

Fundamentals of Multimedia Encryption Techniques

Borko Furht

Department of Computer Science and Engineering
Florida Atlantic University, 777 Glades Road
Boca Raton, FL 33431-0991, USA
E-mail: borko@cse.fau.edu

Daniel Socek

Department of Computer Science and Engineering
Florida Atlantic University, 777 Glades Road
Boca Raton, FL 33431-0991, USA
E-mail: dsocek@brain.math.fau.edu

Ahmet M. Eskicioglu

Department of Computer and Information Science
Brooklyn College of the City University of New York
2900 Bedford Avenue, Brooklyn, NY 11210, USA
E-mail: eskicioglu@sci.brooklyn.cuny.edu

1. INTRODUCTION

The recent advances in technology, especially in computer industry and communications, allowed potentially enormous market for distributing digital multimedia content through the Internet. However, the proliferation of digital documents, multimedia processing tools, and the worldwide availability of Internet access has created an ideal medium for copyright fraud and uncontrollable distribution of multimedia content [1]. A major challenge now is how to protect the intellectual property of multimedia content in multimedia networks.

To deal with the technical challenges, the two major multimedia security technologies are being developed:

- (1) Multimedia encryption technology to provide end-to-end security when distributing digital content over a variety of distributions systems, and

- (2) Multimedia watermarking technology as a tool to achieve copyright protection, ownership trace, and authentication.

In this chapter, we present the current research efforts in multimedia encryption, while the Chapter 7 gives an overview of multimedia watermarking techniques.

2. OVERVIEW OF MODERN CRYPTOGRAPHY

The three most important objectives of cryptography with respect to the multimedia information security include (1) confidentiality, (2) data integrity, and (3) authentication [3].

Confidentiality refers to the protection of information from unauthorized access. An undesired communicating party, called adversary, must not be able to access the communication material. *Data integrity* ensures that information has not been manipulated in an unauthorized way. Finally, *Authentication* methods are studied in two groups: *Entity* authentication and *message* authentication. Message authentication provides assurance of the identity of the sender of a message. This type of authentication also includes an evidence of data integrity because if the is modified during transmission, the sender cannot be the originator of the message. Entity authentication assures the receiver of a message of both the identity of the sender and his active participation.

Modern cryptographic techniques provide solutions for these three objectives. In general, there are two types of cryptosystems: (1) symmetric (private) key cryptosystems and (2) asymmetric (public) key cryptosystems.

2.1 SYMMETRIC KEY CRYPTOSYSTEMS

All classical cryptosystems (that is cryptosystems that were developed before 1970s) are examples of symmetric key cryptosystems. In addition, most modern cryptosystems are symmetric as well. Some of the most popular examples of modern symmetric key cryptosystems include AES (Advanced Encryption Standard), DES (Data Encryption Standard), IDEA, FEAL, RC5, and many others. The AES algorithm will be further discussed in Section 2.4.

All symmetric key cryptosystems have a common property: they rely on a shared secret between communicating parties. This secret is used both as an encryption key and as a decryption key (thus the keyword "symmetric" in the name). This type of cryptography ensures only confidentiality and fails to provide the other objectives of cryptography. Even more importantly, the disadvantage of symmetric key cryptography is that it cannot handle large communication networks. If a node in a communication network of n nodes needs to communicate confidentially with all other nodes in the network, it needs $n - 1$ shared secrets. For large value of n this is highly impractical and inconvenient. On the other hand, an advantage over public key cryptosystems is that symmetric cryptosystems require much smaller key sizes for the same level of security. Hence, the computations are much faster and the memory requirements are smaller.

2.2 PUBLIC KEY CRYPTOSYSTEMS

In public key cryptosystems there are two different keys: a public key, which is publicly known, and the secret key, which is kept secret by the owner. The system is called "asymmetric" since the different keys are used for encryption and decryption— the public key and the private key. If data is encrypted with a public key, it can only be decrypted using the corresponding private key,

and vice versa. Today, all public key cryptosystems rely on some computationally intractable problems. For example, the cryptosystem RSA relies on the difficulty of factoring large integers, while El-Gamal cryptosystem relies on the discrete logarithm problem (DLP), which is the problem of finding a logarithm of a group element with generator base in a finite Abelian group.

Public key cryptosystems do not need to have a shared secret between communicating parties. This solves the problem of large confidential communication network introduced earlier. In addition, public key cryptography opened door for ways of implementing technologies to ensure all goals of cryptography. By means of combining public key cryptography, public key certification, and secure hash functions, there are protocols that enable digital signatures, authentication, and data integrity.

Due to the increase in processor speed and even more due to smart modern cryptanalysis, the key size for public key cryptography grew very large. This created a disadvantage in comparison to symmetric key cryptosystems: public key cryptography is significantly slower, and requires large memory capacity and large computational power. As an example, a 128-bit key used with DES cryptosystem has approximately the same level of security as the 1024-bit key used with RSA public key cryptosystem [5].

To solve these problems, researchers introduced different approaches. In order to decrease the key size so that public key cryptography can be used in smaller computational environments (such as smart cards or handheld wireless devices), Neil Koblitz introduced the idea of using a more exotic group in the public key underlying algebraic structure: the elliptic curve group. Elliptic curve cryptography (much of whose implementation is credited to CertiCom) enables smaller key sizes of public key cryptosystems that rely on DLP. The elliptic curve group algebra is much more complex so the related cryptanalysis is much harder, resulting in smaller key

requirements. Another solution came from public key cryptosystems that initially use more complex computational problem, such as the lattice reduction problem. The relatively new cryptosystem NTRU based on the algebra of a ring of truncated polynomials, relies on the lattice reduction problem and it is the only public key cryptosystem that has the speed, memory, and computational complexity comparable to the symmetric key cryptosystems. However, the security aspects of NTRU are yet to be investigated [19, 20]. The most common implementation solution is to combine symmetric key cryptosystems with public key cryptography. Namely, to overcome the problems related to applying the symmetric key encryption only, the plaintext is encrypted using a fast symmetric key scheme, and only the secret key used for the symmetric encryption is encrypted with the slow public key scheme such as RSA. This way all goals of cryptography can be achieved with much better performance.

2.3 CRYPTANALYSIS

Cryptanalysis is an art of deciphering an encrypted message in whole or in part, when the decryption key is not known. Depending on the amount of known information and the amount of control over the system by the adversary (cryptanalyst), there are several basic types of cryptanalytic attacks:

Ciphertext-only attack: The adversary only has access to one or more encrypted messages. The most important goal of a proposed cryptosystem is to withstand this type of attack.

Brute force attack: This is a type of ciphertext-only attack. It is based on exhaustive key search, and for well-designed cryptosystems should be computationally infeasible. By today's standards, 128-bit keys are considered secure against the brute force attack.

Known-plaintext attack: In this type of attack an adversary has some knowledge about the plaintext corresponding to the given ciphertext. This may help determine the key or a part of the key.

Chosen-plaintext attack: Essentially, an adversary can feed the chosen plaintext into the black box that contains the encryption algorithm and the encryption key. The black box spits out the corresponding ciphertext, and the adversary may use the accumulated knowledge about the plaintext-ciphertext pairs to obtain the secret key or at least a part of it.

Chosen-ciphertext attack: Here, an adversary can feed the chosen ciphertext into the black box that contains the decryption algorithm and the decryption key. The black box produces the corresponding plaintext, and the adversary tries to obtain the secret key or a part of the key by analyzing the accumulated ciphertext-plaintext pairs.

2.4 ADVANCED ENCRYPTION STANDARD (AES)

The AES algorithm is essentially Rijndael [38] symmetric key cryptosystem that processes 128-bit data blocks using cipher keys with lengths of either 128, 192, or 256 bits. Rijndael is more scalable and can handle different key sizes and data block sizes, however they are not included in the standard. Table 1 shows the basic AES parameters.

	Key Length (Nk words)	Blocksize (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Table 1. The basic AES parameters.

The AES algorithm's internal operations are performed on a two-dimensional array of bytes called the *State*. The State consists of 4 rows of bytes, each containing *Nb* bytes, where *Nb* is the blocksize divided by 32 (since the size of a word is 32 bits), which is in the case of AES equal to 4. At the beginning of the encryption or decryption procedure, the input array *in*, is copied to the State array *s* according to the following rule: $s[r, c] = in[r + 4c]$, where $0 = r < 4$ and $0 = c < Nb$. Similarly, at the end of each procedure, the State is copied to the output array *out* as follows: $out[r + 4c] = s[r, c]$, where $0 = r < 4$ and $0 = c < Nb$. Clearly, in this notation *r* stands for row and *c* for column.

2.4.1 AES Encryption

At the beginning of the AES encryption procedure, the input is copied to the State array using the conventions described above. After an initial Round Key addition (see next section), the State array is transformed by implementing a round function 10, 12, or 14 times (depending on the key length), with the final round differing slightly from the first $Nr - 1$ rounds. The final State is then copied to the output as described above. Figure 1 shows the pseudo code for the AES encryption.

```

AesEncrypt(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]

    state = in

    AddRoundKey(state, w[0, Nb-1])

    for round = 1 step 1 to Nr-1
        SubBytes(state)
        ShiftRows(state)
        MixColumns(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
    end for

    SubBytes(state)
    ShiftRows(state)
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

    out = state
end

```

Figure 1. Pseudo code for the AES encryption algorithm.

The `SubBytes()` transformation is essentially an S-box type of transform which is a non-linear byte substitution that operates independently on each byte of the State. The simplest representation of this S-box function is by the lookup table. The lookup table associated with `SubBytes()` is here shown in Table 2.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Table 2. AES Encryption S-box: substitution values for the byte xy (in hexadecimal representation).

In the `ShiftRows()` transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes, while the first row is not shifted. The `ShiftRows()` transformation is defined as follows: $s'[r, c] = s[r, (c + \text{shift}(r, Nb)) \bmod Nb]$ for $0 = r < 4$ and $0 = c < Nb$, where the shift value $\text{shift}(r, Nb)$ is defined in the following way: $\text{shift}(1, 4) = 1$, $\text{shift}(2, 4) = 2$, and $\text{shift}(3, 4) = 3$.

The `MixColumns()` transformation operates on the State column-by-column. The columns are considered as polynomials over the Galois Field $\text{GF}(2^8)$ and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$. Here, the notation $\{h_1h_2\}$ denotes a byte in

the hexadecimal format. This can be accomplished by applying the following four equations to the State columns:

1. $s'[0, c] = (\{02\} \bullet s[0, c]) \otimes (\{03\} \bullet s[1, c]) \otimes s[2, c] \otimes s[3, c]$
2. $s'[1, c] = s[0, c] \otimes (\{02\} \bullet s[1, c]) \otimes (\{03\} \bullet s[2, c]) \otimes s[3, c]$
3. $s'[2, c] = s[0, c] \otimes s[1, c] \otimes (\{02\} \bullet s[2, c]) \otimes (\{03\} \bullet s[3, c])$
4. $s'[3, c] = (\{03\} \bullet s[0, c]) \otimes s[1, c] \otimes s[2, c] \otimes (\{02\} \bullet s[3, c])$

Here, the \bullet operation denotes multiplication in $GF(2^8)$ modulo the polynomial $x^4 + 1$, while the \otimes denotes the usual bitwise XOR operation.

