# VIDEO ENCRYPTION

**Undergraduate Research Project**

**Fatih Levent YILMAZ**
**880925-H815**

**Supervisor**
**Professor Sven Nordebo**

Linnaeus University
Faculty of Engineering
Department of Electrical Engineering
Växjö-Sweden

May 2011

**I certify that this report is a product of my own work and no parts of this report have been taken from any other published or unpublished work without giving proper reference.**

**Fatih Levent YILMAZ**

**(02.05.2011)**

# ABSTRACT

Video Encryption is nearly the best method for blocking unwanted seizures and viewing of any transmitted video or information. There are several useful techniques that are available for encryping videos. However, one of the unique speciality for human eye is spotting the irregularity in videos due to weak video decoding or weak choice of video encryption hardware. Because of this situation, it is very important to select the right hardware or else our video transmissions may not be secured or our decoded video may be un-watchable. Every technique has advantages and disadvantages over other technical methods.

Line-cut and rotate video encryption method is maybe the best way of acquiring safe, secured and good quality encypted videos. In this method, every line in the video frame cuts and rotates from different points and these cut points are created from a random matrix. The advantage of this method is to supply a coherent video signal, gives an excellent amount of darkness, as well as good decode quality and stableness. On the other hand it's disadvantages is to have complex timing control and needs specialized encryption equipment.

# ACKNOWLEDGMENT

I would like to express my deepest gratitude to Professor Göran Ewing my supervisor Professor Sven Nordebo for their guidance, advice, criticism, encouragements and insight throughout the project.

I would like to thank to my parents for their love and the support they have provided for my education.

# CONTENT

## FIGURES

## TABLES

# CHAPTER 1
# INTRODUCTION

Many digital visual data are stored on different media and transfered over different type of networks today. Generally, these visual data covers special or secret informations or are connected with financial interests. Eventually, techniques are required to give security optionalities like privacy, integrity, or authentication especially matched for this kind of data types. A comparativily new field, introduced as “Multimedia Security”, is targeted towards these developing, rising technologies and applications. [1]

In addition to watermarking, steganography, and techniques for assessing data completeness and accuracy, providing secrecy and privacy for visual data is between the most important subjects in the field of multimedia security, applications place from digital rights management (DVB, DVD and pay-TV) to delicated personal communications (e.g. video conferencing with encryption). In the following there are given some tangible examples of applications which needs support of some encryption types to accomplish the desired in order of functionalities [1] :

**Telemedicine:** Todays health systems generally has problems from the reason that different doctors cannot access to each others patient data. The big waste of resources for multiple inspections, analyses, and check-ups are an urgent result. Especially, multiple purchases of the same medical image data and loss of previous data of this type has to be refrained to save and store them and to give a timecontiguous medical report for each patient. One solution to these kind of problems is to compose a scattered database groundwork where each doctor has electronic entrance to all existing medical resource associated to a patient, in specific to all medical image data obtained over the years. Further more, alot of medicalists are persuaded that the future of medical health care will be designed by teleradiology and technologies like telemedicine. These reasons presents clearly that there is

immediate need to give and shield the privacy of patient connected medical image resource when saved in databases and transmitted over any kind networks. [1]

**Video Conferencing:** Communication systems which used today generally visual data is related in order to increase the more conventional clearly systems based on audio. However video conferencing (VC) has been existing since to serve such intentions for quite a while and is managed on personal computers over computer networks, video telephony is a technology that has been rising quite recently in the market of mobile cell phone technology.Attempts had been done earlier to marketise videophones operating over traditional phone lines have not been much successful. No matter which technology assists these kind of communication systems, the boundary of possible content exchanged is very broad and possible including private communication between friends to conversate about present developments in their own relationships alongside video conferences among companies to have a talk about their brand-new product emplacement strategies. However, each continuity needs the theme to be secured for obvious reasons against potential eavesdroppers. [1]

**Security:** The required plane of security apparently changes a lot between the six given patterns. The first example group which contains VC, Telemedicine and Surveillance is more related about citizen rights and covering telecommunication moves, yet the second applications group which contains DVD, VOD and Pay-TV News comes from the field of multimedia entertainment where the principle interest is the income stream of the substance owners. Based on this categorisation of applications, one may urgently derive that the first applications group containing a higher range of security as compared to the second one. While the entertainment industry would disagree to this clarification at first vision, “level of security” is not meant in the classical cryptographic sense. However the data content is critical and has therefore to be secured in the case of the first group applications this is not the situation for the entertainment applications. Therefore, it is generally enough and acceptable to set down the quality to an area that an inexact user is not interested to glance the material. In definite applications, this condition is even more tractive as compared to “classical encryption” since users might become interested to get access

to the full quality when compared with encrypted but apparent material. One other important matter is the question how long encrypted visual data has to lean on against possible attacks. Again, the first application group has elevated necessities as the second one, where one could possibly specify that DVD and VOD films have to be secured only as long as they are comparatively new. A remarkable condition are Pay-TV News where the information loses its worth after some hours already. In other case it is ofcourse not true that entertainment applications do involve a much lower “classical” securace level in general – a possible discussion might be that it does not matter for the income stream if some hundred specialists globaly are able to decode encrypted entertainment substance (since their share of the entire payments is negligible). The following reasons shows us this is not true [1] :

- As from peer to peer music dispersion networks, effective techniques cosist to transfer digital media information to a wide caliber of possible clients over the internet at a lower cost. As the increasing network bandwidth in mind, peer to peer video dispertion is currently taking off and might soon become a menace to the income of content owner as it is already the case for audio. [1]

- As we already know from attacks against CSS,DVD and Pay-TV systems, the internet is a good means to disperse key data, decryption software, or even definitions how to build illegal smartcards. [1]

- Because of writable DVDs a medium is at utilization to disperse once cracked entertainment material over classical dispersion channels. [1]

As a result, it is clear that also for entertainment applications even if it may be reasonable to only cast down the material, this casting down must not be convertable. This dispenses encryption schemes relying on poor cryptographic systems from being applied in this field. As long as there are no other limitations (e.g. as effected by complexity or data format restrictions, see below), security must not be sacrificed. [1]

**Speed:** There is one expressive gap among the encryption of visual data and the encryption of data encryption is classically applied to (e.g. text data, web documents): visual data is usually much larger, especially in the case of video encryption. Given this fact together with possible timing constraints or real-time requirements it becomes clear that speed might be an important issue. In telemedicine, a certain delay caused the security mechanisms might be acceptable under certain circumstances as well it might be for surveillance. However, when using telemedicine to control remote surgery or the surveillance system is used to trigger actions of security personnel, significant delay is of course not desirable. VC should by performed under real-time constraints of course. DVD encryption is not time critical at all, decryption must not reduce the frame rate of the video when displayed. In the general Pay-TV environment the situation is similar to the DVD case, whereas for on-line live broadcast (as it is the case is News broadcast) encryption has to be done in real-time as well. However, as long as we have point to point connections or a broadcast scenario as in the examples discussed so far, each involved encryption/decryption module has to process a single data stream. The situation is much worse in the VOD application. A video on demand server has to process several streams (corresponding to the clients' requests) concurrently under real-time constraints. Since each stream has to be encrypted separately, this is the "killer application" with respect to speed in the area of visual data encryption. When discussing speed, two issues that often go hand in hand with execution speed are power consumption and memory requirements. Especially in case the target architecture the encryption has to be performed on is a mobile device low power consumption is crucial for not exhausting the batteries too fast. This could be the case for almost any of the applications discussed except for surveillance. For hardware implementations in general memory is an important cost factor and therefore the corresponding requirements have to be kept to a minimum. [1]

**Bitstream Compliance:** When encrypting visual data given in some specific data format (e.g. video as MPEG,MPEG-2) with a classical cipher (e.g. AES), the result has nothing to do with an MPEG,MPEG-2 stream any more, it is just an unstructured

bitstream. An MPEG player can not decode the video of course, it will crash immediately or, more probably, not even start to process the data due to the lack of header information. While this seems to be desirable from the security viewpoint at first, it becomes clear quickly that causing a common player to be unable to decode has nothing to do with security and provides protection from an unskilled consumer only since a sincere attacker will do much more than just trying to decode encrypted material with a standard decoder. This kind of security is more of the “security by obscurity” type. In order to really assess if the content of a video is protected (and not only the header structures defining how the content has to be interpreted), it can be desirable that the video can be decoded with a standard viewer. Consequently, this requires the encryption to deliver a bitstream which is compliant to the definition of an MPEG video stream. This can be achieved only by keeping the header data intact and by encrypting only the content of the video. Additionally, care has to be taken about the emulation of markers when encrypting video content – the output of the cipher will generally produce symbols which are reserved for bitstream markers of header information in the MPEG definition which will cause a decoder to interpret the following data incorrectly which will cause the viewer to crash eventually.
Assessment of encryption security is not the most important reason for bitstream compliant encryption. Consider the transmission of visual data over a network where the network characteristic (e.g. bandwidth) changes from one segment to the other. The bitrate of the data to be transmitted has to be changed accordingly. Such QoS requirements can be met by modern bitstreams like MPEG,MPEG-4 or JPEG 2000 due to layered or embedded encoding, “older” bitstreams need to be transcoded. In case the bitstream is encrypted in the classical way, it has to be decrypted, the rate adaptation has to be performed, and finally the bitstream is re-encrypted again. All these operations have to be performed at the corresponding network node which raises two problems [1]:

1 The processing overhead at the network node increases and might delay the delivery of the data. [1]

2 The network node must have access to the key data necessary to decrypt the video which is an enormous key management problem. [1]

On the other hand, in case the encryption leads to a compliant bitstream, most adaptation operations can be done directly at the bitstream level without the necessity to decrypt any part of the bitstream. This is simple to implement in the case of layered or embedded bitstreams, if real transcoding has to be performed it is challenging but not impossible to develop techniques for doing this at bitstream level. From our set of example applications, consider accessing a VOD server from a mobile client or using a modem connection in teleradiology as scenarios where this kind of "network-friendliness" is important. [1]

Chapter 2 explains visual data formats.

