STUDY AND DESIGN OF SECURITY SYSTEM FOR AUDIOGRAPH MULTIMEDIA TEACHING SYSTEM

A thesis presented in partial fulfillment of the requirements
for the degree of

Master of Science
in
Computer Science

at Massey University, Palmerston North,
New Zealand.

Shaohuai Zhang

2002

ABSTRACT

The AudioGraph is a project developed in the Institute of Information Sciences and Technology of Massey University as copyrighted software tools. The AudioGraph courseware files produced by the AudioGraph Recorder may be copyrighted. In order to protect them from been played back or copied by unauthorized parties, a security system to protect the AudioGraph courseware files is required.

This thesis presents the study, design and implementation of the AudioGraph security system. The security system proposed in this thesis consists of three parts: *Copy Protection Record* inside the AudioGraph courseware files; a Key Insertion Tool to detect, extract, insert and update the *Copy Protection Record* in AudioGraph courseware files; and a scheme of usage control embedded into the AudioGraph Plug-in.

The issues covered in this thesis include all relevant aspects. In order to select good encryption algorithms for the AudioGraph security system, this thesis introduces the concept of cryptography and describes some of the most important conventional and public-key encryption algorithms. It also investigates and compares various aspects of some of the conventional cryptography algorithms and chooses very strong, simple and suitable encryption algorithms to be used in the AudioGraph security system. A scheme to protect AudioGraph courseware files is described in this thesis, this scheme meets the requirements of the AudioGraph security system, and it is strong enough to withstand brute-force attack and all known cryptanalysis.

The implementation of the AudioGraph security system is also been described in this thesis. The result from system testing demonstrates that this AudioGraph security system works well and had achieved its goal to protect the AudioGraph courseware material.

ACKNOWLEDGEMENT

This Master Thesis is for my new life in New Zealand. There has been so much encouragement and support over the past few years, I am really grateful for it.

A special thanks goes to my supervisor, Professor Chris Jesshope. Thank you for your support and encouragement. I have never gained so much self-confidence before.

A hearty thank you goes out to my wife and my parents for their continued support. Without them, I could not have finished this thesis.

Thanks goes to the staff in computer science department of Massey University, without their encouragement and support, I could not have finished the research work over the past few years.

I also want to thank all my colleagues in the New Zealand Educational Software unit. It makes me feel happy working in such a good team.

Shaohuai Zhang
Master of Science (Computer Science) Candidate,
Massey University

Computer Science,
Institute of Information Science and Technology
Massey University
Palmerston North
New Zealand

LIST OF ACRONYMS

AES	Advanced Encryption Standard
API	Application Programming Interface
CA	Certification Authority
DES	Data Encryption Standard
DLL	Dynamic-Link Library
GSM	Global System for Mobil telecommunication
HTML	HyperText Makup Language
HTTP	HyperText Transfer Protocol
IANA	Internet Assigned Numbers Authority
IDEA	International Data Encryption Algorithm
IETF	Internet Engineering Task Force
ITU	International Telecommunications Union
ITU-T	ITU Telecommunications Standardization Sector
KDC	Key Distribution Center
LFSR	Linear-Feedback Shift Register
MIME	Multipurpose Internet Mail Extensions
NIST	National Institute of Standards and Technology of America
NSA	National Security Agency (the official security body of the U.S. government)
PNG	The Portable Network Graphics format
SQA	Software Quality Assurance
SSL	Secure Sockets Layer
TCP/IP	Transmission Control Protocol/Internet Protocol
TEA	Tiny Encryption Algorithm
TLS	Transport Layer Security

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1: Introduction

This chapter briefly describes the concept of multimedia teaching and the AudioGraph project in the Institute of Information Sciences and Technology of Massey University, and introduces some of the technologies used in the AudioGraph project. This project has looked at the problem of introducing protection system into the playback of the Audiograph multimedia files. The concepts of computer security and cryptography, which are related to the protection system of AudioGraph, are introduced and the requirements for the protection system of the AudioGraph are both described in this chapter.

1.1 Introduction of Multimedia Teaching

Teaching is an interactive goal-directed activity, which includes presentations given by the lecturer. In a presentation, the lecturer usually explains abstract concepts or theories and demonstrates ideas by using all available tools – oral description, drawing a diagram, showing a working model etc. Usually the students will ask questions about the presentation, do exercises or experiments and the lecturer will answer questions, correct the mistakes of the students on their exercises, instruct on the experiments etc.

It is obvious that all of these teaching activities would need to include interactive audio, visual, text communication and animation. With the invention of the computer and the widespread use of multimedia and computer networks, especially the Internet, all of these activities connected with teaching can be realized locally or distantly. Therefore the student and the lecturer do not need to meet together at a specific place and time in order to be instructed on a specific topic. Self-paced, individual-time-and-space-independent learning becomes possible. Moreover, multimedia computer presentations make it possible to visualize abstract concepts and simulate a process, thus making it easier to understand. With artificial intelligence integrated into the multimedia education system, a student may carry out experiments and interact with virtual worlds and an intelligent agent may act as if it were a lecturer. There are a growing number of electronic interactive textbooks that are now available on the net. More recently there

has been a large interest, in the university sector, in the establishment of virtual universities (HART & MANSON, 1996).

