Authentication for Multimedia Documents

Fan Chen and Ernst L. Leiss
Department of Computer Science
University of Houston
Houston Texas 77204-3475
coscel@cs.uh.edu

Abstract

We discuss problems of data security for multimedia documents. While many techniques have been developed for text data, the more interesting aspects of multimedia, namely still images and video, require very different approaches because the data files are several orders of magnitude larger and it is consequently very wasteful of computing resources if one must decrypt the entire file in order to use it.

Computer-based multimedia systems allow the integration of various types of information in a computer. In addition to traditional textual and numeric data, other types of information including graphics (bit-mapped or vector-based), pixel maps, audio- and video clips can be handled in such a multimedia system in digital form. Successful applications include real-time conferencing, group working, multimedia electronic mail, computer-aided distance learning, and interactive video [WILL91]. This paper focuses on security concerns in multimedia, especially authentication, a relatively neglected aspect of multimedia.

Multimedia data can be subverted in several ways. Of interest for us is the fact that transmitted data can be easily copied, distorted, or subverted in other ways by an intruder. For example, a picture sent via satellite it is accessible to anyone who has a TV set and an appropriate antenna [KWYU89] and can be intercepted and subverted during its transmission.

We will address only authentication and verification of multimedia information. While authentication of text is well known, that of image and video information has been ignored in current research area even though image and video are most important in multimedia information. We outline authentication schemes for all three. Sound is not explicitly covered here, but the schemes for still image authentication can be directly applied.

Attacks on the security of a system are best characterized by viewing the function of the system as providing information. There is a flow of information from a source, such as a file or a region of main memory, to a destination, such as another file or a user. This normal flow is depicted in Figure 1a. The other parts of the figure show four general categories of attack [BHAS93]:

- *Interruption*: An asset of the system is destroyed or becomes unavailable or unusable. Examples are destruction of hardware, cutting communication lines, or disabling the file management system.

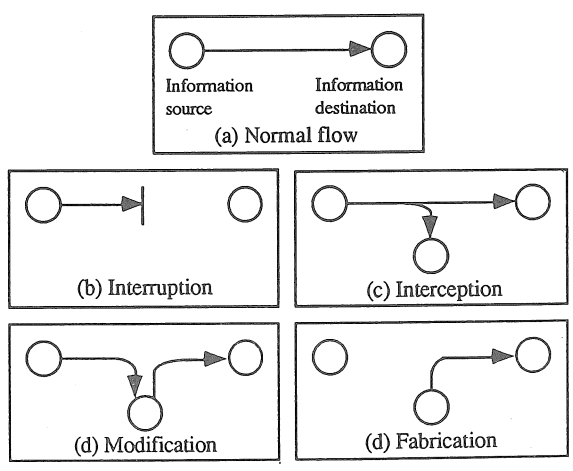


Figure 1. Security Threats

- *Interception*: An unauthorized party gains access to an asset. The unauthorized party could be a person, a program, or a computer. This is an attack on confidentiality. Examples are wiretapping to capture data in a network and illicit copying of files.
- *Modification*: An unauthorized party tampers with an asset. A message or some portion of the message is altered, delayed or recorded, to produce an unauthorized effect. This is an attack on integrity. Examples are changing values in a data file, replacing an image with a fake one in multimedia file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.
- *Fabrication*: An unauthorized party inserts complete counterfeit objects into the system. This is an attack on authenticity. Examples are inserting spurious messages in a network or adding records to a file.

We list the major security issues.
Confidentiality: The protection of transmitted data from passive attacks.
Authentication:The assurance that a communication is authentic.
Integrity: The assurance that the information was not modified.
Nonrepudiation: Nonrepudiation prevents either sender or receiver from denying a sent message. Thus, the receiver of a message can prove that it was in fact sent by the alleged sender.

In this paper, we outline methods of how to detect a modification, distortion, or subversion of a multimedia document at the receiver's end. To this end, we present algorithms for generating an authentication code and authentication protocols for text, still image, and video so that the receiver can do the verification and authentication of the received data.

We first give an overview of approaches to text authentication; then authentication algorithms for still image and video are presented. We distinguish the following attacks [SHAF94]:

1. *Disclosure of messages*.
2. *Traffic analysis*:
3. *Masquerade*: Insertion of messages into the network from a fraudulent source.
4. *Content modification*:
5. *Sequence modification*: Any modification in the sequencing of messages.
6. *Timing modification*: Delay or replay of messages.
7. *Repudiation*:

Measures against the first two attacks are in the realm of message confidentiality. Measures against attacks 3 through 6 are generally message authentication. Mechanisms against attack 7 come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all the attacks 3 through 6.

Authentication Functions for Text

Any message authentication mechanism can be viewed as having fundamentally two levels. At the lower level, we have some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as primitive in a higher-level authentication protocol enabling a receiver to verify a message's authenticity.

We outline functions that can be used to produce an authenticator for text message. These functions may be grouped into two classes, as follows:

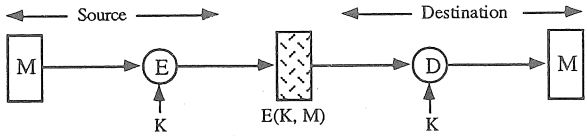
- *Message encryption*: The ciphertext of the entire message serves as its authenticator.
- *Generation function of authentication code:* A public function of the message with a secret key, or a secret function that produces a fixed-length value that serves as the authenticator.

