

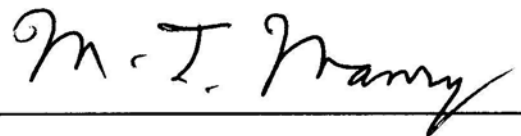
A VIDEO AUTHENTICATION SCHEME USING DIGITAL SIGNATURE
STANDARD AND SECURE HASH ALGORITHM
FOR H.264/AVC MAIN PROFILE

The members of the Committee approve the master's
thesis of Nandakishore Ramaswamy

Kamisetty R. Rao
Supervising Professor



Michael T. Manry



Soontorn Oraintara



Copyright © by Nandakishore Ramaswamy 2004

All Rights Reserved

A VIDEO AUTHENTICATION SCHEME USING DIGITAL SIGNATURE
STANDARD AND SECURE HASH ALGORITHM
FOR H.264/AVC MAIN PROFILE

by

NANDAKISHORE RAMASWAMY

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

August 2004

ACKNOWLEDGEMENTS

I am grateful to Dr. K.R. Rao for his constant encouragement and support which made this thesis possible. He has been both a mentor and a friend during my time in graduate school. The author would also like to express his thanks to Dr. Manry and Dr. Oraintara for being on his committee and wishes to acknowledge the members of the Multimedia Processing Lab for their inputs which went a long way in completing this thesis. A special note of thanks to Gautam, the author's room-mate for his moral support.

Finally, I am indebted to my family for their love, encouragement, moral and financial support. This thesis is dedicated to them.

July 09, 2004

ABSTRACT

A VIDEO AUTHENTICATION SCHEME USING DIGITAL SIGNATURE STANDARD AND SECURE HASH ALGORITHM FOR H.264/AVC MAIN PROFILE

Publication No. _____

Nandakishore Ramaswamy, MS

The University of Texas at Arlington, 2004

Supervising Professor: Kamisetty R. Rao

Multimedia authentication techniques are required to prove the validity of legal multimedia content and establish the identity of the content creator. Digital signature is one way of authenticating multimedia content. Typically digital signature uses cryptographic techniques to ensure the integrity of the multimedia bitstream. In this thesis we propose a digital signature scheme to authenticate the video bitstream generated by H.264/AVC Main Profile by using features extracted from the video and identify the video originator. This signature is unique to every coded video sequence and is generated using the Digital Signature Standard (DSS). Also, we developed a technique to identify the tampered locations and point out sender forgery in case of an authentication failure.

Results demonstrate that our proposed algorithm is robust to temporal and spatial manipulations.

LIST OF ILLUSTRATIONS

Figure	Page
1.1 Common video authentication system	3
1.2 Common video verification system.....	3
2.1 Encoder of the trustworthy digital camera	6
2.2 Decoder of the trustworthy digital camera	6
2.3 Signature generation process using histogram.....	7
2.4 Signature verification process using histogram.....	8
2.5 Digital signature generation using block DC value	10
2.6 Signature verification using block DC value	11
2.7 Flow diagram of feature extraction proposed by Lou et al [17].....	12
2.8 Semi-fragile watermark generation.....	13
2.9 Semi-fragile watermark verification	13
2.10 Signature generation of JPEG-2000 code-block	14
3.1 Transformed luma coefficients used in the authentication process for Intra 4x4 and Inter blocks of a macroblock.....	16
3.2 Intra 16x16 luma macroblock after integer 2D-DCT	17
3.3 AC coefficients ((0,1) and (1,0) of every 4x4 block in the integer 2D-DCT domain) of the Intra 16x16 macroblock included in the authentication process.....	18
4.1 Frames 0 and 26 encoded using QP 29	37
4.2 Frames 0 and 26 encoded using QP 35	37

4.3 Reordering of frames 1 and 2	39
4.4 Original foreman frame	42
4.5 Foreman frame after DC attack.....	42
4.6 Number of feature coefficients collected for every frame	44
4.7 Original Foreman frame	45
4.8 Cropped frame in Foreman sequence.....	45
4.9 Rotated frame in Foreman sequence	45
4.10 Inverted frame in Foreman sequence	46
4.11 Tampered frame in Foreman sequence	46

LIST OF TABLES

Table	Page
4.1 DC Attack.....	40
4.2 Comparison of proposed algorithm with previous work [17]	43

LIST OF FLOWCHARTS

Flowchart	Page
3.1 Encoder for calculating digital signature.....	20
3.2 Decoder for verifying the digital signature of the received video.....	23
3.3 Hash computation process for every picture of the video sequence	26
3.4 Process of identifying tampered locations and sender forgery.....	29

LIST OF ACRONYMS

1D	One dimensional
2D	Two dimensional
DCT	Discrete Cosine Transform
IDCT	Inverse Discrete Cosine Transform
HT	Hadamard Transform
IHT	Inverse Hadamard Transform
SHA	Secure Hash Algorithm
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
RSA	Rivest, Shamir and Adleman
QCIF	Quarter Common Intermediate Format
CIF	Common Intermediate Format
SIF	Source Image Format
MPEG	Moving Picture Experts Group
VCEG	Video Coding Experts Group
AVC	Advanced Video Coding
SEI	Supplemental Enhancement Information
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission

ITU-T	International Telecommunication Union Telecommunications standardization sector
QP	Quantization Parameter
JPEG	Joint Photographic Experts Group

TABLE OF CONTENTS

ACKNOWLEDGEMENTS.....	iv
ABSTRACT	v
LIST OF ILLUSTRATIONS.....	x
LIST OF TABLES.....	xii
LIST OF FLOWCHARTS.....	xiii
LIST OF ACRONYMS	xiv
Chapter	
1. INTRODUCTION.....	1
1.1 Thesis Outline	2
2. MULTIMEDIA AUTHENTICATION.....	5
2.1 Introduction	5
2.2 Hard Authentication schemes.....	5
2.3 Related Work.....	10
2.4 Summary.....	14
3. A VIDEO AUTHENTICATION SCHEME USING DIGITAL SIGNATURE STANDARD AND SECURE HASH ALGORITHM FOR H.264/AVC MAIN PROFILE	15
3.1 Introduction	15
3.2 Digital signature computation of the coded video sequence	16
3.3 Digital signature verification of the coded video sequence.....	21

3.4 Identifying tampered locations and sender forgery	24
3.5 Storage requirement for the algorithm.....	29
3.6 Advantages and limitations of using multiple signatures	31
3.7 Summary.....	31
4. SIMULATION RESULTS	32
4.1 Introduction	32
4.2 Digital Signature Algorithm Parameters	32
4.3 Results for one coded video sequence	33
4.3.1 Without video tampering or forgery	33
4.3.2 Locating tampered frames	33
4.3.3 Detecting malicious activity	34
4.3.4 Special case of frame tampering and sender forgery.....	35
4.3.5 Robustness to quantization	36
4.3.6 Frame reordering attack.....	38
4.3.7 DC attack	40
4.3.8 Comparison with previous work.....	43
4.4 Results for multiple coded video sequences.....	46
4.4.1 Without video tampering or forgery	48
4.4.2 Locating tampered frames	48
4.4.3 Detecting malicious activity	50

4.4.4 Identifying malicious user and tampered locations simultaneously for multiple video sequences	51
4.5 Summary.....	51
5. CONCLUSIONS AND FURTHER STUDIES.....	52
5.1 Conclusions	52
5.2 Further Studies.....	53
Appendix	
A. DIGITAL SIGNATURE STANDARD	54
B. H.264/AVC.....	64
C. VIDEO AND SAMPLING FORMATS	79
D. THESIS SOFTWARE.....	83
REFERENCES.....	85
BIOGRAPHICAL INFORMATION	89

CHAPTER 1

INTRODUCTION

The continuous growth of the Internet and its increasing influence on our lives have led to a huge amount of multimedia content such as images, photos and video being made available for download to millions of users [13]. This is both a boon and a bane for multimedia content creators and users. It has created new business opportunities for content creators, opened new avenues for distribution of multimedia content and has made it possible to reach users in every nook and corner on earth. However it is hard to manage and track digital assets as compared to physical assets apart from being easy to copy and manipulate digital content [12].

Recent advances in developing powerful processors and increasing software sophistication have made it easier to alter and forge digital video content without leaving any trace [20]. From the user perspective, the same multimedia content is now available from different sources and legally acceptable methods for authentication (verifying the originator of content and at the same time ensuring that the content has not been manipulated or falsified in any way) need to be established [13].

Multimedia authentication schemes usually apply cryptographic techniques to secure the multimedia content. The advantage of using cryptographic theory for multimedia authentication lies in the fact that its strength and flaws have been analyzed by mathematicians, engineers and likes for decades and it is widely accepted by the

legal community as a means of protecting and verifying digital data [4].

1.1 Thesis Outline

In this thesis we propose a hard authentication algorithm to (1) verify the integrity of the video content and identify the sender/content creator for videos encoded using H.264/AVC [4] by employing a digital signature. H.264/AVC is the latest video coding standard jointly developed by ITU-T and ISO/IEC [8]. It can achieve significant compression efficiency over previous video coding standards and can be used for a wide variety of applications. More information on H.264/AVC can be found in Appendix B. The characteristics of H.264/AVC are taken into account while selecting the authentication criteria. (2) Identify the tampered locations on a frame level and point out if the authentication failure has been due to sender forgery. (3) This algorithm is also robust to some common video attacks such as frame reordering, macroblock reordering and DC attack.

Figure 1.1 shows a common video authentication and verification process for video signals [24]. The features f from a video are extracted and the authentication data S is computed using f in the authentication process. Features here refer to data (usually smaller in size than the video itself) obtained from a particular representation of a video and this data is unique to a video representation. A change in the video would similarly change the features associated with it. Multimedia authentication schemes apply cryptographic principles to protect and verify the features extracted rather than the video as it results in considerable bit savings and computation time. This data is encrypted

using a user key K to give $E_K(S)$ and appended to the video for transmission/storage.

Figure 1.1 shows the authentication process.

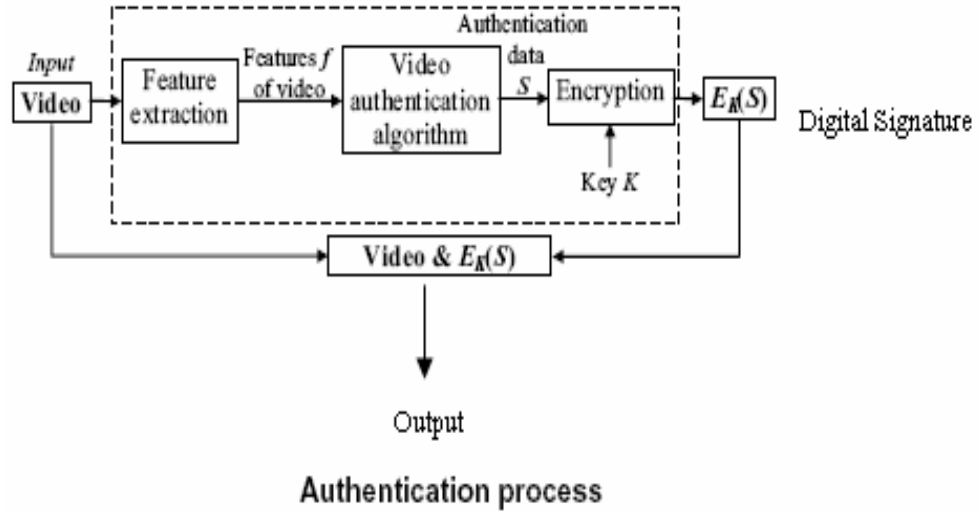


Figure 1.1 Common video authentication system [24].

During the verification process, the new authentication features S' are computed and compared with the original S obtained by decrypting $E_K(S)$ to determine whether the video signal is authentic or tampered. If S and S' match then the video is authentic otherwise it is tampered. Figure 1.2 shows the verification process.

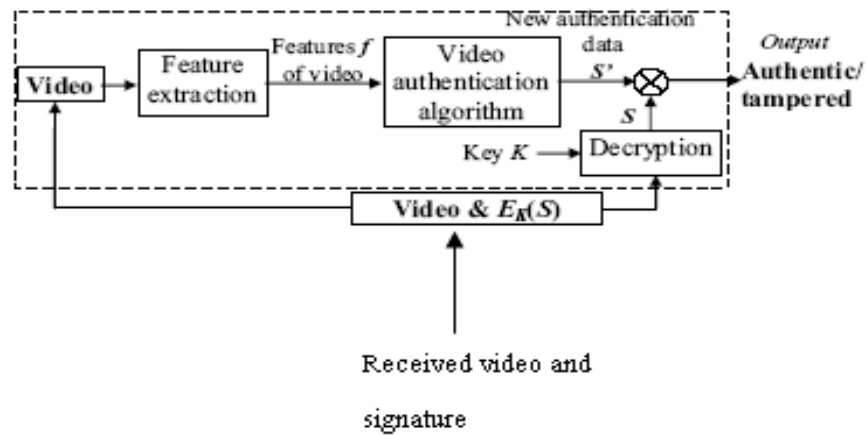


Figure 1.2 Common video verification system [24]

This thesis is outlined as follows: Chapter II describes major multimedia authentication algorithms and related work in the area of video authentication. Chapter III presents our algorithm for authenticating video content. Results are presented in Chapter IV for single and multiple coded video sequences. Chapter V outlines conclusions and further research. Appendix A describes the digital signature standard and secure hash algorithm. Appendix B gives an overview of H.264/AVC, Appendix C lists the common video formats and sampling structure used in the representation of digital video and finally Appendix D is a useful guide on using the software.

CHAPTER 2

MULTIMEDIA AUTHENTICATION

2.1 Introduction

Authentication in this thesis means to verify the integrity and authenticity of the signal without reference to the original signal. Multimedia authentication schemes can roughly be classified into two types: (1) Hard Authentication and (2) Soft Authentication [14]. In hard authentication schemes, no alteration of the pixel values is allowed. These schemes embed the authentication information directly into the multimedia signal or accompany the multimedia signal as additional data. On the other hand soft authentication scheme accepts manipulations to the multimedia signal which maintain the semantic structure and perceptual quality. Our focus here is on the hard authentication schemes.

2.2 Hard Authentication schemes

One of the earliest authentication schemes was proposed for verifying the images produced by a digital camera [15]. The authors propose that the hash of the image taken by the digital camera be computed and later sign this hash key using the digital camera's unique private key to produce a signature file. The key is stored in a secure microprocessor which does allow for the contents to be read outside the factory which

produced them. To verify the images, the signature file and the image whose credibility has to be verified are recovered and a hash of the alleged image is performed. The digital camera's serial number which acts as the public key is used to decrypt the signature and matched with the hash computed. For the image to get verified the decrypted signature has to be the same as the hash of the image whose authenticity has been challenged. Figures 2.1 and 2.2 show the encoder and decoder blocks respectively.

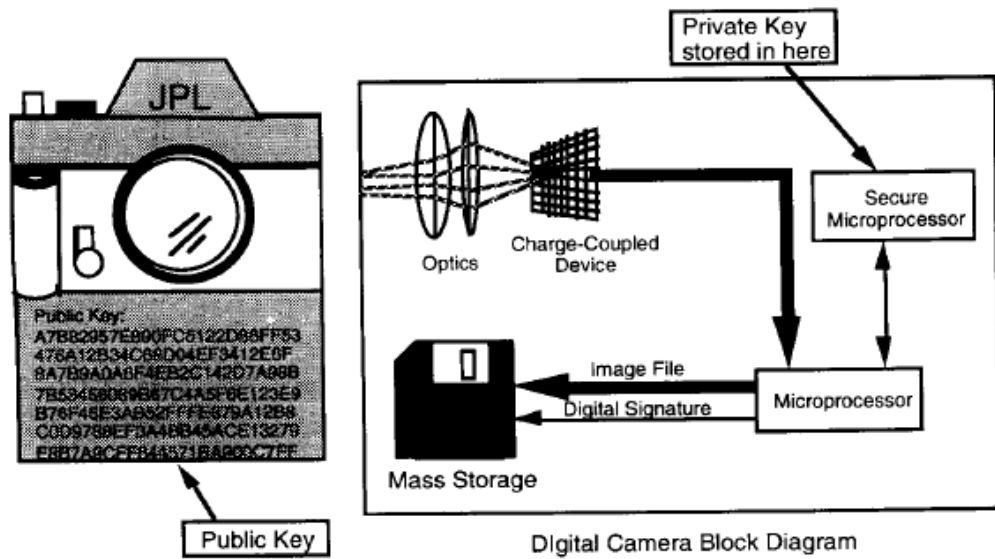


Figure 2.1 Encoder of the trustworthy digital camera [15]

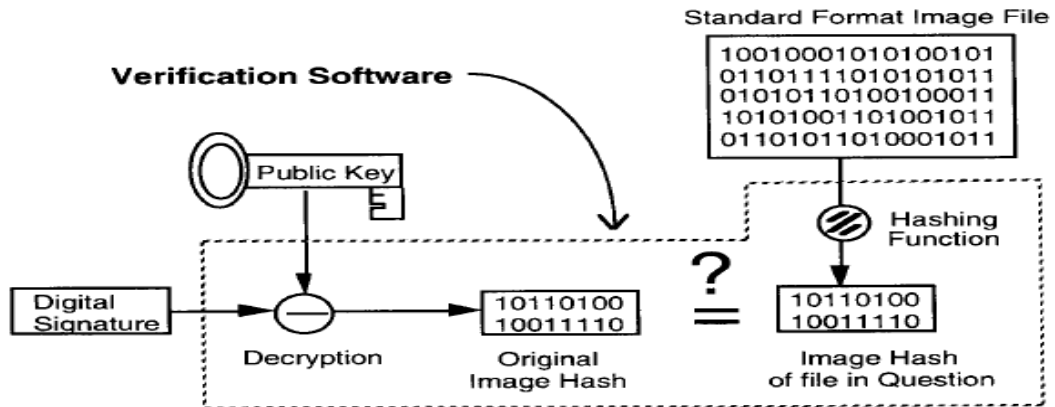


Figure 2.2 Decoder of the trustworthy digital camera [15]

One of the disadvantages of the trustworthy digital camera is that if the private key is exposed by a discontented employee then the public key needs to be changed which means that the camera's serial number cannot be used anymore as a public key.

In [16] the authors propose to generate content based digital signature for image and video signals. A set of features is extracted from the image/video and a hash is computed over these feature values. This hash is signed by the user's private key. The signature generation process is shown in Figure 2.3.