The success of Multimedia Teaching systems depends critically on available hardware technologies: high performance CPU and I/O bus architectures, high-quality audio & visual input/output devices, large capacity data-storage devices and high-speed networks. In addition, these systems require a much higher level of system software support in comparison to past computational and communicational environments. The recent personal computer system has advanced so fast that it now makes multimedia teaching possible. However, there are still problems with network speed, because most potential 'virtual' students do not sit on an intranet but are at the end of a, possibly quite slow, modem line. Multimedia teaching courseware and authoring tools therefore must take account of this problem.

The AudioGraph project addresses this problem by providing tools to develop interactive multimedia presentations, which stream over low bandwidth connections (Jesshope et al., 1998).

1.2 Overview of AudioGraph Multimedia Teaching Tools

AudioGraph is a project that was initially developed in the Computer Science Department of Massey University. The AudioGraph system includes multimedia tools that have been developed for recording audio-graphic presentation material for publication in an HTML reference environment. At present, the tools comprise Macintosh applications (Audiograph Recorder) and Netscape plug-ins. Figure 1.1 shows the AudioGraph tools and their relationship to each other.

The AudioGraph Recorder will be used to record the lectures including text, audio and graphic information and produce AudioGraph courseware files (".aep" files and corresponding HTML documents). The Plug-ins support the playback of presentations in the AudioGraph file format, in Web browsers that support Netscape's Plug-in interface.

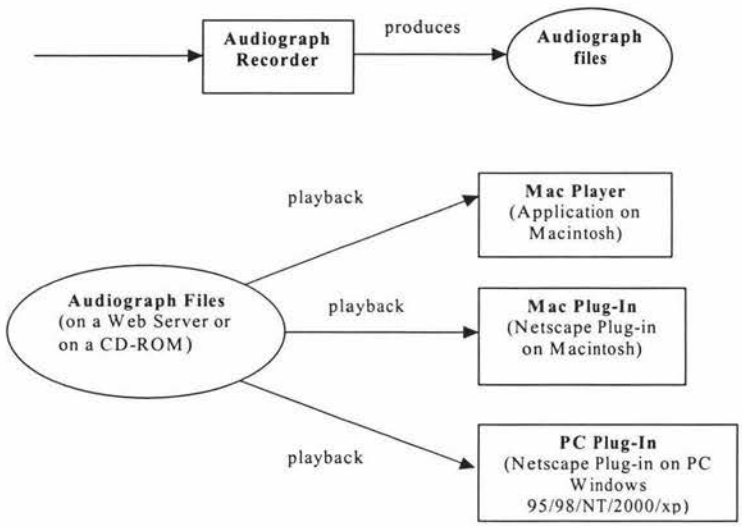

Figure 1.1: Tools AudioGraph

In order to playback AudioGraph courseware, the AudioGraph Plug-In should be put into the "Plugins" directory of Netscape browser or corresponding plug-in directory of other Web Browser that supports the Netscape Plug-in interface. The AudioGraph courseware files can be made available in one of two formats:

- on CD-ROM
- on the Web Server

As show in the following diagram:

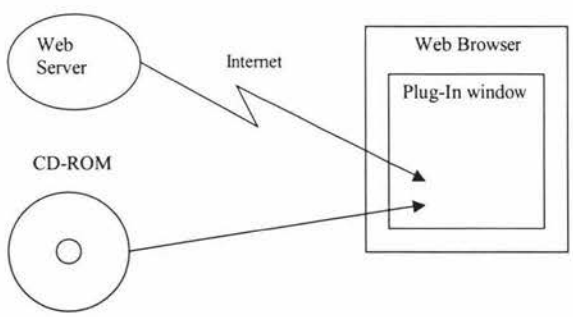

Figure 1.2: Courseware Delivery

This project is concerned with introducing Protection Control into the playback of AudioGraph presentations, so that only authorized or registered users may be able to access the material. Because the playback can be from CD, as well as from the Web, then the normal web-server security techniques are not sufficient to provide the usage

protections that we require. Before the specific issues of Protection Control are discussed, this section looks at techniques and standards used in the AudioGraph Plug-in. In particular the Netscape plug-in interface and two compression standards used in delivering AudioGraph files, namely PNG image compression and GSM sound compression. More details of the AudioGraph project are available in (New Zealand Educational Software, 2000).

1.3 Overview of the Techniques used in the AudioGraph

The PC Plug-in is a tool to playback the recorded AudioGraph courseware on the Web browser. The PC Plug-in extends the supported MIME types of the Web browser in this case to support the playback of the AudioGraph file type. In January of 1996, Netscape Communications Corp. introduced the Plug-in API, which allowed the Web browser to be extended to support user defined MIME types. The Netscape Plug-In API is discussed in this section.

As has been mentioned before, one of the problems with distance, multimedia education is the need to transmit large amounts of courseware over the network, and the available network speed for the 'virtual' student is relatively slow. In order to let the 'virtual' student sit at the end of a slow modem line and still to be satisfied with the delivery speed of the courseware, it is necessary to compress the courseware as much as possible to maintain the user's satisfaction from the quality of sound, text and picture information. The techniques used to compress sound and picture are also reviewed in this section.