Message encryption by itself can provide a measure of authentication. The analysis differs for conventional and public-key encryption schemes.

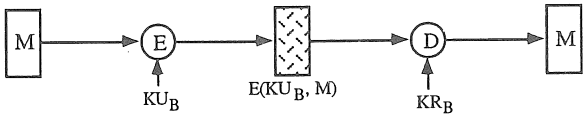
Conventional Encryption

Conventional encryption method, also called symmetric encryption, is the use of the same (or similar) keys for encryption and for decryption: knowing one of the keys enables us to determine the other with little effort. Both keys must be kept secret in conventional encryption. Many conventional encryption methods are available, and one of the best-known is the Data Encryption Standard (DES) proposed by the National Bureau of Standards [LEI82].

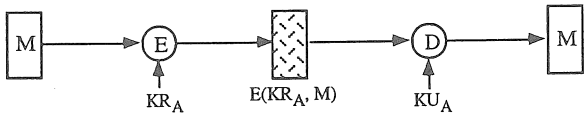
Consider the straightforward use of conventional encryption (Fig. 2a). A message transmitted from source A to destination B is encrypted using a secret key K shared only by sender A and recipient B. If no other party knows K, confidentiality is provided: no other party can recover the plaintext of the message. B is assured that the message was generated by A since A is the only other party that possesses K and thus the only other party with the information necessary to construct ciphertext that can be decrypted with K. Also, if M is recovered, B knows that no bit of M has been altered, because an opponent that does not know K would not know how to alter bits in a ciphertext to produce desired changes in the plaintext. -- Simple public-key encryption (Fig. 2b) provides confidentiality but no authentication. The other schemes in Figure 2 outline the corresponding variations.



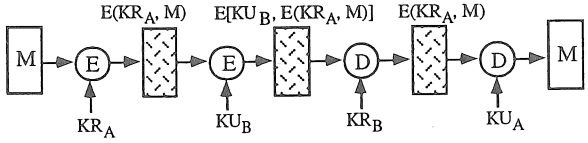
(a) Conventional encryption: confidentiality and authentication



(b) Public-key encryption: confidentiality



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Figure 2. Basic Uses of Message Encryption

So, conventional encryption provides authentication as well as confidentiality. However, consider exactly what is happening at B. Given a decryption function D and a secret key K, the destination will accept any input X and produce the output $Y = D(K, X)$. If X is the ciphertext of a legitimate message M produced by the corresponding encryption function, then Y is some plaintext message M. Otherwise, Y will be a meaningless sequence of bits. There must be some automated means of determining at B whether Y is legitimate plaintext and therefore must have come from A. The implications of this are profound for authentication. Suppose the message M can be any arbitrary bit pattern. In that case, there is no way for the receiver to determine automatically whether an incoming message is the ciphertext of a legitimate message or not. In general, we require that only a small subset of all bit patterns is a legitimate plaintext. For example, if only one in 10^6 bit pattern is legitimate, the probability that any randomly chosen bit pattern, treated as ciphertext, will produce a legitimate plaintext message is 10^{-6}. For a text file, this is obviously true, because text is inherently redundant. Thus, an automatic tool (based on the probabilities of n-grams in a specific language) can be developed to check automatically, i. e., without human intervention, whether the decrypted message is "meaningful". For more details, we refer to [CHEN95].

Message Authentication Code

An alternative authentication technique uses a secret key to generate a small fixed-size block of data, known as message authentication code (MAC) that is appended to the message. It assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key: C(k, M). The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated authentication code (Figure 3a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received authentication code matches the calculated authentication code, then:

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is

assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assumed that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. If the message includes a sequence number (as used with X.25, HDLC, TCP, and the ISO transport protocol), the receiver is assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A MAC generation function is similar to encryption. One difference is that the algorithm need not be reversible, as it must be for decryption. It turns out that because of mathematical properties of the authentication function, it is less vulnerable to being broken than is encryption [DAVI89].

This process provides authentication but not confidentiality, because the message is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure 3b) or before (Figure 3c) the MAC generation algorithm. In both cases, two separate keys are needed, each of which is shared by the sender and the receiver. Typically, it is preferable to tie the authentication directly to the plaintext (Figure 3b).