Finally, in the `AddRoundKey()` transformation, a Round Key is added to the State by a simple bitwise XOR operation. Each Round Key consists of Nb words from the key schedule (to be described in the next section), which are each added into the columns of the State.

```

KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp

    i = 0
    while (i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while
    i = Nk
    while (i < Nb * (Nr+1))
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if

        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end

```

Figure 2. Pseudo code of the key schedule stage of the AES algorithm.

2.4.2 AES Key Schedule

In the AES algorithm, the initial input symmetric key is expanded to create a key schedule for each round. This procedure, given in Figure 2, generates a total of $Nb(Nr + 1)$ words that are used in the `AddRoundKey()` procedure. In this algorithm, `SubWord()` is a function that takes a four-byte input word and applies the S-box to each of the four bytes to produce an output word. The function `RotWord()` takes a word $[a_0, a_1, a_2, a_3]$ as input, performs a cyclic permutation, and returns the word $[a_1, a_2, a_3, a_0]$. The round constant word array, `Rcon[i]`, contains the values given by $[x^{i-1}, \{00\}, \{00\}, \{00\}]$, with x^{i-1} being powers of x (x is denoted as $\{02\}$) in the field $GF(2^8)$.

```
AesDecrypt(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
    byte state[4,Nb]

    state = in

    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

    for round = Nr-1 step -1 downto 1
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
        InvMixColumns(state)
    end for

    InvShiftRows(state)
    InvSubBytes(state)
    AddRoundKey(state, w[0, Nb-1])

    out = state
end
```

Figure 3. Pseudo code of AES decryption algorithm.

2.4.3 AES Decryption

AES decryption, the inverse of AES encryption, also consists of four stages in each round, which are simply the inverses of the transformations described above. Each AES decryption round

performs the following four transformations: `InvShiftRows()`, `InvSubBytes()`, `InvMixColumns()`, and `AddRoundKey()`. The pseudo code for the AES decryption process is given in Figure 3.

The `InvShiftRows()` is the inverse of the `ShiftRows()` transformation, which is defined as follows: $s[r, c] = s'[r, (c + \text{shift}(r, Nb)) \bmod Nb]$ for $0 = r < 4$ and $0 = c < Nb$. Similarly, `InvSubBytes()` is the inverse of the byte substitution transformation, in which the inverse S-box is applied to each byte of the State. The inverse S-box used in the `InvSubBytes()` transformation is presented in Table 3.

		Y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Table 3. AES Decryption S-box: substitution values for the byte xy (in hexadecimal representation).

The `InvMixColumns()` is the inverse of the `MixColumns()` transformation. The transformation `InvMixColumns()` operates on the State column-by-column, treating each column as a polynomial over the field $\text{GF}(2^8)$ modulo $x^4 + 1$ with a fixed polynomial $a^{-1}(x)$, given by $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$. This can be accomplished by applying the following four equations to the State columns:

1. $s'[0, c] = (\{0e\} \bullet s[0, c]) \otimes (\{0b\} \bullet s[1, c]) \otimes (\{0d\} \bullet s[2, c]) \otimes (\{09\} \bullet s[3, c])$
2. $s'[1, c] = (\{09\} \bullet s[0, c]) \otimes (\{0e\} \bullet s[1, c]) \otimes (\{0b\} \bullet s[2, c]) \otimes (\{0d\} \bullet s[3, c])$
3. $s'[2, c] = (\{0d\} \bullet s[0, c]) \otimes (\{09\} \bullet s[1, c]) \otimes (\{0e\} \bullet s[2, c]) \otimes (\{0b\} \bullet s[3, c])$
4. $s'[3, c] = (\{0b\} \bullet s[0, c]) \otimes (\{0d\} \bullet s[1, c]) \otimes (\{09\} \bullet s[2, c]) \otimes (\{0e\} \bullet s[3, c])$

Finally, the `AddRoundKey()` is identical to the one from the AES encryption procedure, since this transformation is its own inverse.

The security level of AES is to this day substantial since there are no known effective attacks discovered thus far. For more information about AES please refer to [38].

3. INTRODUCTION TO MULTIMEDI SECURITY

Multimedia security in general is provided by a method or a set of methods used to protect the multimedia content. These methods are heavily based on cryptography and they enable either communication security, or security against piracy (Digital Rights Management and watermarking), or both. Communication security of digital images and textual digital media can be accomplished by means of standard symmetric key cryptography. Such media can be treated as binary sequence and the whole data can be encrypted using a cryptosystem such as AES or DES [6]. In general, when the multimedia data is static (not a real-time streaming) we can treat it as a regular binary data and use the conventional encryption techniques. Encrypting the entire multimedia stream using standard encryption methods is often referred to as the *naive approach*.

However, due to variety of constraints (such as the near real-time speed, etc.), communication security for streaming audio and video media is harder to accomplish. Communication encryption

of video and audio multimedia content is not simply the application of established encryption algorithms, such as DES or AES, to its binary sequence. It involves careful analysis to determine and identify the optimal encryption method when dealing with audio and video data. Current research is focused on modifying and optimizing the existing cryptosystems for real-time audio/video. It is also oriented towards exploiting the format specific properties of many standard video and audio formats in order to achieve the desired speed and enable real-time streaming. This is referred to as the *selective encryption*.

For textual data, most applications of still images, and some low-quality audio and video streaming multimedia, we can still apply real-time packet encryption by means of SRTP (Secure Real-time Transport Protocol) [31], which is based on AES and encrypts the entire multimedia bit stream. This is essentially an application of the naive approach.

Deciding upon what level of security is needed is harder than it looks. To identify an optimal security level, we have to carefully compare the cost of the multimedia information to be protected and the cost of the protection itself. If the multimedia to be protected is not that valuable in the first place, it is sufficient to choose relatively light level of encryption. On the other hand, if the multimedia content is highly valuable or represents government or military secrets, the cryptographic security level must be the highest possible.

For many real-world applications such as pay-per-view, the content data rate is typically very high, but the monetary value of the content may not be high at all. Thus, very expensive attacks are not attractive to adversaries, and light encryption (often called *degradation*) may be sufficient for distributing MPEG video. For these applications, DRM is of much more interest. In contrast, applications such as videoconferencing or videophone (or even Internet phone) may require much higher level of confidentiality. If the videoconference is discussing important industrial,

governmental or military secrets, then the cryptographic strength must be substantial. Maintaining such high level of security and still keeping the real-time and limited-bandwidth constraints is not easy to accomplish.

Comprehensive survey studies of the multimedia encryption techniques are given in [2, 4, 34]. In [4], Liu and Eskicioglu present a comprehensive classification that summarizes most of the presented algorithms. An updated version of that classification is shown in Table 4.

Table 4. Classification of Selective Encryption Schemes, adapted from [4].

Type of data	Domain	Proposal	Encryption Algorithm	What is encrypted?
Image	Frequency domain	Cheng & Li, 2000 [23]	No algorithm is specified.	Pixel and set related significance information in the two highest pyramid levels of SPIHT
		Droogenbroeck & Benedett, 2002 [28]	DES, Triple DES and IDEA	Bits that indicate the sign and magnitude of the non-zero DCT coefficients
		Pommer & Uhl, 2003 [37]	AES	Subband decomposition structure
	Spatial domain	Cheng & Li, 2000 [23]	No algorithm is specified.	Quadtree structure
		Droogenbroeck & Benedett, 2002 [28]	Xor	Least significant bitplanes
		Podesser, Schmidt & Uhl, 2002 [35]	AES	Most significant bitplanes
Video	Frequency domain	Meyer & Gadegast, 1995 [9]	DES, RSA	Headers, parts of I-blocks, all I-blocks, I-frames of the MPEG stream
		Spanos & Maples, 1995 [7,8]	DES	I-frames, sequence headers and ISO end code of the MPEG stream
		Tang, 1996 [10]	Permutation, DES	DCT coefficients
		Qiao & Nahrstedt, 1997 [12]	xor, permutation, IDEA	Every other bit of the MPEG bit stream
		Shi & Bhargava, 1998 [15]	xor	Sign bit of DCT coefficients
		Shi, Wang & Bhargava, 1999 [16]	IDEA	Sign bit of motion vectors
		Alattar, A-Regib and Al-Semari, 1999 [22]	DES	Every n^{th} I-macroblock, headers of all the predicted macroblocks, header of every n^{th} predicted macroblock
		Shin, Sim & Rhee, 1999 [32]	Permutation, RC4	Sign bits of DC coefficients of I pictures. Random permutation of the DCT coefficients
		Cheng & Li, 2000 [23]	No algorithm is specified.	Pixel and set related significance information in the two highest pyramid levels of SPIHT in the residual error
		Wen, Severa, Zeng, Luttrell & Jin, 2002 [26]	DES, AES	The information-carrying fields, either fixed length code (FLC) codewords, or variable length code (VLC) codewords
		Zeng & Lei, 2002 [27]	Permutation, xor	Selective bit scrambling, block shuffling, block rotation of the transform coefficients (wavelet and JPEG) and JPEG motion vectors
		Wu & Mao, 2002 [33]	Any modern cipher, random shuffling on bit-planes in MPEG-4 FGS	Bitstream after entropy coding, quantized values before run length coding (RLC) or RLC symbols, intra bit-plane shuffling in MPEG-4 FGS
	Spatial domain	Cheng & Li, 2000 [23]	No algorithm is specified.	Quadtree structure of motion vectors and quadtree structure of residual errors

	Entropy codec	Wu & Kuo, 2000 [24]; Wu & Kuo, 2001 [25]	Multiple Huffman tables, multiple state indices in the QM coder	Encryption of data by multiple Huffman coding tables and multiple state indices in the QM coder
Speech	Compressed domain	Wu & Kuo, 2000 [24]	DES, AES	Most significant bits of the LSP codebook indices, the lag of pitch predictor, the pitch gain vectors, the fixed codebook gains, and the VAD mode flag coefficients of the G.723.1 speech codec
		Servetti & De Martin, 2002 [29]	Not specified	1 st stage vector of LSP quantizer, 2 nd stage lower vector of LSP quantizer, pitch delay 1 st subframe, pitch delay 2 nd subframe, gain codebook (stage 1) 1 st subframe, gain codebook (stage 2) 1 st subframe, gain codebook (stage 1) 2 nd subframe, gain codebook (stage 2) 2 nd subframe
Audio	Compressed domain	Servetti, Testa & De Martin, 2003 [30]	Not specified	File header (that contains the original <i>table_select</i> and <i>big_value</i> values) at the beginning of the audio track; a subset of region1 bits and a subset of region2 bits.
		Thorwirth, Horvatic, Weis & Zhao, 2000 [36]	Not specified	Audio quality layers associated with the compressed audio data

4. VIDEO ENCRYPTION TECHNIQUES

In this section we discuss different research directions, which were taken in the area of video communication encryption. As discussed earlier, a naive approach can be used to encrypt multimedia content. The whole multimedia stream is encrypted using a symmetric key cryptosystem. However, even the fastest modern symmetric schemes, such as DES or AES, are computationally very expensive for many real-time video and audio data.