1.3.1 Netscape Plug-In API

Netscape's Plug-In API was first introduced by Netscape Communicator corp. along with the beta release of its Navigator 2.0 Web browser (Oliphant, 2000). Since then, many major companies have accepted this standard and have created plug-ins to support their file formats. Now, both the Netscape Navigator and Microsoft Internet Explorer support Netscape's Plug-In API. (Note: for some reasons, Microsoft no longer supports Netscape's Plug-In API after its Internet Explorer 5.5 SP2).

The Web browser is related to the Multipurpose Internet Mail Extensions (MIME). In the early days of ARPANET, email comprised only text messages. Nowadays email on the Internet includes audio, video message and support for other languages like French, Russian and Chinese as well. The MIME standard was created for supporting data types including images, audio, video, binary data, fonts, and text in the electronic mail. MIME is now widely used not only in email but also over the Internet to identify any file that can be transmitted over a network. Today's Web browser supports many MIME types. HyperText Makup Language (HTML) is one of the most popular MIME type. The MIME type is usually associated with one or more file extensions. For instance, a HTML files will have an extension of .HTML or .HTM. When a file is sent over a network, the server uses a MIME table to associate a MIME type with the file extension, the MIME type is then sent with the data stream over the network. Our AudioGraph files will have a file extension ".aep" and MIME type "application/vnd.audiograph". A Netscape Plug-in will extend the supported MIME types of a Web browser to include this type. When the corresponding Plug-in is registered with the browser, the browser extends its capability to the new MIME type. The complete description of MIME is in RFC1341 (Borenstein & Freed, 1992) and RFC1521 (Borenstein & Freed, 1993).

In Windows 95/98/NT, a Plug-in is implemented as a Dynamic-Link-Library (DLL) that has a name starting with NP. The MIME type information for a Plug-in is defined by the file version resource information. In the file version resource, the resource key of MIMEType defines the MIME type, and the resource key of FileExtents defines the associated file extension of the MIME type. In the version resource, some other information can be defined, for instance, the ProductName will define the name of the Plug-In, and the FileDescription will give a description of the Plug-In.

A Windows Plug-In DLL has only three official library entry points: NP_GetEntryPoints, NP_Initialize and NP_Shutdown. Inside the NP_GetEntryPoints definition, more plug-in entry points are defined. The NP_Initialize will initialize the DLL, and NP_Shutdown is called before the DLL is shutdown. Netscape has designed entry points of the Plug-in to simplify the cross-platform code. The Plug-in implemented entry points, or methods as Netscape calls them, are prefaced with NPP_. Table 1.1 is the currently available Plug-in implemented methods.

Table 1.1: Plug-In implemented method (Netscape Communications Corporation, 1997a)

Method Name	Description
NPP_Destroy	Deletes a specific instance of a plug-in.
NPP_DestroyStream	Tells the plug-in that a stream is about to be closed or destroyed.
NPP_GetJavaClass	Returns the Java class associated with the plug-in.
NPP_GetValue	Allows Communicator to query the plug-in for information.
NPP_HandleEvent	Delivers a platform-specific window event to the instance.
NPP_Initialize	Provides global initialization for a plug-in.
NPP_New	Creates a new instance of a plug-in.
NPP_NewStream	Notifies a plug-in instance of a new data stream.
NPP_Print	Requests a platform-specific print operation for an embedded or full-screen plug-in.
NPP_SetValue	Sets information about the plug-in.
NPP_SetWindow	Tells the plug-in when a window is created, moved, sized, or destroyed.
NPP_Shutdown	Provides global deinitialization for a plug-in.
NPP_StreamAsFile	Provides a local file name for the data from a stream.
NPP_URLNotify	Notifies the instance of the completion of a URL request.
NPP_Write	Delivers data to a plug-in instance.
NPP_WriteReady	Determines maximum number of bytes that the plug-in can consume.

The Navigator-implemented Plug-in methods are prefaced with NPN_. Table 1.2 is the currently available Navigator-implemented Plug-in methods.

A detailed description of the Netscape Plug-in API is available from Web Site of Netscape Communicator Corp. (Netscape Communications Corporation, 1997b).

Table 1.2: Navigator-implemented Plug-In method (Netscape Communications Corporation, 1997a)

Method Name	Description
NPN_DestroyStream	Closes and deletes a stream.
NPN_ForceRedraw	Forces a paint message for a windowless plug-in.
NPN_GetJavaEnv	Returns a pointer to the Java execution environment.
NPN_GetJavaPeer	Returns the Java object associated with the plug-in.
NPN_GetURL	Requests Communicator to create a stream for the specified URL.
NPN_GetURLNotify	Requests creation of a new stream with the contents of the specified URL; gets notification of the result.
NPN_GetValue	Allows the plug-in to query Communicator for information.
NPN_InvalidateRect	Invalidates specified drawing area prior to repainting or refreshing a windowless plug-in.
NPN_InvalidateRegion	Invalidates specified drawing region prior to repainting or refreshing a windowless plug-in.
NPN_MemAlloc	Allocates memory from Communicator's memory space.
NPN_MemFlush	Requests that Communicator free a specified amount of memory.
NPN_MemFree	Deallocates a block of allocated memory.
NPN_NewStream	Requests the creation of a new data stream produced by the plug-in and consumed by Communicator.
NPN_PostURL	Posts data to a URL.
NPN_PostURLNotify	Posts data to a URL, and receives notification of the result.
NPN_ReloadPlugins	Reloads all plug-ins in the `Plugins` directory.
NPN_RequestRead	Requests a range of bytes for a seekable stream.
NPN_SetValue	Sets windowless plug-in as transparent or opaque.
NPN_Status	Displays a message on the status line of the browser window.
NPN_UserAgent	Returns Communicator's `user agent` field.
NPN_Version	Returns version information for the Plug-in API.
NPN_Write	Pushes data into a stream produced by the plug-in and consumed by Communicator.

