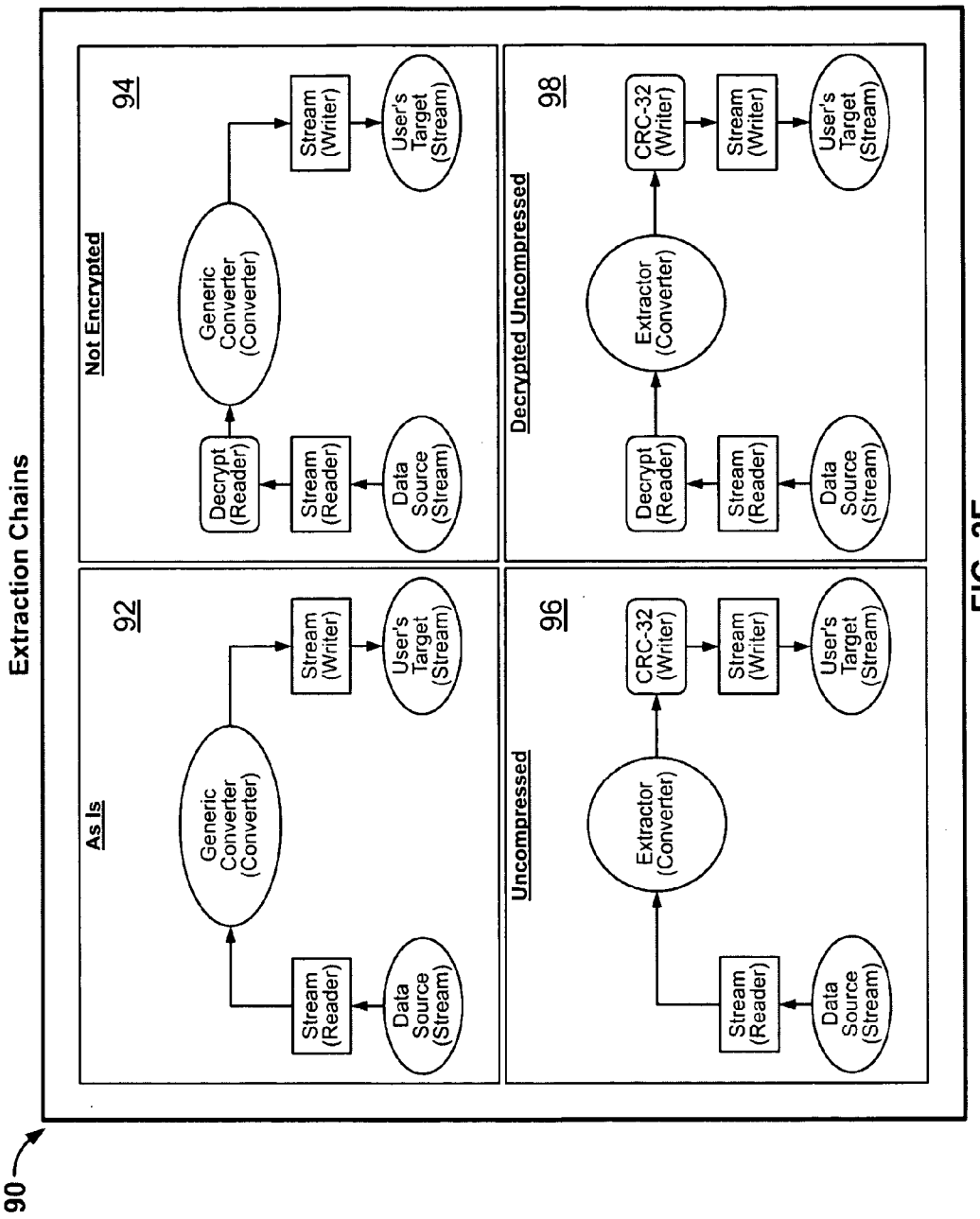

90
Extraction Chains
As Is
92
Data Source (Stream)
Stream (Reader)
Generic Converter (Converter)
Stream (Writer)
User's Target (Stream)
Not Encrypted
94
Data Source (Stream)
Stream (Reader)
Decrypt (Reader)
Generic Converter (Converter)
Stream (Writer)
User's Target (Stream)
Uncompressed
96
Data Source (Stream)
Stream (Reader)
Extractor (Converter)
CRC-32 (Writer)
Stream (Writer)
User's Target (Stream)
Decrypted Uncompressed
98
Data Source (Stream)
Stream (Reader)
Decrypt (Reader)
Extractor (Converter)
CRC-32 (Writer)
Stream (Writer)
User's Target (Stream)

FIG. 2E

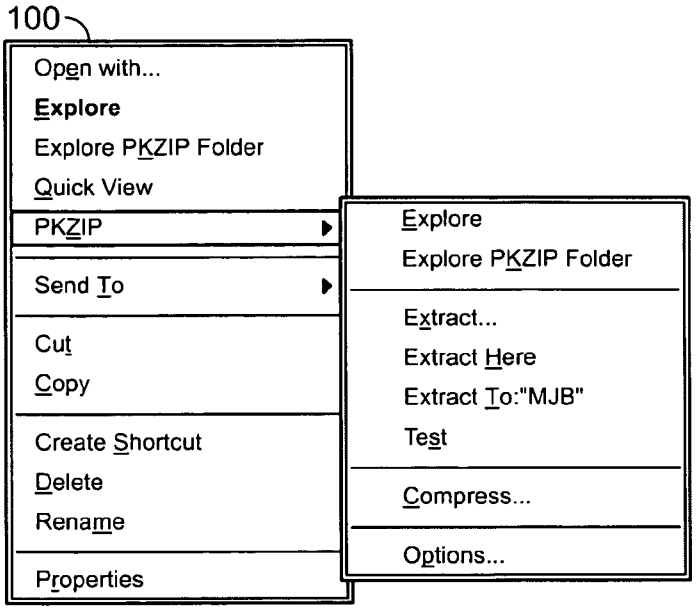

100
Open with...
Explore
Explore PKZIP Folder
Quick View
PKZIP
Send To
Cut
Copy
Create Shortcut
Delete
Rename
Properties
Explore
Explore PKZIP Folder
Extract...
Extract Here
Extract To:"MJB"
Test
Compress...
Options...