4.1 VIDEO SCRAMBLING

Scrambling is a popular method of applying fast, yet very insecure distortion of the video signal. In general, scrambling was the product of immediate industrial need by cable companies for a fast solution to make the free viewing of paid cable channels more difficult than doing nothing to the video signal.

Even though there is no distinct definition of scrambling, it refers to the simplest possible encryption attempts such as simple substitution or simple transposition ciphers whose security in

the modern world is the lowest possible. Early work on signal scrambling was based on using an analog device to permute the signal in the time domain or distort the signal in the frequency domain by applying filter banks or frequency converters [25]. However, these schemes seriously lack security, and they are extremely easy to crack using modern computers. As far as the attacker is concerned, those methods represent a temporary inconvenience in getting the original video or audio signal.

In many instances, this is still a solution that serves cable companies around the world quite effectively since most people lack the knowledge and engineering skills to crack the scrambled video signal, but the modified cable boxes that can unscramble all scrambled material are quite easy to manufacture. However, scrambling is out of the question for the applications that require more serious level of security.

4.2 SELECTIVE VIDEO ENCRYPTION

In order to meet real-time constraint for audio and video multimedia playback, *selective encryption* techniques have been proposed. The basic idea of selective encryption is to encrypt only a portion of the compressed bit-stream. A common approach in selective encryption is to integrate compression with encryption as shown in Figure 4.

For example, we can select only the most important coefficients from either final or intermediate steps of a compression process and encrypt those coefficients. Less important coefficients are left unencrypted. Still, some schemes prefer to lightly encrypt even these less important coefficients. By “light” encryption we mean applying a simple encryption scheme whose speed is high and whose security level is low (there is always a trade-off between the speed and security). In this manner, we define *variable encryption* as an approach that applies different encryption schemes

with different security strengths. Selective encryption can be treated as a special case of variable encryption [34].

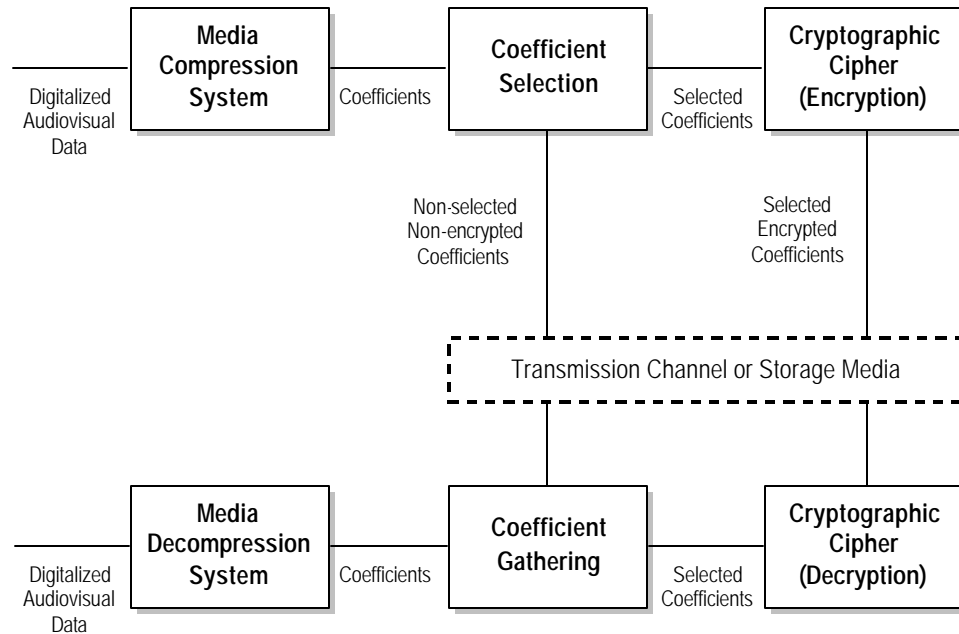


Figure 4. A common selective encryption scheme.

4.2.1 SEC MPEG by Meyer and Gadegast, 1995

In 1995 Meyer and Gadegast introduced the selective encryption method called *Secure MPEG*, or shortly *SEC MPEG*, designed for the MPEG-1 video standard [9]. The SEC MPEG contains four different levels of security. At the first level, SEC MPEG encrypts the headers from the sequence layer to the slice layer, while the motion vectors and DCT blocks are unencrypted. At the second level, most relevant parts of the I-blocks are additionally encrypted (upper left corner of the block). At the third level, SEC MPEG encrypts all I-frames and all I-blocks. Finally, at the fourth level, SEC MPEG encrypts the whole MPEG-1 sequence (the naive approach). The authors chose Data Encryption Standard (DES) symmetric key cryptosystem, which was the natural choice, given that this cryptosystem had been around since 1976 and was the official symmetric

encryption algorithm standardized by NIST and adopted by the US Government. Since DES is a symmetric key cryptosystem, it could only be used to achieve confidentiality. Meyer and Gadegast targeted solving the problem of data integrity as well. For that reason, the Cyclic-Redundancy-Check (CRC) was incorporated as a low-level solution to the integrity. The real data integrity mechanisms that included public key cryptography and cryptographically good hash functions such as MD4, MD5, or SHA were left for further research.

The selective encryption in SECMPEG (levels 1, 2, and 3) has some weaknesses [11,13]. It is shown that even though single P- or B-frame on its own carries almost no information without the corresponding I-frame, a series of P- or B-frames can tell a lot if their base I-frames are correlated. The experiments by Agi and Gong showed that encrypting I-frames only might leave visible I-blocks in other frames [11]. The same authors then propose a few trade-off improvements: increasing the frequency of the I-frames and/or encrypting all P- and B-frames. These improvements decrease speed, and further disrupt the compression ratio. Since SECMPEG introduces changes to the MPEG-1 format, a special encoder and decoder is needed to handle SECMPEG streams.

Nevertheless, the SECMPEG paper [9] and implementation by Meyer and Gadegast was one of the first important research initiatives for selective encryption of multimedia streams. These authors were among the first to realize the benefits of encrypting only selected bits in a video bit-stream. Their experiments confirmed that the visual disruption was substantial when only encrypting the DC coefficients and the first 3-8 AC coefficients in the 8x8 JPEG block of the MPEG-1 I-frames.

4.2.2 Aegis by Maples and Spanos, 1995

Another important research achievement on selective encryption of MPEG video stream appeared in 1995 by Maples and Spanos [7,8], and it introduced the secure MPEG video mechanism called *Aegis*. *Aegis* was initially designed for MPEG-1 and MPEG-2 video standards. *Aegis* method encrypts I-frames of all MPEG groups of frames in an MPEG video stream, while B- and P-frames are left unencrypted. In addition, *Aegis* also encrypts the MPEG video sequence header, that contains all of the decoding initialization parameters that include the picture width, height, frame rate, bit rate, buffer size, etc. Finally, this method also encrypts the ISO end code; i.e., the last 32 bits of the stream. As a result, the bit-stream's MPEG identity is further concealed. The *Aegis*' behind-the-scenes encryption engine chosen by Maples and Spanos was DES. In a way, *Aegis* is very similar to the level 3 of SECMPEG by Meyer and Gadegast, and the security weaknesses that are applicable to SECMPEG are carried over to *Aegis*.

As discussed earlier, Agi and Gong criticized encrypting only I-frames by *Aegis* and SECMPEG, in view of the fact that their experiments showed that it is possible to discern some aspects from the scene from the decoded P- and B-frames [11]. For example, they were able to conclude that the talking head was present at the scene. Alattar and Al-Regib presented the similar argument regarding the security of SECMPEG in [21]. It is important to notice that this type of security is not particularly good for applications where the security is one of the top priorities (such as the important military or business teleconferencing) but it might be sufficient as a quality degradation of the entertainment videos, such as the pay TV broadcast.

Aegis is also considered as one of the pioneering research accomplishments in the area of selective video encryption. The experimental results for the selective encryption published by

Maples and Spanos in [7,8] confirmed the enormous gain in speed over the naive approach, by applying the selective encryption.

4.2.3 Zigzag Permutation Algorithm by Tang, 1996

Tang's *Zigzag permutation algorithm* [10] is based on embedding the encryption into the MPEG compression process. The JPEG images and the Iframes of MPEG video undergo a zigzag reordering of the 8x8 blocks. The zigzag pattern forms a sequence of 64 entries that are ready to enter entropy-encoding stage. The main idea of this approach is to use a random permutation list to map the individual 8x8 blocks to a 1x64 vector. The algorithm consists of three stages:

- *Stage 1:* A list of 64 permutations is generated.
- *Stage 2:* Splitting procedure on an 8x8 block is performed as follows: we denote DC coefficient by an 8 digit binary number with digits $b_7b_6b_5b_4b_3b_2b_1b_0$. This binary sequence is then split into two equal halves: $b_7b_6b_5b_4$ and $b_3b_2b_1b_0$. Finally, we place number $b_7b_6b_5b_4$ into DC coefficient and the number $b_3b_2b_1b_0$ as the AC_{63} (the last AC coefficient), which is the least important value in the block. This will result in no visible degradation of the quality.
- *Stage 3:* The random permutation is applied to the split block.

However, the zigzag permutation algorithm is not particularly secure [13,14]. Qiao and Nahrstedt introduced two types of attacks on zigzag permutation: a known-plaintext attack, and a ciphertext-only attack. A known-plaintext attack is particularly applicable to the videos with known clips such as blank screens, the movie rating screens, MGM roaring lion, etc. If these known frames are used as a comparison, the adversary can easily generate the secret permutation list.

Since Tang himself realized large vulnerability to known-plaintext attack, he introduced an improvement of the zigzag permutation technique by using the binary coin flipping sequence method together with two different permutation lists. Two different permutation lists are generated and for each 8x8 block a coin is randomly flipped. The outcome event determines which list to apply; i.e., if the head is flipped we apply one list and if the tail is flipped we apply the other list. As shown in [13,14], this addition turns out to be useless. If we know some of the original frames in advance (known-plaintext) then by simple comparison both permutation lists can be found. Since of the certain statistical properties of the blocks (upper left corner gathering of the AC coefficients within a block), we can easily determine which list of permutations is used.

In addition, Qiao and Nahrstedt showed that the Tang's splitting method is just a subset of SECMPPEG with second level, and thus its security is not good. Finally, Qiao and Nahrstedt showed that the Tang's zigzag permutation algorithm is susceptible to the ciphertext-only attack. The attack relies on the statistical properties of the DCT coefficients where most non-zero terms are gathered in the top left corner of the block. Statistical analysis shows the following observation [13]: the DC coefficient always has the highest frequency value, the frequencies of AC_1 and AC_2 coefficients are among top 6, and frequencies of AC_3 and AC_4 are among top 10. Applying these observations, one can reconstruct the original video a relatively the good accuracy using only five DCT coefficients. Therefore, zigzag permutation cipher seriously lacks the desired level of security.