After successfully building a Netscape Plug-in DLL, to install it, just copy the DLL files into Netscape's Plug-in subfolder – the *Plugins* subfolder. When Netscape Navigator is launched, it enumerates all the files whose name start with NP in the *Plugins* subfolder and queries each Plug-in for its MIME type, description and suffixes. Once Netscape Navigator is running, the user can see what Plug-ins are available by selecting the *Help | About Plug-ins* menu, then all active Plug-ins including their MIME type are displayed on the Netscape Navigator. Figure 1.3 show MIME type information of PC Plug-in of AudioGraph from Netscape Navigator:

Figure 1.3: MIME type information of AudioGraph PC Plug-in from Netscape Web Browser

The MIME type must be added to the Web server so that the Web server can recognize this MIME type and files of this type based on the file extension. When a Web browser requests a file, the Server responds to this by sending the MIME type of the file and then streaming file's data to the browser.

The PC Plug-in of AudioGraph is an embedded plug-in. When a Web browser makes a request to the Web Server, the AudioGraph HTML document must contain a TAG like:

```
<EMBED SRC="bvn1.aep" WIDTH="663" HEIGHT="492" ALIGN="BOTTOM">
```

to request the display of AudioGraph courseware file "bvn1.aep" using the Plug-in. This causes the browser to request file "bvn1.aep" from the Web server and the server firsts looks-up the MIME type based on the file extension. Then, if it finds the file extension ".aep" has been entered in the MIME type table, it starts sending file "bvn1.aep" with the MIME type to the browser. The browser then checks whether it can handle this type. It will find that the file with extension ".aep" will be handle by a Plug-in DLL "NPaep.dll", then it will load the DLL to handle this file. The result is to playback the

AudioGraph course material in file "bvn1.aep" within the region specified inside the <embed> tag.

1.3.2 PNG image compression

The Portable Network Graphics (PNG) format was designed to replace the older and simpler GIF format and, to some extent, the much more complex TIFF format. It is used in two major fields: on displaying images over the Web as well as in editing and storing images. It provides a portable, legally unencumbered, well-compressed, well-specified standard for lossless, bitmapped image files.

The PNG supports three image types: true-colour (up to 48 bits per pixel), grayscale (up to 16 bits per pixel) and 8 bits palette-based images. GIF only supports palette-based image types, and JPEG only supports true-colour and grayscale image types. Compared with GIF, PNG has four advantages: full alpha channels (i.e. variable transparency, GIF has only binary transparency), image gamma correction (supports automatic cross-platform control of image brightness/contrast), a faster initial presentation using a two-dimensional interlacing (a method of progressive display), and straightforward detection of file corruption. PNG also compresses around 5% to 25% better than GIF in almost every case. However, GIF supports multi-images, but PNG only support a single image format. The following section overviews some of the technical details of PNG, the full specification of PNG is available on (Boutell & Lane, 1996).

File Structure

A PNG file consists of a *File Signature* followed by a series of *Chunks.*

The File Signature contains the following 8 byte fixed value (given here as a decimal encoding): 137 80 78 71 13 10 26 10. Its purpose is to detect file corruption due to file transmission over the Network especially across different systems, where different standards, such as for end of line markers, may have been automatically applied to the data.

The structure of each *Chunk* in a PNG file is the same, it consists of four parts:

1. *Length*: 4-bytes unsigned integer, giving the number of bytes in the *chunk data* field of a chunk.
2. *Chunk type*: 4-bytes, giving the type of a chunk.
3. *Chunk data*: the actual data appropriate to the chunk type, variable length. The byte number of this field is giving by the *Length* field of this chunk.
4. *CRC*: the cyclic redundancy check calculated on *chunk type* code and *chunk data* field.

The chunk series will start with IHDR chunk and end with IEND chunk. There are two different chunk categories: Critical Chunk and Ancillary Chunk. Some of the critical chunks are as following:

- IHDR – Image header
- PLTE – Palette chunk
- IDAT – Image data
- IEND – Image trailer

Some of the Ancillary chunks are as following:

- bKGD – Background colour
- cHRM – Primary chromaticities and white point
- gAMA – Image gamma
- hIST – Palette histogram
- iCCP – Embedded ICC profile
- pHYs – Physical pixel dimemsions
- sBIT – Significant bits
- sPLT – Suggested palette
- sRGB – Standard RGB colour space
- tEXt – Textual data
- tIME – Image last-modification time
- tRNS – Transparency
- zTXt – Compressed textual data

Compression

PNG compression method 0 is the only compression method currently supported. It is a Lempel-Ziv 77 (LZ77) (Emeterio, 1999) derivative used in the popular zip, gzip, pkzip. It uses the zlib compression engine which is defined in RFC1950 (Deutsch, 1996).