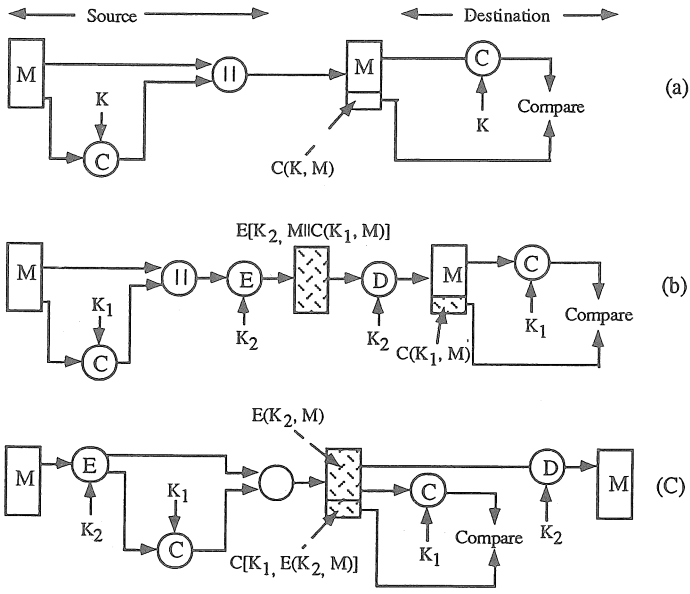


Figure 3. Basic Uses of Message Authentication Function

[DAVI89] suggests three situations in which a message authentication code may be used:
1. The same message is broadcast to a number of destinations, with only one destination responsible for monitoring authenticity. Thus, the message must be broadcast in plaintext with an associated MAC.
2. One party has a heavy work load and cannot afford the time to decrypt all incoming messages.
3. Authentication of a computer program in plaintext is attractive.

Three other rationales may be added, as follows:
4. Messages need not be kept secret, but they must be authenticated. E. g., the Simple Network Management Protocol Version 2 (SNMPv2) separates the functions of confidentiality and authentication.
5. Separation of authentication and confidentiality functions affords architectural flexibility. One may want to perform authentication at the application level but provide confidentiality at a lower level, e. g., transport layer.
6. One may wish to extend the protection beyond the time of reception and yet allow processing of message contents. With message encryption, protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.

Requirements for Message Authentication Codes

A message authentication code (MAC) is generated by a function C of the form $MAC = C(K, M)$ where M is a variable-length message, K is a secret key shared only by sender and receiver, and C(K, M) is the fixed-length authenticator. The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the MAC.

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require 2^{k-1} attempts for a k-bit key. Thus, for a cipher-only attack, the opponent, given ciphertext C, would perform $P_i = D(K_i, C)$ for all possible key values Ki until a Pi was produced that matched the form of acceptable plaintext.

In the case of a authentication code, the considerations are different. The MAC function is many-to-one. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys. If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible messages with $N >> 2^n$. Furthermore, with a k-bit key, there are 2^k possible keys. Using brute-force methods, how would an opponent attempt to discover a key? If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n, i. e., the key size is greater than the MAC size. Then, given a known M1 and MAC1, with $M1 = C(K_1, MAC1)$, the cryptanalyst can perform $Mi = C(Ki, MAC1)$ for all possible key values Ki. At least one key is guaranteed to produce a match of Mi = M1. A number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key! On average a total of $2^k/2^n = 2^{k-n}$ keys will produce a match. Thus the opponent must iterate the attack. On average, a round will be needed if k = a * n. For example, if an 80-bit key is used and the MAC is 32 bits, the first round will produce about 2^{48} possible keys, the second will narrow the possible keys to about 2^{16} possibilities, and the third should produce only a single key, which must be the one used by the sender. Thus, a brute-force attempt to discover the authentication key is no less effort, and may be more effort, than that required to discover a decryption key of the same length.

Other attacks that do not require the discovery of the key are possible. In assessing the security of a cryptographic MAC function, we must consider what attacks may be mounted against it. Assume that an opponent knows the MAC function C but not K; then the MAC function should have the following properties:

1. If an opponent observes M and C(K, M), it should be computationally infeasible for the opponent to construct a message M' such that C(K, M') = C(K, M).
2. C(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K, M) = C(K, M') is 2^{-n}, where n is the number of bits in the MAC.
3. Let M' be equal to some known transformation on M. That is, M' = f(M). For example, f may involve inverting one or more specific bits. In that case, Pr [C(K, M) = C(K, M')] = 2^{-n}.

A number of data authentication algorithms have been proposed. One of the most widely used cryptographic MACs is based on the DES. This algorithm is both a FIPS publication (FIPS PUB 113) and an ANSI standard (X9.17) and appears to meet the requirements specified earlier [SHAF94].

A MAC Based on a Hash Function

A hash value is generated by a function H of the form

$$h = H(M)$$

where M is a variable-length message, and H(M) is the fixed-length hash value. All hash functions operate using the following general principles: the input (message, file, etc.) is viewed as sequence of n-bit blocks and is processed one block at a time to produce an n-bit hash function. The hash value is appended to the message at the source at a time when the message is assumed to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (see Figure 4).

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. For message authentication, a hash function H must have the following properties (see [NECH92]):

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is easy to compute for any given x, making both hardware and software implementations practical.
4. For any given code m, it is computationally infeasible to find x such that H(x) = m.
5. For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).
6. It is computationally infeasible to find any pair (x,y) such that H(x) = H(y).

A hash function that satisfies the first five properties in the preceding list is refereed to as weak hash function. If the sixth property is also satisfied, then it is refereed to as a strong hash function. Various hash functions for data authentication purpose have been proposed. For example, a hash function based on using a cipher block chaining technique was described in [RABI78]. An algorithm called Snefru was developed by Ralph Merkle for providing good security and ease of implementation on 32-bit processors [MERK90]. However, many algorithms still have weaknesses. Much research [JASP95] is being done in this area.