FIG. 3

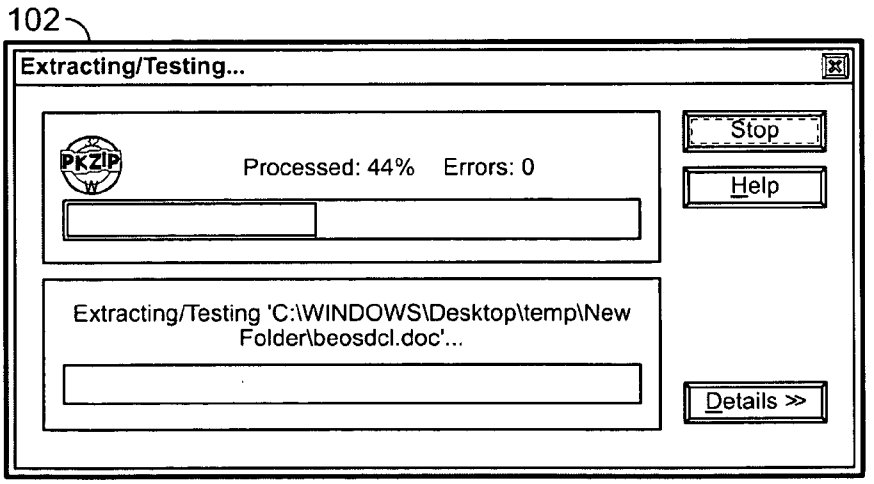

102
Extracting/Testing...
Processed: 44% Errors: 0
Stop
Help
Extracting/Testing 'C:\WINDOWS\Desktop\temp\New Folder\beosdcl.doc'...
Details >>

FIG. 4

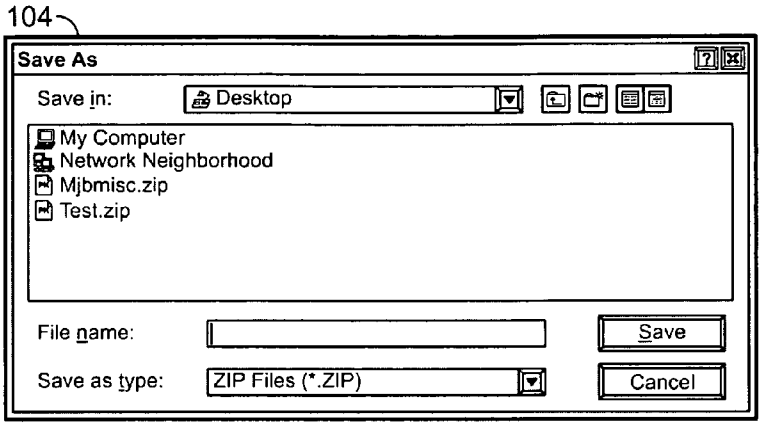


FIG. 5

106

Button:	Description:
	Save the contents of the .ZIP file.
	Extract file(s) via the Extract dialog.
	Add file(s) via the Add dialog.
	Display Log file. Green color indicates that previous PKZIP Explorer operation completed without errors.
	Display Log file. Red color indicates that errors were encountered during the previous PKZIP Explorer operation.
	Invoke the Options/Properties tab dialog where you may specify several PKZIP Explorer configuration parameters.

FIG. 6A

108

Item (Menu):	Description:
Test (File)	Test the contents of the .ZIP file.
Extract (File)	Extract file(s) via the Extract dialog.
Add (File)	Add file(s) via the Extract dialog.
Save (File)	Save the contents of the .ZIP file.
Save As (File)	Save the contents of the .ZIP file to another file and open that file in PKZIP Explorer.
Save Copy As (File)	Save the contents of the .ZIP file to another file but remain in the current PKZIP Explorer window.
Columns (View)	Specify which information columns (e.g. CRC32) you want PKZIP Explorer to display in the right pane.
Show Log (View)	Invoke the Log dialog.
PKZIP Options (View)	Invoke the Options/Properties tab dialog where you may specify several PKZIP Explorer configuration parameters.