4.2.4 Video Encryption Algorithm by Qiao and Nahrstedt, 1997

The *Video Encryption Algorithm* (VEA) by Qiao and Nahrstedt [12] is constructed with the goal to exploit the statistical properties of the MPEG video standard. The algorithm consists of the following four steps:

- *Step 1:* Let the $2n$ byte sequence, denoted by $a_1a_2\dots a_{2n}$, represent the chunk of an I-frame
- *Step 2:* Create two lists, one with odd indexed bytes $a_1a_3\dots a_{2n-1}$, and the other with even indexed bytes $a_2a_4\dots a_{2n}$.
- *Step 3:* Xor the two lists into an n -byte sequence denoted with $c_1c_2\dots c_n$
- *Step 4:* Apply the chosen symmetric cryptosystem E (for example DES or AES) with the secret key $KeyE$ on either odd list or even list, and thus create the ciphertext sequence $c_1c_2\dots c_n E_{KeyE}(a_1a_3\dots a_{2n-1})$ or $c_1c_2\dots c_n E_{KeyE}(a_2a_4\dots a_{2n})$ respectively.¹

Clearly, the decryption mechanism at the other end of the communication channel consists of two easy steps: Apply the symmetric cryptosystem E with the appropriate key to the second half of the ciphertext to obtain the first half of the original sequence, and xor this result with the first half of the ciphertext to obtain the other half of the original sequence.

Even though the security of the system is based on the security of the chosen underlying cryptosystem, if an adversary obtains either even or odd list, the system is completely broken. Thus, the authors suggest an even more secure approach. Instead of dividing the list into the even and odd lists, the random $2n$ -bit key, called $KeyM$, is generated and used to split the $2n$ -byte chunk of MPEG stream into two lists. The method of partitioning the $2n$ -byte MPEG sequence

¹ The authors' choice was to encrypt the even sequence creating the ciphertext $c_1c_2\dots c_n E_{KeyE}(a_2a_4\dots a_{2n})$

into two subsequences is as follows: if i^{th} bit in $KeyM$ is 1 then the i^{th} byte of the MPEG sequence goes into the first subsequence, and similarly, if i^{th} bit in $KeyM$ is 0 then the i^{th} byte of the MPEG sequence goes into the second subsequence. It is required that $KeyM$ is a binary sequence with exactly n ones and n zeros, in order to ensure that the two lists are of the equal length. The suggested values for n are 64 or 128. By applying this approach, the attacker will have a very hard time ordering the resulting sequence of bytes in the case that the encrypted subsequence is somehow cracked, assuming that the binary sequence $KeyM$ remained secret. This means that $KeyM$ must be securely transmitted to the receiver before the decryption can take place.

Another consideration is that a possible pattern repetition in the $2n$ stream chunk represents a significant security hole in the method proposed by Qiao and Nahrstedt. According to the statistical analysis of MPEG stream sequences, it was observed that the non-repeating patterns have a lifetime over only one 1/16 chunk [12]. Therefore, to obtain the non-repeating pattern with a length of 1/2 frame, which is consisted of 8 such chunks, it is sufficient to shuffle only these 8 chunks. However, each of these chunks must be shuffled by using different keys, which are here denoted by $Key_1, Key_2, \dots, Key_8$. The authors suggest using the permutations of degree 32 (even though the degree 24 was shown to be sufficient), so that each Key_i is stored as a random permutation of degree 32. Hence, the length of a Key_i is 20 bytes, resulting in the total length of 160 bytes for $Key_1, Key_2, \dots, Key_8$. Once the keys are generated, Key_1 permutation is applied to the first 32 bytes of even list, Key_2 permutation is applied to the next 32 bytes, and so on, repeating this process after the 8^{th} key. This procedure will ensure the non-repetition in the MPEG stream, thus significantly decreasing the vulnerability of the system to the repetitive pattern attacks. Furthermore, the increase in the bit-stream size caused by including these extra keys is at a negligible level.

Finally, Qiao and Nahrstedt proposed the use of one more additional key, denoted by $KeyF$. The experiments have shown that the pattern of selecting the even list and the odd list should be changed for every frame in order to ensure the anticipated security level [12]. For this reason, a 64-bit key $KeyF$ is randomly generated for every single frame in the MPEG sequence. However, this key is subject to the constraint that every 4 consecutive bits represent a unique number from 0 to 15; i.e., so that any $KeyF$ represents a permutation of degree 16. To get the new keys for each frame, we permute $KeyM$ and Key_i 's repeatedly using the permutation defined by $KeyF$ assigned to that frame.

In order to securely transmit these additional keys ($KeyM$, Key_i 's and $KeyF$'s), they must be encrypted using the encryption function E with $KeyE$. As a final point, the secret key $KeyE$ has to be securely transmitted to the decoder's side (the receiver), possibly by using the public key cryptography, or by using the separate secure channel. Alternatively, the separate secure channel can be used for the transmission of keys $KeyM$ and Key_i 's. The diagram of the proposed algorithm is shown in Figure 5.

The experiments that were originally performed in 1997 were mainly using DES and IDEA cryptosystems; however, the newer choice for the underlying cryptosystem E would probably be the recently standardized AES cryptosystem (Rijndael). The security of method proposed by Qiao and Nahrstedt is very close to the security of encryption scheme E that is internally used. The speed of this algorithm is roughly a half of the speed of naive algorithm, but that is arguably still the large amount of computation for high quality real-time video applications that have high bit rates [25].

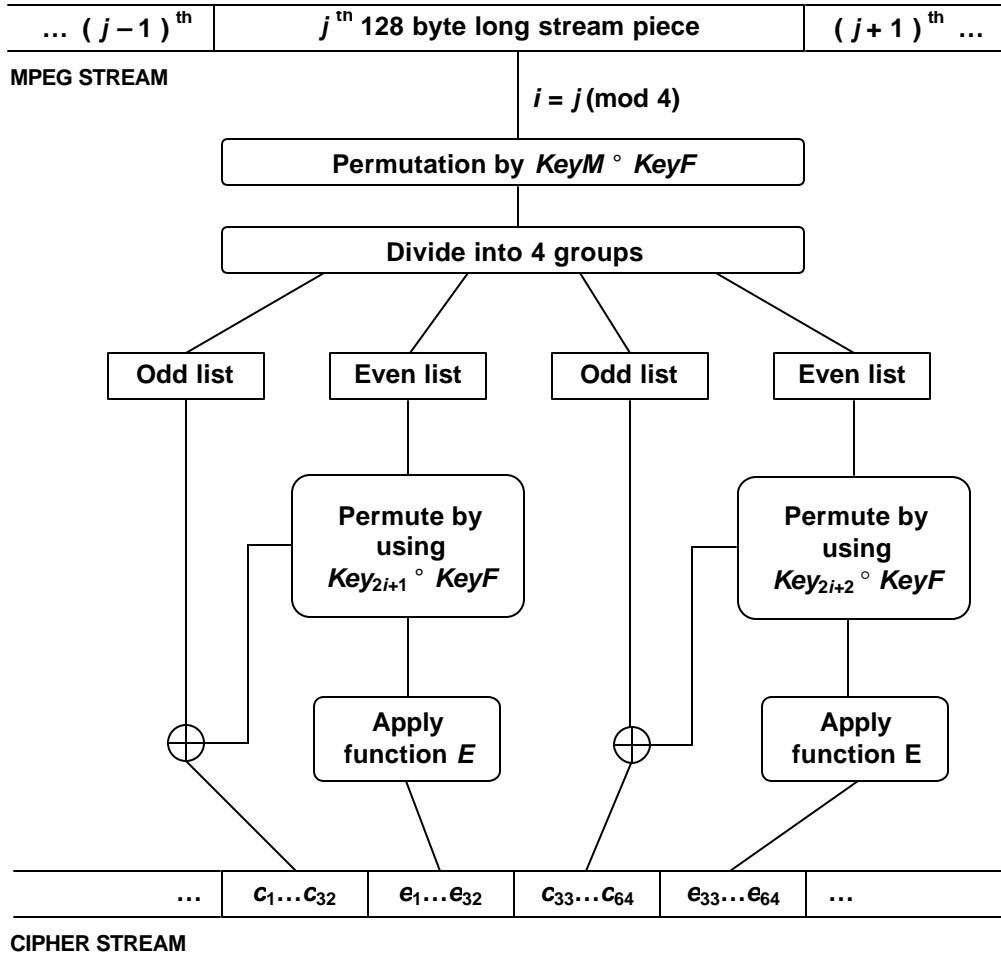


Figure 5. Selective Video Encryption Algorithm by Qiao and Nahrstedt [12].

4.2.5 Video Encryption Algorithms by Shi, Wang and Bhargava, 1998 and 1999

In [17], Shi Wang and Bhargava classified their previous work into four different video encryption algorithms: *Algorithm I*, *Algorithm II (VEA)*, *Algorithm III (MVEA)*, and *Algorithm IV (RVEA)*. The original work was published by the same authors years earlier [15,16]. The basic ideas of these algorithms is to apply selective encryption on selective coefficients in the JPEG/MPEG schemes, as illustrated in Figure 6.

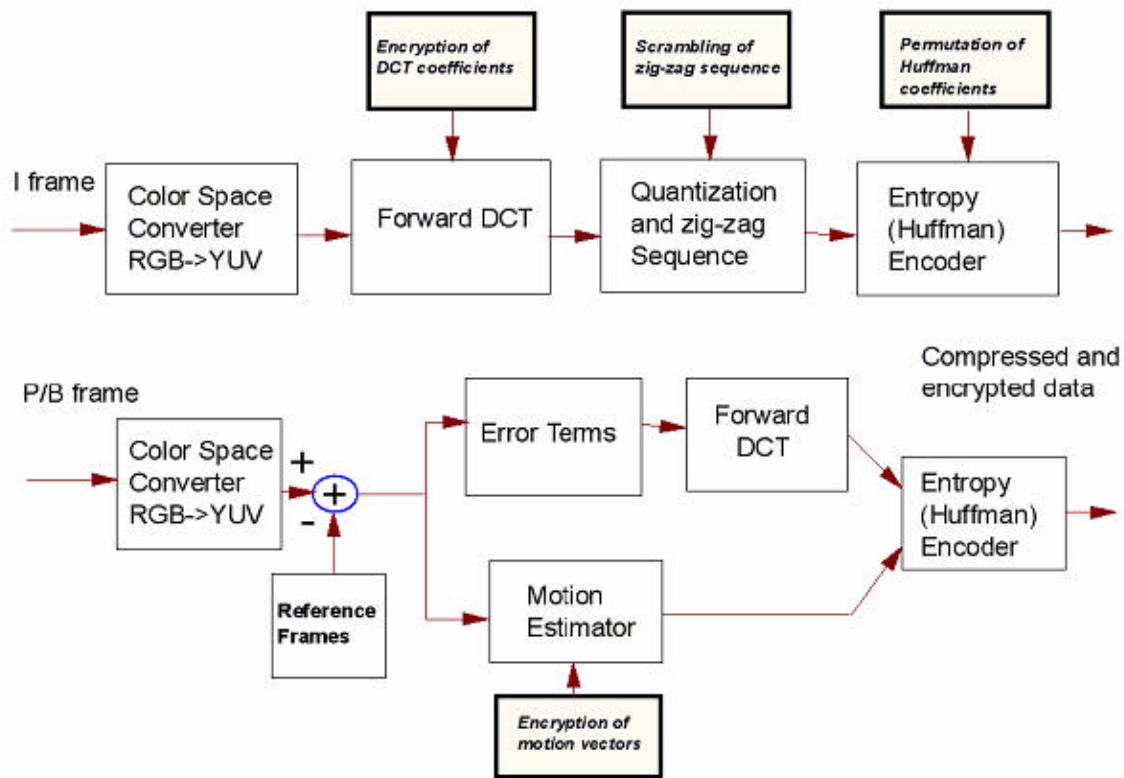


Figure 6. Selective encryption of selective coefficients in MPEG/JPEG coding schemes: DCT and its zig-zag sequence, coefficients in the Huffman table, and motion vectors.