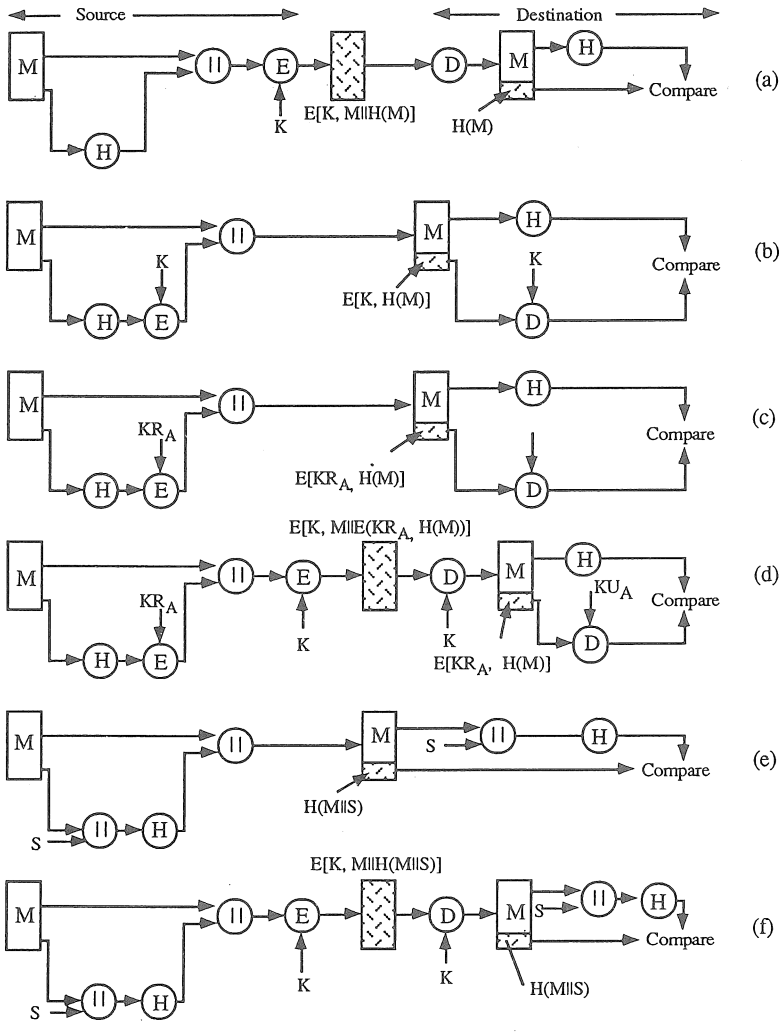


Figure 4. Basic Uses of Hash Functions

Authentication Functions of Image and Video

Usually, image and video files are much larger than text file, so it is impractical to encrypt the entire image and video only for authentication purpose since it would require the sender and the receiver to encrypt and decrypt the message every time a image or a video message is transmitted.

Still Image Authentication

Random Generation

The basic idea of random generation is to choose pixels at random over the whole still image file. The goal is to identify any subversion with a probability approaching 1. We will treat subversions as errors. Thus, a subversion happens when an intruder changes a part (or parts) of a picture such as replacing a head. A subversion usually changes a piece of pixels, but errors could happen in a piece or the errors distribute randomly over the whole image. We assume that the error distribution is even (uniform) on the whole image. We define the error rate r as the average

error per pixel; r = number of errors in the image divided by its total number of pixels. An image is subverted if at least one error is found by checking the witness file. The probability of subversion detection follows the Poisson distribution [JANO88]:

$$P(x;\ \lambda,T)=\frac{(\lambda T)^x\, e^{-\lambda T}}{x!}$$

where T is event internal (number of pixels of the witness file), λ is the arrival rate on average (error rate), x is the number of events (number of errors), and P(x; λ,T) is the probability that x errors are detected. Then, by Poisson's theorem:

$$Pr\ (A\ subversion\ is\ detected) = Pr\ (At\ least\ one\ error\ is\ found) = P(x \geq 1;\ \lambda,T) = 1 - P(0;\ \lambda,T) = 1 - e^{-\lambda T} \quad (1)$$

Suppose a subversion changed 500 pixels in a 640*480=307,200 pixels image. Then $\lambda = 500/640*480 = 0.0016$.

- If the witness file (MAC) is reduced 10,000 times

  $T = 307{,}200 / 10{,}000 \approx 31$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016 * 31} = 0.048$
- If the witness file is reduced 1,000 times

  $T = 307{,}200 / 1{,}000 \approx 308$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016 * 308} = 0.389$
- If the witness file is reduced 100 times

  $T = 307{,}200 / 100 = 3072$ (pixels) and $Pr\ (A\ subversion\ is\ detected) = 1 - e^{-0.0016*3072} = 0.993$

In Formula (1), we see the important relationship between r, T and Pr. We get:

$$e^{-\lambda T} = 1 - Pr \quad or \quad -\lambda T = ln(1-Pr) \quad or \quad \lambda T = -ln(1-Pr) \qquad (2)$$

Thus, given a required probability of subversion detection, its witness file size depends on the error rate. If the error rate is small, the witness file has to be large, if the error rate is large, then the witness file can be much smaller. The witness file size increase with the decrease of λ with the speed of factor ln(1-Pr). Pr is the required probability standard for subversion detection. For instance, if we require the probability of subversion detection needs to be 99.9 percent (acceptable in most cases), then $\lambda T = -ln(1-0.999) = 6.908$. If the error rate r is 0.001, then T = 6.908 / 0.001 = 6908 (pixels). We conclude that the random generation of witness file is a reasonable and acceptable algorithm. Because of (2), we get a higher detection probability and at the same time keep the witness file relatively small.