FIG. 6B

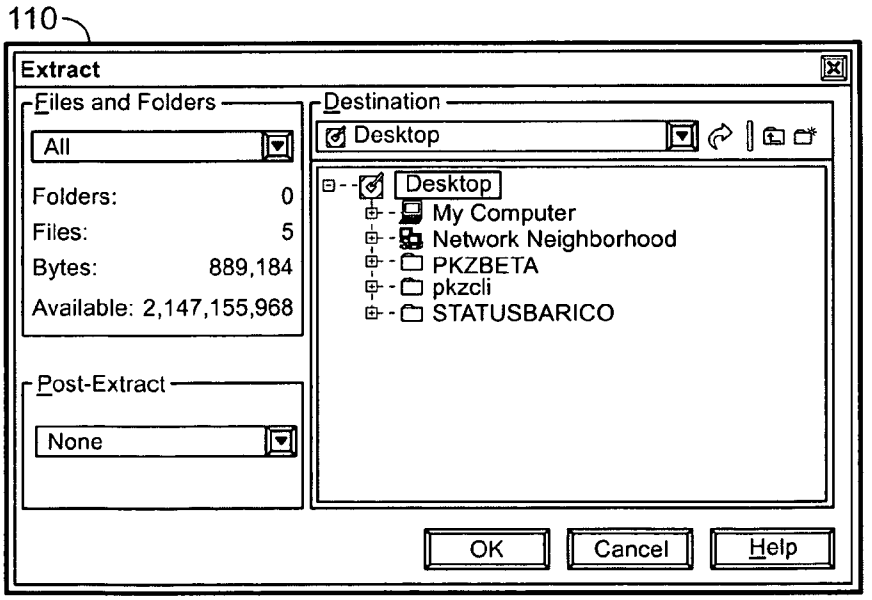


FIG. 7

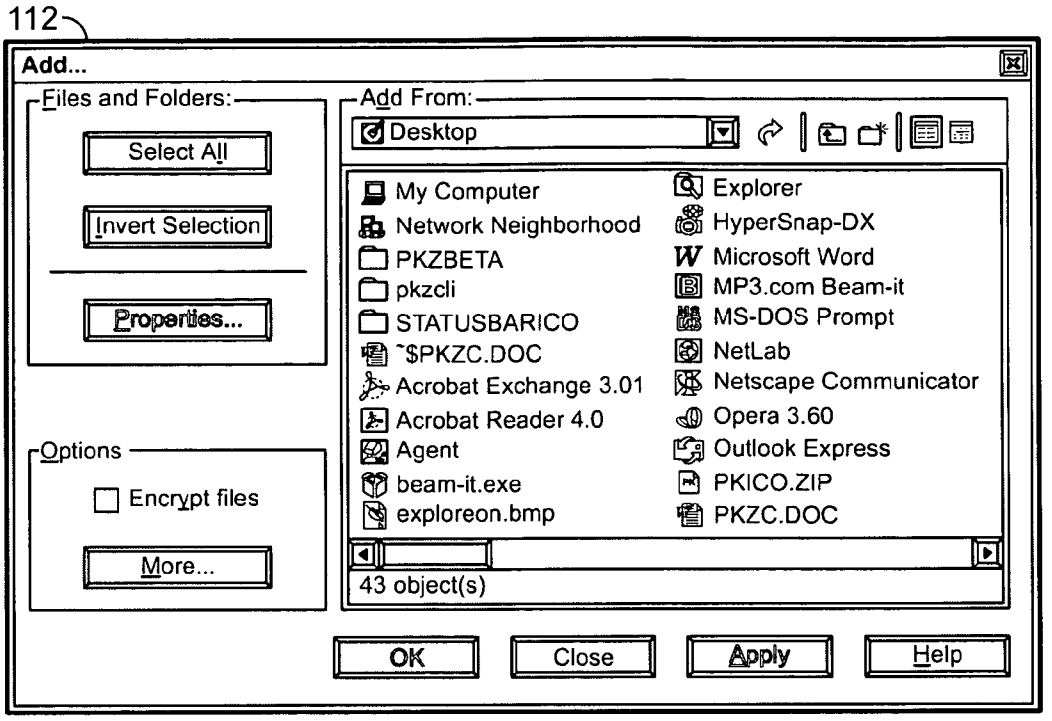

112
Add...
Files and Folders:
Select All
Invert Selection
Properties...
Options
Encrypt files
More...
Add From:
Desktop
My Computer
Network Neighborhood
PKZBETA
pkzcli
STATUSBARICO
~$PKZC.DOC
Acrobat Exchange 3.01
Acrobat Reader 4.0
Agent
beam-it.exe
exploreon.bmp
Explorer
HyperSnap-DX
Microsoft Word
MP3.com Beam-it
MS-DOS Prompt
NetLab
Netscape Communicator
Opera 3.60
Outlook Express
PKICO.ZIP
PKZC.DOC
43 object(s)
OK
Close
Apply
Help

FIG. 8

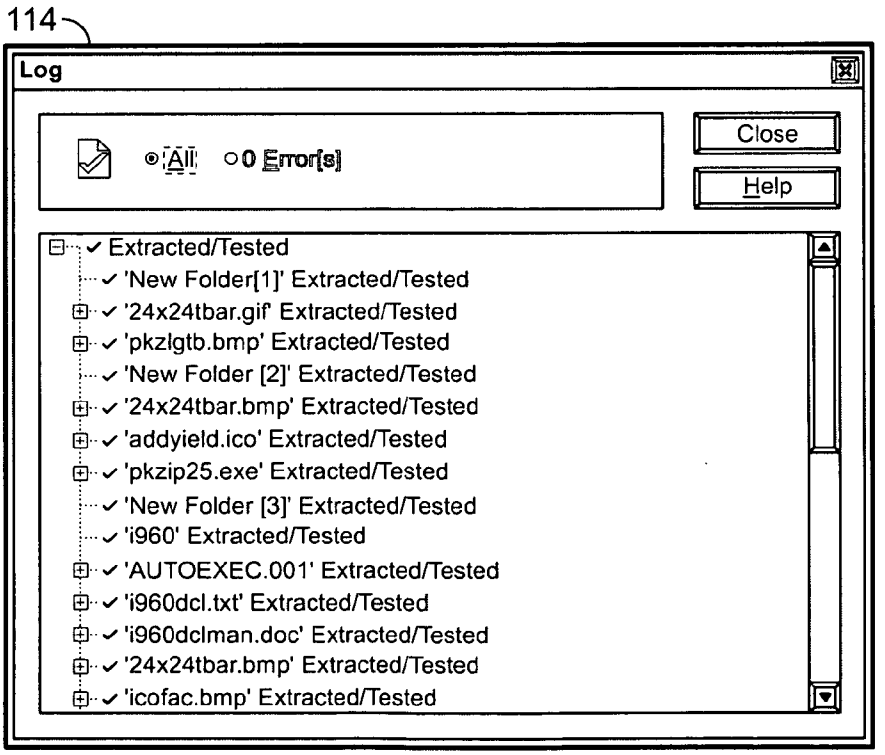

114
Log
All
0 Error[s]
Close
Help
Extracted/Tested
'New Folder[1]' Extracted/Tested
'24x24tbar.gif' Extracted/Tested
'pkzlgtb.bmp' Extracted/Tested
'New Folder [2]' Extracted/Tested
'24x24tbar.bmp' Extracted/Tested
'addyield.ico' Extracted/Tested
'pkzip25.exe' Extracted/Tested
'New Folder [3]' Extracted/Tested
'i960' Extracted/Tested
'AUTOEXEC.001' Extracted/Tested
'i960dcl.txt' Extracted/Tested
'i960dclman.doc' Extracted/Tested
'24x24tbar.bmp' Extracted/Tested
'icofac.bmp' Extracted/Tested