Chapter 3 is about to cryptography primer.

Chapter 4 explains the image and video encryption

Chapter 5 gives an overall evaluation of the work.

# CHAPTER 2
# VISUAL DATA FORMATS

## 2.1. Image and Video Data

Digital visual data is usually organised in rectangular arrays denoted as frames, the elements of these arrays are denoted as pixels (picture elements). Each pixel is a numerical value, the magnitude of the value specifies the intensity of this pixel. The magnitude of the pixels varies within a predefined range which is classically denoted as “bitdepth”, i.e. if the bitdepth is 8 bit, the magnitude of the pixel varies between 0 and $2^8$-1 (8 bpp means 8 bits per pixel). Typical examples are binary images (i.e. black and white images) with 1 bpp only or grayvalue images with 8 bpp where the grayvalues vary between 0 and 255. Colour is defined by using several frames, one for each colour channel. The most prominent example is the RGB representation, where a full resolution frame is devoted to each of the colours red, green, and blue. Colour representations closer to human perception differentiate among luminance and colour channels (e.g. the YUV model). Video adds a temporal dimension to the purely spatially oriented image data. A video consists of single frames which are temporally ordered one after the other.A single video frame may again consist of several frames for different colour channels. [1]

Visual data constitutes enormous amounts of data to be stored, transmitted, or processed. Therefore, visual data is mostly subjected to compression algorithms after capturing (or digitisation). Two big classes of compression algorithms exist [1]:

- Lossless compression: after having decompressed the data, it is numerically identical to the original values. [1]
- Lossy compression: the decompressed data is an approximation of the original values. [1]

Lossy algorithms achieve much higher compression ratios (i.e. the fraction between original filesize and the size of the compressed file) as compared to the lossless case. However, due to restrictions imposed by some application areas, lossless algorithms are important as well (e.g. in the area of medical imaging lossless compression is mandatory in many countries due to legislative reasons). However, in the multimedia area lossy compression algorithms are more important, the most distinctive classification criterion is whether the underlying integral transform is the discrete cosine transform (DCT) or the wavelet transform. [1]

## 2.2. DCT-based Systems

### 2.2.1. JPEG

The baseline system of the JPEG standard operates on 8 × 8 pixels blocks onto which a DCT is applied. The resulting data are quantised using standardised quantisation matrices, subsequently the quantised coefficients are scanned following a zig-zag order (which orders the data in increasing frequency), the resulting vector is Huffman and runlength encoded. [1]

### 2.2.2. MPEG Video Coding (MPEG-1,2,4)

The main idea of MPEG motion compensated video coding is to use the temporal and spatial correlation between frames in a video sequence for predicting the current frame from previously (de)coded ones. Some frames are compressed in similar manner to JPEG compression, which are random access points to the sequence, these frames are called I-frames. All other frames are predicted from decoded I-frames – in case a bidirectional temporal prediction is done the corresponding frames are denoted B-frames, simple unidirectional prediction leads to P-frames. Since this prediction fails in some regions (e.g. due to occlusion), the residual between this prediction and the current frame being processed is computed and additionally stored after lossy compression. This compression is again similar to JPEG compression but a different quantisation matrix is used. Because of its simplicity and effectiveness block-matching algorithms are widely used to remove temporal correlation. In block-

matching motion compensation, the scene (i.e. video frame) is classically divided into nonoverlapping “block” regions. For estimating the motion, each block in the current frame is compared against the blocks in the search area in the reference frame (i.e. previously encoded and decoded frame) and the motion vector (d1 , d2) corresponding to the best match is returned.The “best” match of the blocks is identified to be that match giving the minimum mean square error (MSE) of all blocks in search area defined as

$$MSE(d_1, d_2) = \frac{1}{N_1 N_2} \sum_{(n_1,n_2)\in\mathcal{B}} [s_k(n_1, n_2) - \hat{s}_{k-l}(n_1 + d_1, n_2 + d_2)]^2$$

where denotes a $N_1$ * $N_2$ block for a set of candidate motion vectors (d1 , d2) , s is the current frame and the reference frame. [1]

The algorithm which visits all blocks in the search area to compute the minimum is called full search. In order to speed up the search process, many techniques have been proposed to reduce the number of candidate blocks. The main idea is to introduce a specific search pattern which is recursively applied at the position of the minimal local error. The most popular algorithm of this type is called “Three Step Search” which reduces the computational amount significantly at the cost of a suboptimal solution (and therefore a residual with slightly more energy). The block giving the minimal error is stored describing the prediction in term of a motion vector which describes the displacement of the block. The collection of all motion vectors of a frame is called motion vector field. [1]

**2.3. Wavelet-based Systems**

Image compression methods that use wavelet transforms have been successful in providing high compression ratios while maintaining good image quality, and have proven to be serious competitors to DCT based compression schemes. A wide variety of wavelet-based image compression schemes have been reported in the literature, ranging from first generation systems which are similar to JPEG only replacing the DCT by wavelets to more complex techniques such as vector quantisation in the

wavelet domain, adaptive transforms, and edge-based coding . Second generation wavelet compression schemes try to take advantage of inter subband correlation – the most prominent algorithms in this area are zerotree encoding and hybrid fractal wavelet codecs. In most of these schemes, compression is accomplished by applying a fast wavelet transform to decorrelate the image data, quantising the resulting transform coefficients (this is where the actual lossy compression takes place) and coding the quantised values taking into account the high inter-subband correlations.
The fast wavelet transform (which is used in signal and image processing) can be efficiently implemented by a pair of appropriately designed Quadrature Mirror Filters (QMF). Therefore, wavelet-based image compression can be viewed as a form of subband coding. A 1-D wavelet transform of a signal s is performed by convolving s with both QMF's and downsampling by 2; since s is finite, one must make some choice about what values to pad the extensions with. This operation decomposes the original signal into two frequency-bands (called subbands), which are often denoted as coarse scale approximation (lowpass subband) and detail signal (highpass subband). Then, the same procedure is applied recursively to the coarse scale approximations several times. [1]

The classical 2-D transform is performed by two separate 1-D transforms along the rows and the columns of the image data, resulting at each decomposition step in a low pass image (the coarse scale approximation) and three detail images . [1]

### 2.3.1 JPEG 2000

The major difference between previously proposed wavelet-based image compression algorithms such as EZW or SPIHT is that EBCOT as well as JPEG 2000 operate on independent, nonoverlapping blocks which are coded in several bit layers to create an embedded, scalable bitstream. Instead of zerotrees, the JPEG 2000 scheme depends on a per-block quad-tree structure since the strictly independent block coding strategy precludes structures across subbands or even code-blocks. These independent ode-blocks are passed down the "coding pipeline" and generate separate bitstreams . Transmitting each bit layer corresponds to a certain distortion

level. The partitioning of the available bit budget between the code-blocks and layers ("truncation points") is determined using a sophisticated optimisation strategy for optimal rate/distortion performance. [1]

The main design goals behind EBCOT and JPEG 2000 are versatility and flexibility which are achieved to a large extent by the independent processing
and coding of image blocks, and of course to provide a codec with a beter rate-distortion performance than the widely used JPEG, especially at lower bitrates. The default for JPEG 2000 is to perform a five-level wavelet decomposition with 7/9-biorthogonal filters and then segment the transformed image into non-overlapping code-blocks of no more than 4096 coefficients which are passed down the coding pipeline. [1]

# CHAPTER 3
# CRYPTOGRAPHY PRIMER

## 3.1. Introduction, Terminology

A message readable by humans or computers is usually called *plaintext.* During a transformation called *encryption* the *ciphertext* is generated out of the plaintext. The intention is that this ciphertext cannot by deciphered except by the legitimate recipient. The recipient performs the inverse transformation called *decryption* to recreate the original plaintext. The main purpose of cryptography is to provide means to create *confidentiality,* to keep data secret. Besides that cryptography gives us *authentication,* the ability to prove the sender and not someone else sent a message, *integrity,* the inability to modify a message without detection and *nonrepudiation,* the inability to falsely deny that the sender sent the message. Many different variants to encrypt data have been invented, these algorithms are also called *ciphers.* [1]

## 3.2. Secret key vs. Public key Cryptography

Modern cryptography gives us two classes of encryption methods [1]:

**symmetric ciphers:** the family of secret-key cryptography. This is the traditional way of secret communications: both the sender and the receiver agree on a single key for their communication, the key is used to encrypt and to decrypt the data. Typical key sizes are 56 bits (DES), 64 bits (CAST or Blowfish), 128 bits (IDEA, AES), 192 and 256 bits (both AES). [1]

**asymmetric ciphers:** also called public-key cryptography. In this case two different keys are involved, they are related by some mathematical property. Everybody who is in possession of the public key can encrypt data, but cannot decrypt it, just the person holding the secret key can decrypt this data. Typical key lengths are 768, 1024, 2048 bits (RSA, ElGamal). Keys based on elliptical curves have typical

lengths of 150 to 200 bits. Key lengths of different algorithms cannot be compared directly because they rely on different mathematical properties, e.g. some researchers say that 170 bits for elliptical curves provide roughly the same security as 1024 bits for RSA. Both classes have their advantages and disadvantages. Symmetric ciphers are fast, but every party must keep the key absolutely secret, this becomes more dangerous with an increasing number of involved parties. Additionally, for each pair of persons who want to communicate a different key must be agreed upon. This makes key management very cumbersome. Asymmetric ciphers are computationally much more expensive, but the public key can be distributed to every one, just one person has to guard the secret part of the key. Besides that there is a multitude of hints that asymmetric ciphers are practically secure but a mathematical proof for any individual cipher is still missing. [1]

### 3.3. Block Ciphers

Secret-key ciphers can be partitioned into two groups: block ciphers and stream ciphers. The unit of operation is a block of data, its size depends on the actual cipher, common values are 64 and 128 bits of data, sometimes larger, older ciphers use smaller blocks, whereas modern ciphers prefer larger block sizes. Block ciphers process one block of input data, transform it to another block of output data (based on some key) of the same size, then proceed to the next block. Stream ciphers operate on a continuous stream of undetermined size, some ciphers process the data bit after bit, other ciphers process the data byte-wise. [1]

One advantage of a block cipher is their speed. Modern processors possess large register banks with long registers, they can process large portions of such blocks at once with single statements. Additionally it is computationally very expensive to mount an exhaustive attack on blocks with size 64 or even 128 bits, compared to blocks of 8 bits: as each bit in the original data influences the output the number of possible outputs doubles with every bit. Therefore the probability of correctly “guessing” the plaintext data is much lower with larger block lengths. [1]

The drawback is that the underlying data must be organised in chunks which have the size of the encryption blocks. If this is not the case, the data must be padded with some additional data. An example for this would be an encrypted remote login session where the user types the keys on the local keyboard and transmits them over some network to the remote host: one byte typed by the user and 7 bytes added for padding is not efficient of course. Sometimes padding is not possible at all. Imagine an application using a record set of data with fixed lengths, when one wants to encrypt pieces of data smaller then the blocksize, it must be padded, but the result does not fit into the record set any more. [1]

### 3.3.1 sidestep: XOR

In the following we will often refer to the XOR-operation It is verypopular in cryptology because it is its own inverse function [1]:

| input1 | input2 | XOR output |
|---|---|---|
| **0** | **0** | **0** |
| **0** | **1** | **1** |
| **1** | **0** | **1** |
| **1** | **1** | **0** |

Given a plaintext bitstream *T* and some secret bitstream *S* then the application of the bitwise XOR-operation gives $C = T \oplus S$. The inverse operation to recompute the original is $T' = C \oplus S = (T \oplus S) \oplus S = T \oplus (S \oplus S) = T \oplus 0 = T$, therefore $T' = T$. These bitstreams *T*, *S* and *C* can be fixed length blocks of bits or a continuous stream of bits. The basic principle is the same: at a certain point during encryption a common secret stream is XORed with the input data, and during decryption the same secret stream is XORed again to produce the original plaintext. Sometimes XOR is also referred to as “addition modulo 2”. [1]

### 3.3.2 Operation Modes for Block Ciphers

Block ciphers can be deployed in different so-called “Operation Modes”. Depending on external requirements or threats to avoid a suitable mode should be chosen. The most common modes are [1]:

**ECB = electronic codebook mode:** the most obvious mode. Here each block of data is encrypted/decrypted completely independent of all data before or after this block. This has the advantage that encryption of multiple blocks in parallel is possible, transmission errors are confined to the current block. The disadvantage is that this mode is susceptible for replay attacks or traffic analysis: a block containing constant data is encrypted every time to the same cipher block provided the same key is used.[1]

**CBC = cipher block chaining mode:** This mode can be a solution against the replay attacks on the ECB mode. Here the output (the ciphertext) of the previous block and the plaintext of the current block are XOR-ed and subsequently encrypted and transmitted/stored. The output of the current block is used for the XOR-operation with the next plaintext block. The decryption works in reverse order: the current ciphertext block is decrypted, the result XOR-ed with the previous ciphertext block, the result is the original plaintext block. A careful reader might have noticed that there is a problem at the beginning, to solve this a dummy block must be transmitted first to start the chain, this block is also called “initialisation vector”. Transmission errors have a slightly larger impact, a flipped bit leads to completely different plaintext version of the current block (as in ECB), and it changes a single bit of plaintext in the next block. [1]

**CFB = cipher feedback mode:** This mode and the following modes have been invented to transform a block cipher into a stream cipher. These modes can be used when data must be encrypted with a size less than the block length. Again, at the beginning an initialisation vector is used, it is put into a shift register (in the following we assume that it shifts from right to left). The contents of this register is

encrypted, the left-most n bits are used for an XOR-operation with plaintext of size n . The resulting cipher data (again size n ) is sent or stored and additionally put back into the shift register on the right side. The decryption operation is almost identical to the encryption operation: it is initialised with the same IV. The received cipher data of size n is put into the shift register, its contents encrypted and the left-most n bits of its output XOR-ed with the just-received cipher data. Care must be taken that the IV is unique for every message, but there are no absolute requirements to keep it secret. [1]

**OFB = output feedback mode:** is similar to CFB. Whereas in CFB-mode the result of the XOR-function is put back into the shift register, in OFB the loop does not involve data from a user, the feedback-loop comprises just the shift register and the encryption function: the result of the encryption function is fed back to the shift register. This has the advantage that the key stream can be computed independently of the data that must be encrypted. [1]

**CTR = counter mode:** Similar to OFB, but here no shift register is used. Instead a counter value is used as the input to encryption function, after each encryption the counter is changed, usually it is incremented by one. An advantage of this mode is that random access mode to some data is possible, without the drawbacks of ECB. [1]

### 3.4. Stream Ciphers

Stream ciphers are different to block ciphers, they do not transform blocks of data to another block of data. Instead, based on a key a (theoretically infinite length) key stream is generated, like a pseudo-random number generator. This key stream is used to XOR it with the plain text. The decryption operation is identical to the encryption operation. Many stream ciphers are based on linear feedback shift registers (LFSR) because of the ease of implementation in hardware. The downside is that in general they are insecure. Examples for LFSR ciphers are the first version of A5, the algorithm used in GSM phones (currently there are three versions A5/1 - A5/3), or the cipher which encrypts the GPS signal. Other ciphers which rely on non-linear or

clock-controlled generators (an example for the latter is the shrinking generator) seem to perform better than the linear generators. A commercial example for a stream cipher is RC4 which is officially a trade secret but many implementations have been published which are said to interoperate with the original. RC4 has some minor weaknesses, some bits of the key leak into the first bits of the encrypted stream, an overview of publications about RC4 is available on the WWW5. RC4 comes with many products, currently the most popular is the WEP (wired equivalent protocol) encryption used in wireless LAN devices, it contains RC4 with a 40 bit key. WEP incorporates the weaknesses of RC4 and has some of its own, the short key does not help either. Another approach to generate a stream cipher is to use a cryptographically strong pseudo random number (or bit) generator like Blum-Blum-Shub. [1]

### 3.4.1 One Time Pad — OTP

The OTP is different to all other stream ciphers because there is a requirement that the key stream must not be repeated, i.e. all kind of pseudo-random number generators are not qualified for OTP. It can be proved that this method is ultimately secure, but there are some other problems associated with it: key distribution and randomness. Since the key must be at least as long as the plaintext Alice and Bob must exchange the same amount of key data before they exchange this amount of encrypted message. The key must be absolutely random, therefore generators which are based solely on software cannot be used, it is necessary to use some source of entropy outside of a computer, like atomic decay and radiation. [1]

# CHAPTER 4
# IMAGE AND VIDEO ENCRYPTION

Image and video encryption are of course closely related by the fact that raw video data consists of a sequence of still images. However, compressed video like the MPEG format is composed of different types of data which can be treated in specific ways by special encryption schemes. As a consequence, we first discuss (still) image encryption techniques which can be applied to still images or single frames (e.g. I-frames in MPEG) in a video. Subsequently, we describe procedures that exploit video specific properties. Additionally, we distinguish between techniques applied during the compression stage (compression oriented schemes) and techniques applied to an already given bitstream (bitstream oriented schemes). [1]

After describing and discussing the algorithms suggested so far in literature, we provide a final assessment for each technique. For this assessment, we evaluate the following properties [1]:

**- Time demand:** The additional time demand required for the encryption consists of two parts – the time required to perform the actual encryption (denoted as "time(E)") and the time required to identify the parts of the data which are subject to encryption (which may include bitstream parsing or any other preprocessing technique and is denoted as "time(P)"). The time demand is rated as 0, low, medium, high, and may have the additional property of being scalable if depending on the amount of data encrypted. [1]

**- Security:** The security of an entire image and video encryption approach has two aspects. First, the security of the cipher in use itself, and second, the importance and suitability of the data subject to encryption. The security is rated as low, medium, high, and may have the additional property of being scalable if depending on the amount of data encrypted. In accordance to the two aspects of security, two entirely different types of attacks against image and video encryption systems are possible.