To make the image compression better, compression filters may be used to transform the image before it is compressed. Each horizontal line can be associated with one of the five possible filters. The filter itself doesn't reduce the file size of the image, but it can improve the compressibility. The article in (Roelofs, 1996a) gives an example – an extreme case – to reduce image file size by using a compression filter:

"a 512 x 32,768 image containing all 16,777,216 possible 24-bit colours compressed **over 300 times better** with filtering than without. The uncompressed image was 48 MB in size; the compressed-but-unfiltered version was around 36 MB; but the filtered version is only 115,989 bytes (0.1 MB)."

In PNG, the compression method and compression filter is specified in the IHDR chunk.

Alpha Channels

An Alpha channel is used to represent the transparency of a pixel. All of the three image types can use alpha values, but it is most often used with true-colour or grayscale images. For true-colour images, a pixel with an alpha channel is represented as RGBA (Red, Green, Blue, Alpha). An alpha value of zero means full transparency, i.e., the image is fully transparent (invisible) and all of the background is fully visible. An alpha value of $2^{bitdepth-1}$ represents a fully opaque pixel. Therefore the user can use the Alpha channel to combine the displayed image with the background image and to yield a composite image. Compared with GIF's support of only fully transparency or fully opaque pixels, PNG's variable transparency is a great advantage to the user.

Gamma Correction

Images on different computer monitors may have different setting for brightness/contrast, for instance, a Macintosh generated image tends to look too dark on PC, even a image generated on a PC may not look right on other PC.

In PNG, the gAMA chunk is used to specify a power function for relating the desired display output. In the gAMA chunk, a gamma value is specified and the display output intensity and sample value has the following relation:

```
Sample = light_out ^ gamma
```

However, the gamma value is only an approximation of the display device. The better approximation is to use the so-called chromaticity values, which are supported in PNG by a cHRM chunk. The best solution is to use the Colour Management System, which is supported in PNG by a sRGB chunk.

Interlacing

PNG supports interlacing. The user can decide to store an image file in interlaced order to allow progressive display. The interlacing method 0 is actually no interlacing, it just stores the pixels in a scanline left to right, and the scanlines from top to bottom. The interlace method 1 is known as Adam7. Adam7 consists of 7 distinct passes over the image, each pass transmits a subset of the pixels in an image according to the following 8×8 pattern over the whole image:

```
1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
```

The number in the above pattern means the pass number to transmit this pixel. By using this interlacing method, the features of the image are presented faster. Usually any embedded text is readable after pass two and that only requires transmitting 2/64 of the whole image.

File Integrity Checks

PNG uses three methods for file integrity check: *File Signature*, *CRC-32* and *Adler-32 Checksum*.

The *File Signature* is a fixed 8 bytes containing the following value:

```
Decimal: 137   80   78   71   13   10   26   10
ASCII  : \211  P    N    G    \r   \n   \032 \n
```

The purpose of this *File Signature* is to quickly check if the file is corrupted due to transmission in Text mode. For example, over a Unix system, the "\r" byte in the *File Signature* will be lost and over a Macintosh system, the \n byte will be lost. In this way, the decoder will be able to detect the corruption of a file by just reading these 8 bytes.

The *CRC-32* checksum is a 4-bytes field (32 bits) in every chunk of a PNG file. The *CRC-32* algorithm is defined by ISO3309 and is popular used in network protocol for the purpose of error detection. In the PNG format, the CRC-32 field is calculated over the chunk type and chunk data inside a chunk. With CRC-32, the decoder of the PNG file will be able to detect any error in the chunk type or chunk data.

The *Adler-32 checksum* is calculated over the uncompressed stream data to make sure the integrity of the whole raw image data stream. It is a lower level checksum.

This section introduces some aspect of PNG format, the interested reader can find more details about PNG images in (Roelofs, 1996b).

1.3.3 GSM sound compression

The GSM sound compression algorithm is the sound compression algorithm adopted in Global System for Mobil telecommunication (GSM). Its original purpose is to compress speech signals for digital cellular phones. Now GSM has been used in many multimedia software applications to compress sound messages (Degener, 2000).

The GSM sound compression algorithm is a lossy, telephone-quality sound compression algorithm. The algorithm will compress a buffer of 160 bytes of raw sound data block into a GSM frame which is 32.5 bytes long. In practice, the GSM encoding algorithm usually takes 320 bytes of raw sound data block and compresses it into a double GSM frame which is 65 bytes long.

More detail on GSM speech compression algorithm is available from article on (Degener & Bormann, 2000a). The source code of the GSM algorithm in C language is unlicensed and freely available in (Degener & Bormann, 2000b). The source code in Java language is also freely available in (Degener & Bormann, 2000c) but, unlike the C library, the Java code is licensed under the Free Software Foundation's General Public License.