Transmission Protocol

A message authentication code (witness file) can be used in different ways to provide still image authentication. Suppose A is the sender, B is the receiver. A transmits still image plus authentication code to B. We define:

M:	Original still image file.	M':	Received image file.
C:	Message authentication algorithm (random pixel generator).	C':	Received witness file.
C'':	Generated witness file.	X:	Seed for C.
X':	Received seed.	E:	Encryption algorithm.
R:	Received message.	D:	Decryption algorithm.
K:	Secret key shared by A and B.	KR_A:	A's secret key.
KU_A:	A's public key.	KR_B:	B's secret key.
KU_B:	B's public key.	‖:	Message concatenation

a) $A \rightarrow B:\quad M \parallel E(K, X) \parallel C(X, M)$

When B receives a message M' ‖ R(K,X) ‖ C'(X, M), he does the following:

X' = D(K, R(K,X)); C'' = C(X', M'); if (C'' == C') then "The received image is correct" else "A subversion happened"

In this transmission protocol, only the seed is encrypted using conventional encryption. It is kept secret, so it is computationally infeasible to get M' ≠ M such that C(x, M') = C(X, M) (x is a random seed and x ≠ X). This protocol only provides authentication; no signature is provided. Because K is shared by A and B, B can also "sign" the image.

b) $A \rightarrow B:\quad M \parallel E(KU_B, X) \parallel C(X, M)$

When B receives a message M' ‖ R(KU_B, X) ‖ C'(X, M), he does the following:

X' = D(KR_B, R(KU_B, X)); C'' = C(X', M'); if (C'' == C') then "The received image is correct" else "A subversion happened"

In this protocol, only the (secret) seed is encrypted using public-key encryption. It is computationally infeasible to find M'≠M with C(x, M') = C(X, M) (x is a random seed and x ≠ X). This protocol only provides authentication. Only B can decrypt the seed using his private key. No signature exists because everybody can "sign" a message using B's public key.

c) $A \rightarrow B:\ M \parallel E(KU_B,\ E(KR_A, X)) \parallel C(X, M)$

When B receives a message M' ‖ R(KU_B, E'(KR_A, X)) ‖ C'(X, M), he does the following:

X' = D(KR_A, D(KR_B, R(KU_B, E(KR_A, X)))); C'' = C(X', M');

if (C''==C') then "The received image is correct" else "A subversion happened"

In this protocol, the seed is encrypted first using A's private key, then B's public key to provide secrecy.This protocol provides authentication and signature. Only B can decrypt the seed and only A can "sign" the seed using his private key.

d) A->B: M || E(K, X || C(X, M))

When B receives a message M' || R(K,X || C(X, M)), he does the following:

X' || C'(X, M) = D(K, R(K,X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted using conventional encryption. This transmission protocol only provides authentication, no signature is provided. Because K is shared by A and B, B can also "sign" the image.

e) A->B: M || E(KU_B, X ||C(X, M))

When B receives the message M' || R(KU_B, X || C(X, M)), he does the following:

X'|| C'(X, M) = D(KR_B, R(KU_B, X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted using public-key encryption. This protocol only provides authentication. Only B can decrypt the seed and the witness file using his private key. No signature is provided because everybody can "sign" a message using B's public-key.

f) A->B: M || E(KU_B, E(KR_A, X || C(X, M)))

When B receives a message M' || R(KU_B , E(KR_A , X || C(X,M))), he does the following:

X' || C'(X, M) = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X || C(X, M))))); get X' and C'(X, M); C'' = C(X', M');

if (C'' == C') then "The received image is correct" else "A subversion happened"

The concatenation of seed and witness file is encrypted first using A's private key, then B's public-key to provide secrecy. The seed and witness file are kept secret. This transmission protocol provides authentication and signature. Only B can decrypt the seed and the witness file and only A can "sign" a message using his private key.

When a subversion happens in a picture, the block (or blocks) in the subverted part have higher error rates than others, or higher values than the usual transmission error rate. We divide the whole image block by block, all the blocks have approximately the same size. Each block is looked upon as one single unit. We choose pixels randomly in a block. For the details, we refer to [CHEN95]. We have two standards to define that a image was correct and no subversion happened:

- The error rates for all the blocks are close to a known usual transmission error. E. g., if a network's average transmission error rate is 0.01%, we may define as unacceptable error rates in excess of 0.01%.
- Error rates for all blocks are distributed uniformly, which means that all the error rates are almost the same, or in a small, acceptable range.

If an entire picture gets changed, the first standard can detect this subversion, but the second standard may not. In the following, we use the first standard. Define Pr(i) to be the probability that a subversion is detected in block i, β to be the predefined transmission error rate, and τ to be a defined acceptable small changing range. It follows that the more blocks an image is divided in, the more block detection probability terms are added to the subversion detection probability for the image. The more blocks in an image, the smaller the size of each block, so the witness file for each block gets smaller.