On the one hand, the cipher in use may be the target of an attack. In this case, common cryptanalytic results about the security of specific ciphers in general apply. On the other hand, in the case of partial or selective encryption, it is possible to reconstruct the visual content without taking care of the encrypted parts. Depending on the importance of the encrypted data for visual perception,the result may range from entirely incomprehensible to just poor or reduced quality. In any case, when conducting such a "direct reconstruction", the high frequency noise originating from the encrypted portions of the data is propagated into the reconstructed frame. In order to avoid this phenomenon, "error-concealment attacks", "perceptual attacks",or "replacement attacks" have been suggested. These types of attacks either try to conceal the quality reduction caused by encryption by treating unbreakable data as lost and then trying to minimise the impact on quality as a result of loss (error-concealment attacks) or simply replace the encrypted parts of the data by either artificial data mimicking simple nonstructured content (replacement attacks) or data minimising the perceptual impact of the incorrect data (perceptual attacks). [1]

**- Bitstream compliance:** An image or video encryption scheme is said to be bitstream compliant, if the resulting bitstream is compliant to the bitstream definition of the compression system in use. Bitstream compliance is inherent to any compression oriented encryption scheme if it is based on somehow manipulating coefficient data. On the other hand, bitstream oriented schemes often do not take care about this property at all. Bitstream compliance is rated yes or no. [1]

**- Compressed domain processing:** An important property of image and video encryption schemes is whether they may be applied directly to a given bitstream or the bitstream needs to be decoded before being able to apply encryption. This property is denoted "Bitstream processing" and rated yes or no. [1]

**- Compression performance affected:** As we shall see, quite a lot of image and video encryption schemes increase the file size as compared to applying compression without encryption. This property is denoted as "affecting R/D" (rate-distortion) and is rated yes, moderately, and no. [1]

### 4.1 DCT-based Techniques

#### 4.1.1 Image Encryption

##### 4.1.1.1 Compression Oriented Schemes

**Zig-zag Permutation Algorithm.** The historically first MPEG encryption proposal is due to Tang [1] and is called " zig-zag permutation algorithm". The idea is to substitute the fixed zig-zag quantised DCT coefficient scan pattern by a random permutation list. Consequently, in the terminology introduced in the previous section this is a soft encryption approach. Additional security measures are proposed as follows [1]:

- The 8 bit DC coefficient is to be split into two 4 bit halves, out of which the MSB part replaces the original DC coefficient and the LSB part replaces the smallest AC coefficient. The reason is that the DC coefficients could be identified immediately by their size thus revealing a low-pass approximation of the image. [1]

- Before DC-splitting, the DC coefficients of eight 8 × 8-pixels blocks are concatenated, DES encrypted and written back byte-oriented. [1]

- Instead of using one permutation list a cryptographically strong random bit generator selects one out of two permutation list for each 8 × 8-pixels block. [1]

There have been several shortcomings of the zig-zag permutation algorithm identified in literature [1]:

**Security**

- Known plaintext and chosen ciphertext attacks: Permutations are known to be vulnerable against known plaintext attacks. Assuming a certain frame of the video is known (the plaintext) – by comparing original and permuted coefficients the permutation list can be retrieved. Even in case two lists have been used the correct one can be found on a per block basis by applying both and using the block with most non-zero coefficients in the upper left corner as decrypted. [1]

- Ciphertext only attack: The ciphertext only attack is the weakest attack available to an adversary. In the context of the zig-zag permutation algorithm it is based on the fact that non-zero AC coefficients tend to gather in the upper left corner of the considered image block. Once the non-zero coefficients have been identified in the block, they are shifted to the upper left corner of the block and only a relatively small number of combinations among the non-zero coefficients is required to be tested. Based on some statistical analysis which involves the frequency of non-zero occurrence. [1]

**Decrease of compression performance:** The zig-zag scan as included in the JPEG and MPEG standards orders the coefficients with respect to increasing frequency and decreasing magnitude. As a consequence, long runs of zeros occur in the high frequency areas of a block. The JPEG and MPEG Huffman tables are optimised with respect to those properties – therefore, by changing the zig-zag pattern, some compression performance is expected to be lost. [1]

The Huffman codeword list as defined in the MPEG standard is permuted using a secret key, and subsequently used in the compression and decompression process. In order to prevent the algorithm from affecting compression performance only permutations are admissible which maintain the length of the codewords. This limits the number of possible permutations significantly and therefore reduces the available keyspace (and with it the security of the system). Note that contrasting to this approach zig-zag permutation potentially also changes the number of required Huffman codewords (due to different amount and length of zero-coefficient runs) in addition to the resulting permutation of the Huffman table. [1]

Table 1 Overall assesment of the Zig-zag permutation algorithm

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| low | 0 | low | yes | no | yes |

**Frequency-band Coefficient Shuffling.** In order to limit the drop in compression efficiency as seen with zig-zag permutation, Zeng and Lei [2] propose not to permute the coefficients within a single 8 × 8 pixels block but to group the coefficients from an entire set of such blocks together and perform permutation to DCT coefficients within a frequency band (i.e. with similar frequency location). This strategy reduces the bit overhead significantly, but still a file size increase of 10 - 20%. Whereas the security problems as induced by the use of permutations remain valid in principle, the situation is improved as compared to the pure *zig-zag* permutation approach since additional key material may be employed to define which blocks are used to select coefficients from and the described ciphertext only attack is much more difficult since more blocks are involved. [1]

Table 2 Overall assesment of Frequency-band Coefficient Shuffling

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| low | low | low | yes | no | moderately |

**Scalable Coefficient Encryption.** It is proposed to encrypt the low-frequency DCT coefficients only and leave the high-frequency ones unencrypted in the JPEG environment. This approach is therefore a partial encryption technique. It is pointed out that the concentration of signal energy to a few coefficients as done by most orthogonal transforms does not necessarily imply that the same is true for intelligibility, which is often scattered among all frequency components. Whereas this general observation questions all partial encryption techniques in the DCT domain in principle, wavelet based techniques show different characteristics in this respect. [1]

Table 3 Overall assessment of Scalable Coefficient Encryption (in coefficient domain)

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium+scalable | 0 | low+scalable | yes | yes | no |

**Coefficient Sign Bit Encryption.** It is suggested to encrypt the sign bit of each DCT coefficient only (which is a partial encryption approach). The rationale behind this idea is that the sequence of sign bits already exhibits high entropy, consequently, a further increase of entropy caused by encryption (and with it a file size increase after compression) should not be expected. Experimental results involving a H.263 codec even show a small bitrate reduction when applying sign bit encryption.
The reduction of computational amount with respect to full encryption is significant since only 13 - 20% of all data need to be encrypted. Encryption of the sign bits can be implemented in many ways: one possibility is to XOR the sign bits with a key stream coming from a cryptographically secure stream cipher, another way would be to apply a block cipher to a set of sign bits where the order of the set equals the block size. [1]

**Secret Fourier Transform Domain.** An approach significantly different from those discussed so far is to conceal the transform domain into which the image data is transformed by the compression scheme. The underlying idea is that

Table 4 Overall assessment of Coefficient Sign Bit Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium | medium | low | yes | yes | no |

if the transform domain is not known to a hostile attacker, it is not possible to decode the image. Fractional Fourier domains have been used in earlier work to embed watermarks in an unknown domain. In particular, the input plane, the encryption plane, and the output plane of the proposed method are fractional Fourier domains related to each other by a fractional Fourier transform. While the security and the

size of the corresponding keyspace is discussed the complexity is not. While being an interesting approach in principle, it seems that this technique might be used only in very specific environments and the advantage over a classical full encryption is not obvious. [1]

Table 5 Overall assessment of Secret Fourier Transform Domain

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| high | high | high | no | no | yes |

**Secret Entropy Encoding.** Based on the observation that both, cipher and entropy coder, turn the original data into redundancy-free bitstreams which cannot be decoded without certain information. The information required for decoding is the key in the cryptographic case and the statistical model in the entropy coder case.
Wu and Kuo propose to use multiple Huffman coding tables (MHT) out of which a specific one is selected based on random decisions for encoding a given symbol. In order to cope with maintaining the compression ratio, it is suggested to use a different set of training images for each table. Huffman tree mutation can also be used to create a large number of Huffman tables out of 4 initially given tables. [1]

Table 6 Overall assessment of Secret Entropy Encoding

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| low | 0 | high | no | no | moderately |

#### 4.1.1.2 Bitstream Oriented Schemes

**Header Encryption.** The most straightforward to encrypt an image or video bitstream is to encrypt its header. Bitstream compliance is immediately lost and the image or video cannot be displayed any longer using a common player. However, in case an attack is conducted against this kind of encryption, most header informations can simply be guessed or extracted from the syntax of the bitstream itself. As a consequence, the security of such a scheme depends on the type of header data encrypted – if the protected header data can not be guessed or computed by analysing the bitstream, and if this data is crucial for the decoding of the visual data, this approach could be secure. In any case, header encryption is an interesting approach since it requires minimal encryption effort only. [1]