Algorithm I

The first algorithm, denoted simply by the Algorithm I, uses the permutation of Huffman codewords in the I-frames. This method incorporates encryption and compression in one step. The secret part of the algorithm is a permutation p , which is used to permute standard JPEG/MPEG Huffman codeword list. In order to save compression ratio, the permutation p must be such that it only permutes the codewords with the same number of bits. In addition, the distance between the original and the permuted codeword list must be greater than the encryption quality parameter b , which is chosen by the user.

The security of Algorithm I is not particularly good. In [18], it is shown that the Algorithm I is highly vulnerable to both known-plaintext attack, and ciphertext-only attack. If some of the video frames are known in advance (such as standard introductory jingles and similar), one can reconstruct the secret permutation p by comparing the original and encrypted frames. Vulnerability to this type of attack was also discussed in [17]. However, Algorithm I is also subject to ciphertext-only attack. The low-frequency error attack can be applied on ciphertext produced by Algorithm I [18]. Basically, since permutation p is of the special form; i.e., it only shuffles codewords with the same length, the most security comes from shuffling the 16-bit codewords in the AC coefficient entropy table. However, since there is a very limited number of codewords with length of less than 16 bits, it is very easy to reconstruct all of the DC coefficients and most frequent AC coefficients (since these will be encoded with less than 16-bit codewords). In other words, the only hard part would be to figure out how does the permutation p shuffle the 16-bit codewords. But these are appearing extremely rare, and the reconstructed video may be of almost the same quality as the original. Furthermore, the rare pixels that do contain 16-bit codewords can easily be interpolated to get rid of the noise effect and to create the reconstruction image that, at least to the human visual system, appears flawless as the original.

Algorithm II (VEA)

The Algorithm II (VEA) uses the following selective encryption observation: it is sufficient to encrypt only the sign bits of the DCT coefficients in an MPEG video. The Algorithm II simply xors the sign bits of the DCT coefficients with a secret m -bit binary key $k = k_1k_2\dots k_m$. The effect of this scheme is that the encryption function randomly changes the sign bits of the DCT coefficients depending on the corresponding bit value of k . Let us denote the sign bits of the DCT coefficients in an MPEG stream that belong to the same GOP (Group Of Pictures) by $s_1s_2\dots s_n$. Then, a bit s_i will not be changed if the value of the key bit $k_{i \pmod m}$ is 0, and it will be

flipped if the value of the key bit $k_{i \pmod{m}}$ is 1. The next GOP would simply reuse the secret key, which serves for the resynchronization purposes. The ability to resynchronize the video stream is important in the case of the unreliable network, transmission errors, and VCR-like functionality such as fast forwarding or rewinding.

The security of Algorithm II depends on the length of the key. The authors encourage the use of a long binary key; however, too long key may be infeasible and impractical. On the other hand, if the key is short, the system can be easily broken. If the key is as long as the video stream and it is unique and used only once that would correspond to Vernam cipher (also referred to the one-time pad), which is known to be absolutely secure. However, this is highly impractical for mass applications such as VOD (Video on Demand) and similar. On the other hand, if the key is too short, the whole method simply becomes the Vigenère-like cipher, a well-studied classical cipher for which there are many attacks developed (for example, the Kasiski analysis) [18]. In addition, Vigenère is highly sustainable to the known-plaintext attack (one can literally read off the secret key). Authors in [17] suggest the use of the pseudo-random generator that generates a stream of pseudo-random bits to be the key k of arbitrary length. The trouble with this approach is the security, randomness, and speed of this PRNG (Pseudo-Random Number Generator). The additional issues include the synchronization of the PRNG and the secure transmission of the seed (the secret key) for the P-RNG.

Algorithm III (MVEA)

The Algorithm III (MVEA) is an improvement to the Algorithm II (VEA) described above. It includes the following additions: the sign bits of differential values of motion vectors in P- and B-frames can also be randomly changed. This type of improvement makes the video playback more

random and more non-viewable. When the sign bits of differential values of motion vectors are changed, the directions of motion vectors change as well. In addition, the magnitudes of motion vectors change, making the whole video very chaotic. The authors found that the encrypting of sign bits of motion vectors makes the encryption of sign bits of DCT coefficients in B- and P-frames unnecessary.

Furthermore, the original Algorithm III (MVEA) was designed to encrypt only the sign bits of DC coefficients in the I-frames of MPEG video sequence, while leaving the AC coefficients unencrypted. This significantly reduces the computational overhead, but it opens up new security risks. Namely, because the DC coefficient and the sum of all AC coefficients within the block are related, an adversary may use the unencrypted AC coefficients to derive the unknown (encrypted) DC coefficients [16,17]. For that reason, the authors recommend encrypting all DCT coefficients in the I-frames for applications that need higher level of security.

Just like the Algorithm II (VEA), the algorithm III (MVEA) relies on the secret m -bit key k . Also, the resynchronization is done at the beginning of a GOP. Unfortunately, the fundamental security issues and problems that are applicable to VEA are also applicable to MVEA.

Algorithm IV (RVEA)

Finally, the Algorithm IV (RVEA) is significantly more secure approach than the previous three algorithms. This approach is considered to be robust under both ciphertext-only attack and known-plaintext attack. The difference between RVEA and MVEA/VEA algorithms is that RVEA uses conventional symmetric key cryptography to encrypt the sign bits of DCT coefficients and the sign bits of motion vectors. The conventional cryptosystems are well mathematically understood, and thoroughly tested by the experts in the field, which definitely

adds on to the security aspect of RVEA. The selective approach significantly speeds up the process of conventional encryption by only encrypting certain sign bits in the MPEG stream. The experiments show that the encryption quality is quite good considering the amount of information changed.

In the Algorithm IV (RVEA), the sign bits of DCT coefficients and motion vectors are simply extracted from the MPEG video sequence, encrypted using a fast conventional cryptosystem such as AES, and then restored back to their original position in the encrypted form. The effect of this is similar to VEA/MVEA where the sign bits are either flipped or left unchanged. The authors limited the number of bits for encryption to at most 64 for each MPEG stream macroblock, for the purposes of reducing and bounding the computation time. Next, we describe exactly how these sign bits are selected for encryption.

Each MPEG video slice consists of one or more *macroblocks*. The macroblock corresponds to a 16x16 pixel square, which is composed of four 8x8 pixel luminance blocks denoted by Y_1, Y_2, Y_3 and Y_4 , and two 8x8 chrominance blocks C_b and C_r . Each of these six 8x8 blocks can be expressed as a sequence $ba_1a_2\dots a_n$, where n is at most 64 since b is the code for DC coefficient and a_i is the code for a non-zero AC coefficient. Then, for each such 8x8 block, we can define the sequence $\mathbf{ba_1a_2\dots a_n}$, that corresponds to the sign bits of the matching DCT coefficients b, a_1, a_2, \dots , and a_n . Figure 7 shows the order of selection of the sign bits for each macroblock. The obvious reason for this selection order is that the DC and higher order AC coefficients are carrying much more information than the low-order AC coefficients.

In conclusion, RVEA only encrypts the fraction (typically about 10%) of the whole MPEG video by using the conventional secure cryptographic schemes such as DES, IDEA, AES, etc. Therefore, the Algorithm IV (RVEA) is a much better method than the previous three algorithms

in terms of security. Furthermore, it saves up to 90% of the computation time comparing to the naive approach.

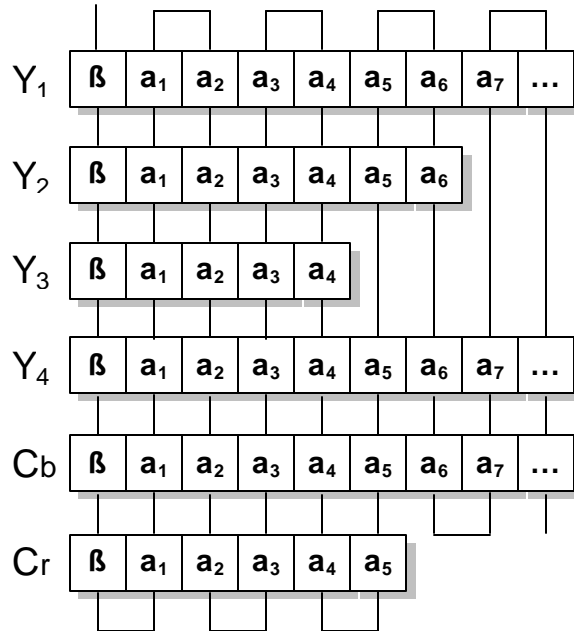


Figure 7. The bit selection order in RVEA [16,17].

4.2.6 Video Encryption Methods by Alattar, Al-Regib and Al-Semari, 1999

In 1999, Alattar, Al-Regib and Al-Semari presented the three methods for selective video encryption based on DES cryptosystem [22]. These methods, called simply *Method I*, *Method II* and *Method III*, were computationally improved versions of the previous work from two of the co-authors [21], which is referred to as *Method 0*.

The first algorithm (Method 0), proposed by Alattar and Al-Regib in [21], essentially encrypts all macroblocks from I-frames and the headers of all prediction macroblocks using DES cryptosystem. This method performs relatively poorly because encryption is carried out on 40%-79% of the MPEG video stream [34].

In Method I, the data of every n^{th} macroblock from the I-frame of MPEG video stream is encrypted using DES cryptosystem, while the information from the all other I-frame macroblocks is left unencrypted. The value of n was not specified, and it can be chosen depending on the application needs. If the value of n is 2 then the encryption is performed on approximately a half of all I-frame macroblocks, but the security level is higher. On the other hand, if the value of n is higher, the computational savings are bigger, yet the security level is lower. An important observation is that even though the certain number of I-macroblocks is left unencrypted, they are not expected to reveal any information about the encrypted ones [22]. However, this method does not hide the motion information, and the authors suggest an improved algorithm, which will be discussed next.