1.4 Introduction to Computer Security

Before the widespread use of computer systems and computer networks, sensitive information within an organization was held in the format of paper files stored in filing cabinets, as well as in the minds of the staff. Therefore information security was provided by physical means such as a lock to protect the paper files, and the administrative means to hire and manage reliable people.

With the widespread use of computer systems, the requirement to protect sensitive data, software and other valuable assets within the computer systems of an organization has become evident. The generic name for the collection of tools designed to protect data and to thwart hackers is *Computer Security*. The following introduces some concepts concerned with computer security.

1.4.1 Security Service

There is no standard definition of security services. In the opinion of the author, the definitions in (Stallings, 1995b) are accurate and complete. According to (Stallings, 1995b), the security service can be classified as:

Confidentiality: Requires that the information in a computer system and transmitted information be accessible for reading by only the authorized parties.

Authentication: Requires that the origin of a message be correctly identified, with an assurance that the identity is not false.

Integrity: Requires that computer-system assets and transmitted information can only be modified by authorized parties.

Nonrepudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

Access Control: Requires that access to information resources be controlled by or for the target system.

Availability: Requires that computer system assets be available to authorized parties when needed.

1.4.2 Security Attack

A security attack is the action that compromises the security of information in the computer or on the network. Security attacks can be classified into two major categories: active attack and passive attack.

An active attack includes *interruption*, *modification* and *fabrication*. *Interruption* is an attack on *availability*, example includes cutting off a communication line between the source and destination so that the information is not available by the receiver. *Modification* and *fabrication* are attacks on *integrity*. Examples include modifying the contents of the message on the network, insertion of spurious messages in a network.

Passive attack includes *Interception* and *traffic analysis*. *Interception* is an attack on *confidentiality*, e.g., the unauthorized party gains access to an asset for example a password file stored in the system diskette.

The party that makes the security attack may be a person or a piece of software (include attack tool, worm, virus etc.).

1.4.3 Cryptography in Computer Security

Cryptography plays a major role in computer security. Cryptographic algorithms serve nearly all the security services. Using a cryptographic algorithm to encrypt the sensitive information means that only the parties who have the correct key can decrypt the information and read it, therefore *Confidentiality* is achieved. In the UNIX system, *Access Control* is achieved by a password system which uses a variant of DES encryption algorithm. Security Services are achieved by using *cryptographic protocol*. *Cryptographic protocol* is the protocol that uses cryptography algorithm to archive a security task. A *cryptographic protocol* is a series of steps, involving two or more parties, designed to accomplish a security task. There are many different cryptographic protocols, designed to achieve different security services. For *Cryptographic protocol*, the interesting readers can refer to (Schneier, 1996).

In cryptography, the original message is called *plaintext* (or *cleartext*). The process of disguising a message so that its substance is hidden is called *encryption*. After being encrypted, the message becomes *ciphertext*. The process of turning ciphertext back into plaintext is called *decryption*. In the past, the security of an algorithm was based on keeping the algorithm secret, i.e., keeping the mechanism of the algorithm secret. The problems with secret algorithms are obvious, for instance, when someone accidentally reveals the secret of the algorithm, everyone using that algorithm must change it. It would take much time to choose a good algorithm and to changeover to that algorithm. Another problem with secret algorithms is that every product that does not share the secret should use their own secret algorithm, as a result, there would be too many different algorithms, and that makes it difficult to manage. Modern cryptography solves this problem by using a key, where the security of the cryptography depends on the secret of the key. Following diagram shows the encryption and decryption model with keys.

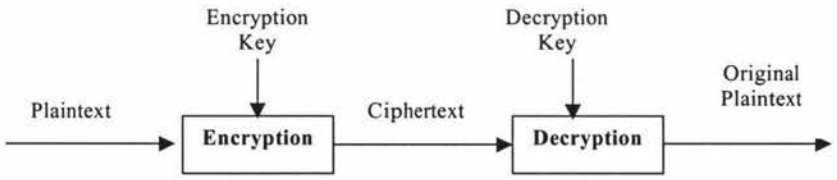

Figure 1.4: Encryption and Decryption with a key

There are two general categories of encryption algorithms: *conventional encryption* and *public-key encryption*.

In conventional encryption, the encryption and decryption use the same key. One of the most important encryption algorithms in the history of modern cryptography is the Data Encryption Standard (DES) algorithm. DES is a conventional block encryption algorithm using a 56 bits length key. It was adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as a standard cryptography algorithm. DES has been a worldwide standard for over 20 years, and it is nearly at the end of its life.

Public-key encryption is the greatest and perhaps the only true revolution in the entire history of modern cryptography. The concept of public-key encryption was invented by Whitfield Diffie and Martin Hellman in 1976 (Diffie & Hellman, 1976). In a public key encryption algorithm, encryption and decryption use different keys, one key is called the *public key* and it can be known to the public (that is why this kind of encryption algorithm is called public-key encryption). The public key is used by the party who want to encrypt the information, but any party who only knows the public key can not decrypt the ciphertext. The other key is the *private key,* and it is used to decrypt the ciphertext encrypted by the corresponding public key. The public key and private key should be used as a key pair, only the correct private key can decrypt the ciphertext encrypted by its corresponding public key. The purpose of a public-key cryptography algorithm is to hide the information in the plaintext. That is to say, the opponent parties can have access to the ciphertext and public key as well, but for the opponent, it should be very difficult to guess the contents of plaintext and private key by studying the ciphertext, public key and encryption algorithm.