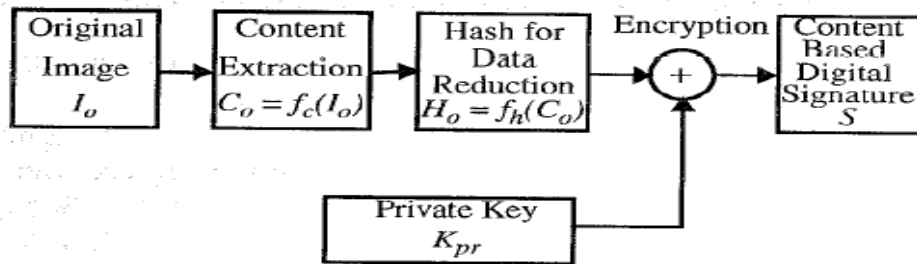


Figure 2.3 Signature generation process using histogram [16]

The authors divide the image into blocks and calculate the features of the block to protect every block individually. The feature extraction used in [16] is the histogram of the luminance blocks as the histogram gives information about spatial information of the image. The image whose authenticity has to be verified undergoes a feature extraction process to extract the same feature as the encoder and then these feature values are hashed and matched with the decrypted signature calculated of the original and unmodified image. Figure 2.4 shows the extraction process.

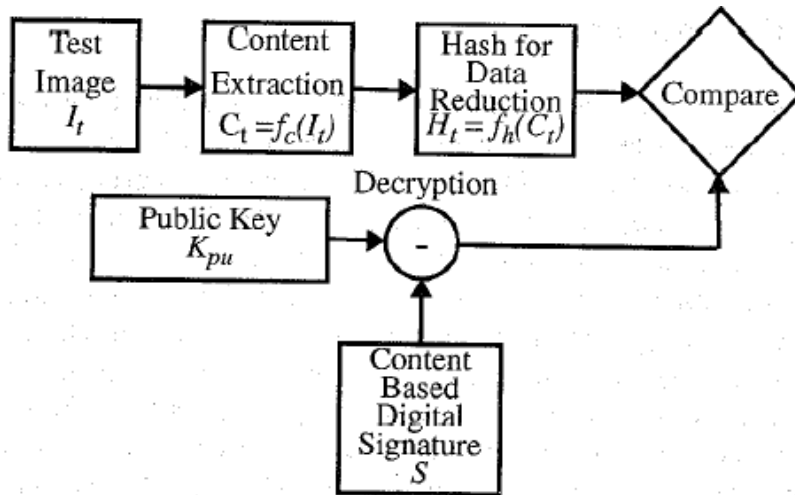


Figure 2.4 Signature verification process using histogram [16]

Schneider et al [16] propose an extension of their scheme to video wherein the individual frames are hashed using their scheme for authenticating images and then all the hashes are collected and arranged in the correct temporal order. A hash is performed of these hash values. This serves two purposes: (1) It authenticates a clip rather than a frame and (2) Prevents frame reordering attack. However the calculation of the histogram of every block and hash computation is a computationally expensive process. It is also quite possible to alter the block values without affecting the histogram [17].

A similar algorithm has been proposed in [18] by Dittmann et al where the Canny edge detector is used on every frame of the video to obtain the edge characteristics which acts as the feature values.

Another landmark paper featuring digital signatures robust to JPEG compression [3] but rejecting malicious manipulations has been published by Lin and Chang [19]. The authors have come up with two theorems for describing the relationship between DCT coefficients of two different blocks at the same position when using JPEG compression.

Let the DCT coefficient vectors of two non overlapping 8x8 blocks **a** and **b** be given as \mathbf{X}_a and \mathbf{X}_b respectively. Let **Q** represent the JPEG quantization matrix and define $Q(v) = \text{Integer Round}(\mathbf{X}_a(v)/Q(v)) \cdot Q(v)$ where $v \in [1, 2, \dots, 64]$. Also let $\mathbf{X}_{a,b} = \mathbf{X}_a - \mathbf{X}_b$ and $Q_{a,b} = Q_a - Q_b$. Additionally let k be a threshold signifying the maximum difference allowed between DCT coefficients set by the user and \mathbb{N} be defined as $\text{Integer Round}(k/Q(v))$.

Theorem 1 gives the sign invariance property of DCT coefficients in the same position at different blocks [19].

- 1) if $\mathbf{X}_{a,b} > 0$ then $Q_{a,b} = 0$
- 2) if $\mathbf{X}_{a,b} < 0$ then $Q_{a,b} = 0$
- 3) if $\mathbf{X}_{a,b} = 0$ then $Q_{a,b} = 0$

Theorem 2 gives the difference relationship between the DCT coefficients after compression [19]. If $\mathbf{X}_{a,b} > k$

$$Q_{a,b} = \begin{cases} \mathbb{N} \cdot Q(v) & k/Q(v) \in \mathbb{Z} \\ (\mathbb{N} - 1) \cdot Q(v), & \text{elsewhere} \end{cases}$$

If $\mathbf{X}_{a,b} < k$

$$Q_{a,b} = \begin{cases} \mathbb{N} \cdot Q(v) & k/Q(v) \in \mathbb{Z} \\ (\mathbb{N} + 1) \cdot Q(v), & \text{elsewhere} \end{cases}$$

If $\mathbf{X}_{a,b} = k$

$$Q_{a,b} = \begin{cases} \mathbb{N} \cdot Q(v) & k/Q(v) \in \mathbb{Z} \\ (\mathbb{N} \text{ or } \mathbb{N} \pm 1) \cdot Q(v), & \text{elsewhere} \end{cases}$$

The above set of theorems is used to extract feature codes Z after quantization which maintain the image structure but yet are robust to compression. These feature codes are then encrypted using a public key encryption algorithm such as the RSA [27] to obtain the digital signature. During the authentication process the image is transformed and then the corresponding relationship between the DCT coefficients in different blocks is found out. This relationship has to match with that obtained by decrypting the signature.

2.3 Related Work

Lou et al [17] propose a scheme for verifying and locating tampered locations for images. The authors propose that the image be divided into 8×8 blocks and the mean i.e. DC value of every luminance block after quantization be written into a file along with some parameters such as the compression scheme used, the byte size of the image and threshold for image authentication. This file is later encrypted using public-key cryptosystem to give the digital signature. The encoder of this scheme is shown in Figure 2.5.

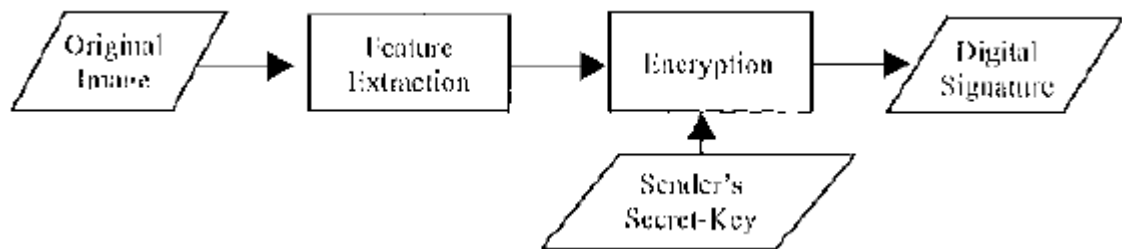


Figure 2.5 Digital signature generation using block DC value [17]

Here, the feature refers to the quantized DC value of every 8×8 block. The signature verification process is the inverse process of the encoder. The DC

values before quantization of the 8x8 blocks are extracted and matched with the decrypted signature. If the values match, then the signature is verified. A block diagram of the decoder process is shown in Figure 2.6.

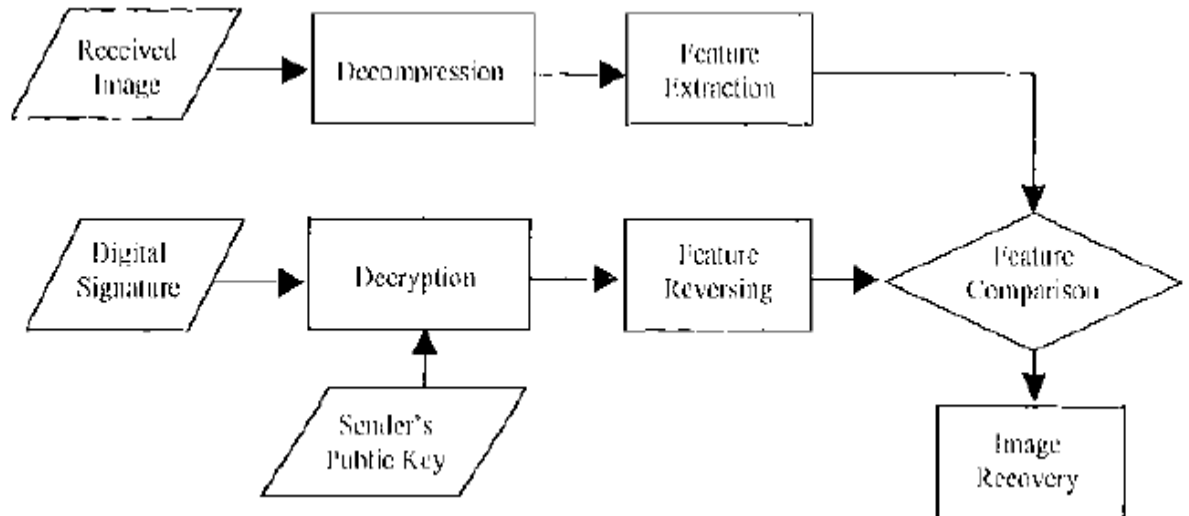


Figure 2.6 Signature verification using block DC value [17]

This scheme is also robust to compression and can identify the tampered locations. The original image is compressed using the maximum compression ratio allowed to give a corrupted version of the original image and then the difference between the DC values of the original image and corrupted image is encoded. If the received image's DC value lies within the range specified by the difference value then the image is authenticated. Figure 2.7 shows the feature extraction and threshold calculation process [17].

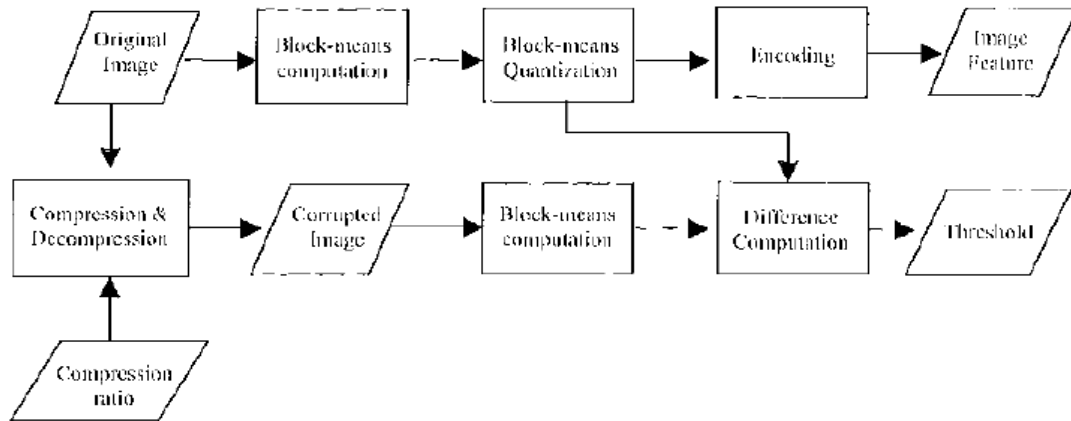


Figure 2.7 Flow diagram of feature extraction proposed by Lou et al [17]

A shortcoming of their scheme is that their feature extraction process is prone to the DC attack wherein the 8x8 block of pixels is replaced completely without any change in the mean value.

A semi-fragile watermarking system for MPEG video authentication has been developed by Yin and Yu [21]. They propose that two watermarks be embedded: a robust watermark to survive content preserving signal processing manipulations and a fragile watermark to prevent malicious attack. The robust watermark embedded is the hashed DC values of every frame. The fragile watermark is inserted by modifying the LSB of quantized DCT AC coefficients to output an even number if '0' is to be embedded and a '1' if an odd number. The fragile watermark is also hashed and encrypted. Figure 2.8 shows the watermark generation algorithm.

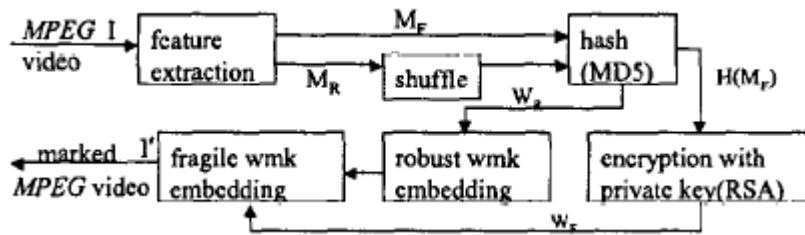


Figure 2.8 Semi-fragile watermark generation [21]

To verify the video the same feature is extracted from the test MPEG video and compared with the extracted watermark. The fragile watermark is compared first to detect if any malicious manipulations have taken place. The robust watermark is compared only if the fragile watermark has been destroyed. The watermark verification process is shown in Figure 2.8.

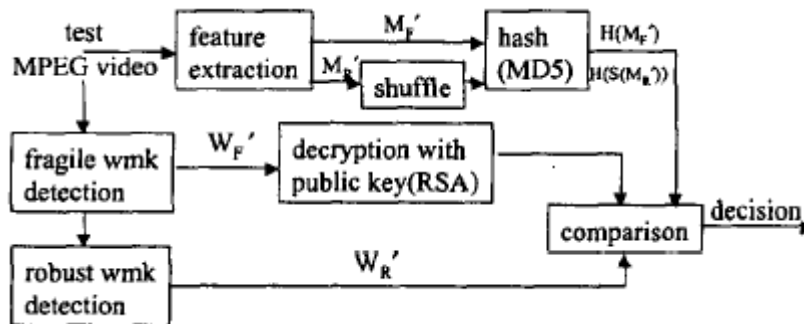


Figure 2.9 Semi-fragile watermark verification [21]

This scheme also suffers from the DC attack first pointed out in [24]. Grosbois et al have proposed a scheme for verifying the integrity of images compressed by JPEG-2000 [22]. Every code-block of the compressed image is hashed using the secure hash algorithm and encrypted to give the digital signature. This encrypted hash value is inserted directly in the bitstream after the JPEG-2000 termination marker (2 byte value greater than 0xFF8F). By default, the arithmetic decoder in

JPEG-2000 stops reading further values from the bitstream when it encounters the termination marker. Figure 2.9 shows their authentication scheme for every code-block.

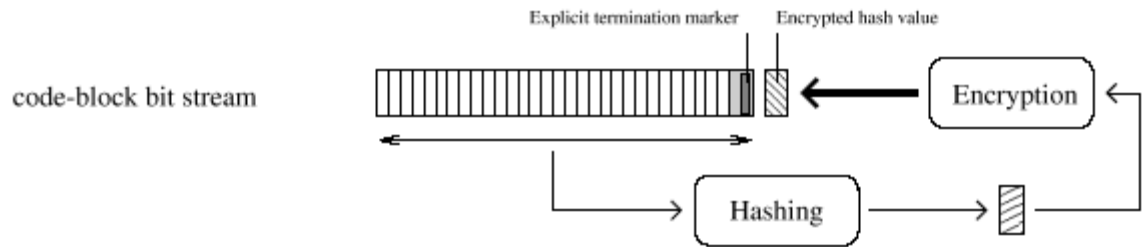


Figure 2.10 Signature generation of JPEG-2000 code-block [22]

To verify the integrity of the image, the test image's code-blocks are hashed and matched with the decrypted hash sent. If both the hash outputs match then the image's integrity is assured.

2.4 Summary

This chapter presented some multimedia authentication schemes which employ cryptographic principles. We also presented the previous work leading to this thesis. Chapter III presents a detailed overview of our proposed algorithm.

CHAPTER 3

A VIDEO AUTHENTICATION SCHEME USING DIGITAL SIGNATURE STANDARD AND SECURE HASH ALGORITHM FOR H.264/AVC MAIN PROFILE

3.1 Introduction

As mentioned earlier in Chapter 1 the goal of this thesis is to identify attacks on video sample values and achieve sender identification. Since an alteration on the frame values in spatial domain in a video translates into a change of the values in transform domain, the authentication scheme to verify the integrity can be applied in the transform domain directly. Sender identification is achieved by allocating a private key to every user, which is used in computing a distinct signature for every user. The proposed algorithm can also identify the tampered locations at frame level in the event of authentication failure, is robust to temporal and spatial attacks and can point out counterfeit videos created by a malicious user.

This proposal is based upon the scheme proposed in [23] to verify the integrity of the images compressed by JPEG-2000. We propose to compute the signature for every coded video sequence [1] (A coded video sequence can be defined as a sequence which can be decoded independently of any other sequence. Every coded video sequence always starts with an I picture.), encrypt it and append it to the H.264/AVC bitstream [8] using the Supplemental Enhancement Information (SEI). SEI is described in detail in section B.2.2.1 of Appendix B. There can be multiple signatures for authenticating the

video if there is more than one coded video sequence present in the video; each signature authenticating a single coded video sequence.

3.2 Digital signature computation of the coded video sequence

In our algorithm we have used the Secure Hash Standard (SHS) [25] and Digital Signature Standard (DSS) to compute the digital signature [26]. The DSS is split into two parts: (1) Digital Signature Algorithm (DSA) and (2) Digital Signature Verification (DSV). SHS and DSS are described in Appendix A. The human visual system is more sensitive to luminance than to chrominance [2] and a modification of the luminance values would affect the perceptual quality. This information is kept in mind while designing our proposal and the signature is computed over luminance frames only. Both Intra as well as Inter frames are used in the authentication process.

For macroblocks using Intra 4x4 and Inter prediction, the DC values and the 2 AC values which are to the immediate right of and below the DC coefficient for every 4x4 block of a macroblock obtained after the integer 4x4 DCT and quantization are used as the authentication criteria. Figure 3.1 shows a 4x4 luminance block in the transform domain. The DC and AC values used are shown in boldface.

DC (0,0)	AC (0,1)	AC (0,2)	AC (0,3)
AC (1,0)	AC (1,1)	AC (1,2)	AC (1,3)
AC (2,0)	AC (2,1)	AC (2,2)	AC (2,3)
AC (3,0)	AC (3,1)	AC (3,2)	AC (3,3)

Figure 3.1 Transformed luma coefficients used in the authentication process for Intra 4x4 and Inter blocks of a macroblock.

H.264/AVC employs another intra prediction method called as Intra 16x16 for coding macroblocks with smooth areas that contain sufficient correlation [5]. The macroblock is divided into 16 blocks of size 4x4 and the integer 2D-DCT is applied to each block. Figure 3.2 shows the macroblock divided into 4x4 blocks. The small rectangle in the upper left corner of every block is the DC value obtained after the integer 2D-DCT.

<small>00</small> 0	<small>01</small> 1	<small>02</small> 4	<small>03</small> 5
<small>10</small> 2	<small>11</small> 3	<small>12</small> 6	<small>13</small> 7
<small>20</small> 8	<small>21</small> 9	<small>22</small> 12	<small>23</small> 13
<small>30</small> 10	<small>31</small> 11	<small>32</small> 14	<small>33</small> 15

Figure 3.2 Intra 16x16 luma macroblock after integer 2D-DCT (The numbers 0-15 indicate the sixteen 4x4 blocks of a luminance macroblock. The numbers 00-33 represent the DC coefficient of every 4x4 block) [7].