FIG. 9

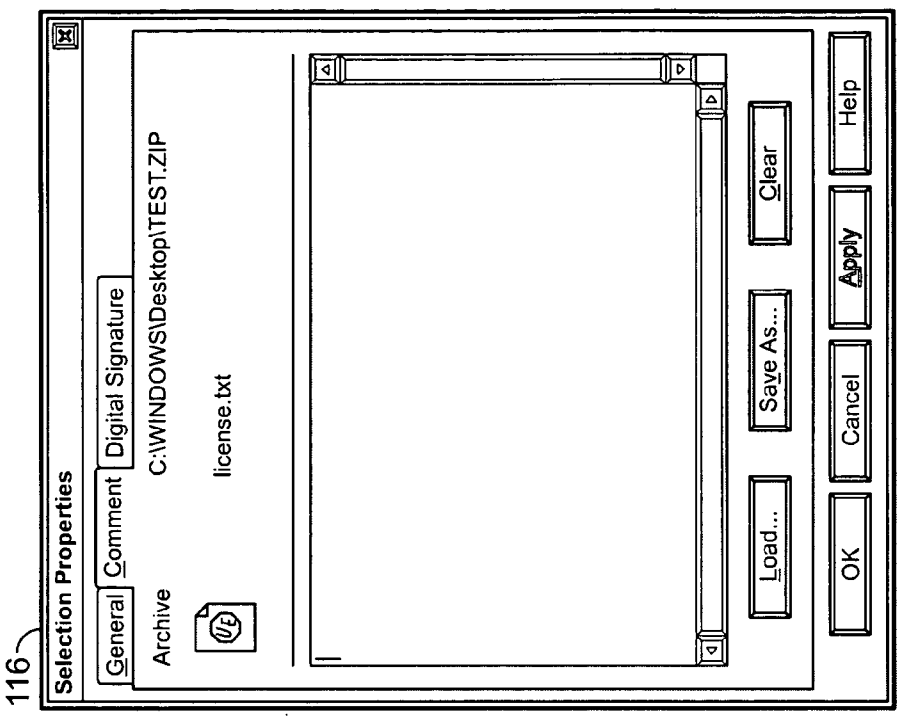


FIG. 10A

FIG. 10B

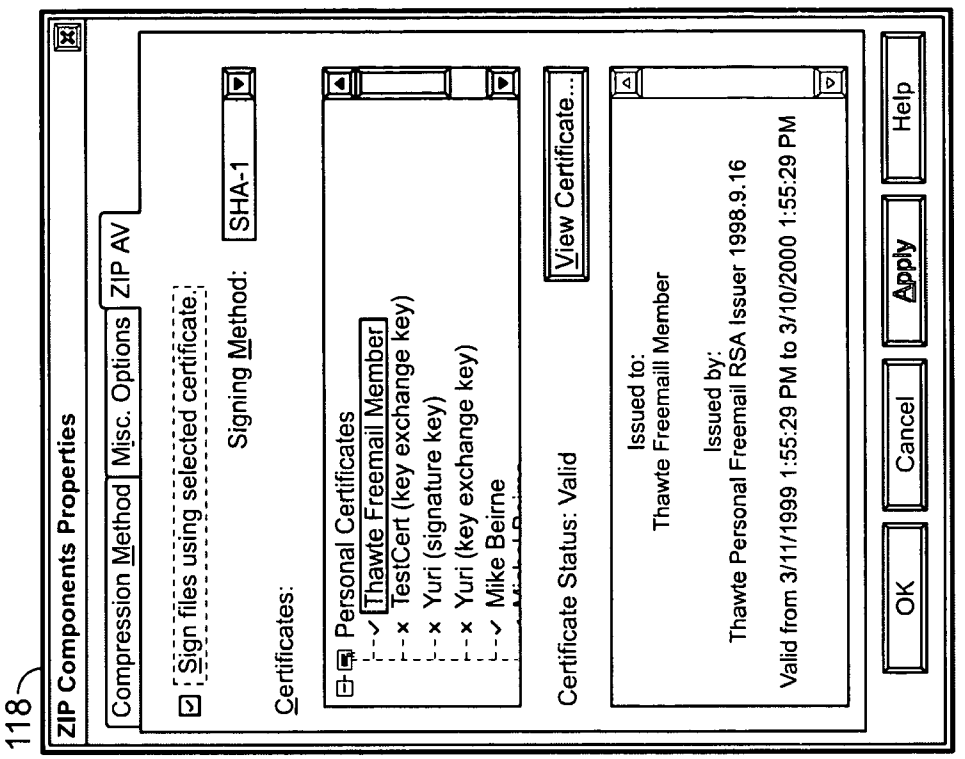


FIG. 11A

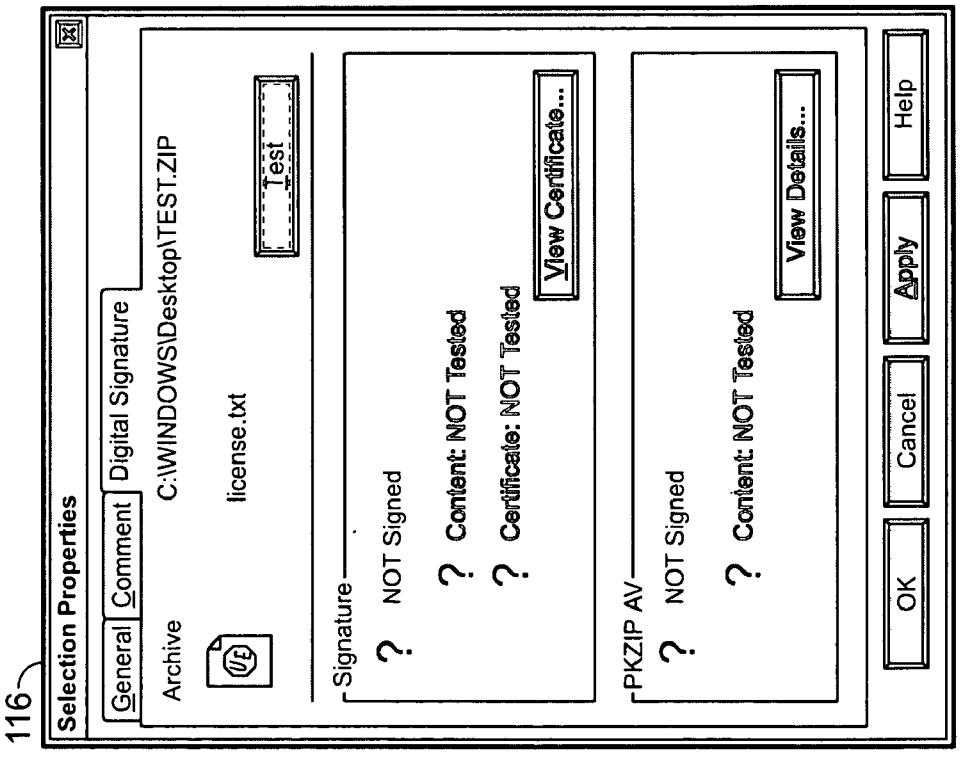


FIG. 10C

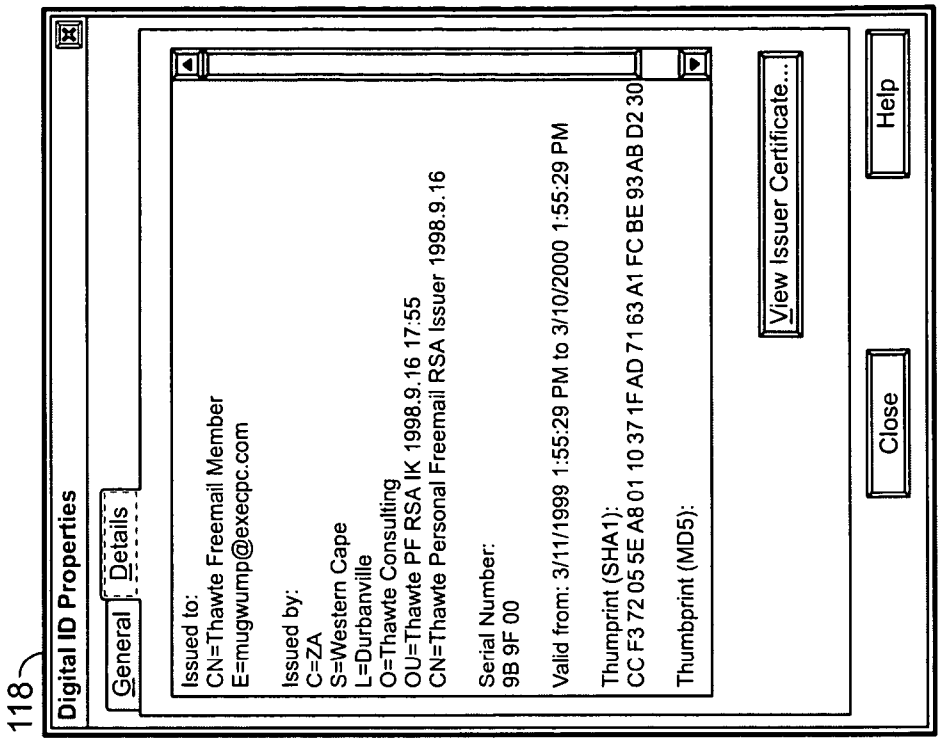


FIG. 11C

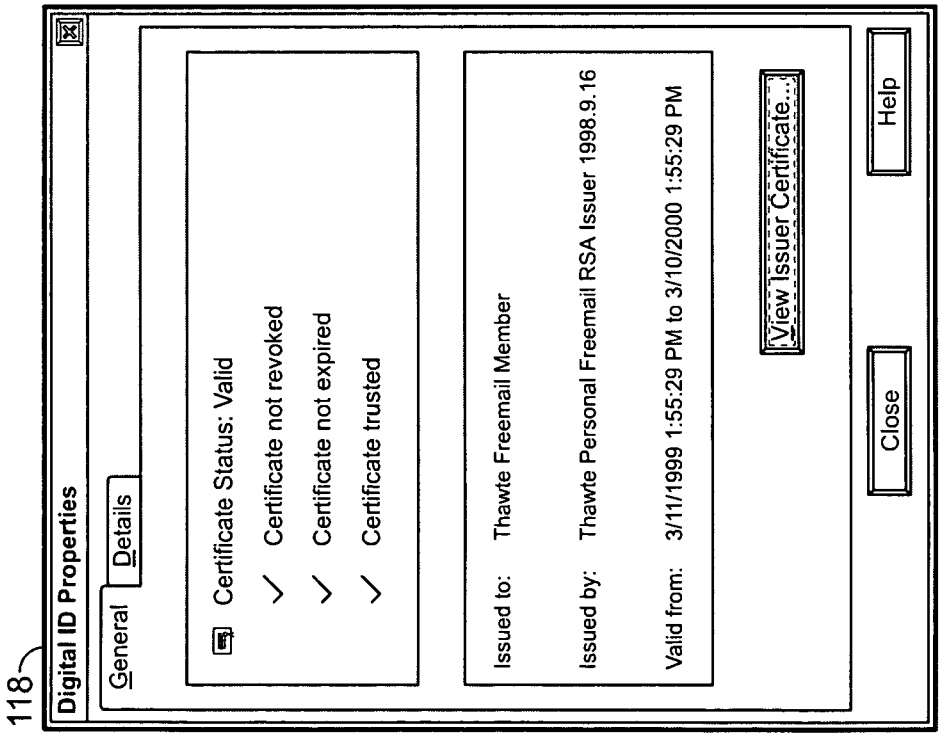


FIG. 11B

SYSTEMS AND METHODS FOR MANIPULATING AND MANAGING COMPUTER ARCHIVE FILES

RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 10/970,897, filed on Oct. 21, 2004, which is a continuation of U.S. patent application Ser. No. 09/803,355, filed on Mar. 9, 2001, now U.S. Pat. No. 6,879,988, which claims the benefit of U.S. Provisional Patent Application No. 60/187,969, filed Mar. 9, 2000.

FEDERALLY SPONSORED RESEARCH OR DEVELOPMENT

[0002] [Not Applicable]

MICROFICHE/COPYRIGHT REFERENCE

[0003] [Not Applicable]

BACKGROUND OF THE INVENTION

[0004] The present invention relates to managing and manipulating computer archive files, and more particularly to a system and method for managing and manipulating zip files through a computer program integrated into Microsoft Windows Explorer.

[0005] Compression of computer files has been around for years. Compressing files can save tremendous amounts of disk space, and transfer time when downloading files from the Internet or transferring files through email. These days, almost any file one downloads from the Internet is compressed in some way. A standard compressed file or folder as it is sometimes called contains one or more files that were compressed into a single file or folder. A lot of different compression formats have been developed over the years. The zip format, created by the assignee of the present invention, is the most common compressed file format for the personal computer, but there are many others in use today. Any file with a ".zip" extension is most likely a compressed file in the zip format. Zipping a file means compressing the file into the zip format archive so that it occupies less disk space, and unzipping a file means uncompressing a compressed file in the zip format. A zip file is a file which has been compressed with PKZIP®, from PKWare, Inc., or another compatible archiver. Zip files are indicated by a ".zip" filename extension.

[0006] A computer file is compressed through the use of one or more compression algorithms. A compression algorithm is essentially a mathematical formula that scans the data in the file for compressible information. For example, compressible information may be any repeating pattern or string that can be represented once. The compression algorithm will then represent the repeated patterns in a coded manner to save space. For standard compression, most of the compression algorithms work basically the same way. Some are just more efficient or faster than others.