To improve the security of Method I, Alattar, Al-Regib and Al-Semari suggested Method II, which additionally encrypts the headers of all predicted macroblocks using DES. Since DES is a block cipher that operates on 64-bit blocks, a 64-bit segment starting from the header of a predicted macroblock is processed in the beginning. This segment may include exactly the whole header (which is the rare case when header size is equal to 64 bits), a part of the header (when header size is larger than 64 bits), or the whole header along with the part of the macroblock data (when the header size is smaller than 64 bits). In the case when the encrypted segment contains a part of the header, an adversary would have serious problems with synchronization, which adds to the security regarding motion vectors [22]. The security is further increased if the encrypted segment also contains a part of the macroblock data. The computation performed using Method II is clearly faster than that of the Method 0, but slower than that of Method I.

Finally, Alattar, Al-Regib and Al-Semari proposed Method III to reduce the amount of computation from Method II. Namely, instead of encrypting all predicted macroblocks, the

encryption in Method III is performed on every n^{th} predicted macroblock, along with encrypting every n^{th} I-macroblock.

4.2.7 Partial Encryption Algorithms for Videos by Cheng and Li, 2000

The partial encryption schemes for still images introduced by Cheng and Li are also further extended to the videos [23]. The approaches proposed by Cheng and Li are not suitable for JPEG image compression, and thus naturally also not suitable for the MPEG video compression standard. Instead, the partial encryption algorithms are designed for the video compression methods, which use either quadtree compression or wavelet compression based on zerotrees for the video sequence intraframes, motion compensation, and residual error coding. For example, the partial encryption is applicable to the videos that are based on the Set Partitioning In Hierarchical Trees (SPIHT) image compression algorithm, which is an application of zerotree wavelet compression.

Cheng and Li's partial encryption algorithms are designed to disguise the intraframes (I-frames), the motion vectors, and the residual error code of the given video sequences. In both quadtree compression and wavelet compression based videos, all I-frames are encrypted using the previously discussed methods for partial encryption of still images by Cheng and Li [23].

In addition, it is also important to encrypt the motion vectors. If the motion vector information is unencrypted, the adversary may be able to use an image frame to obtain approximations to the successive frames. Almost all motion estimation algorithms divide the frame into blocks and try to predict their movement (the position in the next frame) by constructing the estimated motion vector for each block. The blocks that belong to the same large object often have identical motion vectors and it is efficient to encode these vectors together. The authors restrict to those

video encryption algorithms that use a quadtree for merging these blocks [23]. Then, quadtree partial encryption is used to encrypt the motion vectors.

Finally, for the security purposes it is important to encrypt the residual error as well. Unencrypted residual error may reveal the outline of a moving object. The residual error is often treated as an image frame and then compressed using some standard image compression algorithm. Again, we restrict ourselves to video compression algorithms that use either quadtree or wavelet based image compression algorithm to compress the residual error frames. Thus, partial encryption schemes for both quadtree and wavelet compression can be applied to the residual error encryption.

4.2.8 MHT-Encryption Scheme and MSI-Coder by Wu and Kuo, 2000 and 2001

In their recent work [24,25], Wu and Kuo proposed two selective encryption algorithms for MPEG video sequences: *MHT-encryption scheme* (where MHT stands for Multiple Huffman Tables), and *MSI-coder* (where MSI stands for Multiple State Indices).

The MHT-encryption scheme is based on changing the standard Huffman table into a different one based on a secret key. Shi, Wang and Bhargava's Algorithm I, which was discussed above, proposed the similar idea. The basic procedure for generating the entropy code based on different Huffman tree (the "encryption") is summarized in the following three steps:

- (1) Generate 2^k different Huffman tables, numbered from 0 to $2^k - 1$.
- (2) Generate a random vector $P = (p_1, p_2, \dots, p_n)$, where each p_i is a k -bit integer ranging from 0 to $2^k - 1$.

- (3) For the i^{th} symbol in the Huffman entropy encoding stage, use the Huffman table indexed by $p_{(i-1 \pmod{n})+1}$ to encode it.

For the purposes of not affecting the compression ratio, it is important to select optimal Huffman tables that perform similarly to the standard one. This, of course, creates additional constraint that may be a security risk [18]. The authors propose creating a number of so-called training multimedia (images or audio data) that can be used to represent majority of the multimedia data in the world, so that the associated Huffman code that can be easily generated is optimal for the general use. The authors' experimental results show relatively small increase in the output images on average, which means that in this approach the compression performance will be satisfactory, even though the symbols are chosen from the multiple Huffman tables.

Another method for obtaining optimal Huffman trees is proposed in [24,25]. Namely, only 4 base optimal Huffman trees are generated, while the other optimal tables are produced by what is referred to as a *Huffman tree mutation process*. The label pairs are permuted using the secret permutation, as illustrated by the example in Figure 8. Clearly, there is no change in the optimality. In addition, the authors propose two slight security enhancements for the MHT-encryption scheme to increase the resistance against known-plaintext and chosen-plaintext attacks (see [25]).

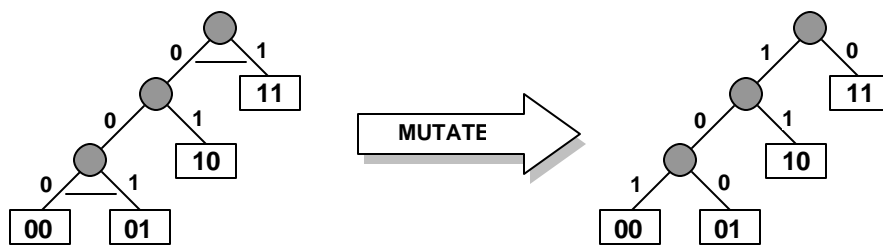


Figure 8. The Huffman tree mutation process [24,25].

The second method (MSI-coder) is designed for video codecs that use QM coding instead of the Huffman coding. The QM coder is an adaptive coder based on low-cost probability estimation. It dynamically adjusts the statistical model to a received binary sequence symbols, whose probability is updated at a low computational cost via table look-up. In the standard QM coder there is a total of 113 possible values for the state index, whose initial value is 0 indicating the equal probabilities of 0's and 1's. Since there are only 113 possibilities to initialize the state index, initializing it to a different "secret" value is pretty useless. Therefore, the MSI-coder uses 4 indices, which are initially set to different secret values, and used alternatively in a secret order. MSI-coder consists of the following 4 steps:

- (1) Generate a random key $K = \{(s_0, s_1, s_2, s_3), (p_0, p_1, \dots, p_{n-1}), (o_0, o_1, \dots, o_{n-1})\}$, where each s_i is a 4-bit integer, while each p_i and o_i is a 2-bit integer.
- (2) Initialize four state indices (I_0, I_1, I_2, I_3) to (s_0, s_1, s_2, s_3) .
- (3) To encode the i^{th} bit from the input, we use the index $I_{p_i \pmod{n}}$ to determine the probability estimation value Q_e .
- (4) If the state update is required after encoding the i^{th} bit from the input, all state indices are updated except for $I_{o_i \pmod{n}}$.

The not-so-obvious step 4 is included to ensure that the state indices will become different even if they become the same at some point. By experiments, Wu and Kuo showed that the MSI encoded data is increased in size by only 0.82% - 4.16% when compared to the data encoded with the original QM coder. Like the previous scheme, the MSI-coder can similarly be slightly enhanced against the known-plaintext and chosen-plaintext attacks [25].

4.2.9 Format-Compliant Configurable Encryption by Wen et al., 2002

This approach, introduced in [26], is generalizing the ideas of selective encryption into format-compliant method. It stresses the importance of maintaining the compliance to a standard (such as MPEG-4 codec). In this model, the data is divided into information-carrying and not information-carrying portions. We only need to encrypt the information-carrying fields. These fields are either fixed length code (FLC) codewords, or variable length code (VLC) codewords. To achieve the format compliance, we extract the bits that are chosen to be encrypted, concatenate them, and then encrypt with secure scheme such as DES. Finally, we put the encrypted bits back to their original positions. For FLC coded fields, this operation most likely creates compliant bit-stream. If it doesn't, we apply same approach for VLC coded fields. For VLC coded fields, encrypting concatenated codewords may not result in another valid codeword. In order to still maintain compliance, we assign a fixed length index to each codeword in VLC codeword list, encrypt the index concatenation, and map it back to the codewords. This, however, introduces the overhead, but the percentage of overhead is content-dependent. In their proposal, Wen et al. stress the importance of preserving the format of standard multimedia codecs when including the additional filters such as the encryption and decryption.

4.2.10 Selective Scrambling Algorithm by Zeng and Lei, 2002

Zeng and Lei proposed the frequency domain scrambling algorithm that groups the transform coefficients into blocks or segments that undergo some or all of the following three operations: (1) selective bit scrambling, (2) block shuffling, and (3) block rotation [27].

Most of the video/image compression algorithms are based on either wavelet transform, or discrete cosine transform (DCT). Therefore, the selective scrambling algorithm proposed by

Zeng and Lei was designed for two types of compressions: wavelet transform based and 8x8 DCT based.

In the case of the wavelet transform based codecs, the selective bit scrambling is performed on the sign bits since they are not highly compressible. Other kinds of bits are refinement bits and significance bits. Significance bits are the worst candidates for selective scrambling since they have the lowest entropy, and therefore they are highly compressible. On the other hand, refinement bits are somewhere in the middle as far as the compressibility, and thus they can be scrambled too, but the degradation level is not as high as in the sign bit scrambling. In the block shuffling procedure, each subband is divided into a number of blocks of equal size, but for different subbands the blocksize can vary. These blocks can be shuffled using the table that is generated by some secret key. Visual effects of this procedure are dramatic, while the output data increases by only about 5% [27]. Finally, the set of blocks can further be rotated by a cyclic shift controlled by the secret key.

For 8x8 DCT based algorithms, each I-frame is divided into segments, each containing several macroblocks. Within the each segment DCT coefficients with the same frequency location are shuffled. In addition, sign bits of each coefficient can be encrypted. They are flipped “randomly” according to some secret key. The authors suggest that scrambling the P- and B-frames, as well as the motion vectors, is a good idea for security purposes, since they give some information about the original I-frames. If that is too computationally expensive, at least the I-blocks in P- and B-frames should be scrambled.

5. IMAGE ENCRYPTION TECHNIQUES

When dealing with still images, the security is often achieved by using the naive approach to completely encrypt the entire image. However, there are number of applications for which the naive based encryption and decryption represents a major bottleneck in communication and processing. For example, a limited bandwidth and processing power in small mobile devices calls for a different approach. Additionally, different image encryption techniques are used as a basis for selective video encryption. In this section, we introduce a few newly proposed techniques for image encryption, based on encrypting only certain parts of the image in order to reduce the amount of computation. This is referred to as *selective image encryption*. A similar idea was already explored in the previous section.

5.1 Partial Encryption Algorithms by Cheng and Li, 2000

In [23], Cheng and Li proposed partial encryption methods that are suitable for images compressed with two specific classes of compression algorithms: (1) quadtree compression algorithms, and (2) wavelet compression algorithms based on zerotrees. The encryption/decryption function is not specified, but in general, the user can select the conventional cryptosystem such as IDEA, AES, etc.