The sixteen DC coefficients obtained after the integer 2D-DCT of each 4x4 block in a macroblock have sufficient correlation among them and thus a second transform is applied to these DC coefficients to decorrelate them. The transform used is the 4x4 Hadamard transform. The integer 2D-DCT, Hadamard transform and Intra prediction modes are discussed in detail in Section B.3.1 and B.3.2 in Appendix B.

All the nonzero coefficients obtained after the 4x4 Hadamard transform and quantization are used as the authentication criteria. Apart from the DC coefficients, the

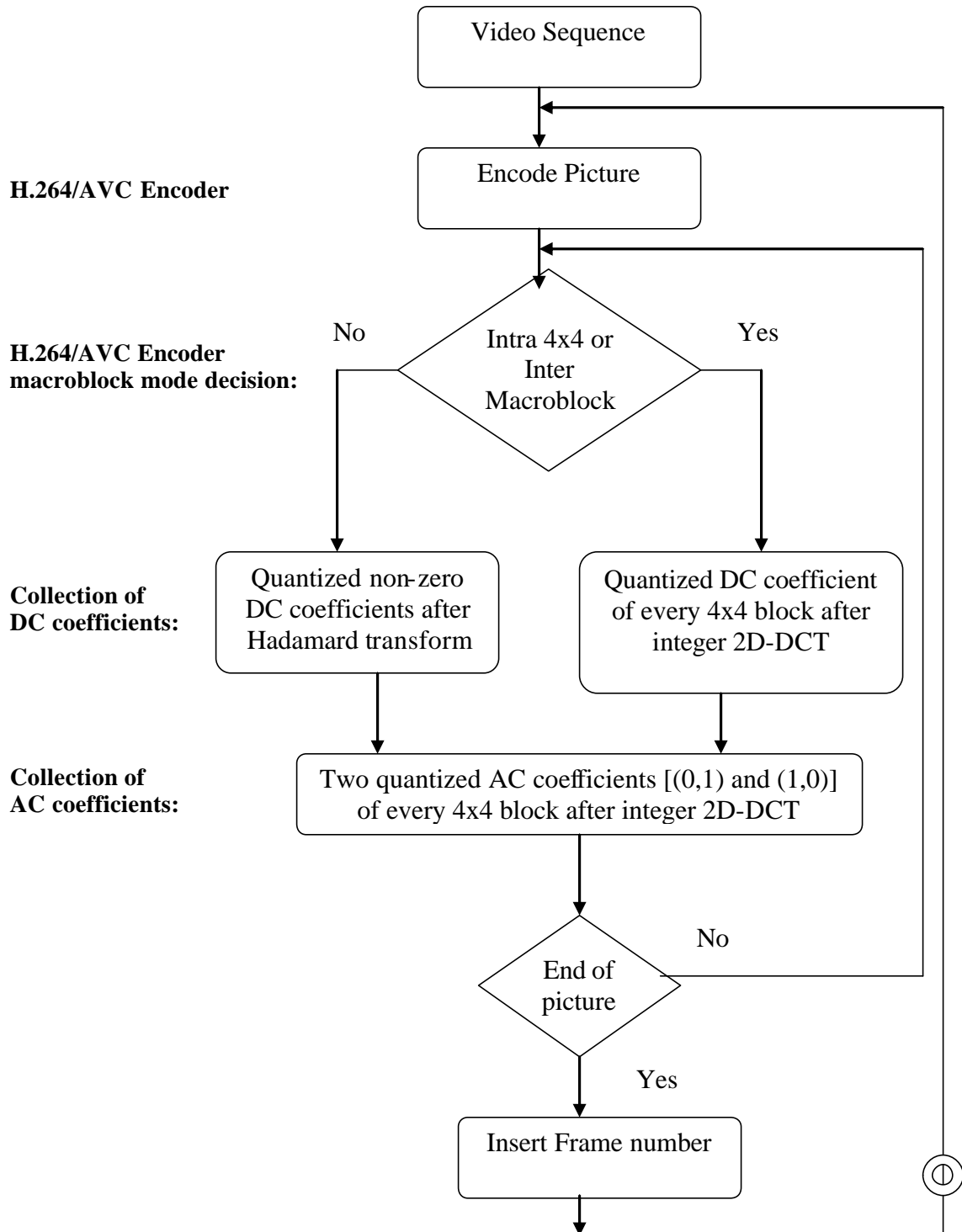
two AC coefficients located at (0, 1) and (1, 0) of every 4x4 block of the INTRA 16x16 macroblock are included in the authentication process. Thus a total of thirty-two AC coefficients are included for the INTRA 16x16 macroblock. Figure 3.3 shows the two AC coefficients of every block that are used in the authentication process in boldface.

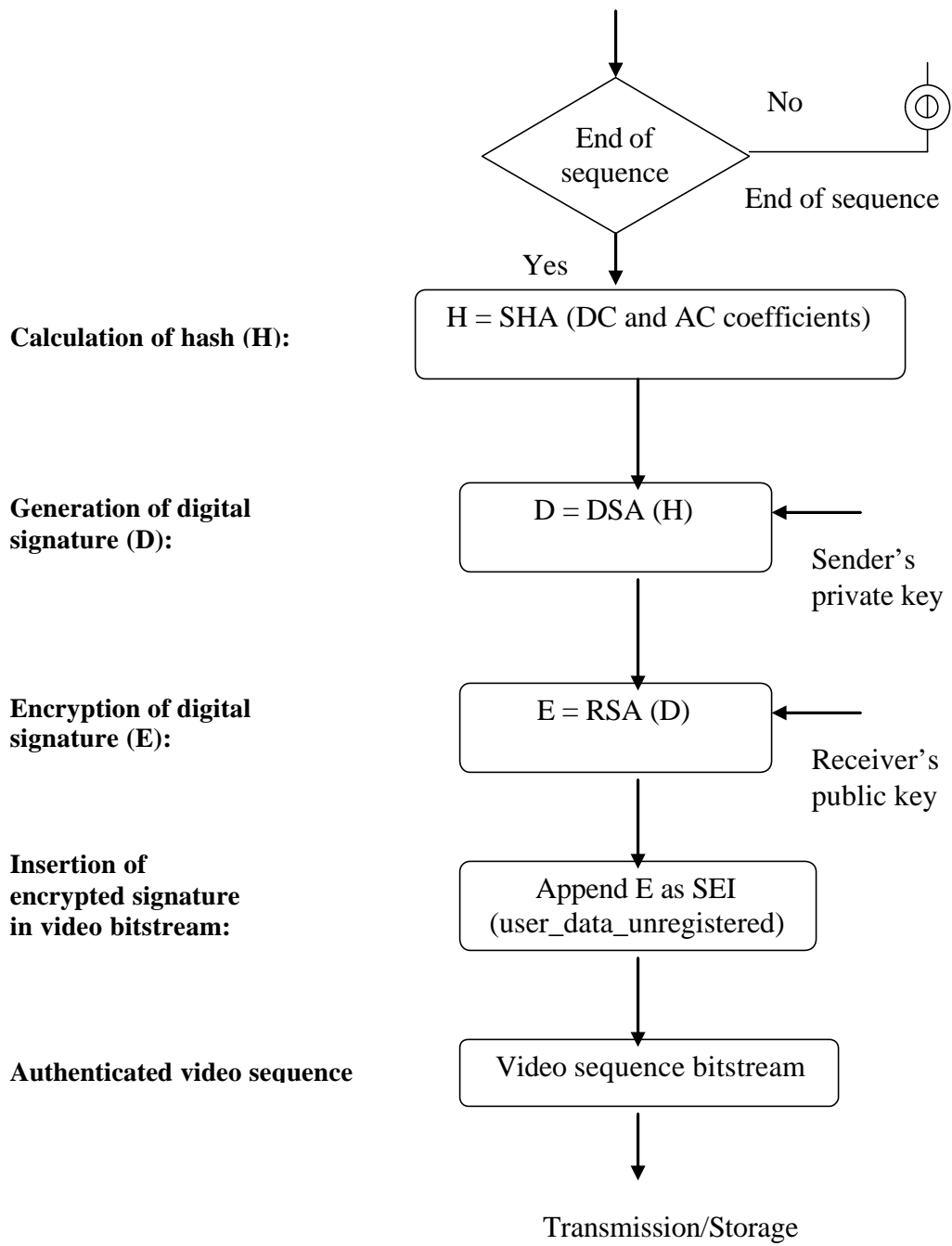
	<u>AC (0,1)</u>	AC (0,2)	AC (0,3)
<u>AC (1,0)</u>	AC (1,1)	AC (1,2)	AC (1,3)
AC (2,0)	AC (2,1)	AC (2,2)	AC (2,3)
AC (3,0)	AC (3,1)	AC (3,2)	AC (3,3)

Figure 3.3 AC coefficients ((0, 1) and (1, 0) of every 4x4 block in the integer 2D-DCT domain) of the Intra 16x16 macroblock included in the authentication process.

The absolute picture number of every frame encoded using H.264/AVC is also inserted in the data to be authenticated to prevent the frame reordering attack in which the temporal ordering of frames can be changed. This picture number is inserted after all the authentication data for a picture has been collected.

When the end of the coded video sequence is detected, the entire authentication data collected is hashed using SHS. The SHS outputs a 160 bit fixed message digest which is given to the DSA. The DSA uses the hash and the user's private key to generate a unique signature consisting of two numbers, each having a length of 160 bits. This signature acts as the authenticated information of a video sequence and is appended to the H.264/AVC as SEI [4] in the ITU-T unregistered data section after encryption [27]. A flowchart of the encoder is shown on the next page.

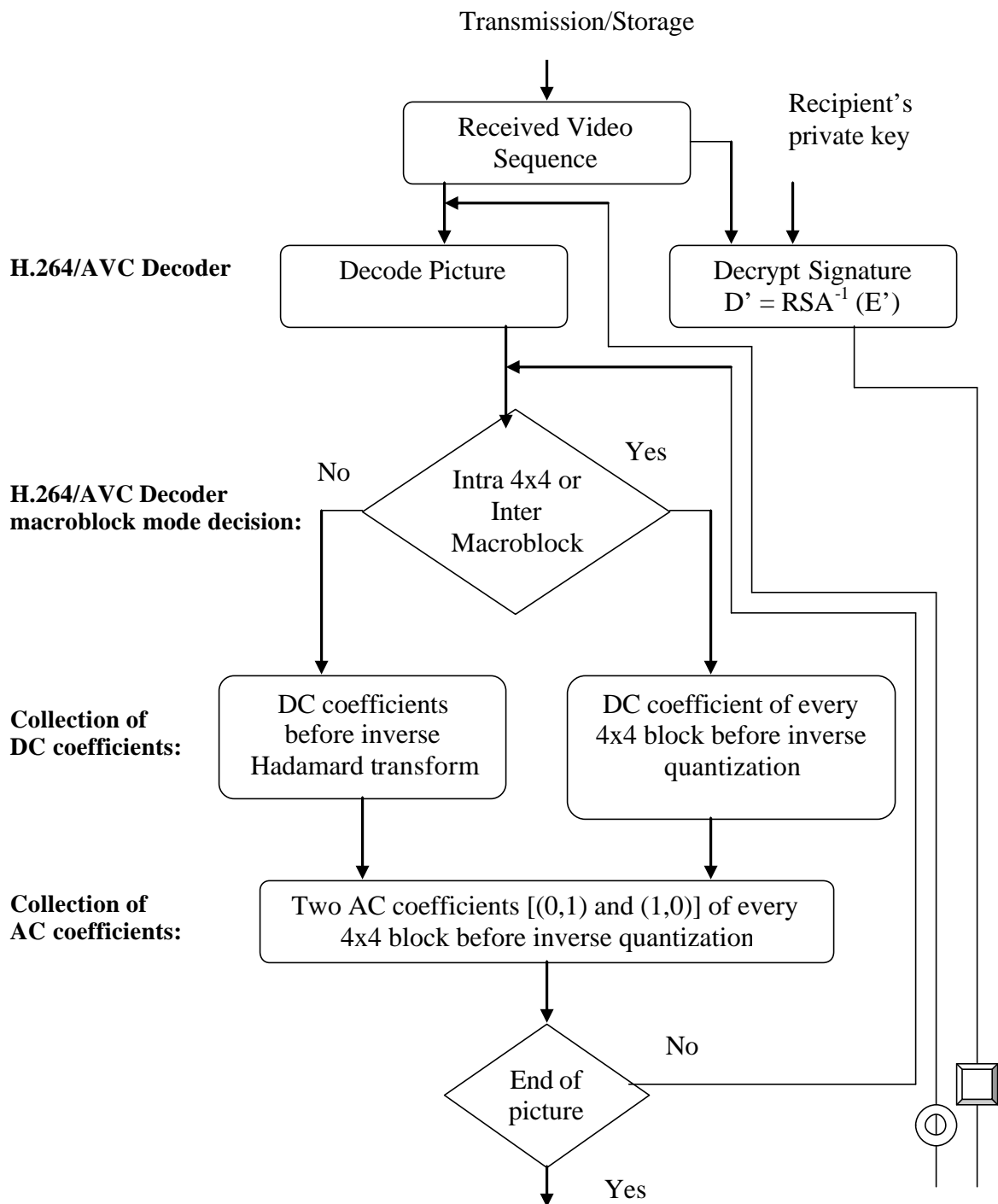


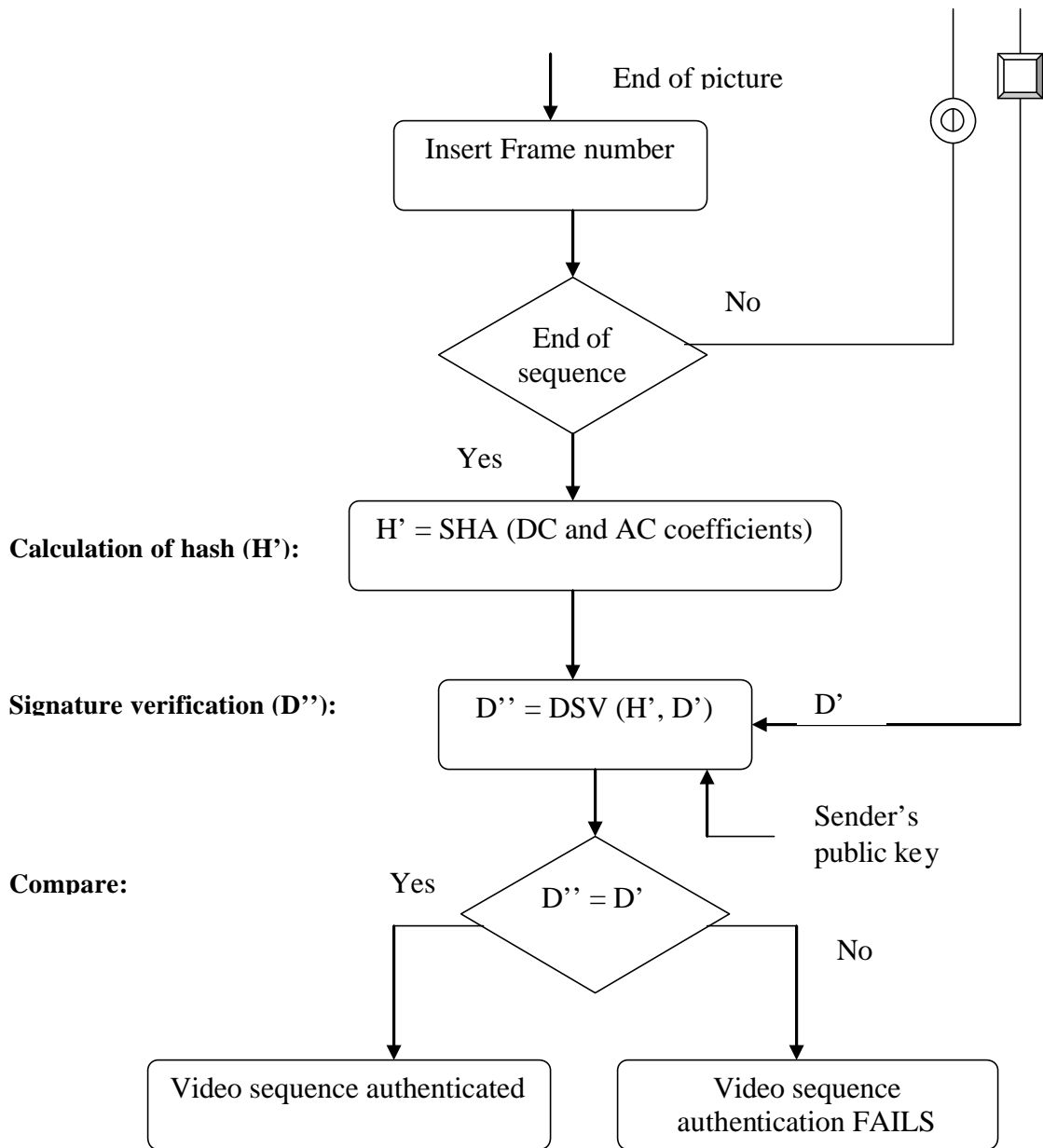


Flowchart 3.1 Encoder for calculating digital signature

3.3 Digital signature verification of the coded video sequence

The flowchart on the next page depicts the decoder process for verifying the digital signature. The compressed video bit stream is decoded and the signature extracted from the SEI is decrypted using the private key of the receiver. The decoder process for collecting the authentication data is the inverse of the encoder. For Inter and Intra 4x4 blocks, the DC and two AC coefficients to the immediate right and below the DC coefficient before inverse quantization are used in the authentication process. For Intra 16x16 macroblock, the DC coefficients before inverse Hadamard transform and inverse quantization and the AC coefficients of every 4x4 block located at positions (0,1) and (1,0) before inverse quantization are used in the signature verification process.