1.4.4 Security of Encryption Algorithm and Cryptanalysis

The encryption algorithm should be strong enough to withstand the attack from cryptanalysts. The security degree of a cryptographic algorithm depends on how hard it is to break it. If the cost to break the algorithm is greater than the value of the encrypted data, then the algorithm is possibly strong enough. If the time to break the algorithm is

longer than the encrypted data must remains secret, then the algorithm is possibly strong enough. If the amount of data encrypted by a key is less than the required data to break the algorithm, then the system using this algorithm is possibly safe. However, there is always a chance of new breakthroughs in cryptanalysis. Therefore, none of the cryptographic algorithm is ever absolutely secure. To study security, one should also study the attack on the security, therefore some aspects of attack on encryption algorithms are discussed below.

According to Lars Knudsen (Knudsen, 1994), breaking an algorithm means:

(1) *Total break.* A cryptanalyst finds the key. Therefore the cryptanalyst can decrypt all the ciphertext using this key.

(2) *Global deduction.* A cryptanalyst finds an alternate algorithm without knowing the key. The alternate algorithm is equivalent to the decryption operation. In this way, the cryptanalyst can also get the plaintext of corresponding ciphertext.

(3) *Instance (or local) deduction.* A cryptanalyst finds the plaintext of an intercepted ciphertext.

(4) *Information deduction.* A cryptanalyst gains some information about the key or plaintext. This information could be a few bits of the key, some information about the form of the plaintext, and so forth. This information may be used in the future to break the algorithm.

In the cryptanalysis of an algorithm, it is always correct to assume that a cryptanalyst has knowledge of the encryption algorithm. Apart from the algorithm, the cryptanalyst needs to know some information about the ciphertext and plaintext before he/she will be able to break the algorithm. According to what the cryptanalyst known, the attack can be classified as:

(1) *Ciphertext only attack.* The cryptanalyst has got some ciphertext encrypted by the algorithm using the same key. The task of cryptanalyst is to recover as much as possible the plaintext, or even to deduce the key (*Total break* of the algorithm) in order to decrypt other ciphertext.

(2) *Known plaintext attack.* The cryptanalyst has got some piece of the ciphertext. Apart from that, the cryptanalyst also has access to one or more plaintext-ciphertext pairs

formed with the same key. The cryptanalyst's job is to find out the key used to encrypt the message.

(3) *Chosen plaintext attack.* The cryptanalyst has got some ciphertext. Apart from that, the cryptanalyst can also chose the plaintext to be encrypted and its corresponding ciphertext formed with the same key. The purpose of cryptanalyst is to find out the key.

(4) *Chosen ciphertext attack.* The cryptanalyst has got some ciphertext. Apart from that, the cryptanalyst can also chose purported ciphertext, together with its corresponding decrypted plaintext. The purpose of cryptanalyst is to deduce the key.

(5) *Chosen text attack.* The cryptanalyst has got some ciphertext to be decoded. Apart from that, the cryptanalyst can also chose a plaintext message together with its corresponding ciphertext, as well as chose purported ciphertext together with its corresponding decrypted plaintext. The purpose of cryptanalyst is to deduce the key

There are at least two other types of cryptanalytic attack:

(6) *Chosen-key attack.* The cryptanalyst has got some ciphertext to be decoded. Apart from that the cryptanalyst has some knowledge about the relationship between different keys. The purpose of the cryptanalyst is to find out the key. Some recent developed cryptanalysis deploys the weakness on key-schedule using chosen-key attack.

(7) *Rubber-hose cryptanalysis attack.* This is an attack on key management. The cryptanalyst get the key by using threats, blackmail, or torture someone or by bribery. These are all very powerful attacks and are often the best way to break an algorithm.

Most of the current conventional encryption algorithms are designed to withstand a *known-plaintext attack.* Only relatively weak algorithms fail to withstand a *ciphertext-only attack.*

All the cryptographic algorithms are breakable in a ciphertext-only attack, simply by exhaustively trying every possible key until a meaningful plaintext is obtained. This is called a *brute-force attack.* The number of operations required to break an algorithm by

brute-force attack depends on the length of a key. A key length of 56 bit of DES is too short to withstand a brute force attack by today's powerful computer (Hellman, 1976).

Cryptanalysis is a fast-moving field. Recent advances in cryptanalysis have revealed many weaknesses in some of the most recently developed cryptographic algorithms. One of the most significant advances in cryptanalysis in recent years is *differential cryptanalysis* (Biham & Shamir, 1990, 1992). Other recent developed cryptanalysis methods include *Linear Cryptanalysis* (Matsui, 1993), *Related-key cryptanalysis* (Biham, 1994), *differential-linear cryptanalysis* (Langford & Hellman, 1994).