Transmission Protocol

Suppose A is the sender, B is the receiver. A wants to transmit the still image plus the authentication code to B. We define:

M:	Original still image file.	M':	Received image file.
C:	Authentication algorithm (random pixel generator).	m:	Numbers of blocks in the image
C'i:	Received witness for block i ($1 \le i \le m$).	C':	Received witness file.
C":	Generated witness file.	Xi:	Seeds of C for each block.
X'i:	Received seeds of C for each block.	E:	Encryption algorithm.
R:	Received message.	D:	Decryption algorithm.
K:	Secret key shared by A and B.	KR_A:	A's secret key.
KU_A:	A's public key.	KR_B:	B's secret key.
KU_B:	B's public key.	||:	Message concatenation
Mi:	Block i in M. $M_1 \| M_2 \| ... \| M_m = M$	X = X1 || X2 || ... || Xm	
X' =	X'1 || X'2 || ... || X'm	C(X,M)=C(X1,M1)||C(X2,M2)||...||C(Xm,Mm)	
C'(X,M) =	C'(X1,M1) || C'(X2, M2) || ... || C'(Xm, Mm)	C"(X,M)=C"(X1,M1)||...||C"(Xm,Mm)	
β:	Transmission error rate.	τ:	A predefined small range value.

Ri: error rate of block i = the number of different pixels between C"i and C'i / the number of pixels in C'i.

a) *A->B: M || E(K, X) || C(X, M)*

When B receives M' || R(K,X) || C'(X, M), he does the following:

X' = D(K, R(K,X)); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

b) *A->B: M || E(KU_B, X) || C(X, M)*

When B receives M' || R(KU_B, X) || C'(X, M), he does the following:

X' = D(KR_B, R(KU_B, X)); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

c) *A->B: M || E(KU_B, E(KR_A, X)) || C(X, M)*

When B receives M" || R(KU_B, E(KR_A, X)) || C'(X, M), he does the following:

X' = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X)))); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

d) *A->B: M || E(K, X || C(X, M))*

When B receives M' || R(K,X || C(X, M)), he does the following:

X' || C'(X, M) = D(K, R(K,X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

e) *A->B: M || E(KU_B, X ||C(X, M))*

When B receive the M' || R(KU_B, X || C(X, M)), he did the following:

X'|| C'(X, M) = D(KR_B, R(KU_B, X || C(X, M))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > τ) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

f) *A->B: M || E(KU_B, E(KR_A, X || C(X, M)))*

When B receives the M' || R(KU_B, E(KR_A, X || C(X,M))), he does the following:

X' || C'(X, M) = D(KU_A, D(KR_B, R(KU_B, E(KR_A, X || C(X, M))))); get X' and C'(X, M); C'' = C(X', M');
for (i=1;i≤m; i++)
{ Ri = the number of different pixels between C"i and C'i / the number of pixels in C'i;
if (abs(Ri - β) > t) { printf("A subversion happened"); exit(1); }
}
printf("The received image is correct");

The strengths and weaknesses of these transmission protocols are the same as the corresponding ones for random generation method, discussed earlier. Protocols a, b, d, and e only provide authentication, Protocols c and f provide a signature as well as authentication.

Video Authentication

Video authentication methods are similar to still image authentication methods, except that the redundancy between successive frames should be considered.

Random Access Frame by Frame

The basic idea is to choose pixels randomly frame by frame. Each frame is looked at as a single unit. Because in video, there is much redundant information between frames, we must not choose the same pixels in a frame; the seeds of different frames need to be different. Suppose a video has S frames, each frame has M*N pixels, and the witness file is T. Define

C: Random number (index) generator.
X_i: Seed for frame i ($1 \le i \le S$).
T_i: The i-th frame ($1 \le i \le S$).
T_{ij}: The j-th pixel in i-th frame ($1 \le i \le S$, $1 \le j \le M*N$).

Then we define

$T_{i,j+1} = C(X_i, T_{ij}) \bmod (M*N)$ $T_i = T_{i1} // T_{i2} // ... // T_{ij} // T_{ij+1} // ...$ $T = T_1 // T_2 // ... // T_S$

where $X_i \neq X_j$ and $|X_i - X_j| \neq$ the distance of two adjacent pixels chosen by the random generator C for all $1 \leq i,j \leq S$ and $i \neq j$.

Probability of Subversion Detection

Assume the error distribution is uniform over the whole video. The error rate r is the average error per pixel,

r = changed number of pixels / total number of pixels in the video.

We say a subversion happened if at least one error is found by checking the witness file. As before we get

$$P(x; \lambda,T) = \frac{(\lambda T)^x e^{-\lambda T}}{x!}$$

$Pr(\text{A subversion is detected}) = Pr(\text{At least one error is found}) = P(x \geq 1; \lambda,T) = 1 - P(0; \lambda,T) = 1 - e^{-\lambda T}$

Suppose a subversion happened in a video, a part of the video was changed. Assume that 10 pixels were changed per frame on average. The video lasts 2 hours, there are 30 frames per sec, each frame has 640 * 480 = 307,200 pixels. Then,

$\text{error rate } \lambda = 10 / 640 * 480 = 3.26 * 10^{-5}$

- If the witness file is reduced 10,000 times

  $T = 7200 * 30 * 307{,}200 / 10{,}000 \approx 6.64 * 10^6$ (pixels)