Flowchart 3.2 Decoder for verifying the digital signature of the received video

When the end of a picture is detected the absolute picture number is inserted at the end. Once the end of video sequence is detected, all the coefficients collected are hashed using SHS. The hash along with the user's public key and the signature sent by the encoder are used in the signature verification process. The output of the verification

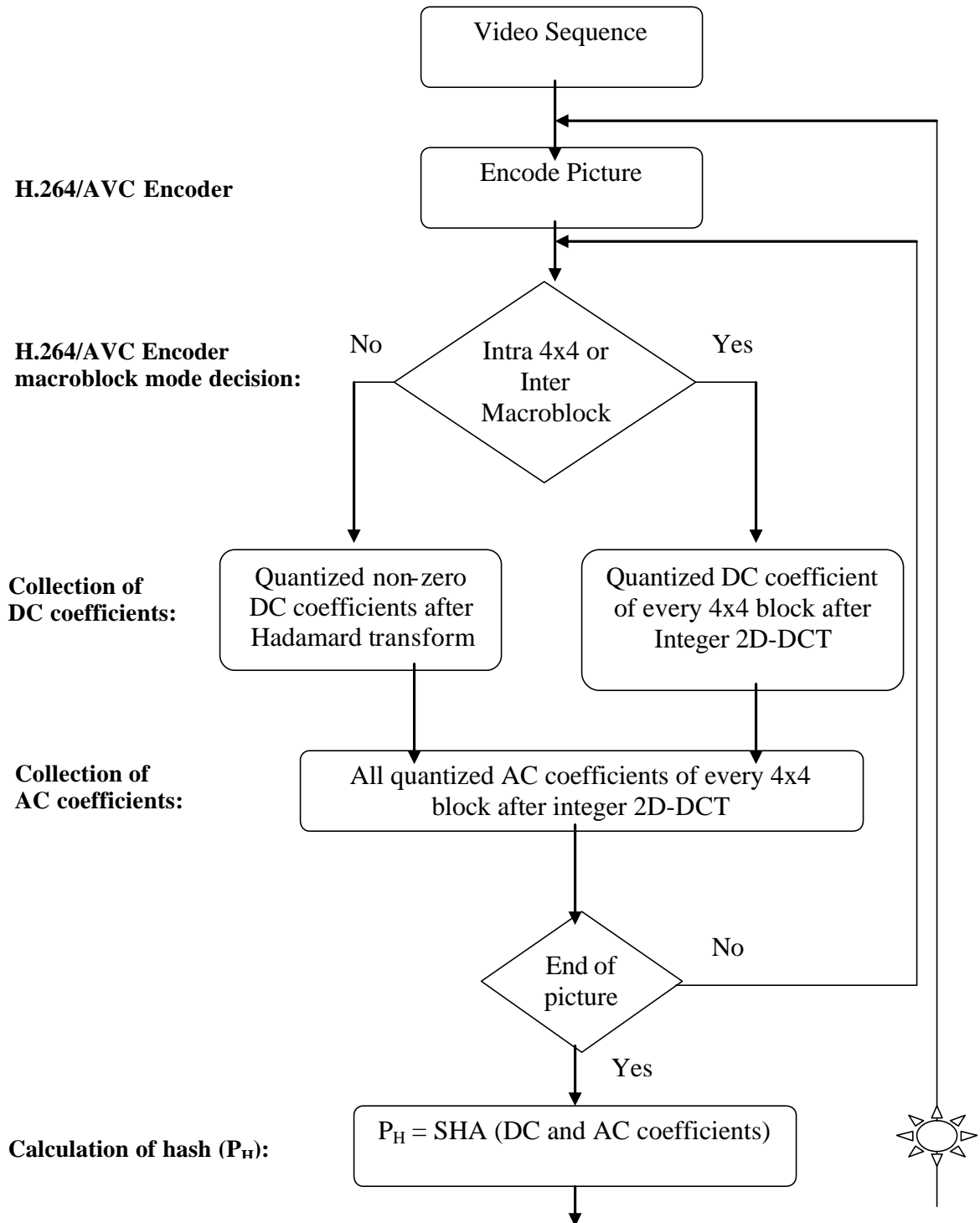
process is a binary value indicating whether the video has been authenticated or not. The authentication process will fail if there has been a modification of the transform coefficients, if the frames have been interchanged or if the originator of the video cannot be verified. Forgery may try to occur when a malicious user signs the video sequence using a different private key than the original one and tries to verify the video content using the public key of the original user. Since there is a unique relationship between the private and public keys of the user, the signatures shall not be verified in case of forgery. However it is not possible to determine the cause of authentication failure in a hard authentication scheme such as this one. To identify the tampered locations and point out malevolent intentions in the case of an authentication failure we have proposed some additions that help in overcoming the two above mentioned shortcomings.

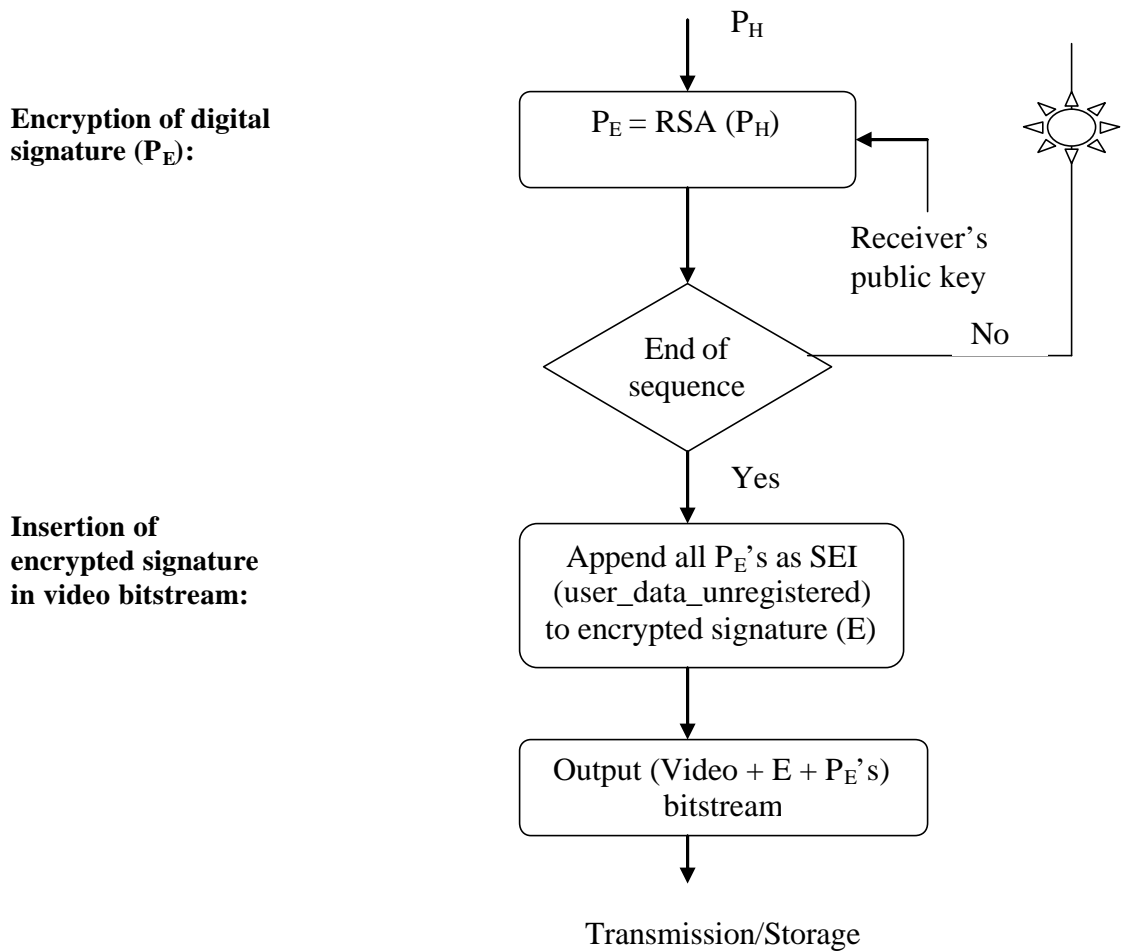
3.4 Identifying tampered locations and sender forgery

The hash of a message is a one-way function [25]. The original message cannot be recovered from the hash and even a minor modification in the original message produces a completely different hash. This information is taken into account while designing our algorithm. We propose that the hash of every picture (luminance component only) be computed in addition to the signature. The encoding procedure is depicted as a flowchart in the following page.

These hash outputs (one for every picture) are appended to the signature after encryption by RSA algorithm in the SEI section before transmission of the video. As shown in the above figure, for Intra 4x4 and Inter macroblocks, the quantized DC and AC coefficients are used as input to the SHS. For Intra 16x16 all the nonzero DC coefficients

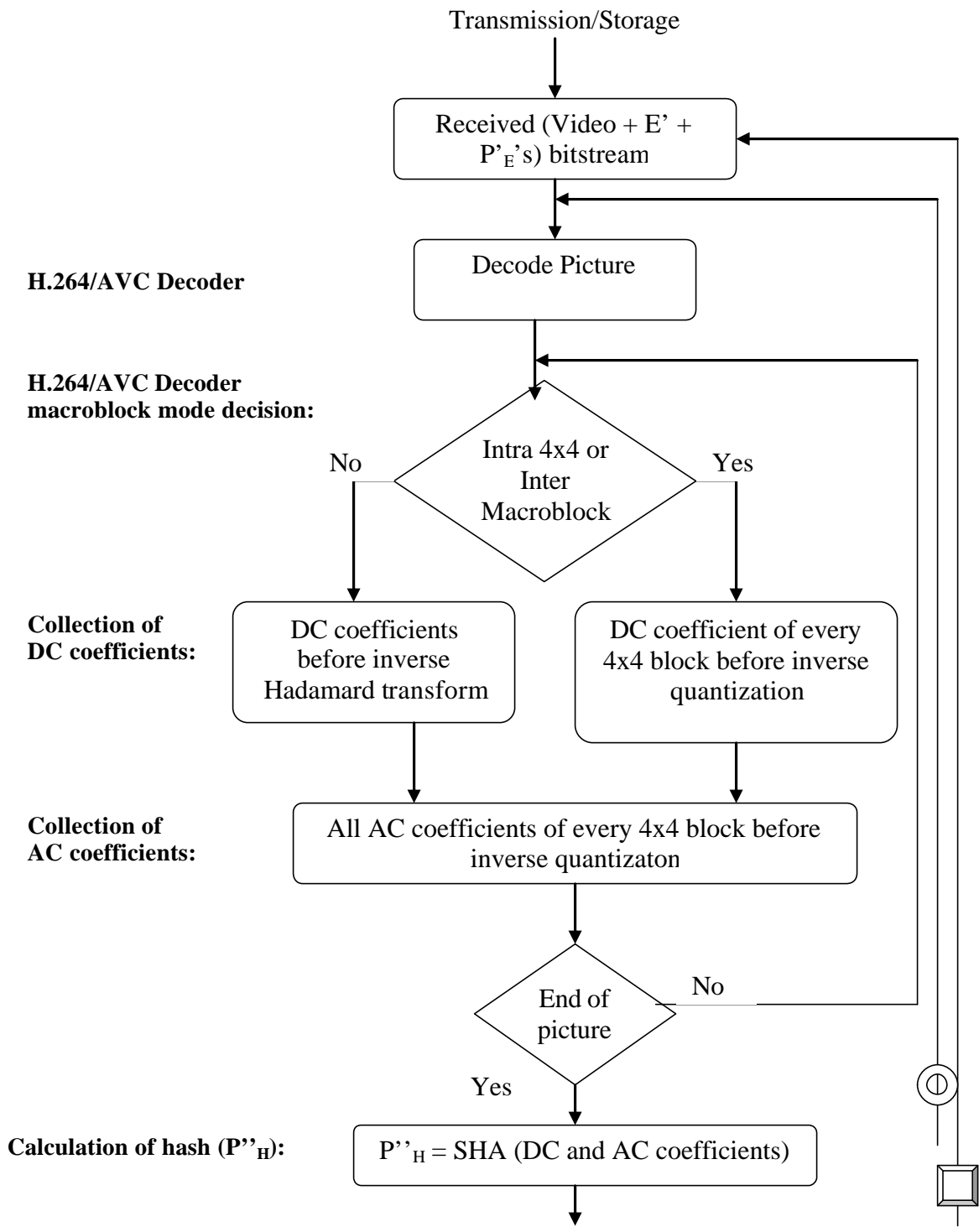
after Hadamard transform and quantization and all the AC quantized coefficients of the macroblock are used in computing the hash.

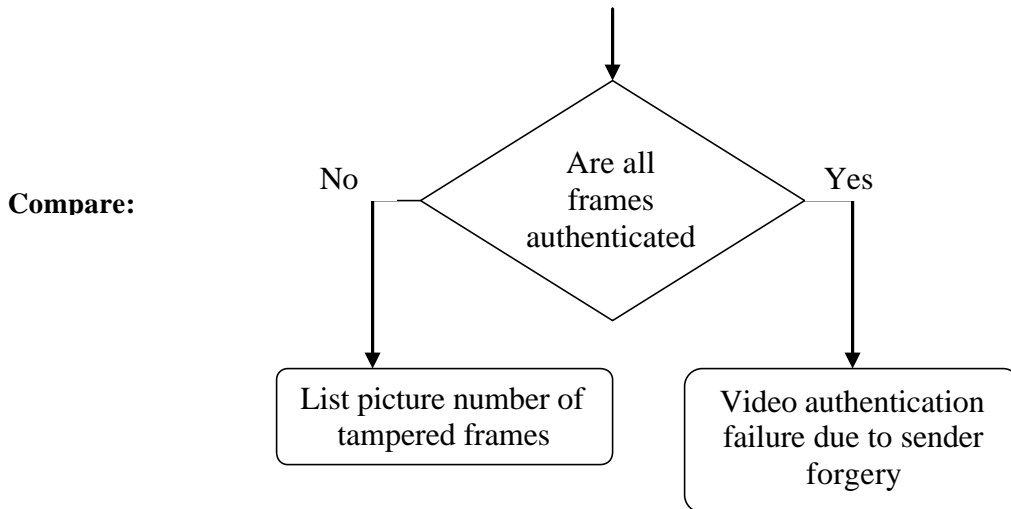




Flowchart 3.3 Hash computation process for every picture of the video sequence

At the decoder side, if the signature verification fails, then the hash of every frame is computed and matched to the one sent by the encoder so that the tampered frames may be identified. The next flowchart shows the decoding procedure.





Flowchart 3.4 Process of identifying tampered locations and sender forgery

As shown in the above figure, for Intra 4x4 and Inter macroblocks, the DC and AC coefficients of every block of a macroblock before inverse quantization are used in computing the hash. For Intra 16x16 macroblock, the DC coefficients before inverse Hadamard transform and quantization are used in the authentication process. All the AC coefficients of every 4x4 block of the Intra 16x16 macroblock are also used in computing the hash. In case the signature fails to verify then the hash outputs from the decoder are matched with the respective ones sent by the encoder. If tampering has occurred in frames then the hash of the corresponding frame will not match with the one sent by the encoder and the tampered frame can be identified. If the hash outputs of the decoder match with the ones sent by the encoder in spite of an authentication failure then it can be inferred that the failure has been due to sender identity forgery.

3.5 Storage requirement for the algorithm

The digital signature consists of two numbers; each of the numbers is 160 bits in length. A total of 320 bits are required for storing the signature. An additional 160 bits

are required to store the hash of every picture of a video sequence. If there are n pictures in a video sequence, $160*n$ bits will be required to store the hash outputs. Thus a total of $320+160n$ bits are required for authenticating a single video sequence.

However the hash outputs and the signature cannot be sent unprotected over the channel, as it will be susceptible to tampering. They need to be encrypted. In this thesis a standard encryption algorithm for encryption known as the Rivest (R) Shamir (S) and Adleman (A) algorithm [27] or RSA is used. 1024 *bits* are required for encryption of the signature and another 1024 *bits* are required for encryption [28] of each hash output. If there are n pictures in a video sequence, $1024*(n+1)$ bits will be required after encryption for authenticating a video sequence.

If frames are transmitted at 30 *fps* and a coded video sequence consists of a second of a video i.e. 30 frames in a video sequence then the authentication bits required would be $1024*(30+1) = 31744$ *bits/second* or approximately 31 *kbps*. This high data rate is attributed to the extra bits added due to encryption of signature and hash outputs.

The Main Profile ranges from 1 to 8+ Mbps [5]. Assuming that the compressed bitrate is 2Mbps and there are 30 frames per second, then the authentication bits required is 31Kbps (calculated in the previous paragraph) which is equivalent to 1.55% of the compressed video. This rate of 31 kbps will be constant for all data rates supported by the Main Profile assuming that the frame rate per second is 30 and 1 coded video sequence per second. As the frame rate increases the authentication bits required increases and vice-versa.

3.6 Advantages and limitations of using multiple signatures

The advantages of using multiple signatures for the video are that each coded sequence can be transmitted separately e.g. a short clip of a movie. It increases robustness to errors as an authentication failure for a single video sequence would mean transmitting only that particular sequence in the worst case.

One of the disadvantages is that the authentication bits required increases as multiple signatures are required and with robustness in identifying tampered locations.

3.7 Summary

We presented the details of the proposed algorithm for computing and verifying the integrity of the video sequence coded by H.264/AVC main profile using digital signature. The algorithm can also identify tampered locations at frame level and identify sender forgery at the expense of extra bits by computing the hash of every picture and matching it with the encoder hash. The signature and hash outputs are sent in the H.264/AVC bitstream as SEI.

Chapter IV presents the results of the proposed algorithm for single and multiple coded video sequences. Some video attacks are also performed in order to evaluate the robustness of the algorithm to these attacks.

CHAPTER 4

SIMULATION RESULTS

4.1 Introduction

This chapter presents the simulation results of the proposed algorithm. The test sequence used is Foreman in QCIF and with 4:2:0 sampling format [10]. More information about video and sampling formats can be found in Appendix C. The results are presented for one signature as well as multiple signatures. Results of robustness of the video to some attacks are also presented. The signature and hash digests sent by the encoder are encrypted using the RSA.

4.2 Digital Signature Algorithm Parameters

Some parameters are kept common for generating and verifying signatures and are unchanged throughout the sequence except when explicitly mentioned. These parameters are listed as follows:

p (in HEX)

A47BA090E6B7EA3181FEFD168BD6BD85342AC4F788CAC2DA1E5938152C9393D
B8D202547FDC849448CD82A9F16B22FE6DB7AE24D7CEEEEB35CC50D259111C0
4BA203E67722A60FD91B54490202624E2E8D6EB4B59DC58204B1C3676668672168
BE478314246A1F8020A3AC793BEB A662050E3E2BAD50245304E465DAD6769815

g (in HEX)

5D1EB2EDC50ADA4970C788A41597483FF569A13011656C2F164B53D5714DB7C7
31EC3B61D7CA1A87FF56ACB4A50DD98AA162534F73DD472A53445CCCD356AC
7235E30ABE170AF7CDF38710C4AD7F7B84AC5C1329546180B3CF816C3F5DF99D
970C5B42010DACD14EE18B25B39802F994E737610759F1E986D3D4591A6F0C8AC
E

q (in HEX)

C3249FAAE671EF3B4EB08D58B98DC640BCB3CD91

Private Key (in decimal)
932308816105877193346324924405238358561797363453

Public Key (in decimal)
492503645189767602019356120312426256649703007844889992066289138986848855
532795488852114767691790638480862659813905528288607709954116167390074290
693300330640086973845494733846378760609158085317472705796081640165701659
252979090706950522104464269188240136128146745079944665789155750285972825
19755964472561590380

4.3 Results for one coded video sequence

A total of fifty nine frames were encoded using the H.264/AVC reference software (version JM 7.5c) [9]. The total number of coefficients over which the signature was calculated was 12501. The signature which consists of two 160 bit numbers (r and s) obtained at the encoder are shown in base 10 below.