[0007] Generally, the contents of a compressed file cannot be accessed unless the archive is uncompressed. In order to uncompress a file, a user needs to either use the same program used to compress the file, or use another program that is compatible with the particular compression format. That meant that users were required to use standalone programs to compress and uncompress their files. The same problem occurs when trying to work with and manipulate compressed archived files. For example, a user wanting to open an existing compressed file, modify the file, or extract data from the file and transfer it to another file would have to command a standalone program to uncompress the original file and command the standalone program to compress the modified file. This process is often burdensome and inconvenient to the user. Therefore, it would be beneficial to create a product that would eliminate the need for separate standalone compression programs, and eliminate the need to separately command a file to be uncompressed or compressed each time the file is opened, modified, or saved.

[0008] Such products have been developed by many companies, including products used in a Microsoft Windows Explorer environment. Microsoft Windows Explorer is a browser program in Windows for exploring directories, files, and folders in a computer system. In connection with Windows Explorer, Microsoft provides a shell name space extension application program interface (API) for software developers to use to integrate other software utility programs into Windows Explorer. Several companies have developed compression file manipulation programs using the Microsoft Windows Explorer interface. Some of these products include: ArjFolder by Raphael Mounier; Cab Viewer by Microsoft Corporation; CleverZip by Cleverness, Inc.; Zip Explorer Pro by Aeco Systems; Internet Neighborhood by KnoWare, Inc.; Net Explore; ZipMagic by Mijenix Corporation; and Netzip Classic by Netzip Inc. The Internet Neighborhood and Net Explore products are file transfer protocol (FTP) products which integrate FTP sites into Windows Explorer. ZipMagic and Netzip Classic are device driver products.

[0009] ZipMagic, patented under U.S. Pat. No. 5,907,703, is directed to a device driver for accessing computer files. The ZipMagic patent utilizes a device driver implemented in the operating system of Windows Explorer that makes all zip files appear to be folders.