Table 7 Overall assessment of Header Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| Low+scalable | low | low+scalable | No | yes | no |

**Permutations.** Permutations are a class of cryptographic systems well suited for designing soft encryption schemes and have been as well proposed to be applied at the bitstream level, however, all these schemes are extremely vulnerable against a known plaintext attack as described in the context of the Zig-Zag Permutation Algorithm. The entities subject to permutation are different when comparing the suggestions made so far. [1]

**1 Bytes** Depending on the security equirements the permutation lists in use can be made longer or shorter. Whereas this approach is extremely fast in terms of the actual encryption process and in terms of parsing the bitstream to identify the data subject to encryption, the semantics of the MPEG stream are entirely destroyed and no bitstream compliance is obtained. [1]

**2 VLC codewords** Codewords from different 8 × 8 pixels blocks are grouped together according to their codeword index and permuted with one permutation list. The number of groups and the range of codeword indices within one group can be adjusted according to security requirements. A problem with this approach is that different 8 × 8 pixels blocks usually contain a different number of non-zero coefficients which can be resolved by controlling the "last field" of each block. Format compliance may be guaranteed by truncating codewords eventually exceeding 64 coefficient per 8 × 8 pixels blocks, but there is of course significant processing overhead as compared to the Pure Permutation Algorithm induced by identifying VLC codewords and grouping of codewords. [1]

**One-time pad VEA.** **A**nother partial encryption VEA (**V**ideo **E**ncryption **A**lgorithm) which operates on MPEG streams at the byte level. Odd-numbered and even-numbered bytes form two new byte streams, the *Odd List* and the *Even List.* These two streams are XORed, subsequently the Even List is encrypted with a strong cipher (DES was suggested at that time). The result of the XOR operation and the encrypted Even List are the resulting cipher text. As a consequence, the DES encrypted half of the byte stream serves as a one-time pad for the other half which makes the system fairly secure, because there exists low correlation between bytes and pairs of bytes in MPEG streams. This exploits the fact that both compression and encryption decorrelate the data. In order to further increase the security the following improvements are suggested [1]:

- The fixed odd-even pattern is replaced by two randomly generated byte lists (where this is controlled by a 128 – 256 bit key). [1]
- Each set of 32 bytes is permuted, 8 different permutation lists are employed which are used in fixed order. [1]
- The generation of the two byte lists is changed for every frame (of a video). [1]

This algorithm exhibits very high security but this is achieved at a relatively high computational cost (as compared to full DES encryption, only 47% of the overall computations are saved). Additionally, operating at the byte level the semantic of the

MPEG stream is entirely destroyed, bitstream markers may be emulated, and there exists no bitstream compliance after encryption of course. [1]

The Even List which would be subject to encryption in the original scheme is instead again split into an Odd2 List and an Even2 List which are XORed. Now, the Even2 List (one quarter of the original data) can be encrypted and the procedure terminates or it is again split into an Odd3 List and an Even3 List. This strategy of course reduces the encryption effort significantly, but the security is also weaker since the one-time pad is no longer one-time in a strict sense . Bitstream compliance is equally destroyed as in the original scheme. [1]

Table 8 Overall assessment of One-time pad VEA

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| high | low | high | no | yes | no |

**Byte-Encryption.** It is proposed to randomly destroy bytes in an MPEG stream for free distribution, while the original bytes at the corresponding positions are transferred in encrypted form to legitimate users. This is actually equivalent to encrypting bytes at random positions. Encrypting 1 % of the data is sufficient to make a video undecodable or at least unwatchable. However, the cryptanalysis given is entirely insufficient. Consider the worst case where only MPEG header data is encrypted by chance using this approach. It is well known that header data may be reconstructed easily provided the encoder in use is known. Additionally, no attack scenario is considered but only the case of playing the protected video in a standard decoder is covered. In order to guarantee a certain level of security, a higher amount of bytes need to be encrypted and care needs to be taken about which bytes are encrypted. [1]

**Selective Encryption**. This algorithm encrypts X bits, the next Y bits are left in plain-text, the next Z bits encrypted again, and so on. In addition to the abovementioned security problems, both approaches partially destroy the MPEG

bitstream syntax (which is the main security approach of these schemes) and potentially emulate important MPEG markers causing a decoder to crash (which is again desired). [1]

Table 9 Overall assessment of Byte Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium+scalable | low | low+scalable | no | yes | no |

**VLC Codeword Encryption.** Whereas Byte Encryption does not take the syntax of the video into account, VLC codeword encryption does. Contrasting to the permutation of such codewords, strong encryption should be applied in this case. In case bitstream compliance after encryption is not the aim, Byte Encryption applied to a significant fraction of the bitstream is a better choice since VLC codewords need not be identified and therefore bitstream parsing may be avoided to a large extent. In case bitstream compliance after encryption is the aim, when encrypting VLC codewords we face the problem the encryption of a concatenation of VLC codewords leads not necessarily to a concatenation of valid codewords. For example, given the codewords 0, 10, 110, 111, and a possible concatenation 010, a possible encryption of 010 may lead to 001 which is no valid codeword concatenation and would therefore destroy bitstream compliance. al. The technique for solution to this problem works as follows for a VLC table with $N = 2^k$ entries. Before encryption, a fixed length bit index I is assigned to each codeword in the VLC table. After a concatenation of VLC codewords is obtained which should be encrypted, a bit string S is constructed by concatenating the indices I of the corresponding codewords. S is encrypted with a secure cipher which results in S'. S' is than mapped back to codewords using the same index-to-codewords map used before for constructing S. The result will be a different concatenation of valid VLC codewords, which are inserted back into the bitstream at the positions of the original codewords. Non

power-of-two VLC tables can be treated by decomposing them into several power-of-two tables. While this scheme guarantees a standard compliant bitstream it does not preserve the size of the bitstream. In general, the original and "encrypted" concatenation of codewords will not have the same size (although the number of codewords is equal). [1]

Table 10 Overall assessment of VLC codeword Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium+scalable | high | medium+scalable | yes | yes | moderately |

### 4.1.2 Video Encryption

Video encryption based on DCT methods is focused on standardised formats like MPEG-1,2,4 or H.26X, therefore all these techniques try to take advantage of the corresponding data formats and bitstreams. Whereas all the techniques discussed subsequently could as well be applied during the compression stage, they are mostly discussed in the context of directly manipulating the bitstream data (after compression has taken place). [1]

#### 4.1.2.1 Bitstream Oriented Schemes

Most schemes for video encryption combine various ideas and are neither purely frame-based nor purely motion-based. Therefore, we will first discuss the main ideas and their properties, subsequently we will describe some complete proposals as given in literature. [1]

**Header Encryption.** As the second-lowest security level in their SECMPEG scheme, proposed to encrypt all (MPEG) header data of the MPEG sequence layer, group of picture layer, picture layer, and slice layer. The data suggested to be selectively encrypted turns out to be hardly suited for that purpose (except for the quantizer_scale_code field in the slice header). It is suggested to encrypt the macroblock_type field in the macroblock header since this data covers only about 3.5% of the entire bitstream and its encryption poses severe challenges to a decoder since this field specifies how the following bits are to be parsed and it is a VLC field which may cause the decoder to get out of sync with the bitstream in case it is not correct. Encryption of the *Dquant* parameter (difference of quantisation step size QP between current and previous macroblock), which is a very simple approach since many macroblocks simply use the default settings which makes this parameter easy to attack. [1]

**Encryption of I-Frames.** Since P and B-frames are reconstructed based on redictions obtained from I-frames, the main assumption is that if these are encrypted, P and B-frames are expected to be protected as well. There are several problems associated with this approach. First, the percentage of the bitstream which is comprised of encoded I-frames is about 25-50% which means that this approach does not reduce the computational complexity to a satisfying extent. Second, the motion in the video remains visible, especially when replacing the encrypted I-frames by uniform frames. A strategy to cope with this would be to increase the number of I-frames which is on the one hand good for security, on the other hand degrades compression performance. For example, increasing the share of I-frames from 1/3 to 1/6 in the “Miss America” and “Flowers” sequences raises the video file size by 50%. Third, and even more severe, the I-blocks contained in P and B-frames resulting from poor prediction results even deliver texture information of I-frames when collected over several frames. This especially affects high motion sequences since in this case P and B-frames contain many I-blocks. [1]

While the security as compared to pure I-frame encryption is increased, the amount of data to be encrypted increases as well (up to 40 - 80%) and the problem of still visible motion content remains unresolved. Additionally, the bitstream parsing effort to identify all I-blocks in P and B-frames is significant. In order to reduce to computational overhead. [1]

Table 11 Overall assessment of I-frame Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium | medium | low | yes | yes | no |

**Encryption of Motion Vectors.** Motion vectors comprise about 10% of the entire data of an MPEG video , therefore, restricting the encryption to motion vectors might be an interesting idea. However, from a security viewpoint encryption motion vectors alone can never be sufficient since all texture information from I-frames remains in plain text. Consequently, a video with very low temporal rate would be in plaintext in any case. I-frame material needs to be secured additionally to provide reasonable security. Similar to the DCT coefficient case, motion vector sign bits or VLC codewords can be protected. [1]