$r = 258230994199886379618680381531392644375747268307$
 $s = 553921005467375065529673563440116158716908054793$

4.3.1 Without video tampering or forgery

The decoder computes a signature based on the received message and encoder's signature to produce a one sixty bit number v. For the signatures to be verified v should be equal to r. The signature (v) at the decoder end is shown below.

$v = 258230994199886379618680381531392644375747268307$

Since v and r match, the video can be claimed to be from an authentic source.

4.3.2 Locating tampered frames

Picture number 0 which is encoded as an I picture is modified. The DC coefficient located at (0,0) after Hadamard transform and quantization is changed from

the original value of 5 to 15. The rest of the coefficients are kept unchanged. The signature (v) calculated at the decoder is shown below.

v = 363510879679629003962752313295967888200812862057.

Note that v and r do not match in this case which leads to an authentication failure of the video. The hash values of all the frames of the received video sequence are compared with the decrypted hashes sent by the encoder. The hash output of the first picture in the sequence of the received picture is different from the hash output computed at the encoder. The hash values are shown in base 10 below.

Picture no. 0 (I picture)
Encoder Hash= 244521134779192871239422005322433514569730613982
Decoder Hash= 564856327811211747778209339112070483726931840033

As can be seen since the two digests are not the same, conclusions can be drawn that the I picture has been tampered with and that frame tampering is the cause of signature verification failure. In the case that multiple pictures are tampered with, the hash comparison method would be able to identify accurately the tampered frames due to the one way nature of hash functions.

4.3.3 Detecting malicious activity

An imposter may attempt to sign the video by using his private key and then try to pose it as the original video by attempting to verify it with the original public key. For simulation purposes the private key is changed and the rest of the parameters are shown below. The modified private key is shown in base 10 below.

Modified Private Key = 132308816105877193346324924405238358561797363453

The signature generated from the encoded video sequence using the modified private key is shown in base 10 below.

$r = 258230994199886379618680381531392644375747268307$
 $s = 659115584368708655163218960845447884312706027895$

At the decoder, the received video sequence is authenticated using the public key of the legal user. The signature (v) obtained is shown in base 10 below.

$v = 757611328353143787983558147251463660400321700712$

Since the signatures do not match, an authentication failure has taken place. As per the proposed algorithm all the hash values sent by the encoder (used after decrypting) are matched with the ones sent by the encoder. Since in this case the received video has not been tampered with, all the hash outputs match. Based on the above information it can be deduced that the authentication failure has been due to fraudulent intentions of the signatory.

4.3.4 Special case of frame tampering and sender forgery

In this case assume that the imposter signs the video using his private key and encodes it using the H.264/AVC encoder [8]. If the received video sequence has also undergone frame(s) tampering, then the proposed algorithm can accurately identify the tampered frames but it is not possible to find out if signatory identification failure has also been the cause of signature failure. The example below demonstrates this case.

The modified private key and the corresponding signature (r and s) generated are shown in base 10 format.

Modified Private Key = **1**32308816105877193346324924405238358561797363453

r = 258230994199886379618680381531392644375747268307
s = 659115584368708655163218960845447884312706027895

For simulation purposes frames 2 and 3 are tampered with by replacing them with frame 1 coefficients. The decoder signature obtained thus is shown in base 10.

v = 47286697397561271786668886776492547694937656005

When the signature verification fails the decoder checks the hash output of every received frame with the decrypted hash outputs sent by the encoder. In this case the hashes of frames 2 and 3 do not match as illustrated below.

Encoder hash for frame 2
383954013647539530350428693598443584716009297999
Decoder hash for frame 2
244521134779192871239422005322433514569730613982

Encoder hash for frame 3
487219263382588711046147491230767672185026923094
Decoder hash for frame 3
244521134779192871239422005322433514569730613982

It can be noted that frames 2 and 3 have been identified as the tampered locations of the video. However it is not possible to state conclusively whether sender forgery is also the cause of video authentication failure. Therefore identifying video forgery is a special case when none of the frames in the received video sequence has been tampered with and signature verification fails.

4.3.5 Robustness to quantization

Quantization is employed to code the video at a lower bit rate and yet maintain the perceptual quality e.g. in transcoding, where the video bit stream is coarsely quantized during intermediate stages of transmission to reduce the bit rate. Due to the

irreversible nature of the hash function, quantization operation on a video bitstream invalidates the signature even though the semantic relationship is maintained. For demonstration purposes, the foreman video sequence was first coded using a QP of 29. A signature is calculated for the resulting video sequence. Frame 0 and Frame 1 are shown below in figure 4.1.



Figure 4.1 Frames 0 and 26 encoded using QP 29.

The QP is now changed from 29 to 35. The decoded frames 0 and 26 are shown in Figure 4.2

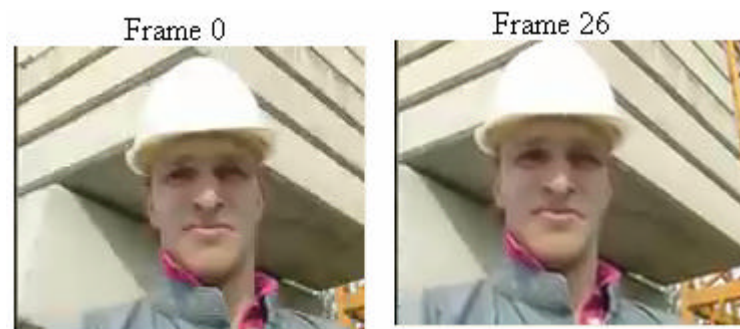


Figure 4.2 Frames 0 and 26 encoded using QP 35

As can be seen from the above two figures perceptual quality is maintained but the signature is not verified by the decoder. The encoder and decoder signatures obtained are shown below.

r = 258230994199886379618680381531392644375747268307
s = 553921005467375065529673563440116158716908054793

v = 450045542578458967124875488958087354781835818707

The proposed algorithm is thus not robust to quantization as it is a hard multimedia authentication scheme.

4.3.6 Frame reordering attack

In frame reordering the temporal order of frames may be changed [6]. This may be of significant consequence when determining the exact timing of the occurrence of an event. To simulate this attack the coefficients of frame 1 and frame 2 are swapped before the signature is verified. Figure 4.3 shows the reordering process.

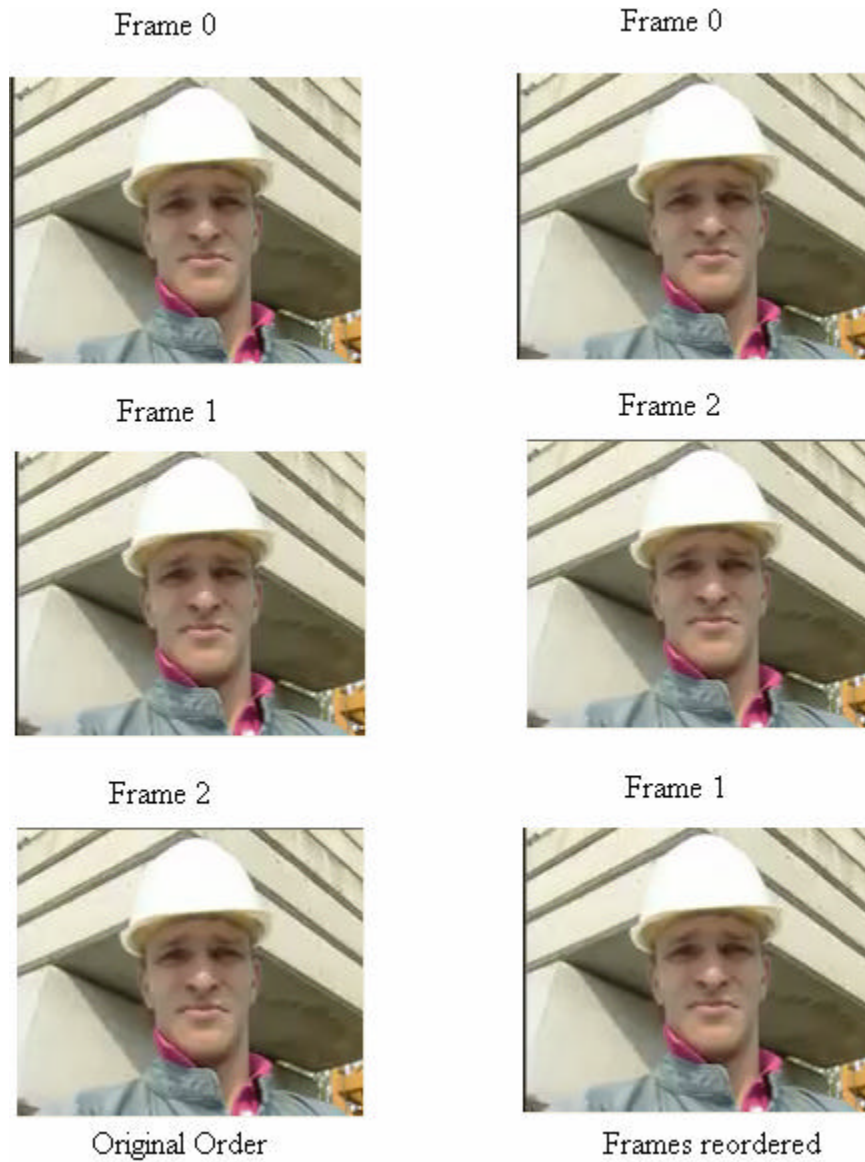


Figure 4.3 Reordering of frames 1 and 2

The encoder and decoder signatures are shown below.

$r = 258230994199886379618680381531392644375747268307$

$s = 553921005467375065529673563440116158716908054793$

$v = 796457292123352592426447215935643118188436156788$

The signatures do not verify and the video sequence will not be authenticated. In the case of multiple video sequences, frame reordering would invalidate only that coded video sequence to which the swapped frames belonged. If the reordering takes place over multiple video sequence boundaries then the corresponding video sequences would not be authenticated.

4.3.7 DC attack

The DC attack modifies the block values without changing the mean of the block. It can introduce perceptual distortion in the final decoded picture. For simulation purposes, six 4x4 block of pixels are modified without changing the mean of each block. The top left location of these six blocks are 1) (0,0) 2) (3,3) 3) (7,7) 4) (11,11), 5) (15,15) and 6) (19,19). One example of block modification is shown in Table 4.1 below.

Table 4.1 DC Attack

<u>Without DC attack</u>				<u>With DC attack</u>			
Original Block of pixels (Location 0,0)				Modified Block of pixels (Location 0,0)			
34	139	219	226	165	154	153	200
34	140	218	225	110	106	104	133
33	139	217	225	120	117	115	132
29	136	218	221	224	221	211	188
Sum of all pixels = 2453				Sum of all pixels = 2453 (unchanged)			

Prediction				Prediction			
40	120	231	197	40	120	231	197
29	140	225	225	29	140	225	225
95	62	186	200	95	62	186	200
29	139	122	237	29	139	122	237

The prediction is the best match found from previously encoded blocks. The residual is the difference between the current block of pixels and the prediction.

Residual				Residual			
-6	19	-12	29	125	34	-78	3
5	0	-7	0	81	-34	-121	-92
-62	-23	31	25	25	-45	-71	-68
0	-3	96	-16	195	82	89	-49

After 4x4 integer DCT				After 4x4 integer DCT			
76	-317	-129	129	76	1482	364	196
-67	301	307	-652	-473	-29	465	-762
138	105	-60	105	726	192	-70	116
-101	-412	11	-251	-219	-567	55	-276

Since the block mean was unchanged, the DC value of the original and modified block remains the same. But the changes in the block structure are reflected in the AC coefficients. Note the AC coefficients in the above two blocks located to the right and below the DC value shown in boldface which are different for the original and modified block.

After quantization (QP 29)				After quantization (QP 29)			
1	-3	-2	1	1	13	5	2
0	1	3	-3	-4	0	4	-4
2	1	-1	1	10	2	-1	1
-1	-2	0	-1	-2	-3	0	-1

After quantization, the DC coefficient of the original and modified block remains the same. If only the DC value of every block was used in the authentication process, then the modified block would get authenticated even though the block of pixels in it are different from the original block.



Figure 4.4 Original foreman frame



Figure 4.5 Foreman frame after DC attack

The figure above on the left shows the original frame of Foreman. When all the six blocks are modified without changing the mean value, as noted above at the start of this section, the frame obtained is shown above on the right. The distortion can be seen in the top left corner of the frame. When only the DC value is used in the authentication process, the above frame on the right will get authenticated.

The current algorithm can detect this attack due to inclusion of two AC coefficients (to the right and below the DC value) of every 4x4 block in the authentication process.

4.3.8 Comparison with previous work

The main differences between the current algorithm and the previous work done in [17] are listed in Table 4.2.

Table 4.2 Comparison of proposed algorithm with previous work [17]

Parameter	Previous Work [17]	Current work
Media	Images compressed by JPEG	Videos compressed by H.264/AVC Main Profile
Features in transform/spatial domain	Transform	Transform
Feature Extraction	Luminance only	Luminance only
Type of feature data for every block	DC only	DC + AC
Number of feature coefficients	Fixed (396 for QCIF) (See Figure 4.6)	Variable (See Figure 4.6)
Digital Signature	1 per 8x8 block	1 per coded video sequence
Detect spatial manipulations	Yes (See Figures 4.8, 4.9, 4.10 and 4.11)	Yes (See Figures 4.8, 4.9, 4.10 and 4.11)
Detect sender forgery	Yes	Yes
Detect tampered location	Yes (Block level)	Yes (Frame level)
Buffer size for coefficients	Fixed	Variable

Self recovery for tampered locations	Yes	No
Robust to quantization	Yes	No
Encryption of digital signature	No	Yes
Encryption passes required for generating digital signature	Fixed (396/image for QCIF)	Fixed (1 for every picture and 2 for the whole video sequence)
Hash calculation	None	1/picture and 1 for the whole video sequence
Computational Complexity	High	Medium

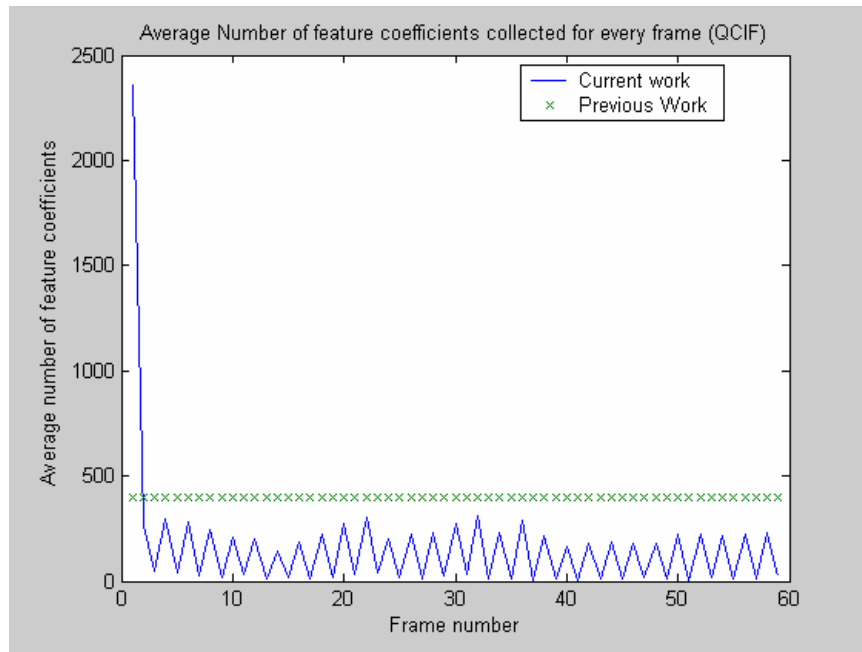


Figure 4.6 Number of feature coefficients collected for every frame

The following figures show the different types of spatial manipulations such as frame cropping, rotation, inversion, etc.



Figure 4.7 Original Foreman frame



Figure 4.8 Cropped frame in Foreman sequence

The cropped figure was obtained by cropping 20 rows from the top and 26 rows from the bottom of the figure. This attack is detected by both the previous work as well as our algorithm.



Figure 4.9 Rotated frame in Foreman sequence

The original frame is rotated by 180 degrees to obtain the above frame. Our current algorithm is robust to this attack as is the case for [17].



Figure 4.10 Inverted frame in Foreman sequence

The inverted frame in case of 8 bit images is the difference between 255 and the current value of every pixel in the image. Inverting the frame leads to a change in the DC value and can be detected by both the current as well as the previous work.



Figure 4.11 Tampered frame in Foreman sequence

The bottom right of the frame is manipulated by removing the construction structure from the original frame. This type of attack is detected easily by both the works due to change in the DC and AC values of the modified blocks.

4.4 Results for multiple coded video sequences

The total number of frames is kept the same as for a single coded video sequence at fifty nine. The number of coded video sequences was set to six, an average of nine frames in a video sequence. The common parameters required for the DSA were

unchanged. There were a total of six signatures obtained, one for every coded video sequence. They are shown below in base 10 format.

First Signature

No of coefficients=4334

r = 258230994199886379618680381531392644375747268307
s = 602843151256385963511141755924782867386260766615

Second Signature

No of coefficients=3724

r = 258230994199886379618680381531392644375747268307
s = 909298954038809272308410772111508639025580732794

Third Signature

No of coefficients= 3896

r = 258230994199886379618680381531392644375747268307
s = 37131761566210379177239004151768998162822235640

Fourth Signature

No of coefficients=4561

r = 258230994199886379618680381531392644375747268307
s = 219871236647467187087134837911922129305756281586

Fifth signature

No of coefficients= 4020

r = 258230994199886379618680381531392644375747268307
s = 908124295867495640787368571873172959524293811336

Sixth signature

No of coefficients= 3860

r = 258230994199886379618680381531392644375747268307
s = 954693410128487399463060653092140043143216469688

4.4.1 Without video tampering or forgery

The decoder computes a signature based on the received video sequence. Since there are multiple video sequences involved, multiple signatures would have to be verified. For each video sequence the decoder verifies the signature and identifies the tampered locations or detects malicious user activity in the case of signature verification failure. The decoder's signature (v) for each of the video sequence in absence of tampering or signatory forgery is presented below.

```
v1 = 258230994199886379618680381531392644375747268307  
v2 = 258230994199886379618680381531392644375747268307  
v3 = 258230994199886379618680381531392644375747268307  
v4 = 258230994199886379618680381531392644375747268307  
v5 = 258230994199886379618680381531392644375747268307  
v6 = 258230994199886379618680381531392644375747268307
```

Here the index i in v_i indicates the signature number. The signatures match and hence the received video can be claimed to be from an authentic source with no tampered frames.