The quadtree image compression produces the quadtree structure and the parameters describing each block in the tree. For the simplicity of argument, it is assumed that the only parameter that describes each block is the average intensity. The intensity for each block does not provide enough information about the original image, but the quadtree decomposition allows the reconstruction of outlines of objects in the original frame [23]. Therefore, the quadtree

encryption method proposed by Cheng and Li encrypts only the quadtree structure, while the block intensities, located at leaf nodes of a quadtree, are left unencrypted. The quadtree partial encryption can be applied to both lossless and lossy image compression. Another consideration is the ordering of the leaf nodes for the transmission. Figure 6 illustrates the different leaf orderings to be discussed. The four branches of each tree node in Figure 6 correspond to NE (North-East), SW (South-West), SE (South-East), and NE (North-East) quadrant in that order, while the black leaf node has value 0 and white leaf node has value 1. Vulnerability to certain cryptanalytical attacks exists if the ordering of leaf nodes is done using the Leaf Ordering I, where leafs are encoded via the inorder traversal of the quadtree. If we would apply Leaf Ordering I to the tree from Figure 6, the result would be binary sequence 0010011011110010. Therefore, the authors suggest using the Leaf Ordering II, where leafs are encoded via the opposite of breadth-first traversal of the tree. This increases the security level [23]. For illustration, applying the Leaf Ordering II to the quadtree from Figure 9 would result in the binary sequence 1111000001100101.

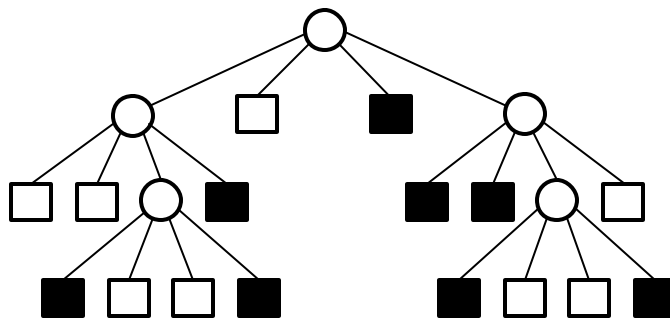


Figure 9. An example of a quadtree for black and white (binary) image [23]

Wavelet compression algorithm based on zerotrees generally transmits the structure of the zerotree, along with the significant coefficients. Typical example of this kind of compression algorithm is the Set Partitioning In Hierarchical Trees (SPIHT) compression algorithm, which transmits the significance of the coefficient sets that correspond to the trees of coefficients. The

SPIHT algorithm uses the significance of the coefficient sets to reconstruct the zerotree structure. In addition, the structure of the tree strongly affects the execution of SPIHT algorithm. The SPIHT encoded image contains different types of data such as the sign bits, the refinement bits, and the significance bits (significance of pixels and significance of sets). Even the small amount of incorrect information at the beginning of the encoded block containing the significance bits causes the failure in the decoding process. Incorrect significance bits often cause the misinterpretation of future bits, unlike the incorrect sign bits or refinement bits, which do not. Therefore, Cheng and Li propose encrypting only the significance information of pixels and sets, as well as the initial threshold parameter n that determines which coefficients are significant [23]. Furthermore, the wavelet transform creates a hierarchy of coefficient bands called *pyramid decomposition*, and the number in the band label is referred to as the *pyramid level*. The authors restrict to encrypting only those significance bits that are in the two highest pyramid levels, since all other pixels and sets are derived by decomposing the sets from the highest two pyramid levels. The information about the significant bits from the first two pyramid levels cannot be deduced just by observing the lower levels.

5.2 Selective Encryption Methods for Raster and JPEG Images by Droogenbroeck and Benedett, 2002

In 2002, Droogenbroeck and Benedett proposed the selective encryption methods for uncompressed (raster) images and compressed (JPEG) images [28].

An uncompressed (raster) grey level image defines 8 bitplanes. The highest (most significant) bitplanes are highly correlated to the original gray level image. On the other hand, the least significant bitplanes appear random-like. Encrypting only the bitplanes that contain nearly uncorrelated values would decrease the vulnerability to the known-plaintext attacks [28]. The

authors propose the encryption scheme that is consisted of xoring the selected bitplanes with a key that has the same number of bits as the bits that are to be encrypted. According to Droogenbroeck and Benedett, at least 45 least significant bitplanes should be encrypted to achieve the satisfactory visual degradation of the image. It is important to note that the partial degradation of the image is not sufficient for applications where high security is a must.

The second method is designed to selectively encrypt the JPEG compressed images. The Huffman entropy encoding creates the symbols based on the run-length coding. In particular, zero coefficients are encoded into the symbols that contain the run of zeros and the magnitude categories for the non-zero coefficients that terminate the runs. These symbols are encoded with the 8-bit Huffman codewords, followed by the appended bits that specify the sign and magnitude of the non-zero coefficients. In the proposed method, only the appended bits that correspond to the selected AC coefficients are encrypted. The DC coefficients are left unencrypted, since their values are highly predictable [28]. On the other hand, the codewords are left unencrypted for the synchronization purposes.

5.3 Selective Bitplane Encryption Algorithm by Podesser, Schmidt and Uhl, 2002

In [35], Podesser, Schmidt and Uhl proposed a selective encryption algorithm for the uncompressed (raster) images, that is quite opposite from the first method by Droogenbroeck and Benedett [28]. In the raster image that consists of 8 bitplanes, Schmidt and Uhl's algorithm encrypts only the most significant bitplanes. The proposed underlying cryptosystem for this method was AES. However, without loss of generality, any fast conventional cryptosystem may be chosen instead.

After performing the experiments, Podesser, Schmidt and Uhl came to the same conclusion in [35] as Droogenbroeck and Benedett did in [28]. That is, encrypting only the most significant bitplane is not secure. Podesser, Schmidt and Uhl argue that MSB bitplane can be reconstructed with the aid of the unencrypted remaining bitplanes. Therefore, they suggest encrypting at least 2 or 4 bitplanes. Encrypting only two bitplanes is sufficient if severe alienation of the image data is acceptable, while encrypting four bitplanes provides high security. It is up to the user to determine whether the degradation method by encrypting only two bitplanes is sufficient for the given application, or more secure approach of encrypting four bitplanes is needed.

6. AUDIO AND SPEECH ENCRYPTION TECHNIQUES

In many applications, the audio sequences need confidentiality during transmission. Sometimes it is enough to apply the naive approach, but in many instances this is too computationally expensive (for example, in small mobile devices). As far as the security is concerned, perhaps the most important type of audio data is speech. Unlike in the case of music files and similar entertainment audio sequences, in many applications speech requires substantial level of security. This section discusses the methods for speech protection, as well as the protection of general audio compressed sequences such as MP3.

6.1 ENCRYPTION OF COMPRESSED SPEECH

Traditional speech scrambling has a long history. Like in the case of analog video signal, the early work was based on analog devices that simply permute speech segments in the time domain

or distort the signal in the frequency domain by applying inverters and filter banks. These schemes are very insecure by today's computing standards [24]. Today's research is shifted towards securing the speech in the digital form, by applying partial or selective encryption [24, 29, 30].

6.1.1 Selective Encryption Algorithm for G.723.1 Speech Codec by Wu and Kuo, 2000

One of the most popular digital speech codecs is the ITU recommendation G.723.1 compression standard. It has a very low bit rate, and it is extremely suitable for voice communications over the packet-switching based networks. It is also a part of the ITU H.324 standard for video-conferencing/telephony over the regular public telephone lines.

The compression is based on the analysis-by-synthesis method. The encoder incorporates the entire decoder unit, which synthesizes a segment of speech according to given input coefficients. The encoder then changes these coefficients until the difference between the original speech and the synthesized speech is within the acceptable range. The decoding is performed with three different decoders: LSP decoder, pitch decoder and excitation decoder, and the G.723.1 coefficients can be categorized depending on the decoder they are fed to. In addition, this codec can work in either 6.3 Kbps or 5.3 Kbps modes.

In [24], Wu and Kuo suggest applying selective encryption to the most significant bits of all important G.723.1 coefficients. They identified the following coefficients as the important ones: the LSP codebook indices, the lag of pitch predictor, the pitch gain vectors, the fixed codebook gains, and the VAD mode flag. The total number of selected bits for encryption is 37 in each frame, which is less than 1/5 of the entire speech stream at the 6.3 Kbps rate, and less than 1/4 of the entire speech stream at the 5.3 Kbps rate.

6.1.2 Perception-Based Partial Encryption Algorithm by Servetti and De Martin, 2002

In 2002, Servetti and De Martin published a perception-based algorithm for partial encryption of telephone bandwidth speech [29]. The algorithm was implemented for the ITU-T G.729 codec for a rate of 8 Kbps.

Servetti and De Martin suggested two methods for partial telephony speech encryption. The first method was aimed to have low-security, but high bit rate (similar to the degradation schemes for videos and images). The goal of this method was to degrade the speech to enough to prevent immediate eavesdropping. However, more substantial cryptanalysis could reveal the encrypted speech. The second algorithm encrypts more of the bit-stream, and its goal is to provide more security. Servetti and De Martin argue that even though it encrypts only about a half of the bit-stream, the algorithm's security level is comparable to that of the naive algorithm.

There are total of 15 segments from the ITU-T G.729 codec output: L_0 (1 bit), L_1 (7 bits), L_2 (5 bits), L_3 (5 bits), P_1 (8 bits), P_0 (1 bit), S_1 (4 bits), C_1 (13 bits), GA_1 (3 bits), GB_1 (4 bits), P_2 (5 bits), S_2 (4 bits), C_2 (13 bits), GA_2 (3 bits), and GB_2 (4 bits). Rather than describing it in words, the selected bits for encryption in Servetti and De Martin's low and high security algorithms are shown in Figure 10. The experiments showed that by applying the high security selective encryption algorithm, the speech is highly scrambled, it cannot be reconstructed from the known bits, and the total computational saving is more than 50%.

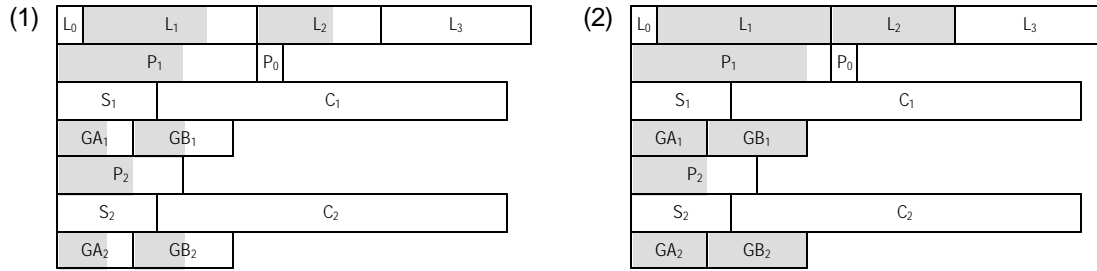


Figure 10. Partial encryption for G.729 speech codec (grayed bits are selected for encryption):
 (1) low security algorithm, and (2) high security algorithm [29].