1.5 Project Definition - Security Consideration of AudioGraph Multimedia Teaching Tools

The AudioGraph is a project developed in the Institute of Information Sciences and Technology of Massey University as copyrighted software tools. The AudioGraph courseware files produced by the AudioGraph Recorder may also be copyrighted. In order to protect the products, there should be some mechanism for preventing an unauthorized access of the courseware files as well as the software tools. This section discusses the security requirement and its associated aspects.

1.5.1 Security Requirement of AudioGraph

To protect the AudioGraph tools and courseware files, it is required that only the authorized user may be allowed to use the AudioGraph tools and access the courseware materials produced by AudioGraph Recorder. The courseware materials are on a CD-ROM or on a Web Server. It should not be easy for the unauthorized user to attack the system in order to access or make a copy of them.

For a courseware file (an .aep file) on a CD-ROM or on the Web Server, there must be some kind of information to mark the file, we call this marking information a "Copy Protection record" in the AudioGraph file. When a user wants to playback the course material in the .aep files via a AudioGraph Player (i.e., a PC Plug-in for the web browser), he/she must provide an identity called an "userId" and a correct password that

corresponds to the userId, otherwise he/she will not be allowed to play it back and the AudioGraph Player will exit execution. Figure 1.5 demonstrates this procedure.

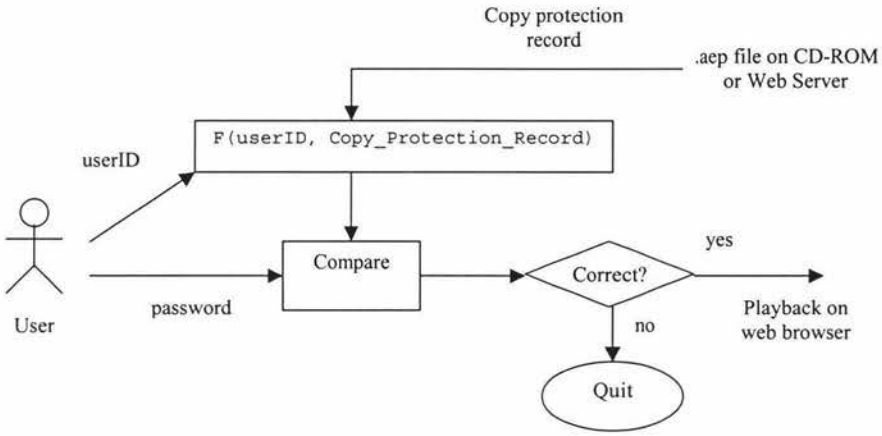

Figure 1.5: Usage Control of .aep file

In above diagram, 'F' is a one-way function, the output of the function is:

```
F(userId, Copy_Protection_Record)
```

The system will compare the output with user's input password, and see if:

```
F(userId,  Copy_Protection_Record) = password
```

If they are equal, then the user is allowed to playback the presentations in courseware file, otherwise, the user is not allowed to playback the presentations and the system will quit.

1.5.2 Copy Protection Record in the AudioGraph Files (.aep files)

The Copy Protection record is embedded in the AudioGraph files (.aep files), it is inserted into the AudioGraph files by a specific tool or by AudioGraph Recorder when it generated the course materials. The Recorder does not need to know about the userId, it must however, generate a unique key for a given unit of presentation, whatever the user decides this may be. The Copy Protection record includes three keys:

Company Key: 4-bytes, to give a unique identity of the company who produces the AudioGraph courseware product

Lower protection key: 4-bytes, used as a protection key

Upper protection key: 4-bytes, used as a protection key.

On encountering this record, the PC Plug-in without the security system embedded in it will terminate with a message asking the user to contact the supplier for a correct Player Plug-in version. For the PC Plug-in with the security system embedded in it, it will ask user to input userId and password and do a security check.

1.5.3 Other consideration of AudioGraph Multimedia Teaching Tools and Courseware files

The protection of AudioGraph software and courseware files from illegal copying depends on the security of the Web Server as well as the administration of the AudioGraph materials, including source code of the software, the Tools and courseware materials.

For the Web Server, it should withstand any kind of attack from intruders. An intruder may try to copy the source code of any AudioGraph materials and study them, and try to make use of them, he/she may possibly try to find out weakness of the AudioGraph security system and learn how to completely attack on them. The Web Server security is beyond the scope of this thesis. Interesting reader can refer to books about Web Security (Garfinkel, 1997; Stein, 1998).

Another important aspect of AudioGraph security is the administration of the AudioGraph materials. No matter how secure the security system and Web server are, if the management of user name and password file has a security bug, then the whole security system is useless. For instance, if the intruder gets access to a file storing all the user name and password pairs due to bad management, then the intruder can access and pass all the security checks which need the user name and password. In another situation, if an intruder is an expert on security programming and gets access to the source code of AudioGraph due to bad management of source code, he/she can study the source code and find out weakness of the security system, or even write his/her own tool such as a variant of PC Plug-in to bypass the security check, and that will make our security system worthless.

1.6 Summary

This chapter has covered the background area concerned with the Security System of AudioGraph project and given the project definition. The PNG image compression and GSM sound compression algorithms have been discussed. The basic concepts concerned with computer security and cryptography have been described. In the following chapter some of the popular used encryption algorithms will be discussed in more detail.