4.4.2 Locating tampered frames

Our proposed algorithm can identify the locations of tampered frames for each coded video sequence separately based on the encrypted hash outputs sent by the encoder. In this example frames 13 (B picture) and 30 (I picture) belonging to the second and fourth video sequence respectively are modified. Frame 13's coefficients are replaced by those of frame 11 and frame 30 is replaced by the previous I frame (picture number 20 of the 3rd video sequence). The decoder successfully verifies the signature for the first,

third, fifth and sixth video sequence. The decoded signatures (v) for the second and fourth video sequence are shown below in the same order.

$$v_2 = 766321293063485944746073327048840451936391521142$$

$$v_4 = 402004964420942097203746248359479943276321484948$$

These signatures do not match with their corresponding counterparts from the encoder. The next step taken by the decoder is to verify the hash outputs against the decrypted hash outputs from the encoder for each of the video sequences. For the second video sequence frame no. 13's hash outputs from the encoder and decoder are shown.

Encoder Hash=1264207254626377624026298464974576363754052487452

Decoder Hash=1205948196737049705496575512093566763134799073160

Since the hash outputs do not match, we can come to the conclusion that frame 13 has been tampered with and is the reason for the cause of authentication failure for the second coded video sequence. Similarly frame 30's encoder and decoder hash outputs are shown below.

Encoder Hash = 263664647879112316110830736118735317109027381491

Decoder Hash = 994353145971971509751583421994471357435786107776

As above the hash outputs do not bear any faint resemblance to each other and frame 30 is responsible for the authentication failure of the fourth video sequence. The proposed algorithm can also identify multiple tampered frames for each coded video sequence by comparing the hash outputs from the encoder and decoder.

4.4.3 Detecting malicious activity

The procedure for identifying fraudulent activity is the same as for a single coded video sequence. In this example the identification parameters for the third video sequences are tampered with. The third video sequence is signed using a different private key and then it is passed through the authentication process at the decoder using the original public key of the legal user. The modified private key is shown below in base 10 format.

Modified Private Key = 132308816105877193346324924405238358561797363453

The signature (r and s) obtained at the encoder are shown below.

r = 258230994199886379618680381531392644375747268307
s = 142326340467543968810784401557100723758620208742

The decoded signature (v) obtained is shown below in base 10 format.

v = 562076462702330475705116426621660862036694533173

As can be seen, the signatures do not match and hence the signature verification fails for the third video sequence. The decoder now matches hash of every frame in the third video sequence to the corresponding one sent by the encoder. Since none of the frames in the third video sequence were modified the hashes would all match. It can be thus inferred from the above result that the video authentication failure for the third video sequence has been due to malevolent intentions of the signatory.

4.4.4 Identifying malicious user and tampered locations simultaneously for multiple video sequences

As demonstrated for a single video sequence, it is possible to identify all the tampered frames in the case of multiple signatures where each signature is used as the authentication information for a single coded video sequence. However it is not possible to identify whether sender identification is also the cause for authentication failure when both of them occur simultaneously.

4.5 Summary

We have presented results for a single coded video sequence when the frames are tampered; video is forged by a malicious user, we have tested the algorithm to attacks such as frame reordering attack and showed the algorithm's robustness to quantization. Our algorithm is also robust to the DC attack. Similar results were also presented for multiple coded video sequences.

Chapter V presents conclusions and further studies pertaining to our proposed algorithm.

CHAPTER 5

CONCLUSIONS AND FURTHER STUDIES

5.1 Conclusions

We have proposed a hard authentication scheme in this thesis to verify the integrity and achieve sender identification for videos compressed with H.264/AVC main profile [8]. Every coded video sequence is verified independently by using the DC value and two AC values as the authentication data for every 4x4 luminance block except for Intra 16x16 modes where all the non-zero DC coefficients after Hadamard transform and 32 AC coefficients are used. This scheme is more suitable to video as the integrity of the whole sequence is verified rather than a single frame. The DSS is used for generating the digital signature and RSA is used for encryption of the digital signature. Additionally, the proposed algorithm can identify the tampered locations at frame level and detect sender forgery at the expense of increased authentication bits by calculating the hash of every frame and appending it to the signature.

The simulation results presented in Chapter 4 demonstrate that the algorithm can detect malicious temporal and spatial manipulations to the video and point out the frames where these manipulations have taken place. Also, they can point out the fraudulent intentions of an imposter when attempting to forge the video. The results also show that our algorithm is not robust to quantization due to the one-way nature of a hash function.

The chrominance coefficients are also not taken into account when designing the algorithm. This leads to cases where the signature is verified even though unlawful color manipulations have taken place. The frame hash outputs required in identifying the tampered locations demand extra authentication bits to be transmitted by the encoder thus increasing the bit rate of the coded video.

5.2 Further studies

The proposed algorithm can be extended to include chrominance coefficients apart from luminance frames wherein the DC and a few of the low frequency chrominance coefficients of every block can be used as the authentication criteria. Further study to make the scheme robust to quantization is required. Finally, the proposed scheme requires that the authentication bits be transmitted separately from the video. Watermarking [29] i.e. embedding the hash values produced by this algorithm in the video sequence can be an area of research.

APPENDIX A

DIGITAL SIGNATURE STANDARD

A.1 Introduction

The Digital Signature Standard (DSS) is a public key cryptographic system developed by the National Institute of Standards and Technology [26]. The need for it has been brought about by the intention of developing an efficient way to reduce costs and replace handwritten signatures with digital ones. The digital signature serves two purposes: (1) Identify and authenticate the creator of the message and (2) Verify the message integrity by making sure that the information has not been altered after it has been signed.

The DSS is split into two parts: (1) Secure Hash Standard (SHS) and (2) Digital Signature Algorithm (DSA). To calculate the signature the message to be authenticated is fed to the SHS which produces a 160 bit condensed representation of the message. This 160 bit message is acted upon by the DSA which also takes the user's private key into account and generates a digital signature consisting of two 160 bit numbers. Since there is a unique mapping between the private and public keys of a user, any party with access to the public key, the message to be authenticated and the digital signature can verify the signature using the techniques described in the DSA. Figure A.1 shows the digital signature generation and verification process described in the DSS.

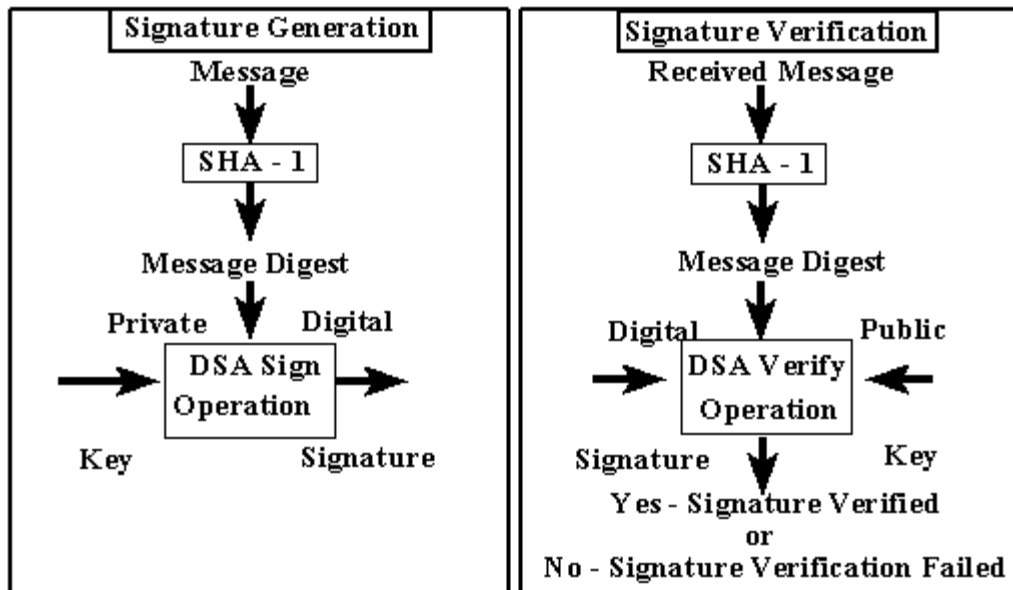


Figure A.1 Signature generation and verification process of DSS [26]

The security provided by a digital signature is also an important point to take into consideration. Some factors are the mathematical soundness of the algorithm, management and distribution of keys (private and public) and the implementation of the system in a given application [26]. The security provided by the DSA is the effort required to compute the discrete logarithm of a very large number. Also, the digital signature cannot be forged if the private parameters (such as the private key) required by the DSA are not available to the adversary. The following sections describe the SHS and DSA.

A.2 Secure Hash Standard [25]

The job of a hash function is to produce an irreversible condensed representation of the message. This condensed representation of the message is called a digest. Here irreversible refers to the property of a hash algorithm that the original message cannot be

recovered from the hash. The secure hash algorithm is called secure because it is computationally infeasible to find the message from a message digest, find two messages having the same digest and a modification in the original message results in a different digest with very high probability. The SHS uses the Secure Hash Algorithm (SHA-1) to produce the message digest. The SHA-1 produces a 160 bit fixed output for message length $< 2^{64}$ bits. The hash generation process is described below.

A.2.1 Representation of bit strings and integers

1) A hex digit belongs to the set (0, 1...9, A, B...F). Since there are 16 possible elements in the set, four bits will be required for unique identification of every element in the set. Therefore a hex digit may be represented by four bits.

2) A 32 bit sequence is called a word and a word is represented by eight hex digits.

3) Any integer between 0 and $2^{32}-1$ can be represented as a word where the least significant four bits of the integer are represented by the right most hex digit.

4) Block = 512 bit string. A block may be represented by a succession of sixteen words where each word is a 32 bit string.

A.2.2 Word operations

Let X and Y both represent words. Then

1) Bitwise logical word operations:

$X \wedge Y$ = bitwise logical “and” of X and Y

$X \vee Y$ = bitwise logical “inclusive-or” of X and Y

$X \text{ XOR } Y$ = bitwise logical “exclusive-or” of X and Y

$\sim X$ = bitwise logical “complement” of X

2) Let X and Y represent integers x and y respectively where $0 \leq (x, y) < 2^{32}$. Then X+Y can be defined as follows. Compute $z = (x+y) \bmod 2^{32}$ where $0 \leq z < 2^{32}$. z is converted to a word Z where Z is defined as X+Y.

3) Let $S^n(X)$ represent the circular left shift operation where n is an integer where $0 \leq n < 32$. Then $S^n(X) = (X \ll n) \text{ OR } (X \gg 32-n)$.

A.2.3 Message Padding

The SHA-1 which produces the message digest requires that the total length of the message (length of a message is the total number of bits in the message) be a multiple of 512 bits. The message is processed sequentially in blocks. Message padding converts the total length of the message to a multiple of 512. The padding process is described below. All padding is done on the right of the message bitstream. Let the length of the message be a 64 bit string represented as k.

a) A '1' is appended. Example: If the original message is 10100111 01100010 01000001, then this is padded to 10100111 01100010 01000001 1.

b) '0' 's are appended. The last 64 bits of the final 512 bit block are reserved for the length of the message. The total number of '0' 's to append is calculated such that the message is a block size of 512. Example: The original message length of the above bitstring is 24 bits. After appending a '1' the message size is now 25 bits. The last 64 bits are reserved for the length. Thus 423 '0' 's are appended. The new message in hex becomes

```

A7624180    00000000    00000000    00000000
00000000    00000000    00000000    00000000
00000000    00000000

```

c) k , the length of the message in bits is represented in a two word format. If $k < 2^{32}$ then the first word contains all zeroes. This two word representation of the length is appended to the message to get the final message over which the hash will be computed. Example: Here in this example k is 24. The two word representation in hex is 00000000 00000018.

Thus the final padded message becomes

```

A7624180    00000000    00000000    00000000
00000000    00000000    00000000    00000000
00000000    00000000    00000000    00000018

```

The padded message in this example contains exactly 16 words. The previous statement can be generalized for any message by noting that the final padded message will contain $16*n$ words where $n > 0$. The padded message is processed in sequence of n blocks M_1, M_2, \dots, M_n where M_1 represents the first bits in the message.

A.2.4 Secure Hash Functions

The following eighty functions f_0, f_1, \dots, f_{79} are used in SHA-1. Each function operates on three words X, Y and Z and produces a single word as output.

$$f_t(X, Y, Z) = (X \text{ AND } Y) \text{ OR } ((\text{NOT } X) \text{ AND } Z) \quad (0 = t = 19)$$

$$f_t(X, Y, Z) = X \text{ XOR } Y \text{ XOR } Z \quad (20 = t = 39)$$

$$f_t(X, Y, Z) = (X \text{ AND } Y) \text{ OR } (X \text{ AND } Z) \text{ OR } (Y \text{ AND } Z) \quad (40 = t = 59)$$

$$f_t(X, Y, Z) = X \text{ XOR } Y \text{ XOR } Z \quad (60 = t = 79)$$

A.2.5 Constants

SHA-1 employs eighty constant words in the hash process. These constants K_0, K_1, \dots, K_{79} are given as follows:

$K_t = 5A827999$ ($0 \leq t \leq 19$)

$K_t = 6ED9EBA1$ ($20 \leq t \leq 39$)

$K_t = 8F1BBCDC$ ($40 \leq t \leq 59$)

$K_t = CA62C1D6$ ($60 \leq t \leq 79$).

A.2.5 Message Digest Computation

The SHA-1 requires two five word buffers, an eighty word buffer and a single word buffer labeled TEMP. The first five word buffer is A, B, C, D and E. The second five word buffer is H_0, H_1, H_2, H_3 and H_4 . The eighty word buffer sequence is given as $W_0, W_1, W_2 \dots W_{79}$.

The sixteen word blocks $M_1, M_2, M_3 \dots M_n$ are processed in sequential order with each block requiring eighty steps. Before processing any of the blocks the five word H buffer are initialized in hex as follows:

$H_0 = 67452301$

$H_1 = EFC DAB89$

$H_2 = 98BADC FE$

$H_3 = 10325476$

$H_4 = C3D2E1F0$

For processing of each block M_i

- a) M_i is split into sixteen words $W_0, W_1, W_2 \dots W_{15}$ where W_0 represents the left-most word.
- b) For $t = 16$ to 79 let $W_t = S^1(W_{t-3} \text{ XOR } W_{t-8} \text{ XOR } W_{t-14} \text{ XOR } W_{t-16})$
- c) Let $A = H_0, B = H_1, C = H_2, D = H_3, E = H_4$.

d) For $t = 0$ to 79 do

$$\text{TEMP} = S^5(A) + f_t(B,C,D) + E + W_t + K_t;$$

$$E = D; D = C; C = S^{30}(B); B = A; A = \text{TEMP};$$

e) Let $H_0 = H_0 + A$, $H_1 = H_1 + B$, $H_2 = H_2 + C$, $H_3 = H_3 + D$, $H_4 = H_4 + E$.

After all the blocks in the padded message are processed the digest is given by the 160 bit buffer $H_0 H_1 H_2 H_3 H_4$. An alternative method of computing the digest when the required buffer space is not available is discussed in [25].

A.3 Digital Signature Standard [26]

The DSA specifies the methods used for the generation and verification of digital signatures. In the signature generation part the signatory signs the message with his/her private key and the public key is employed in the verification part. This calls for a method of associating the private and public keys of the signatory. The DSA is described below.

A.3.1 DSA Parameters

The following parameters are used in the DSA:

- a) $p =$ a prime modulus, where $2^{L-1} < p < 2^L$ for $512 = < L = < 1024$ and L a multiple of 64
- b) $q =$ a prime divisor of $p - 1$, where $2^{159} < q < 2^{160}$
- c) $g = h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < p - 1$ such that $h^{(p-1)/q} \bmod p > 1$ (g has order $q \bmod p$).
- d) $x =$ a randomly or pseudorandomly generated integer with $0 < x < q$
- e) $y = g^x \bmod p$

f) $k =$ a randomly or pseudorandomly generated integer with $0 < k < q$

p , q , and g parameters can be released to the public whereas x and k parameters are supposed to be kept secret by the signatory. A user's private key is x and public key is y . Public keys should be made available to the user for verifying the message from a trusted third party source. k is a random integer generated for every signature.

A.3.2 Signature Generation

The signature consists of two 160 bit numbers denoted by r and s . They are computed as follows:

$$r = (g^k \bmod p) \bmod q$$

$$s = (k^{-1}(\text{SHA}(M) + xr)) \bmod q \text{ where } k^{-1} \text{ is the multiplicative inverse of } k \text{ i.e. } (k^{-1} k) \bmod$$

$$q = 1 \text{ and } 0 < k^{-1} < q \text{ and SHA is the secure hash algorithm.}$$

If either r or s or both are zero then the signature should be recomputed but the probability of either one or both of them being zero are very less.

A.3.3 Signature Verification

Assume that the received message and signature is given by M' , r' and s' respectively. If either or both of the numbers (r' and s') representing the signature are zero then the signature verification part fails. If the above condition is met then further computations shown below are performed.

$$w = (s')^{-1} \bmod q$$

$$u1 = ((\text{SHA}(M')w) \bmod q$$

$$u2 = ((r')w) \bmod q$$

$$v = (((g)^{u1} (y)^{u2}) \bmod p) \bmod q$$

The signature is said to have been verified only if v and r' match. All other cases should be treated as verification failures. However it is not possible to point out whether the failure has been due to incorrect signature calculation or forgery attempt by an imposter.

APPENDIX B

H.264/AVC

B.1 Introduction

H.264/AVC is the latest international video coding standard released by the ITU-T Video Coding Experts Group (VCEG) and ISO/IEC 's Moving Picture Experts Group (MPEG) with the objective of enhanced compression efficiency over previous standards, ease in network representation of the video for interactive e.g. video telephony and non interactive applications such as broadcast, streaming [6], [7]. A full range of applications to which H.264/AVC would be suitable can be found in [6], [7], [5], [2]. The scope of the standardization is limited to the decoding process and is shown in Figure B.1.

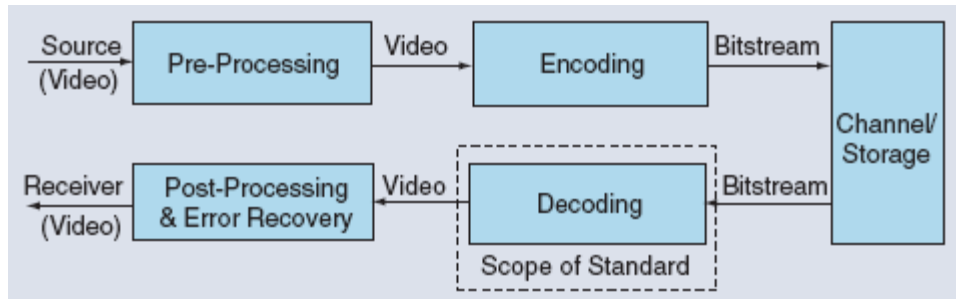


Figure B.1 Scope of standardization [6]

The standard is split into two parts: (1) Video Coding Layer (VCL), which focuses on removing redundancy from the video signal and (2) Network Abstraction Layer (NAL), whose objective is to format the compressed video signal produced by the VCL into a form suitable for transmission over different wired and wireless networks such as circuit switched network (PSTN) and packet switched networks (3G mobile networks, Internet). Figure B.2 shows the VCL and NAL in H.264/AVC.