[0010] However, all of the above products are implemented differently from the present invention, and do not include many of the features of the present invention. Many of the above programs have increased performance overhead in processing (compressing/uncompressing) files continuously in and out, and it is often difficult for a user to determine if he is in a zip file or rather in a folder.

[0011] Accordingly, there is a need for a system and method for easy management and manipulation of archive files. The program of the present invention is intended for use on Microsoft Windows 9x, Me (Millennium Edition), NT 4.0, and 2000 systems. Windows 95 and NT 4.0 systems require Microsoft Internet Explorer 4.0 or greater.

BRIEF SUMMARY OF THE INVENTION

[0012] The present invention provides a software utility program that is seamlessly integrated into Microsoft Windows Explorer. The program allows users to manage and manipulate their zip archive files without leaving the Explorer environment. Users may open, archive, compress, extract, create, modify and add to their zip archive files using Windows Explorer's context and pull-down menus, toolbars, copy and paste operators, and drag and drop operators. A mail compressor attachment module integrates into Microsoft Outlook to automatically archive files sent via

email. An Internet plug-in module works with Internet Explorer 4.0+ or Netscape Communicator 4.0+ to facilitate the handling of downloaded zip files from the Internet. The Internet module allows a user to view and manipulate zip archive files downloaded from the Internet.

[0013] An archive manager provides quick access to a user's zip files stored on the computer. The archive manager can create a hierarchical tree representation of a zip file which allows quick and easy management and manipulation of complex zip archives. Shortcuts may also be optionally created and/or deleted by the archive manager. Double-clicking a shortcut will open a zip file under the archive manager. File shortcuts may be created using copy and paste operators, dragging a file into the archive manager, or via the scan and add function. File shortcuts are deleted by highlighting the shortcut and selecting delete on the keyboard or Windows Explorer menu. Shortcuts may be created that branch to a zip file's contents under a specified working directory as an alternative to working within the archive manager. Archive files may be extensively modified before the actual changes are saved. As a result, system overhead is minimized, as the resources required for such operations are only needed when the archive is actually saved. An edit mode of the archive manager during archive modification illustrates graphic instruction cue icons (indicating Add and Delete states) in the far left column of the Explorer Window.

[0014] In addition to easily opening, extracting, creating and modifying archive files, the present invention also includes several miscellaneous features or functions selectable by the user. These functions include edit-before-saving, digital certificate based file authentication and encryption, selecting compression methods by file type and spanning/splitting of archive files.

[0015] In one aspect of the invention, an edit-before-saving function that is useful during creating, opening, modifying or extracting an archive file. The edit-before-saving function provides graphic instruction cue icons (indicating Add and Delete states) next to archives that have been modified. Archives may be extensively modified before the actual changes are saved. As a result, system overhead is minimized, as the resources required for compressing and uncompressing are used only when the archives are actually saved.

[0016] In another aspect of the invention, a Public Key Infrastructure (PKI) based digital signature, file authentication and encryption function adds a layer of authenticity to the zip archive files. The invention includes a X.509 based authentication and encryption function which allows a user to digitally sign and encrypt individual files archived in a zip file and subsequently authenticate and decrypt those files upon extraction. Digitally signing a zip file allows one to detect whether the integrity of a zip file has been compromised. Encrypting a file denies access to the file's contents by unauthorized users. The ability to store a PKI based digital signature using standard X.509 based certificate (e.g., VeriSign Digital ID) information is a significant enhancement to the zip archive file format. This function allows users to digitally sign an archive file and its contents using a standard X.509 based digital certificate.

[0017] This function also allows a user to digitally sign the central directory of the zip file and to encrypt file names and supplemental information such as, but not limited to, file system security descriptor information.

[0018] In a further aspect of the invention, a user may select a compression method based on the type of file being saved. The compression methods include Store, Deflate, and DCL Implode. By default, the present invention compresses all files using the Deflate algorithm. A user may choose to compress all files using the Deflate algorithm, or may optionally modify the default method of compression, as well as the method to be used on a specified file type. In addition, a user may specify to use the 64 k dictionary version of the Deflate algorithm for improved compression.

[0019] The spanning function of the invention allows a user to span large zip archives over multiple removable media diskettes. The splitting function of the invention allows a user to divide an archive file into specified file segment sizes.

[0020] Various other features, objects, and advantages of the invention will be made apparent to those skilled in the art from the following detailed description and accompanying drawings.

BRIEF DESCRIPTION OF SEVERAL VIEWS OF THE DRAWINGS

[0021] FIG. **1** is a diagram illustrating the software components underlying the system of the present invention;

[0022] FIGS. **2***a*-**2***e* are diagrams illustrating the different compression and extraction chains used in the present invention;

[0023] FIG. **3** displays a right-click context menu of the present invention;

[0024] FIG. **4** displays a progress dialog screen of the present invention;

[0025] FIG. **5** displays a save as dialog screen of the present invention;

[0026] FIG. **6***a* displays a table of toolbar buttons used in the present invention;

[0027] FIG. **6***b* displays a table of menu items used in the present invention;

[0028] FIG. **7** displays an extract dialog screen of the present invention;

[0029] FIG. **8** displays an add dialog screen of the present invention;

[0030] FIG. **9** displays a log dialog screen of the present invention;

[0031] FIG. **10**A displays a "General" screen of a series of selection properties dialog screens of the present invention;

[0032] FIG. **10**B displays a "Comment" screen of a series of selection properties dialog screens of the present invention;

[0033] FIG. **10**C displays a "Digital Signature" screen of a series of selection properties dialog screens of the present invention;

[0034] FIG. **11**A displays a "General" screen of a series of authenticity/certificate dialog screens of the present invention;

[0035] FIG. **11**B displays a "Comment" screen of a series of authenticity/certificate dialog screens of the present invention; and

[0036] FIG. **11**C displays a "Digital Signature" screen of a series of authenticity/certificate dialog screens of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0037] FIG. **1** is a diagram illustrating the software components underlying the system **10** of the present invention. There are three basic components of the underlying software. These components are the user interface (UI) **20**, the file management (FM) component **30**, and the compression/extraction engine (CE) **40**.

[0038] The lowest level component is the compression/extraction engine **40**. The compression/extraction component **40** consists of the actual compression, extraction, and crc-32 algorithms. These are written as a set of portable C language routines, with higher level C++ routines interfacing with the higher level file management component **30**. The file management component **30** consists of the central directory **32** which holds a cached tree-like structure of the archive independent of the actual archive type. Actual archive implementation is used by the central directory **22** to read/write data to the archives **34** and the user interface **20**. The central directory **32** consists of folder objects and file objects **36**. A services object **38** is also part of the file management component **30**. The services object **38** acts as a helper interface between the user interface component **20** and the file management component **30**. The user interface **20** consists of a shell **22**, a graphical user interface (GUI) **24**, and a call level interface (CLI) **26**.