Table 12 Overall assessment of Motion Vector Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| low | medium | low | yes | yes | no |

**SECMPEG.** SECMPEG defines a new bitstream including a header structure which makes it incompatible with respect to common MPEG players. Besides its data integrity and authentication functionality (which we do not discuss here), five levels of security are defined [1]:

1 No encryption. [1]

2 Header data from the sequence layer down to the slice layer is encrypted. [1]

3 Encrypt same data as in level 2 and the low frequency DCT coefficients of all blocks in I-frames. [1]
4 Encrypt all I-blocks (also those in P and B-frames). [1]
5 Encrypt the entire video. [1]

As can be seen, three different basis techniques are combined into SECMPEG and all their properties and restrictions apply correspondingly: Header encryption, I-frame encryption, and Scalable Coefficient encryption. [1]

Scheme with scalable complexity and security using three levels [1]:

1 Encrypt all data associated with every n-th I-macroblock. [1]
2 Encrypt all data associated with every n-th I-macroblock and all header data of every n-th P and B-encoded macroblocks. [1]
3 Encrypt all data associated with every n-th I-macroblock and all header data of P and B-encoded macroblocks. [1]

Again, Header encryption and I-frame encryption is combined. As a third example for an explicitly scalable scheme we describe the security levels of the combined watermarking encryption scheme [1]:

1 Encrypt the DC coefficient of the luminance component of all I frames. [1]
2 Encrypt the luminance and chrominance DC coefficients of all I frames. [1]
3 Encrypt all luminance and chrominance coefficients of all I frames. [1]
4 Encrypt the DC coefficient of the luminance component of all I macroblocks. [1]
5 Encrypt the luminance and chrominance DC coefficients of all I macroblocks. [1]
6 Encrypt all luminance and chrominance coefficients of all I macroblocks. [1]
7 Encrypt the data of all frames (but no header data). [1]

Contrasting to the other two suggestions no header data is encrypted which enables the scheme in principle to deliver compliant bitstreams. Additionally, an explicit

distinction between luminance and colour component is made, where securing the luminance component is of course more important from a perceptual viewpoint. [1]

**MVEA and RVEA.** This technique is denoted MVEA (if sign bits are randomly changed by a secret key – which is vulnerable to a plaintext attack) or RVEA (if sign bits are encrypted by DES or IDEA). Both, DC coefficients and motion vectors are differentially encoded which causes significant impact when the corresponding sign bits are changed. The give a fixed scan order through the data of a macroblock, at most 64 sign bits are encrypted per macroblock. For I-macroblocks, first the luminance and chrominance DC coefficient sign bits are encrypted, subsequently the lowest frequency AC coefficient sign bits and so on. For P and B-macroblocks, the first sign bits to be encrypted are the sign bits of motion vector data, then the scan proceeds as in the case of I-macroblocks. About 10% of the entire video data consists of sign bit data which makes the approach interesting from the less computations viewpoint. [1]

MVEA and RVEA combine Coefficient Sign bit encryption, Scalable Coefficient encryption, and Motion Vector encryption. [1]

**Techniques in MPEG-4 IPMP.** Three different configurations with increasing security [1]:

1 Encrypt the FLC coded DCT sign bit, the parameter DQUANT (two bits determining the difference between the quantisation parameter used for the previous macroblock and the current one), and the I-macroblock DC value. [1]
2 Encrypt the VLC motion vector field. [1]
3 Encrypt the data of both previous suggestions. [1]

This proposal combines several techniques as well: I-frame encryption, Coefficient Sign Bit encryption, Header encryption, Motion vector encryption, and Scalable

Coefficient encryption. Whereas the first two options are said to be useful for entertainment purposes only, the third configuration provides satisfying results. [1]

## 4.2. Wavelet-based Techniques

Wavelet-based techniques devoted to video encryption have not been discussed in literature so far, although all proposals made for image encryption may be applied to each frame of a video independently of course. The lack of a wavelet-based video coding standard explains this situation. [1]

Note that most wavelet-based compression schemes use arithmetic coding as their entropy coding stage which does not provide a one-to-one correspondence among symbols and codewords like Huffman coding as used in DCT-based systems does. Therefore, techniques manipulating single coefficients cannot be employed in the transform domain. [1]

### 4.2.1 Compression Oriented Schemes

#### 4.2.1.1 Coefficient Selective Bit Encryption

In this technic , these are compared refinement, significance, and sign bits with respect to their entropy and compressibility. Based on this analysis, it is suggested to encrypt bits that are not highly compressible due to their high entropy and low intercorrelation. The corresponding selection limits the influence of the encryption process to rate-distortion performance: sign bits and refinement bits. Of course, refinement bit encryption can be used only as an additional security technique since it does not provide enough confidentiality as a standalone approach. Similar doubts with respect to security of sign bit encryption are valid as in the DCT case. [1]

Table 13 Overall assessment of Coefficient Selective Bit Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium | medium | low | yes | no | no |

#### 4.2.1.2 Coefficient Permutation

One obvious advantage as compared to the DCT scenario is that the distribution of wavelet coefficients is image dependent and therefore the vulnerability against ciphertext only attacks does not occur. Also, it is claimed that contrasting to the DCT case the observed drop in compression performance is about 2% only. A system based on randomly permuting wavelet-subbands incorporated in the JPEG 2000 or the SPIHT coder generally delivers much worse results in terms of compression performance . The comparison of JPEG 2000 and SPIHT in this context provides interesting insights with respect to the correctness of the zerotree hypothesis. [1]

**Encryption Using Random Permutation of Wavelet-Subbands.** The basic approach is to permute the wavelet coefficients of different wavelet subbands with dedicated permutation keys. A permutation key is defined as a vector of length n , and n wavelet coefficients can be encrypted using this key. In case permutation keys have to be transmitted along with the compressed image data the used keys have to be protected and therefore be encrypted with a standard encryption scheme like AES. For example, the key data itself can be inserted conveniently into the JPEG 2000 bitstream taking advantage of the so-called termination markers. Encryption based on random permutation lists has been shown to be vulnerable to known plaintext attacks. The use of more different keys increases the overall security of the system. This raises the question how many keys should be used to encrypt the data and what key lengths should be used in order to achieve a satisfying level of security. Additionally it needs to be considered that any key information needs to be stored in

the final bitstream and decreases the compression performance. There are two key management scenarios [1]:

**1** full key scenario: A wavelet subband with pixels is permuted with a “full” key of length n. [1]
**2** key for row scenario: A wavelet subband consisting of n pixels. [1]

(n = r * c , r = rows , c = columns )

is permuted with keys on a per row basis. [1]
Therefore, a number x, 1<= x = r of keys with length equal to one row is used, and the keys are exchanged in a round robin fashion. [1]

#### 4.2.1.3 Coefficient Block Permutation and Rotation

In this technic , It is suggested to divide each subband into a number of blocks of the same size. The size of these blocks can vary from subband to subband. Within each subband, blocks of coefficients are permuted according to a permutation key which should also differ from one subband to another. Since the local statistics of the wavelet coefficients are preserved, the expected impact on coding performance is smaller as compared to the pure coefficient permutation case (the degradation is the smaller, the larger the block size is selected). On the other hand, using large blocks threatens security due to two reasons [1]:

- The permutation key is small. [1]
- A possible attacker might try exploit edge continuity in the high pass subbands and a smoothness constraint in the low pass subband to invert the permutation. [1]

To further increase security without affecting rate-distortion performance it is suggested to additionally use one of eight isometries of each block (which

corresponds to rotating and flipping the block). This makes it harder to invert the permutations based on image properties, however, a higher number of rotated versions would be necessary to provide sufficient security. [1]

Table 14 Overall assessment of Coefficient Block Permutation and Rotation

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| Low | 0 | low | yes | no | moderately |

#### 4.2.1.4 Secret Transform Domains

It is also possible to use secret wavelet transforms for an encryption application. The idea of using secret wavelet domains has also been used to increase the security of watermarking schemes. In this context, filter parameterisations and wavelet packet subband structures have been used to conceal the embedding domain. Contrasting to the Fourier case, all proposals concealing the wavelet transform domain for encryption are integrated into a compression pipeline. There are two ways of generating a large variety of wavelet filters out of which a secret one may be chosen for actual decomposition. All these techniques have a significant advantage: the amount of data subject to encryption is minimal since only information about the transform in use needs to be encrypted. Therefore, these methods may be seen as a special variant of header encryption. [1]

### 4.2.2 Bitstream Oriented Schemes

#### 4.2.2.1 SPIHT Encryption

A partial encryption scheme for SPIHT which can be applied to any zerotree-based wavelet coding scheme. The basic observation is as follows: the compression algorithm produces many different types of bits – sign bits, refinement bits, and significance bits of pixels and sets. The decompression algorithm has to interpret each bit under the correct context. Incorrect significance bits may lead to an incorrect interpretation of subsequent bits, this is not the case when sign bits or refinement bits are decoded incorrectly. As a consequence it is suggested to encrypt the significance information of sets and pixels of the two lowest resolution pyramid levels. The reason for not encrypting all significance information is as follows: the significance information of the low resolution levels is used to initialise the different lists used by the algorithm. If the states of these lists are incorrect right from the start of the decoding, it is hardly possible for the algorithm to recover from the error. The information left unencrypted is of low value for an attacker since without. The significance information the type of bits can not be distinguished from another. The amount of data encrypted is very small in this proposal – less that 7% in case of 256 × 256 pixels images and less than 2% in case of 512 × 512 pixels images. Although the methods seems to be very secure, no experimental attacks have been mounted against the scheme proving its robustness. [1]