6.2 ENCRYPTION OF COMPRESSED AUDIO

There are numerous applications where the general audio data needs to be protected, and the expensive naive approach might be infeasible. This demand created some research initiatives towards developing selective encryption approaches for compressed audio streams.

6.2.1 MP3 Security Methods by Thorwirth, Horvatic, Weis and Zhao, 2000

Thorwirth et al., proposed a selective encryption algorithm for perceptual audio coding (PAC) based compression standards, such as MPEG-I Layer-3 (MP3) [36]. In their proposal, the authors concentrated on analyzing encryption of MP3 encoded audio files.

In the late 1980's, the researchers at Fraunhofer Institute developed the audio coding technique called MPEG-I Layer-3, or simply MP3. This codec achieved the stunning compression ratio of 1:10 - 1:12, while still maintaining the CD sound quality. It is a perception audio codec (PAC) that uses the perceptual masking algorithms to discard the redundancy from the raw audio digital signal. All components of the raw audio signal that are inaudible to the human hearing system

are identified as the redundant data by the perceptual models. The compression in MP3 audio standard is based on the Huffman entropy code.

Hearable frequency spectrum is divided into the segments called the *audio quality layers*. The 20Hz-4KHz range represents the lowest audio quality layer, while the CD quality range is 20Hz-22KHz. The authors propose encrypting the audio quality layers associated with the compressed audio data, in order to enforce audio quality control by means of encryption. The algorithm identifies the frequency spectrum boundaries that are used to determine the audio quality layers. Each known layer is then grouped into the blocks of equal size and a block cipher is applied. After encryption stage, the encrypted data can be easily plugged back to the MP3 bit-stream, thus preserving the format. Such encrypted MP3 sequences are decodable and playable by any valid MP3 player [36].

7. SUMMARY

In this chapter, many of the current important multimedia encryption techniques were presented and analyzed. In brief, the best way of protecting multimedia data is by means of the naive algorithm; i.e., by encrypting the entire multimedia bit sequence using a fast conventional cryptosystem. Yet, in many instances this is not possible due to the expensive computational overhead introduced by the encryption and decryption stages. Much of the past and current research targets encrypting only a carefully selected part of the multimedia bit-stream in order to reduce the computational load, and yet keep the security level high. Many of the proposed schemes only achieved moderate to low security, which may find applications in which quality degradation is preferred over absolute security. However, only few of the proposed methods

promise to achieve substantial security, which is the number one requirement in many multimedia applications. It is also worthwhile mentioning that many of the above proposed audiovisual encryption techniques are flexible in terms of selecting the underlying cryptosystem, regardless of the choice made by the authors. In general, a well-studied, fast and secure conventional cryptosystem should be chosen. Surely, AES is a good candidate, but there are numerous other successful modern cryptosystems with similar and, in some aspects, even better performances.

Further research should be directed towards developing substantially secure encryption schemes for high bit rate and high-quality audiovisual data. Furthermore, the experts should investigate the security aspects of the newly proposed multimedia encryption techniques.

REFERENCES

1. IEEE Transactions on Circuits and Systems for Video Technology: Special Issue on Authentication, Copyright Protection, and Information Hiding, Vol. 13, No. 8, August 2003.
2. B. Furht and D. Socek, "Multimedia Security: Encryption Techniques," IEC Comprehensive Report on Information Security, International Engineering Consortium, Chicago, IL, 2003.
3. A.M. Eskicioglu, "Protecting Intellectual Property in Digital Multimedia Networks," IEEE Computer, July 2003, pp. 39-45.
4. X. Liu and A.M. Eskicioglu "Selective Encryption of Multimedia Content in Distribution Networks: Challenges and New Directions," IASTED International Conference on Communications, Internet and Information Technology (CIIT 2003), Scottsdale, AZ, November 17-19, 2003.
5. P.C. van Oorschot, A.J. Menezes, and S.A. Vanstone, "Handbook of Applied Cryptography," CRC Press, Inc., 1997.
6. D.R. Stinson, "Cryptography Theory and Practice," CRC Press, Inc., 2002.
7. T.B. Maples and G.A. Spanos, "Performance study of selective encryption scheme for the security of networked real-time video," in Proceedings of the 4th International Conference on Computer and Communications, Las Vegas, NV, 1995.
8. G.A. Spanos and T.B. Maples, "Security for Real-Time MPEG Compressed Video in Distributed Multimedia Applications," in Conference on Computers and Communications, 1996, pp. 72-78.
9. J. Meyer and F. Gadegast, "Security Mechanisms for Multimedia Data with the Example MPEG-1 Video," Project Description of SECMPEG, Technical University of Berlin, Germany, May 1995.
10. L. Tang, "Methods for Encrypting and Decrypting MPEG Video Data Efficiently," Proceedings of the 4th ACM International Multimedia Conference, Boston, MA, November 18-22, 1996, pp. 219-230.
11. I. Agi and L. Gong, "An Empirical Study of Secure MPEG Video Transmission," Proceedings of the Symposium on Network and Distributed Systems Security, IEEE, 1996.
12. L. Qiao and K. Nahrstedt, "A New Algorithm for MPEG Video Encryption," Proceedings of the 1st International Conference on Imaging Science, Systems and Technology (CISST '97), Las Vegas, NV, July 1997, pp. 21-29.
13. L. Qiao and K. Nahrstedt, "Comparison of MPEG Encryption Algorithms," International Journal on Computer and Graphics, Special Issue on Data Security in Image Communication and Network, 22(3), 1998.

14. L. Qiao, K. Nahrstedt, and I. Tam, "Is MPEG Encryption by Using Random List Instead of Zigzag Order Secure?" IEEE International Symposium on Consumer Electronics, December 1997. Singapore.
15. C. Shi and B. Bhargava, "A Fast MPEG Video Encryption Algorithm," Proceedings of the 6th International Multimedia Conference, Bristol, UK, September 12-16, 1998.
16. C. Shi, S.-Y. Wang and B. Bhargava, "MPEG Video Encryption in Real-Time Using Secret key Cryptography," 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99), Las Vegas, NV, June 28 - July 1, 1999.
17. B. Bhargava, C. Shi, and Y. Wang, "MEPG Video Encryption Algorithms", August 2002, available at <http://raidlab.cs.purdue.edu/papers/mm.ps>
18. T. Seidel, D. Socek, and M. Sramka, "Cryptanalysis of Video Encryption Algorithms," to appear in Proceedings of The 3rd Central European Conference on Cryptology TATRACRYPT 2003, Bratislava, Slovak Republic, 2003.
19. T. Seidel, D. Socek, and M. Sramka, "Parallel Symmetric Attack on NTRU using Non-Deterministic Lattice Reduction," to appear in Designs, Codes and Cryptography, issue on Proceedings of the Third Pythagorean Conference, Rhodes, Greece, 2003.
20. N. Howgrave-Graham, P. Nguyen, D. Pointcheval, J. Proos, J.H. Silverman, A. Singer, and W. Whyte, "The Impact of Decryption Failures on the Security of NTRU Encryption," in Proceedings of Crypto 2003, Santa Barbara, CA, 2003.
21. A. Alattar and G. Al-Regib, "Evaluation of selective encryption techniques for secure transmission of MPEG video bit-streams," in Proceedings of the IEEE International Symposium on Circuits and Systems, vol. 4, pp IV-340-IV-343, 1999.
22. A. M. Alattar, G. I. Al-Regib and S. A. Al-Semari, "Improved Selective Encryption techniques for Secure Transmission of MPEG Video Bit-Streams," Proceedings of the 1999 International Conference on Image Processing (ICIP '99), Vol. 4, Kobe, Japan, October 24-28, 1999, pp. 256-260.
23. H. Cheng and X. Li, "Partial Encryption of Compressed Images and Video," IEEE Transactions on Signal Processing, 48(8), 2000, pp. 2439-2451.
24. C.-P. Wu and C.-C. J. Kuo, "Fast Encryption Methods for Audiovisual Data Confidentiality," SPIE International Symposia on Information Technologies 2000, Boston, MA, November 2000, pp. 284-295.
25. C.-P. Wu and C.-C. J. Kuo, "Efficient Multimedia Encryption via Entropy Codec Design," Proceedings of SPIE Security and Watermarking of Multimedia Content III, Volume 4314, San Jose, CA, January 2001.
26. J. Wen, M. Severa, W. Zeng, M. Luttrell, and W. Jin, "A Format-Compliant Configurable Encryption Framework for Access Control of Video," IEEE Transactions of Circuits and Systems for Video Technology, Vol. 12, No. 6, June 2002, pp. 545-557.
27. W. Zeng and S. Lei, "Efficient Frequency Domain Selective Scrambling of Digital Video," IEEE Transactions on Multimedia, Vol. 5, No. 1, March 2002, pp. 118-129.
28. M. Van Droogenbroeck and R. Benedett, "Techniques for a Selective Encryption of Uncompressed and Compressed Images," Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS) 2002, Ghent, Belgium, September 9-11, 2002.
29. A. Servetti and J.C. De Martin, "Perception Based Partial Encryption of Compressed Speech," IEEE Transaction on Speech and Audio Processing, Vol. I, No. 8, 2002.
30. A. Servetti, C. Testa and J. C. De Martin, "Frequency-Selective Partial Encryption of Compressed Audio," IEEE International Conference on Acoustics, Speech and Signal Processing, Hong-Kong, April 6-10, 2003.
31. D. McGrew, M. Naslund, K. Norman, R. Blom, E. Carrara, and D. Oran "The Secure Real time Transport Protocol (SRTP)," Internet draft, 2001.
32. S. U. Shin, K. S. Sim and K. H. Rhee, "A Secrecy Scheme for MPEG Video Data Using the Joint of Compression and Encryption," Second International Workshop on Information Security (ISW '99), Kuala Lumpur, Malaysia, November 1999, Lecture Notes in Computer Science, Vol. 1729, pp. 191-201, 1999.
33. M. Wu and Y. Mao, "Communication-Friendly Encryption of Multimedia," 2002 International Workshop on Multimedia Signal Processing, St. Thomas, US Virgin Islands, December 9-11, 2002.

34. T. Lookabaugh, D. C. Sicker, D. M. Keaton, W. Y. Guo and I. Vedula, "Security Analysis of Selectively Encrypted MPEG-2 Streams," Multimedia Systems and Applications VI Conference, Orlando, FL, September 7-11, 2003.
35. M. Podesser, H.-P. Schmidt and A. Uhl, "Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments," 5th Nordic Signal Processing Symposium, on board Hurtigruten, Norway, October 4-7, 2002.
36. N.J. Thorwirth, P. Horvatic, R. Weis and J. Zhao, "Security methods for MP3 music delivery," Conference Record of the Thirty-Fourth Asilomar Conference on Signals, Systems and Computers, 2000, vol. 2, pp 1831-1835.
37. A. Pommer and A. Uhl, "Selective Encryption of Wavelet-Packet Encoded Image Data," ACM Multimedia Systems Journal, Special Issue on Multimedia Security, 2003.
38. J. Daemen and V. Rijmen, "AES Proposal: Rijndael", AES Algorithm Submission, September 3, 1999, available at <http://www.nist.gov/CryptoToolkit>.