[0039] The data object **36** supports one or more standard file formats (Explorer or File Manager drag and drop), and one or more custom formats (zip compressed non-encrypted and zip compressed). When files are dropped from Explorer to the archive, the archive requests available standard data formats to compress the data. When files are dropped from archive to Explorer, Explorer requests available standard data formats. In this instance, the data object will need to uncompress the data. When files are dropped from one zip archive to another zip archive, the target archive will be able to detect the native data and copy it without modification. When files are dropped from an ARJ archive to a zip archive, the zip archive will be able to recognize only standard formats, as a result, the ARJ data object will uncompress the data and the zip archive will compress the data. So it is possible to convert data between different archives.

[0040] FIGS. **2***a*-**2***e* illustrate the different compression and extraction used by the present invention. In FIG. **2***a*, regular compression and extraction chains are shown. In FIG. **2***b*, compression data chains are shown, including the use of a generic converter involving no compression. In FIG. **2***c*, data here compression chains are shown. In FIG. **2***d*, GetData extraction chains are shown. In FIG. **2***e*, GetData-Here extraction chains are shown.

[0041] The compression/extraction engine **40** and the file management component **30** form the data compression library that is used to build applications, such as the present invention, needing zip compatible compression and file management.

[0042] The present invention provides a software utility program that is integrated into Microsoft Windows Explorer for managing and manipulating archive files without leaving the Explorer environment. The invention includes an archive manager which allows a user to open, view, modify (add/delete), and extract data from an existing archive, or create a new archive using modified Windows Explorer right-click context menus, pull-down menus, toolbars, copy and paste operators, or drag and drop operators.

[0043] FIG. **3** displays a right-click context menu **50** of the present invention which may be used to open, modify and extract files from an exiting zip file, or create a new zip file. In opening a zip file, a user may simply double-click the file to view the contents of the file. Alternatively, the following is an example of the steps one might follow to open and view the contents of a zip file. First, the zip archive to be opened is located by using Windows Explorer. Then, the user right-clicks on the zip file he wants to open. A context menu appears. PKZIP/Explore is selected. The contents of the zip file will be displayed in the right pane under the archive manager. As another alternative, a user may select PKZIP/Explore PKZIP Folder to create a folder shortcut under the current folder, and display the contents of the zip file via this folder.

[0044] To extract individual files and/or folders archived in a zip file, a user opens the zip file in Explorer as discussed above and invokes the extract dialog, by selecting the Extract menu item in the right-click context menu. The Extract dialog appears, FIG. **7**, allowing the user to manually specify a destination directory. Alternatively, a user may select PKZIP/Extract Here to extract the contents of the archive into the directory where the zip archive resides. To create a directory (e.g., "Test") under the directory where the zip archive resides, and extract all files in that directory, the user selects the "Extract-to" menu item. Alternatively, files may be extracted using a drag and drop operation. The user highlights the files and/or folders he wishes to extract, drags the files to a destination, and drops the files in the enabled destination. The files and/or folders will be automatically extracted into the drop destination. As the extraction process proceeds, the progress is displayed in a progress dialog, as shown in FIG. **4**. If there is an error encountered during the extraction process, the error is indicated in the progress dialog and a log dialog, shown in FIG. **9**.

[0045] The present invention also allows a user to create a new zip file. The following is an example of the steps one might follow to create a new zip file. First, the user highlights the files and/or folders he wishes to archive. The user then clicks his right mouse button to bring up the context menu. PKZIP/Compress is then selected. The "Save as" dialog appears, FIG. **5**. A name and destination are specified for the zip file, and the save button is clicked to proceed. The progress dialog appears to monitor completion and to indicate errors in the process. The new zip file should now reside in the specified destination directory. A user may alternatively create a new zip file by other means as well. A user may create a new folder in an archive by selecting the New Folder menu toolbar item and specifying a folder name as desired.

[0046] Adding or deleting files in a zip file works somewhat differently than these same operations do in Explorer. For example, in Explorer, when a user highlights a file and

clicks the delete key, that file is immediately deleted. The present invention includes an edit-before-saving function, so that when a user highlights a file and clicks the delete key, a graphic instruction cue icon is displayed directly to the left of the file icon, indicating that this file is to be deleted. Similarly, when a file is added to an archive, the program will display an add icon directly to the left of the file icon indicating that this file is to be added. In other words, a zip file is not actually modified until the user specifically instructs the program to save the zip file.

[0047] The following is an example of the steps one might follow to modify an existing zip file. A user first locates and opens the zip file he wishes to modify. Next, the files and/or folders to add to the archive are specified by clicking the add toolbar button, thereby invoking the add dialog, or by dragging the files and/or folders from their source and dropping them at a destination. A user may alternately use the copy and past operation to specify files and/or folders to add to the archive. The program will display an add icon (such as plus symbol) indicating that these files and/or folders are to be added when the archive is saved.

[0048] In a similar manner, a user may specify files and/or folders to delete in the archive by highlighting the files the user wishes to delete, and clicking the delete key, or by selecting the Delete menu item. The program will display a Delete icon (such as a circle with a slash through it) indicating that these files and/or folders are to be deleted when the archive is saved.

[0049] After the user is finished modifying the zip file, the file may be saved by selecting the Save menu item available under the File menu, or by use of the right-click context menu. A user may also click the Save button on the toolbar. To save the modifications to another zip file, select the Save As or Save Copy As menu items.

[0050] FIGS. **6***a* and **6***b* display modifications and additions to the Explorer toolbar buttons and menu items used in the present invention.

[0051] The present invention may also include many options which may be configured in the options tab dialog accessible via the menu/tool bar or via the right-click context menu. One option is the compression method. Under this option, a user may specify a compression algorithm other than the default algorithm. The compression algorithms to choose from may include Store, DCL Implode, and Deflate. By default, all files are compressed using the Deflate algorithm. As one example, a user may wish to use the Store feature for all JPEG files, since the compression ratios on files of this type are typically negligible. A user may specify a default method or extension specific method under the extension column. Depending on the compression method specified, a user may wish to configure one or more of the storage parameters, as described below.

[0052] There are no available settings for the store method. The program simply archives the specified files without compression. Since the program does not expend time compressing files, this is the fastest method of archival.

[0053] Under the DCL Implode method of compression, the dictionary byte size (i.e., 1024, 2048, 4096) a user wishes to use when compressing files is configurable along with the data type. The binary setting should be selected to optimize compression of program files or other non-text files. The ASCII setting should be selected to optimize compression of text files.

[0054] Most zip utilities use the Deflate algorithm to compress files. Under this algorithm, the compression level may be set using a slide bar to specify the level of compression you wish to apply when archiving files. Moving the slide bar all the way left instructs the program to use the fastest method of compression. Moving the slide bar right increases the time the program expends compressing the file which, as a result, improves compression. Moving the slide bar all the way right instructs the program to apply maximum compression to files. This is the slowest method of file archival because the program must expend time maximizing compression on the files. Typically, applying maximum compression results in the smallest zip file.