#### 4.2.2.2 JPEG 2000 Encryption

For selectively encrypting the JPEG 2000 bitstream we have two general options. First, we do not care about the structure of the bitstream and simply encrypt a part, e.g. the first 10% of the bitstream. In this case, the main header and a couple of packets including packet header and packet data are encrypted. Since basic information necessary for reconstruction usually located in the main header is not available at the decoder, encrypted data of this type can not be reconstructed using a JPEG 2000 decoder. Although this seems to be desirable at first sight, an attacker

could reconstruct the missing header data using the unencrypted parts, and, additionally, no control over the quality of the remaining unencrypted data is possible. Therefore, the second option is to design a JPEG 2000 bitstream format compliant encryption scheme which does not encrypt main and packet header but only packet data. [1]

Table 15 Overall assessment of SPIHT Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| low | low | high | yes | yes | no |

### 4.3. Further Techniques

#### 4.3.1 Raw Image Data

##### 4.3.1.1 Permutations

Applying permutations to the raw image data is the simplest and fastest way to apply encryption technology to visual data. Hybrid Pay-TY systems (i.e. analog signal transmission but encryption and decryption is done in the digital domain) have extensively made use of this technology. The Nagravision/ Syster system for example applies line permutations within blocks of 32 lines, VideoCrypt uses specific permutations within each line of the video frame by cutting each line at a secret position and interchanging the two resulting sub-lines. Although the key material is changed frequently, both systems are vulnerable to a ciphertext only attack by using smoothness constraints (guessing a correct permutation results in a smoother image than guessing incorrectly) and known facts about the generation of the permutation keys. It is proposed to use line permutations in the context of a multiresolution image decomposition which facilitates a good control over the amount of degradation. However, this scheme is not more secure than “pure” permutation in the image domain. [1]

Table 16 Overall assessment of Permutations applied to raw image data

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| Low | 0 | low | yes | no | yes |

#### 4.3.1.2 Chaos-based Systems

Chaos-based encryption of visual data uses the principle of applying chaotic maps with strong mixing properties to the raw image data. The basic idea is that (continuous) chaotic maps exhibit similar properties as (discrete) cryptographic systems. Usually, these systems are hybrids between permutation and substitution ciphers with specific properties. Therefore, they are very fast. These algorithms have also been used to conceal logo-type images for watermarking generation [3]. The most famous example of a chaotic map is the "Baker Map" B (defined on $[0,1]^2$ as follows: B(x,y) = (2x, y/2) for 0 <= x <1/2 and B(x, y) = (2x-1, y/2 + 1/2) for ½ <= x <= 1 The left vertical half of the domain [0, 1/2) × [0,1) is stretched horizontally and contracted vertically to be mapped to the domain [0,1) × [0,1/2). In the same way the right half [1/2,1) × [0,1) is mapped to [0,1) × [1/2, 1). [1]

#### 4.3.1.3 Bitplane Encryption

Assumed an 512 × 512 pixels image to be given in 8bit/pixel (bpp) precision and considered the 8bpp data in the form of 8 bitplanes, each bitplane associated with a position in the binary representation of the pixels. The encryption approach is to e.g. AES encrypt a subset of the bitplanes only, starting with the bitplane containing the most significant bit (MSB) of the pixels. Each possible subset of bitplanes may be chosen for encryption, however, the minimal percentage of data to be encrypted is 12.5 % (when encrypting the MSB bitplane only), increasing in steps of 12.5 % for each additional bitplane encrypted. It is used an AES implementation with blocksize

128 bit and a 128 bit key. The 128 bit block is filled with a quarter of a bitplane line (512/4 = 128 bits). The encrypted bitplanes are transmitted together with the remaining bitplanes in plain text. [1]

Whereas in the case of encrypting the MSB only structural information is still visible, encrypting two bitplanes leaves no useful information in the reconstruction, at least when directly reconstructing the image data. Note the pattern reminiscent of a bar code in the upper right quarter of the image. This phenomenon is due to the fact that AES encryption is used with identical key for all blocks in the image. Consequently, if there are identical plain text quarter-lines directly situated above each other which also adhere to the AES block-border (i.e. starting at pixel positions 0, 128, 256, or 384), these data produce identical ciphertext blocks. Identical blocks of ciphertext are again arranged as identical quarter-lines thereby generating the barcode effect. [1]

### 4.3.2 Quadtrees

Quadtree compression partitions the visual data into a structural part (the quadtree structure) and colour information (the leave values). It is suggested to encrypt the quadtree structure only and to keep the leave values in plaintext. As it is the case with wavelet packets a brute-force attack is not feasible due to the exceedingly high number of possible quadtrees. The only way to attack such a scheme is to try to deduce the quadtree structure from the non-encrypted leave values. There are two variants how the leave values may be organized in the compressed file [1] :

**1** Leaf ordering I: is a depth first scan which starts with the NW quadrant, counterclockwise. [1]

**2** Leaf ordering II: is a line oriented scan within each level of the quadtree, starting with the smallest leaves. [1]

It turns out there exists an attack against leaf ordering I whereas ordering II seems to be secure. The problem with leaf ordering I is that leaves which are neighbours in the quadtree are also neighbours in the bitstream. Based on this observation one notices that runs (i.e. a sequence of identical leave values) provide information about the local quadtree structure, since four identical adjacent leave values can never be situated at the same quadtree level (in this case no quadtree partitioning would have taken place). These facts may be exploited to significantly reduce the admissible quadtrees during an attack. In case of lossy quadtree compression, the quadtree structure covers about 15 – 25 % of the entire data. Therefore, a significant amount of data needs to be encrypted. If the data is not given in quadtree compressed form (which is highly probable), the complexity of compression needs to be considered as well. [1]

Table 17 Overall assessment of Quadtrees Encryption

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| high | high | high | - | yes | no |

### 4.3.3 Fractal-based system

An access control system for fractal encoded visual data which may be operated ranging from a full encryption mode to a variant where the images are only slightly distorted. The main idea is to partially encrypt the binary representation of the luminance scale parameter. Whereas reconstruction or filtering of the corrupted image data is extremely difficult due to the highly non-linear distortions induced by fractal interpolation, the amount of parsing to extract the data subjected to encryption is significant. Further, due to the robustness of the decoding process, this technique is hardly useful to provide real confidentiality. Transparent encryption may be a better application field. [1]

Table 18 Overall assessment of Encrypting Fractal Encoded Data

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| Low | High | low | yes | yes | no |

### 4.3.4 Vector quantisation based system

The embedding stage is performed using a vector quantisation codebook which is derived from both the cover image and the image to be protected. The security relies on the fact that two random vectors are encrypted in secure manner, concatenated repeatedly to generate a keystream which is then XORed with the list of indices from vector quantisation. This simple scheme is of course a threat for the security of the system. Secure encryption of the entire list of indices would be the other choice, however, computational amount for this operation is significant. [1]

Table 19 Overall assessment of Virtula Image Cryptosystem

| time(E) | time(P) | Security | BS comp. | BS proc. | affectsR/D |
|---|---|---|---|---|---|
| medium | high | high | yes | no | no |

### 4.3.5 Base-switching based system

It is used a lossless image codec for encryption scheme. After partitioning the image into 3 × 3 pixels blocks these blocks are transformed involving the “base value” which is the difference between maximum and minimum of the pixel values in the blocks. It is assumed that this value stays in the range (1) and the security of the entire system resides in encrypting this base value. Without having access to the base value the remaining data for each block cannot be decoded and the field containing the base value for the next block cannot be identified. Although it is claimed that there would be 128! mappings from the plain base value to the encrypted one, a brute-force attack only requires 128 guesses per block to test all possible base values. Taking into account several plausibility criteria, an attack could probably be mounted with much less effort. Regarding this observation, taken together with the fact that the compression scheme is not very effective and hardly ever used, the system does not seem to be very practical. Additionally this method is susceptible to known plaintext attacks because then the attacker knows the encrypted und the unencrypted version of the base value, so that the following bits can be interpreted without a chance of error. [1]

**Experiments:**

**Line Cut and Rotate Method**

Several systems which exists to encrypt videos, credit on methods of directly deforming the image in the spatial domain. These encryption techniques are not enough for trasfering digital video signals because generally they will considerably chance the statistical attribution of the original video signal.

Line scrambling is one of the spatial domain approaches, which scrambles the lines of the image. To test spatial domain scrambling influence, line scrambling algorithm is administered on MATLAB, just to test spatial domain scrambling effects. A permutation of the progrssion from 1 to image height is used to rearrange the image lines.

**Scrambling Method**: Each scan line is cut into pieces and re-collected in a varied serie.

For example, if a line like 0123456789

passes the encoder, the output might look like 4567890123

**Advantages**: It supplies a proper video signal, gives an amount of obscurity in great quality, as well as good decode quality and steadiness.