  $T_i = 7200 * 30 * 640 * 480 / T \approx 10{,}000$ (pixels) $(1 \leq i \leq S)$

  $Pr(\text{A subversion is detected}) = 1 - e^{(-3.26 * 10^{**}(-5) * 6.64 * 10^{**}6)} \approx 1$

- If the witness file is reduced 100,000 times

  $T = 7200 * 30 * 307{,}200 / 1{,}000 \approx 6.64 * 10^5$ (pixels) $T_i = 100000$ (pixels) $(1 \leq i \leq S)$

  $Pr(\text{A subversion is detected}) = 1 - e^{(-3.26 * 10^{-5} * 6.64 * 10^6)} = 0.999999999$

- If the witness file is reduced 1,000,000 times

  $T = 6.64 * 10^4$ (pixels) $Pr(\text{A subversion is detected}) = 1 - e^{(-3.26 * 10^{-5} * 6.64 * 10^4)} \approx 0.8855$

We conclude that the random generation of witness file is a reasonable and acceptable algorithm. We can get a higher detection probability and at the same time keep the witness file relatively small.

Transmission Protocol

The transmission protocol for the video is the same as that for the still image. In the video, each frame has a different seed, in a video 2 hours long with 30 frames per second, there are 2 * 3600 * 30 = 216,000 frames, so we need 216,000 seeds. Transferring all these seeds with encryption will take much bandwidth. Instead, we may use only a single seed (the seed of the first frame); all the other seeds are generated by a mathematical algorithm or another random number generator. We have

$$X_i = C'(X_{i-1}) \bmod (M*N)$$

where X_i is the seed for frame i ($1 < i \leq$ total frame number), C' is a mathematical algorithm or a random number generator, and M*N is the number of pixels in a frame. In this case, only the seed for the first frame is to be transferred; after the receiver gets the message, he generates other seeds using the same C'.

Random Access with Interframe Compression

In the above random access process, random indices are chosen frame by frame; each frame is looked at as an independent single unit where the seeds for all the frames are different. What happens if we use only a single seed for all frames? It is easy to determine that when one seed is used for all the frames, the positions of selected pixels in a frame are the same as those in all the other frames. For example, if the algorithm is to select pixels with an interval of 100 pixels, the pixels in index positions 1, 101, 201, ..., are picked out in each frame.

There is much redundant information between frames, especially for adjacent frames. Many pixels between adjacent frames are the same. Interframe compression technique always use this feature to eliminate the redundancy, using key frames to store an entire frame, while delta, or difference frames record only interframe changes [WUXY92]. We can develop a single seed random access algorithm with interframe compression to eliminate interframe redundancy: if a pixel selected by a random number (index) generator is the same as the one in the immediately preceding frame, it is ignored; otherwise, it is selected for the witness file.

Suppose a video has S frames, each frame has M*N pixels, X is the seed, C is the random index generator, T is the witness file, P(i,j) is the pixel index position indicating the j-th pixel in the i-th frame. Here is the pseudocode:

```
T = " "; for (i=1; i≤S; i++)
        { P(i,1) = X;
          do if (i==1 || TP(i,j) != TP(i-1,j)) T = T + TP(i,j); P(i, j+1) = C(P(i,j)) mod (M*N); j++
          until p(i,j) = X
        }
```

Another variant takes the first frame as key frame, all other frames as delta or difference frames. In the random access process, for a key frame, we select pixels among the entire frame uniformly, but for a delta frame, we only consider the pixels that are different from the immediately preceding delta or key frame. For details, see [CHEN95].

Probability of Subversion Detection

We define the error rate λ as the average error per pixel; then λ = changed number of pixels in the video / total number of pixels in the video; a subversion happened if at least one error is found by checking the witness file.

Interframe motion is a determining factor for the witness file size. The witness file size for a video with a high motion sequence is much larger than for a video with a lower motion sequence. For low motion sequences such as a "talking head" video, greater interframe compression is obtained and the witness file size gets much smaller; thus, for a given witness file size, the generated witness file contains more information about the video than for a high motion sequence, and therefore, the probability of subversion detection gets higher . So, interframe motion is the enemy of interframe compression, it decreases not only the video quality and display rate, but also the probability of subversion detection given a required size of authentication code.-- The transmission protocol is the same as the one of random generation for still images.

Detection of Uniform Distribution

When a subversion happens in a video, some frames in a row or some part in a frame have a higher error rate. The uniform distribution detection algorithm is based on this feature. Divide each considered frame (reference frame) in a video into some roughly equal size blocks. Considering each block as a single unit, we choose pixels at random block by block and store them into a witness file. Take the first reference frame as key frame, all the other reference frames as delta frames. Because of redundancy between frames, when we select blocks in these reference frames, we only need to consider the blocks that are different from the one of the immediate preceding reference frame in the same position.

Suppose a video has S frames, each frame has M * N pixels, each frame is divided into m blocks, one of every s frames is selected as a reference frame. Then, in total we have S/s reference frames and each block has M*N/m pixels. Define C to be the random number (index) generator, X its seed, B(i,j) the jth block in the ith frame, T(i,j) the block in the witness file generated by B(i,j) ($1 \le i \le S$, $1 \le j \le m$), and T the witness file. Here is the witness file generation process in pseudo code:

```
T = " ";  for (i=1; i≤S; i = i+s)
            {T = T + "i"; for (j=1; j≤m; j++) if (i==1 || B(i,j) != B(i-s,j)) T = T + "j" + C(X, B(i,j)); }
          }
```

The witness file size is determined by the interval s (among how may frames do we select a reference frame?), the number of pixels in each block, the number of blocks, and the random number generator which decides how many pixels are chosen for the witness file. Again note that high interframe motion is the enemy of the authentication algorithm.