[0055] In addition, the dictionary kilobyte size may be selected when using the Deflate algorithm. The dictionary size is selectable between a 32K dictionary and a 64K dictionary. The 64K dictionary provides slightly better compression ratios, but may not be compatible with all zip utilities.

[0056] The present invention also allows the user to digitally sign and encrypt the individual files archived in a zip file as well as the central end directory, and subsequently to authenticate and decrypt those files upon extraction. The signing and encrypting functionality is based on PKCS No. 7, and related public key encryption standards and is therefore compatible with security functionality in other applications such as Microsoft's Internet Explorer. Signing a zip file allows one to detect whether a zip file's integrity has been compromised. Encrypting a file denies access to the file's contents by unauthorized users.

[0057] Before a user can sign or encrypt files, he must first have a digital certificate with which to sign or encrypt. A digital certificate may be obtained from VeriSign or Thawte or from another certificate authority.

[0058] The present invention also provides a software utility program that integrates the compression/extraction engine into Microsoft Outlook to compress, encrypt and authenticate email attachments without leaving the Outlook environment. The invention includes a toolbar button and a tooltray menu that allows turning the compression of email attachments on or off. The compression of email attachments reduces the storage and transfer time of email messages and can reduce the spread of common email attachment viruses.

[0059] The system of the present invention may further include a more generally applicable mail attachment compressor module. Most email programs support sending file attachments along with the main body of the email message. Most users can choose to send the attached file as it originally exists, or compress it prior to attachment to the mail message so it is smaller and more efficient to send and store. Currently, the file to be attached must be manually compressed outside the email program and then attached using the attachment features of the email program. The mail attachment compressor module of the present invention integrates compression into the standard Microsoft Outlook mail message edit form so compressing attachments can be done automatically as the message is sent. The mail attachment compressor module also provides the ability to digitally sign attachments as they are sent for greater security.

[0060] After installing the program of the present invention, the mail form of Microsoft Outlook will have two additional buttons in its "standard" toolbar. These buttons include a Toggle Compression button and an Options button. If the Toggle Compression button is not depressed (the default state), all mail attachments will be compressed automatically when the standard "send" button is used to send the message. Attachments already compressed when attached will be left as is, while attachments that are not compressed will be compressed into a single ZIP file that will replace the original uncompressed attachments. When the Toggle Compression button is depressed, the compression will not be done and the files will be sent as attached. The Options button will display the Options configuration dialogs from the compression/extraction engine so that the compression actions can be configured. The primary use of this button is to configure digital certificates, but any configurable parameters supported by the compression/extraction engine can be set. These parameters include digital certificates, passwords and compression method options.

[0061] Operation of the mail attachment compressor program is initiated by installing the mail attachment compressor module software onto the users system, and initiating Microsoft Outlook. If the mail attachment compression feature is enabled through the toggle button, the attachments will automatically be compressed when the message is sent using the send button. If the mail attachment compression feature is not enabled, then the attachments will be sent unaltered.

[0062] The components of the mail attachment compressor module provide the functionality to be implemented within Microsoft Outlook and provide integration between this module and the underlying compression/extraction engine. When compressing an attachment, the files to be compressed will be passed off to the compression/extraction engine of the underlying software program. After compression, the compressed file will be reattached to the original message, the original copies of the attachments that are now compressed will be removed from the message, and any temporary files created during compression will be deleted.

[0063] The mail attachments module uses the compression/extraction engine to hook directly into Microsoft Outlook to allow users to compress email attachments into zip files. This module provides an automation hook so that email attachments appended to Outlook mail messages can be automatically compressed when the message is sent.

[0064] The Scan and Add dialog of the archive manager is invoked via the Scan and Add toolbar button or the Explorer File/right-click menu. Once the user is finished specifying files to add to the archive manager list, the user clicks OK to add the selected file shortcuts to his list and return to Explorer, or he clicks Apply to add the selected files and remain in the Scan and Add dialog. The options available via this dialog include Files and Folders, multiple selection scan, and Scan and Add form. The archive manager allows the user to add unopened archives to the archive manager list, and to add multiple files using the multiple selection scan option under the archive manager.

[0065] While the invention has been described with reference to preferred embodiments, those skilled in the art will appreciate that certain substitutions, alterations and omissions may be made without departing from the spirit of the invention. Accordingly, the foregoing description is meant to be exemplary only, and should not limit the scope of the invention set forth in the following claims.

1. A system for providing a user-accessible shortcut to a file in a file archive, said system including:

a file archive including a data file;

a graphical user interface provided by a browser; and

a file archive management application, wherein said file archive management application allows a user to establish a user-accessible shortcut to said data file using said graphical user interface.

2. The system of claim 1 wherein said file archive is a .zip archive and said file archive management application is a .zip archive management application.

3. The system of claim 1 wherein said browser is Microsoft Windows Explorer.

4. The system of claim 1 wherein said file archive is compressed.

5. The system of claim 4 wherein said file archive is decompressed when said user-accessible shortcut is selected.

6. The system of claim 4 wherein said data file is decompressed when said user-accessible shortcut is selected.

7. The system of claim 1 wherein said file archive is encrypted.

8. The system of claim 7 wherein said file archive is decrypted when said user-accessible shortcut is selected.

9. The system of claim 7 wherein said data file is decrypted when said user-accessible shortcut is selected.

10. The system of claim 1 wherein said file archive management application allows a user to establish a user-accessible shortcuts.

11. The system of claim 10 wherein at least one of said user-accessible shortcuts is a shortcut to said data file using said graphical user interface.

12. The system of claim 10 wherein at least one of said user-accessible shortcuts is a shortcut to said file archive using said graphical user interface.

13. A method for providing a user-accessible shortcut to a file in a .zip archive, said method including:

providing a .zip archive including a data file;

providing a graphical user interface using Microsoft Windows Explorer; and

providing a .zip archive management application, wherein said .zip archive management application allows a user to establish a user-accessible shortcut to said data file using said graphical user interface.

14. A method for accessing a file in a .zip archive, said method including:

providing a .zip archive including a data file;

providing a graphical user interface using Microsoft Windows Explorer; and

accessing said data file through said graphical user interface using a user-accessible shortcut.

15. A method for supplementing Microsoft Windows Explorer, said method including:

providing Microsoft Windows Explorer;

providing a .zip archive management application;

installing said .zip archive management application into Microsoft Windows Explorer to form a supplemented version of Microsoft Windows Explorer;

providing a .zip archive including a data file; and

providing a user-accessible shortcut to said data file using said supplemented version of Microsoft Windows Explorer.

* * * * *