**Disadvantages**: It may have a complex timing control and requires specialized scrambling equipment

The cut and rotate video encryption method is presumably the best way of achieving confidential and good quality video encryption, an example of a good praxis of this system is in the Viewlock II.

**Results:**

As for the line and rotate based algorithm that I coded(see appendix), I succeeded obtaining the following encrypted Lena picture(.raw format) results on Matlab where fig. 1 is our original frame and fig. 2, fig. 3 are our encrypted results.

Figure 1 Original Frame

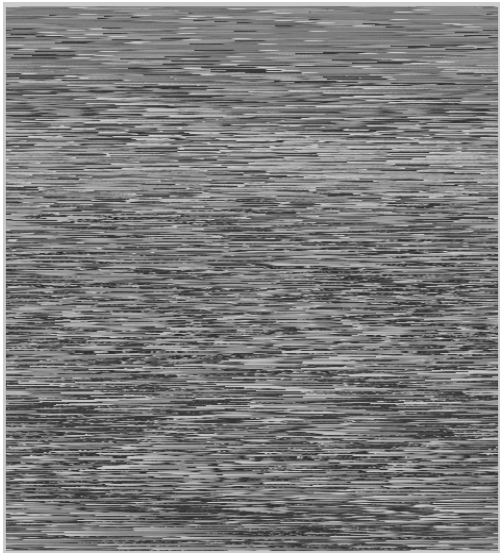

Figure 2 Row Encrypted Frame

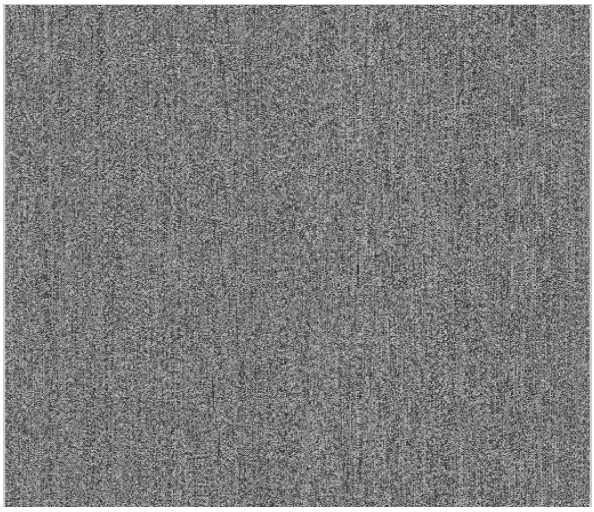

Figure 3 Row and Column Encrypted Frame

No Lena shape is recognizable in the encrypted image. Visually it is very secure.

Likewise, it supplies selective complexity/security range with the calibration of its parameters in the following way:

- Increasing or decreasing number of cut points.
- Using cascade in vertical and horizontal directions.

# CHAPTER 5
# CONCLUSIONS

Image and video encryption is neither in its starting nor it may be noticed aa a old technique.Moreover, many suggestions exist with respect to possible technologies and we have learned a lot since the first prospositions have been emitted in the mid-nineties: There are better ways to obtain privacy for visual data in comparison to AES encrypt the resembling bitstream in case safty is not the only criterion for accomplished distribution. The “There is nor free lunch” theorem also matches to encryption of visual data. However encryption perseverance may be importantly decrease by actuating selective techniques, bitstream compliance may not be succeeded in a simple and time-efficient sort simply. Visual data in scalable formats may be encrypted much more effectively and with much preferable functionalities. Especially, selective encryption can be deployed effectively only when applied to scalable or embedded bitstreams. Highly context-based wavelet compression schemes seem to be better fitted for selective encryption in comparison to classical DCT based compression schemes like MPEG or JPEG. Moreover, international standards exposing express of the art like MPEG IPMP or JPSEC are quite recent or not even finalized. Publicity products like DVD or many Pay-TV systems generally base their security insurance on the “security by obscurity” principle and have effectively been successfully attacked.

As a result, it will be seen exciting improvings in this field in the near future. Exclusively, with respect to real-world utilization it is going to be be interesting to see if the entertainment and telecommunication industries will make broad use of the new standards MPEG IPMP and JPSEC. From the more research oriented observing, the conjagation and interoperability of varied multimedia safty techniques (e.g. encryption and robust watermarking or encryption and fragile watermarking) poses a lot of open questions. For this reason, this area is prospected to remain exciting for some time.

**References :**

[1]. Andreas Uhl and Andreas Pommer. Image and Video Encryption From Digital Rights Management to Secured Personal Communication Proceedings of the Springer Science + Business Media Inc. , Boston, USA, 2005.

[2]. L. Tang. Methods for encrypting and decrypting MPEG video data efficiently. In *Proceedings of the ACM Multimedia 1996,* pages 219–229, Boston, USA, November 1996.

[3]. Wenjun Zeng and Shawmin Lei. Efficient frequency domain video scrambling for content access control. In *Proceedings of the seventh ACM International Multimedia Conference 1999,* pages 285–293, Orlando, FL, USA, November 1999.

[4]. George Voyatzis and Ioannis Pitas. Chaotic mixing of digital images and applications to watermarking. In European Conference on Multimedia Applications, Services and Techniques, ECMAST '96, volume 2, pages 687–695, Louvain-la-Neuve, Belgium, May 1996.

[5]. Internet: Video Encryption
http://ezinearticles.com/?Video-Encryption&id=11025

[6]. Internet: Wikipedia Videocrypt
http://en.wikipedia.org/wiki/VideoCrypt

[7]. Internet: Viewlock II Video Encryption System
http://www.ovation.co.uk/Video-Encryption.html

[8] Internet: Some technical details about Videocrypt
http://www.cl.cam.ac.uk/~mgk25/tv-crypt/details.txt

**Appendix :**

Line Cut and Rotate Algorithm:

```
clear;
clc;

w=512;
h=512;

raw_im=('lena.raw');
file=fopen(raw_im,'r');
I=fread(file);
status=fclose(file);
imlena=reshape(I,w,h);
lena = imrotate(uint8(imlena),-90,'bilinear');
imshow(lena)

msnrmlz1=0;
msnrmlz2=0;


for i=1:1000

%row encryption-------------------------------------------------------
A = ceil(512.*rand(512,1));

    for i=1:512
        a=A(i,1);
        for j=1:a
            enc1(i,j+(512-a))=lena(i,j);     %row encryption
        end
    end

    for i=1:512
        a=A(i,1);
        for j=1:512-a
            enc1(i,j)=lena(i,j+a);           %row encryption
        end
    end

%---------------------------------------------------------------------

lenadbl=double(lena);
enc1dbl=double(enc1);

m=lenadbl-enc1dbl;

m2=m.^2;
mstoplam1=0;
for i=1:512
    for j=1:512
```

```
        mstoplam1=mstoplam1 + m2(i,j);
    end
end

mse1=mstoplam1/(512*512);
msenormalize1=mse1/(255*255);
msnrmlz1=msnrmlz1+msenormalize1;




%column encryption*********************************************
B = ceil(512.*rand(512,1));

    for i=1:512
        b=B(i,1);
        for j=1:b
            enc2(j+(512-b),i)=enc1(j,i);    %column encryption
        end
    end

    for i=1:512
        b=B(i,1);
        for j=1:512-b
            enc2(j,i)=enc1(j+b,i);          %column encryption
        end
    end

%************************************************************

enc2dbl=double(enc2);
mm=lenadbl-enc2dbl;

mm2=mm.^2;
mstoplam2=0;
for i=1:512
    for j=1:512
        mstoplam2=mstoplam2 + mm2(i,j);
    end
end

mse2=mstoplam2/(512*512);
msenormalize2=mse2/(255*255);
msnrmlz2=msnrmlz2+msenormalize2;

end
figure; imshow(enc1)
figure; imshow(enc2)
msenorm1=msnrmlz1/1000;
msenorm2=msnrmlz2/1000;

%decryption_____(firstly columns)_____________________________
%    for i=1:512
%        b=B(i,1);
```

```
%         for j=1:b
%             decr1(j,i)=enc2(j+(512-b),i);   %column encryption
%         end
%     end

%     for i=1:512
%         b=B(i,1);
%         for j=1:512-b
%             decr1(j+b,i)=enc2(j,i);         %column encryption
%         end
%     end
%figure; imshow(decr1)

%decr1dbl=double(decr1);
%mmm=lenadbl-decr1dbl;

%mmm2=mmm.^2;
%mstoplam3=0;
%for i=1:512
%     for j=1:512
%         mstoplam3=mstoplam3 + mmm2(i,j);
%     end
%end

%mse3=mstoplam3/(512*512);
%mesnormalize3=mse3/(255*255);

%decryption______(secondly rows)_____________________________

%     for i=1:512
%         a=A(i,1);
%         for j=1:a
%             decr2(i,j)=enc1(i,j+(512-a));     %row encryption
%         end
%     end

 %    for i=1:512
 %        a=A(i,1);
 %        for j=1:512-a
 %            decr2(i,j+a)=enc1(i,j);           %row encryption
 %        end
 %    end

%figure; imshow(decr2)
```