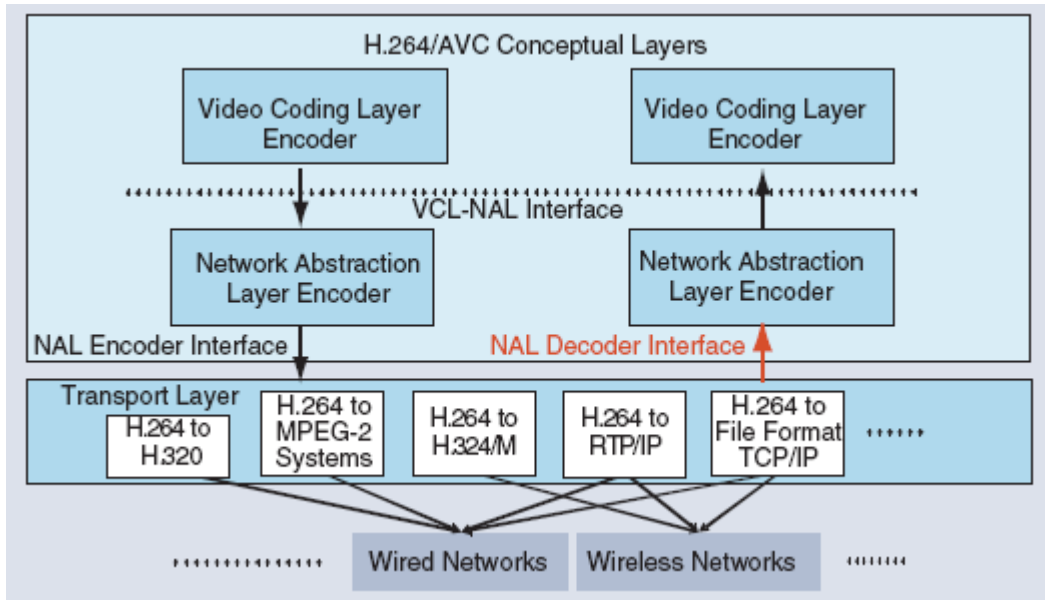


Figure B.2 VCL and NAL layers for H.264/AVC [6]

B.2 Network Abstraction Layer

The output of NAL is NAL units with each of the unit having an integer number of bytes. The first byte in a NAL unit indicates the information contained in the rest of the bytes. A series of NAL units is called a NAL stream. The NAL units are further distinguished as VCL and non VCL NAL units. VCL NAL units contain video picture values whereas non VCL NAL units referred to as parameter sets contain information regarding decoding of VCL NAL units [4]. Typically, the number of non VCL NAL units are very few compared to the number of VCL NAL units. Since parameter sets are required for decoding the video signal they are usually sent out of band reliable channel [7].

B.2.1 Parameter Sets

Parameter sets may be of two types: (1) Sequence parameter sets: - These are used in the decoding of a series of consecutive coded video pictures and (2) Picture parameter sets: - Required for decoding a single picture.

Every VCL data is embedded with a unique identifier that specifies which picture parameter set to use and in turn every picture parameter set contains an identification signifying the sequence parameter set to be used [5]. Figure B.3 shows the parameter set concept.

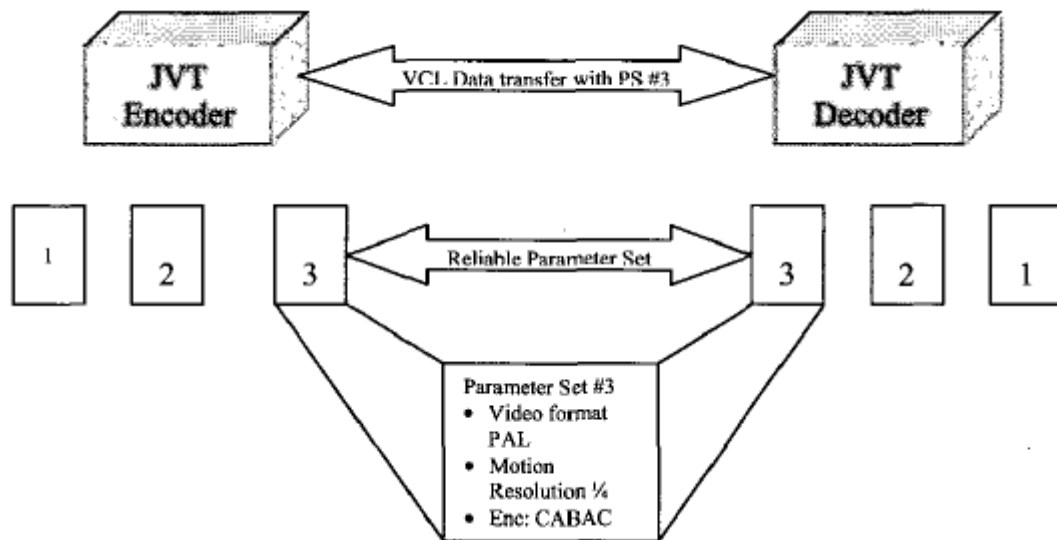


Figure B.3 Parameter set concept [7]

A coded video sequence in H.264/AVC may be defined as a series of consecutively coded pictures that refer to a single sequence parameter set. Every coded video sequence always starts with an Instantaneous Decoder Refresh (IDR) meaning an Intra picture, which is coded without reference to any other previously coded picture. On receiving an IDR, the decoder's short term and long-term buffers are cleared.

B.2.2 NAL Access unit

Reference [5] defines a NAL access unit as a series of NAL units which on being decoded give a single picture of the coded video sequence as output. Access unit delimiters are found at the start of the access unit. Figure B.4 shows the structure of an access unit. The delimiters are used for marking the boundaries of each access unit. Accompanying each access unit may be optional information such as a redundant picture that is used if the primary picture is corrupted. Generally redundant pictures are of low quality and thus occupy fewer bits than the primary coded picture.

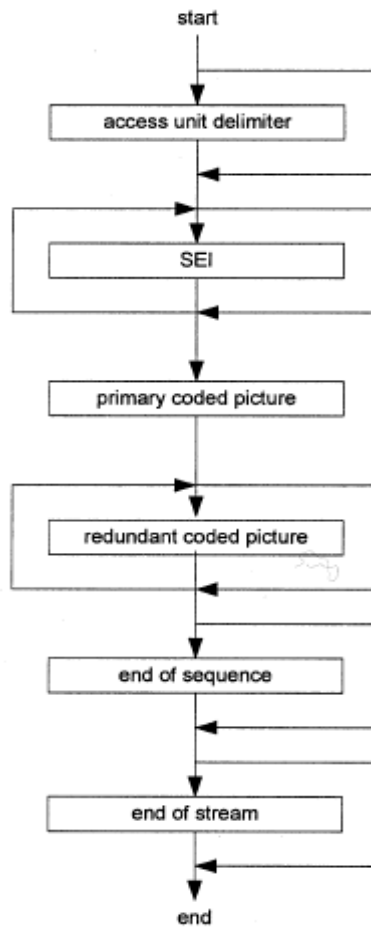


Figure B.4 Structure of an access unit [5]

The primary coded picture consists of a series of VCL NAL units, which on decoding give rise to the video picture.

B.2.2.1 Supplemental Enhancement Information (SEI)

SEI is defined in Annex D of the H.264/AVC standard [8]. It provides supplementary information in order to assist the decoder in decoding the video, display related issues or can carry other redundant information specified below. Conformance decoders have an option of discarding SEI messages except for verifying bitstream conformance. The different types of information that can be carried by SEI are listed in Table B.1.

Table B.1 SEI Messages [8]

SEI message payload type	Payload indicator
buffering_period	0
pic_timing	1
pan_scan_rect	2
filler_payload	3
user_data_registered_itu_t_t35	4
user_data_unregistered	5
recovery_point	6
dec_ref_pic_marking_repetition	7
spare_pic	8

scene_info	9
sub_seq_info	10
sub_seq_layer_characteristics	11
full_frame_freeze	12
full_frame_freeze_release	13
full_frame_snapshot	14
progressive_refinement_segment_start	15
progressive_refinement_segment_end	16
motion_constrained_slice_group_set	17
reserved_sei_message	18

It is interesting to note that the “*user_data_unregistered*” SEI message is used for carrying any data that does not fit in any one of the categories specified above. The payload length in bytes has to be specified before sending any data using this SEI message. This will allow the decoder to know the number of bytes to expect.

B.3 Video Coding Layer (VCL)

The purpose of the video coding layer is to remove temporal and spatial redundancies from the video. Like other previous standards H.264/AVC follows a block based coding approach wherein the video is transformed into Y, Cb, and Cr format. Though the encoder is not standardized, Figure B.5 shows a generic H.264/AVC encoder. Y pictures are divided into 16x16 non-overlapping macroblocks and chroma pictures are divided into 8x8 macroblocks. The human visual system is more sensitive to luminance

than chrominance. Thus chrominance components are subsampled by a factor of two. There are two modes available for prediction: (1) Intra mode and (2) Inter mode. In Intra mode all the macroblocks are predicted without reference to any other previously coded frame. Intra modes are used to remove drifting effects caused by the prediction model and to code pictures that have significant coding complexity than coding them as other types which do not result in significant gain. Every coded video sequence in H.264/AVC always starts with an Intra picture. Inter modes on the other hand allow motion compensated prediction from macroblocks that have been coded previously. This mode plays a significant part in reducing the bit rate of a video signal. Inter pictures are further divided into two types (a) P type and (b) B type. P types allow prediction only from previously occurring pictures whereas B types allow prediction from both past and future occurring pictures.

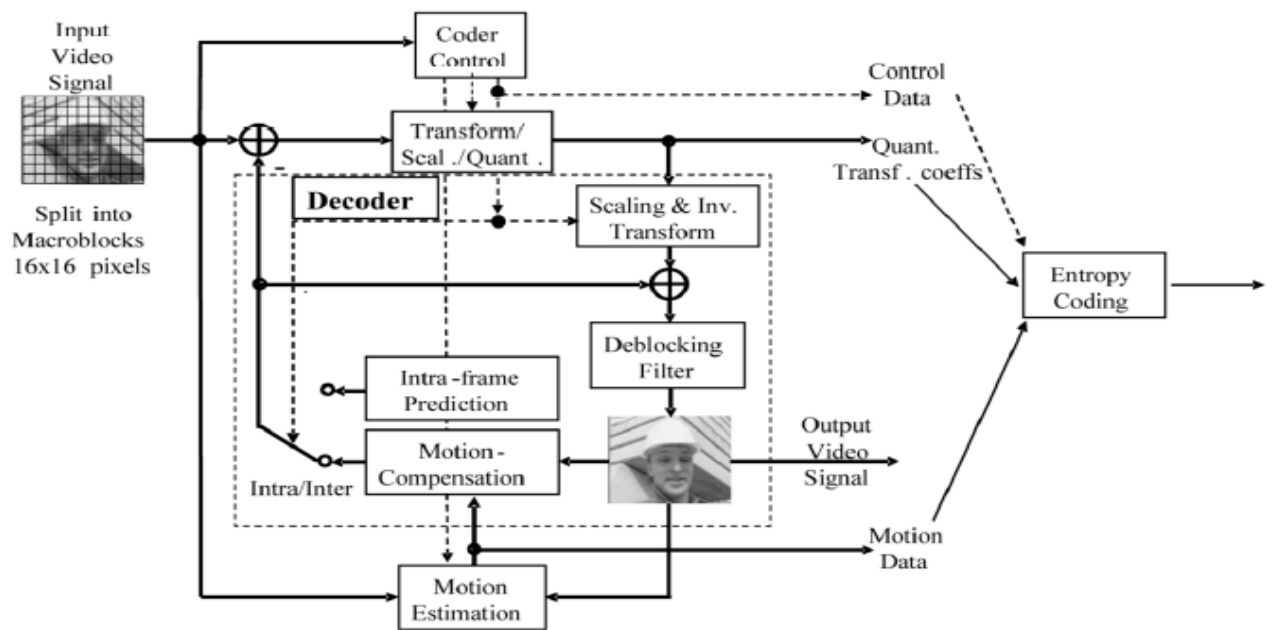


Figure B.5 A generic H.264/AVC encoder [5]

B.3.1 Integer 2D DCT

H.264/AVC employs a new 4x4 integer approximation of the DCT. The advantages of this transform [2] are that all operations can be carried out using integer arithmetic, prevention of inverse mismatch at the decoder and use of only adds and shifts in computing the core of the transform. Let $[X] = \{x_{00} \dots x_{33}\}$ be the input matrix and the output of the 2D-integer DCT be given by $[Y]$. Then the 2D integer DCT computation process is shown in Figure B.6.

$$Y = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & -2 \\ 1 & -1 & -1 & 1 \\ 1 & -2 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_{00} & x_{01} & x_{02} & x_{03} \\ x_{10} & x_{11} & x_{12} & x_{13} \\ x_{20} & x_{21} & x_{22} & x_{23} \\ x_{30} & x_{31} & x_{32} & x_{33} \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 & 1 \\ 1 & 1 & -1 & -2 \\ 1 & -1 & -1 & 2 \\ 1 & -2 & 1 & -1 \end{bmatrix}$$

Figure B.6 Integer 2D-DCT calculation of $[X]$ [7]

This transform is used in the prediction of both Intra and Inter modes.

B.3.2 Intra prediction modes

In Intra mode coding the prediction is obtained from the previously decoded parts of the same picture. A change compared to the previous standard is that the previously decoded blocks of the same picture need not be restricted to an I type picture. This prediction is subtracted from the current block and transformed to compact the energy into the low frequency coefficients. There are four possible types of Intra prediction modes available which are: (1) Intra 4x4 (2) Intra 16x16 (3) Constrained Intra and (4) I_PCM.

There are a total of nine prediction modes for Intra 4x4. Three of the prediction modes are shown below in Figure B.7.

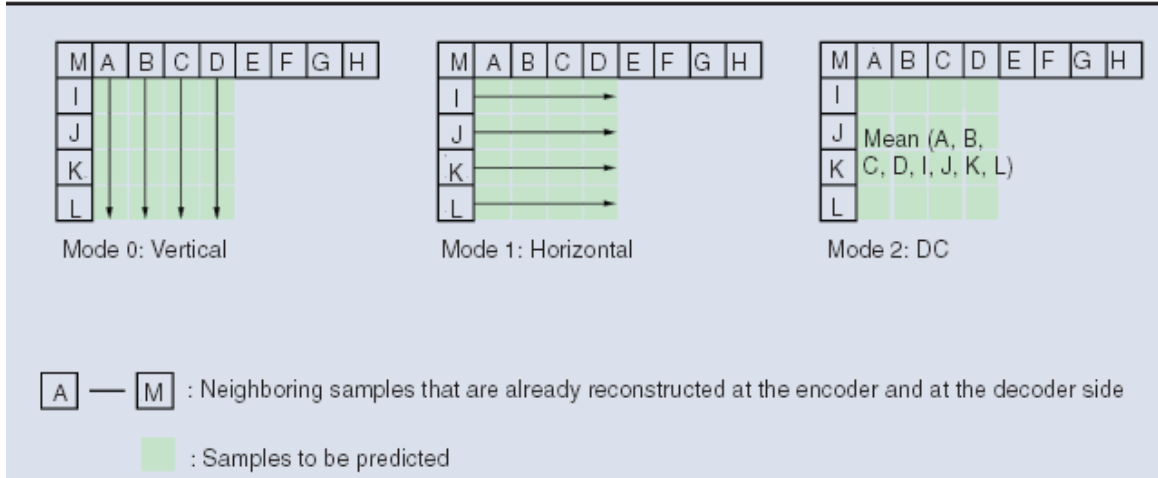


Figure B.7 Three of the nine possible Intra 4x4 prediction modes [6]

There is another Intra prediction method employed for coding smooth areas of a picture. This method is referred to as Intra 16x16 prediction method. In Intra 16x16 mode, the macroblock is divided into 16 blocks of size 4x4. The integer 2D (4x4) DCT is applied to each of these blocks. Figure B.8 shows this transformation process. The little left rectangle (00...33) indicates the DC value of every 4x4 transformed luma block. The numbers 0,1...15 represent the transformed luma blocks.

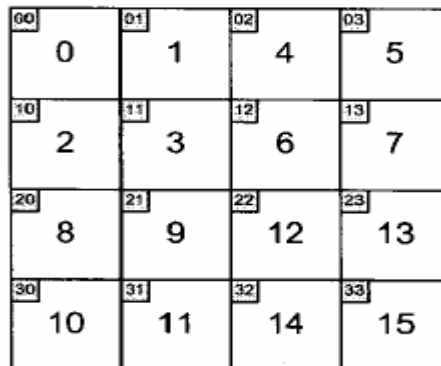


Figure B.8 Two dimensional transformed luma macroblock

The sixteen DC coefficients obtained are highly correlated and a second 4x4 transform (Hadamard transform) is utilized to decorrelate these DC coefficients. The need for a second transform is based on the property of a two-dimensional transform of smooth content that the reconstruction accuracy is proportional to the inverse of the one-dimensional size of the transform [5]. Since INTRA 16x16 is used for coding smooth areas the reconstruction error can be reduced by a factor of half. Let $[X_D] = \{x_{D00} \dots x_{D33}\}$ represent the DC coefficients of the transformed macroblock and $[Y_D] = \{y_{D00} \dots y_{D33}\}$ represent the output matrix after the Hadamard transform. The process for computing $[Y_D]$ is shown in Figure B.9.

$$Y_D = \left(\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_{D00} & x_{D01} & x_{D02} & x_{D03} \\ x_{D10} & x_{D11} & x_{D12} & x_{D13} \\ x_{D20} & x_{D21} & x_{D22} & x_{D23} \\ x_{D30} & x_{D31} & x_{D32} & x_{D33} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \right) / 2$$

Figure B.9 Hadamard transform for Intra 16x16 luma macroblock [7]

For chroma macroblocks, a 2x2 Hadamard transform is applied for coding the four DC coefficients obtained after integer DCT of each of the chroma blocks.

In Constrained Intra prediction mode the prediction is formed from only Intra coded parts of the picture. This is useful for stopping drift errors and a periodic picture of this type would result in better quality but this is at the expense of an increase in the number of bits required to represent the picture. I_PCM mode simply skips the prediction and transform coding process and encodes the pixel values directly [5].

B.4 Changes over previous standards

Other major changes as compared to previous standards are as follows.