The original video and the seed plus authentication code T are transferred. When the receiver receives this, he generates his own witness file T' with the same seed using the same methods described above. The generated witness file is compared to the received witness file block by block and error rates are calculated for each block (here, we define error rate R_i = different number of pixels between T'i and Ti / number of pixels in Ti for block i, where Ti is a received block in the witness file, T'i is the corresponding generated corresponding block by receiver). The receiver can do this for all the blocks in all reference frames, because all the information randomly accessed is included in the witness file. If needed, the receiver can recover authentication information from a previous part in the witness file. Again, there are two standards to define that no subversion happened:

- The error rates for all the blocks are equal to, or close to a usual transmission error rate.
- Error rates for all blocks are distributed uniformly.

The Probability of Subversion Detection

The probability of a subversion detection is the sum of subversion detection probabilities for all the blocks, which is

$$\Pr(\text{A subversion is detected}) = \sum_{i=1}^{m*S/s} \Pr(Ti) = \sum_{i=1}^{m*S/s} \Pr(|Ri - \beta| > \tau)$$

where β is the predefined usual transmission error rate and τ is an acceptable value. From this formula, we see that the more blocks are got from a video, the more block detection probabilities are added to the subversion detection probability, but this doesn't mean that the subversion detection probability gets larger. The more blocks in a witness file, the more reference frames must be considered or the more blocks must be used in a frame, which means a smaller size for each block. How many blocks and how many pixels in a block should be taken, how many reference frames we need to consider, to get optimized values depends on the real problems. Interframe motion is the enemy for this algorithm; the higher the interframe motion in a video, the larger its witness file size becomes.

Transmission Protocol
These are omitted here; they are analogues of those formulated earlier; see [Chen, 1995].

Conclusion

We have discussed security issues with emphasis on multimedia authentication. Unlike image and video, text itself can be encrypted thereby providing authentication. Encryption using conventional (symmetric) methods such as the DES provides authentication, because only the sender and receiver share a secret key, and so no one else can decrypt and change the message meaningfully. Text encrypted with the public key method can provide not only authentication but also a digital signature.

Even though text itself together with encryption does provide authentication, there are applications in which encrypting the entire text is not appropriate. A text authentication code can be used instead for authentication.There are many algorithms for generating a text authentication code. We reviewed two here: authentication code based on the DES and on a hash function.

In contrast to text, authentication of still images and videos is rarely considered even though images and video are an important part in multimedia. We proposed authentication algorithms for still images and videos. Still images and videos are different from text. They are much larger; usually lossy data compression techniques are applied to them so they can be stored on computer disk and played back effectively. Complete encryption is not an effective way to provide authentication since neither sender nor receiver can afford the time to encrypt and decrypt the whole image or video every time. Effective authentication codes were instead developed here to attach to image or video to provide authentication.

References

[BHAS93] K. N. Bhaskar, *Computer security: threats and countermeasures*, NCC Blackwell, Manchester, 1993.
[CHEN95] F. Chen, Multimedia Authentication, M. S. thesis, Dept. of Computer Science, Univ. of Houston, Dec. 1995.
[DAVI89] D. Davies and W. Price, *Security for computer networks.*, New York: Wiley, 1989.
[JANO88] G. Janos, *Advanced Probability Theory*, M. Dekker, New York, 1988.
[JASP95] B. Jaspan, "GSS-API security for ONC RPC", *Proceedings of the System on Network and Distributed System Security*, San Diego, CA, 1995, 144-51.
[KWYU89] K.W.Yu and T.L.Yu, "Data encryption based upon time reversal transformations," *The Computer Journal*, v. 32, n. 3, 1989, 241-244.
[LEIS82] E. L. Leiss, *Principles of Data Security*, Plenum Press, New York, 1982.
[MERK90] R. Merkle, "A fast software one-way Hash function." *Journal of Cryptology*, v. 3, n. 1, 1990, 43-58.
[NELS92] M. Nelson, *The Data Compression Book*, M&T Publishing, 1992.
[NECH92] J. Nechvatal, "Public key cryptography," in [SIMM92a]
[RABI78] M. Rabin, "Digitalized signatures," in *Foundations of Secure Computation*, DeMillo, R.; Dobkin, D., New York: Academic Press, 1978.
[SHAF94] S. L. Shaffer and A. C. Simon, *Network Security*, AP Professional, Boston, 1994
[SIMM92] G. Simmons, *Contemporary Cryptology: The Science of Information Integrity*. Piscataway, NJ: IEEE Press, 1992.
[WILL91] N. Williams, G. S. Blair and N. Davies, "Distributed multimedia computing: an assessment of the state of the art," *Information Services and Use*, 1991, 265-281.
[WUXF92] X. WU, and Y. Fang, "Lossless interframe compression of medical images", *DCC'92 Data Compression Conf.*, Snowbird, UT, 1992, 249-58.