The different block sizes possible for motion compensation are 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4. The motion vector resolution is $\frac{1}{4}$ pel. This results in a better prediction. The deblocking filter which helps in reducing block based artifacts is now mandatory. There are also a couple of new entropy coding methods; Context Adaptive Binary Arithmetic Coding (CABAC) and Context Adaptive Variable Length Coding (CAVLC) which adapt (using a different codetable/symbol table) according to the statistics of the previously coded symbols.

In the previous standards one picture was allowed as a reference (past only) for a P picture and two reference pictures (one past and one future) were allowed for B frames. In H.264/AVC it is possible to have multiple reference frames for each of the picture types. The reference software (JM version 7.5C) [9] allows up to 5 frames for both the picture types. One more notable change is that B frames can be used as reference frames. A picture is divided into slice groups, each slice group containing one or more slices. Unless otherwise specified a slice group consists of macroblocks in raster scan order. Flexible Macroblock Ordering (FMO) allows arbitrary macroblock grouping. Figure B.10 shows an example of slice groups and Figure 11 shows 2 examples of FMO [6].

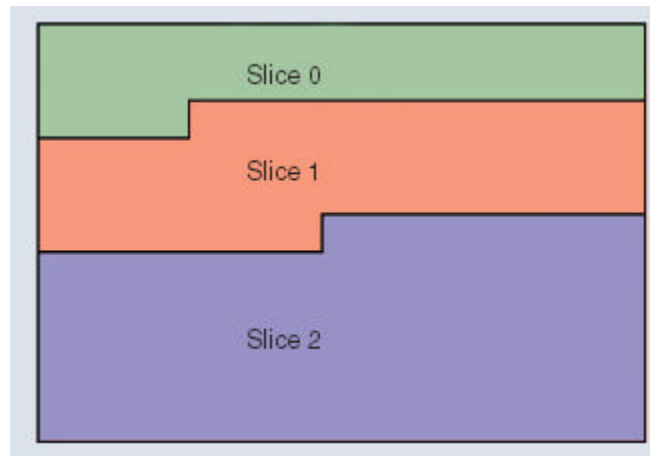


Figure B.10 Slices in an image [6]

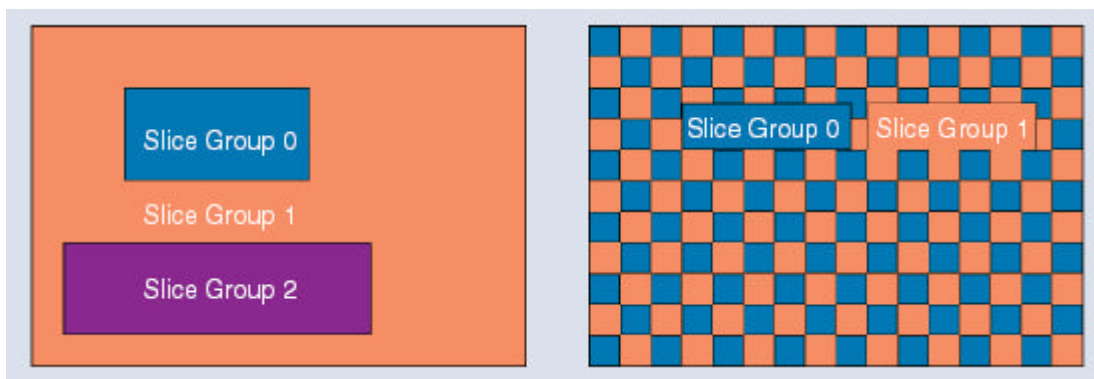


Figure B.11 Division of an image into arbitrary slice groups using FMO [6]

H.264/AVC allows the sequence and picture parameter sets to be sent separately using an out of band reliable channel. This can be used to increase error resilience by ensuring that parameter sets are given more protection than the VCL NAL units as the parameter sets contain information used in decoding pictures. This unequal information protection is known as data partitioning. A complete list of changes can be found in [5] [6], [7], [8].

The standard specifies a set of three profiles based on the application. These profiles are the Baseline, Main and Extended. Baseline profile's applications include

videotelephony, videoconferencing and wireless communications. Main profile is suited for television broadcasting and video storage while Extended profile may be used for streaming applications [6]. The different tools associated with each profile are shown in Figure B.12.

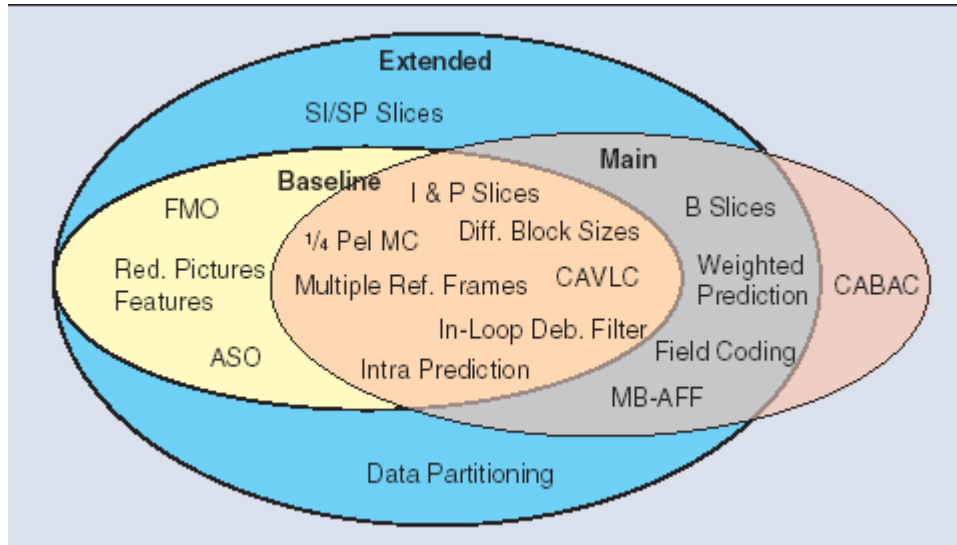


Figure B.12 H.264/AVC profiles [6]

The bit rate savings of H.264/AVC over other video coding standards when coded in Main profile is presented in Figure B.13 and when coded in Baseline Profile is presented in Figure B.14.

Coder	Average Bit Rate Savings Relative To:		
	MPEG-4 ASP	H.263 HLP	MPEG-2
H.264/AVC MP	37.44%	47.58%	63.57%
MPEG-4 ASP	-	16.65%	42.95%
H.263 HLP	-	-	30.61%

Figure B.13 Bit rate savings achieved by H.264/AVC Main profile [6]

Coder	Average Bit Rate Savings Relative To:		
	H.263 CHC	MPEG-4 SP	H.263 Base
H.264/AVC BP	27.69%	29.37%	40.59%
H.263 CHC	–	2.04%	17.63%
MPEG-4 SP	–	–	15.69%

Figure B.14 Bit rate savings achieved by H.264/AVC Baseline profile [6]

The acronyms used in the above two figures are

MP: Main Profile

ASP: Advanced Simple Profile

HLP: High Latency Profile

CHC: Conversational high profile

SP: Simple Profile

It has been observed through empirical results that intraprediction, multiple block size ME/MC, multiple picture weighted prediction, CABAC, B frames, Rate Distortion by Lagrangian optimization, Hadamard transform and Deblocking filter are the main causes of increase in complexity [6].

APPENDIX C

VIDEO AND SAMPLING FORMATS

C.1 Video Formats

The video frame may be displayed with different resolutions and dimensions depending on the application. For example in videoconferencing and video telephony small picture size that results in a small bandwidth is preferred whereas for broadcast quality video higher resolution that provide image detail may be preferred [3]. The different image formats are listed in Table C.1.

Table C.1 Common Video Formats [1], [11]

Video/Image Format	Resolution (no_of_pixels_per_line x no_of_lines_per_frame)
CCIR – 601 (NTSC)	720 x 625
CCIR – 601 (PAL)	720 x 525
Digital TV (NTSC)	720 x 576
Digital TV (PAL)	720 x 480
SIF (NTSC)	360 x 240
SIF (PAL)	360 x 288
16 CIF	1408 x 1152
9 CIF	1056 x 864
4 CIF	704 x 576
CIF	360 (or 352) x 288
QCIF	176 x 144
SQCIF	128 x 96

C.2 Image Sampling Formats

In digital video every frame is usually converted from RGB to YCbCr format. In this mode the human visual system is less sensitive to chrominance than to luminance [2]. A variety of formats each specifying the resolution to be used for Y, Cb and Cr is developed based on the previous fact.

C.2.1 4:4:4 Sampling

In 4:4:4 format, the three components (Y, Cb and Cr) have the same horizontal and vertical resolution. The numbers 4:4:4 indicate the relative sampling of each component in the horizontal direction [2] i.e. for every 4 luma samples in the horizontal directions there are 4 Cb and 4 Cr samples. Figure C.1 shows the 4:4:4 sampling format.

C.2.2 4:2:2 Sampling

In 4:2:2 sampling there are 2Cb and 2 Cr samples for 4 Y components in each line. Thus the horizontal resolution of Cb and Cr is reduced by half. The vertical resolution remains the same. Figure C.1 shows the 4:2:2 sampling format. This format is used in high quality color reproduction.

C.2.3 4:2:0 Sampling

This format signifies that the horizontal and vertical resolution of Cb and Cr are each reduced by half with respect to Y. The numbers 4:2:0 do not signify anything though. It is used in low bit rate communication services. Figure C.1 shows this format.

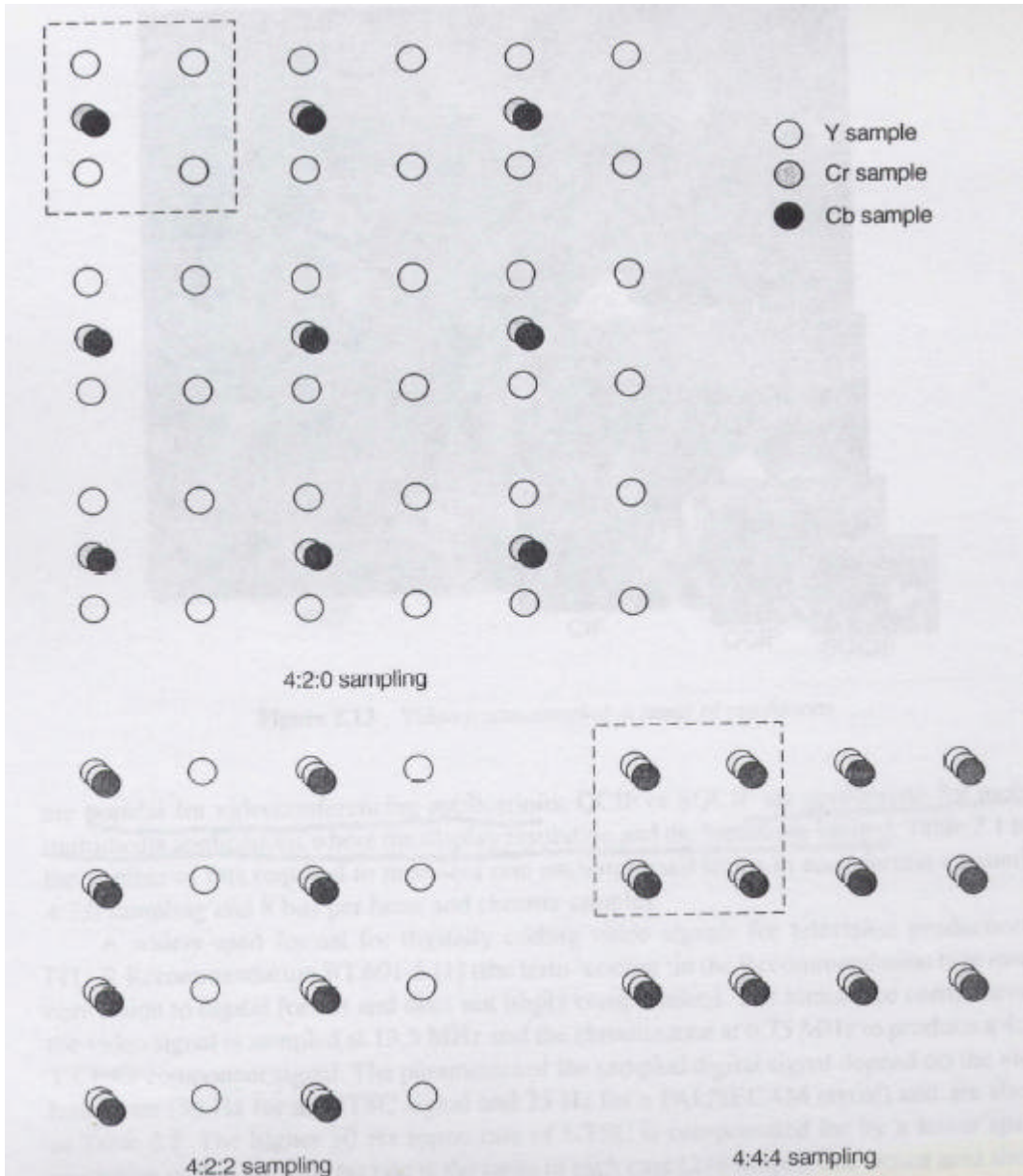


Figure C.1 Sampling formats [2]

APPENDIX D

THESIS SOFTWARE

D.1 Software

A big number C/C++ library was used in developing the software in this thesis. This software can be downloaded from [28]. Several cryptographic routines are provided by this software which eases development time. There are several example algorithms provided by [28] in C/C++ format.

Regarding the software developed in this thesis, the big number library has been integrated in JM 7.5. A list of files modified in JM 7.5 in implementing this algorithm can be found in “*changes-authentication.txt*”. The software has been attached with this thesis.

REFERENCES

- [1] K.R. Rao and J.J. Hwang, Techniques and Standards for Image, Video and Audio Coding. Upper Saddle River, NJ: Prentice Hall, 1996.
- [2] I.E.G. Richardson, H.264 and MPEG-4 Video Compression. Chichester, West Sussex: Wiley, 2003.
- [3] M. Ghanbari, Standard Codecs: Image Compression to Advanced Video Coding London, Institute of Electrical Engineers, 2003.
- [4] Singh, Simon., The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography. Anchor Books/Doubleday, 2000.
- [5] Special Issue on H.264/AVC
IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, pp. 557-725, Jul 2003.
- [6] Ostermann, J. et al “Video coding with H.264/AVC: tools, performance, and complexity”, *IEEE Circuits and Systems Magazine*, Vol. 4, Issue.1. pp. 7-28, 2004.
- [7] Tamhankar, A. and Rao, K.R. “An overview of H.264/MPEG4- Part 10”, *Video/Image Processing and Multimedia Communications*, 4th EURASIP, Zagreb, Croatia, 2003.
- [8] H.264/AVC International Standard
ITU-T Rec. H.264 | ISO/IEC 14496-10 version 3

- [9] H.264/AVC Reference Software <http://bs.hhi.de/~suehring/tml/>
- [10] YUV 4:2:0 Video Sequences
<http://trace.eas.asu.edu/yuv/yuv.html>
- [11] Know LEX, "CIF", <http://www.knowlex.org/lexikon/CIF.html> .
- [12] Srinivasan, S., "On piracy and privacy", *IEEE Computer*, Vol. 36, pp. 36-38. Jul 2003.
- [13] Memon, Nasir. and Wong, Wah-Ping., "Protecting digital media content", *Communications of the ACM*, vol. 41, pp. 35-43, Jul 1998.
- [14] Zhu, B.B., Swanson, M.D., and Tewfik, A.H. "When seeing isn't believing [multimedia authentication technologies]", *IEEE Signal Processing Magazine*, Vol.21, pp. 40- 49, Mar 2004.
- [15] Friedman, G.L., "The trustworthy digital camera: restoring credibility to the photographic image", *IEEE Trans. on Consumer Electronics*, Vol. 39, pp. 905-910, Nov 1993.
- [16] Schneider, M. and Shih-Fu Chang, "A robust content based digital signature for image authentication", In Proc of *IEEE International Conference on Image Processing*, Vol. 3, pp. 227-230, 16-19 Sep 1996.
- [17] Der-Chyuan Lou and Jiang-Lung Liu, "Fault resilient and compression tolerant digital signature for image authentication", *IEEE Trans. on Consumer Electronics*, Vol. 46, pp. 31-39, Feb 2000.

- [18] Dittmann, J., Steinmetz, A. and Steinmetz, R., “Content-based digital signature for motion pictures authentication and content-fragile watermarking”, In Proc of *IEEE International Conference on Multimedia Computing and Systems*, Vol. 2 pp. 209-213, July 1999.
- [19] Ching-Yung Lin and Shih-Fu Chang, “A robust image authentication method distinguishing JPEG compression from malicious manipulation”, *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 153-168, Feb 2001.
- [20] Chin-Yung Lin, “Watermarking and digital signature techniques for multimedia authentication and copyright protection”, PhD Thesis, Columbia University, 2001.
- [21] Peng Yin and Yu, H.H., “Semi-fragile watermarking system for MPEG video authentication”, In Proc of *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 4 pp. 3461-3464, 13-17 May 2002.
- [22] Grosbois, R., Gerbelot, P. and Ebrahimi, T., “Authentication and access control in the JPEG 2000 compressed domain”, In Proc of the SPIE 46th Annual Meeting, *Applications of Digital Image Processing XXIV*, San Diego, July 29-August 3rd 2001.
- [23] Grosbois, R. and Ebrahimi, T., “Secure JPEG 2000-JPSEC”, In Proc of *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 4 pp. 716-719, 6-10 April 2003.

- [24] Atrey, P.K., Wei-Qi Yan, Ee-Chien Chang and Kankanhalli, M.S., “A hierarchical signature scheme for robust video authentication using secret sharing”, In Proc of *IEEE 10th International Multimedia Modelling Conference*, pp. 330-337, 5-7 January 2004.
- [25] Secure Hash Standard
Federal Information Processing Standards Publication-180-1
<http://www.itl.nist.gov/fipspubs/fip180-1.htm>
- [26] Digital Signature Standard
Federal Information Processing Standards Publication-186
<http://www.itl.nist.gov/fipspubs/fip186.htm>
- [27] Rivest, R.L., Shamir, A. and Adleman, L.M., “A method for obtaining digital signatures and public-key cryptosystems”, *Communications of the ACM* vol. 21, p.p 120-126, Feb 1978.
- [28] Shamus Software Ltd, “Multiprecision integer and rational arithmetic C_{C++} library”, Public domain software, <http://indigo.ie/~mscott/>
- [29] Langelaar, G.C., Setyawan, I. and Lagendijk, R.L. “Watermarking digital image and video data. A state-of-the-art overview”, *IEEE Signal Processing Magazine*, Vol.17, pp. 20-46, Sep 2000.

BIOGRAPHICAL INFORMATION

Nandakishore Ramaswamy received the Bachelors degree in Electronics and Telecommunication degree from the University of Pune, India in 2001 and a M.S. degree in Electrical Engineering from the University of Texas at Arlington in 2004. His research interests are in video codec design and embedded systems. He is a student member of ACM and IEEE.