INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Esquema de cifrado para la transmisión de video en sistema operativo LINUX

Tesis que presenta

Ing. Mónica García Cortés

Que para obtener el grado de

Maestro en Ciencias en Ingeniería de Cómputo con Opción en Sistemas Digitales

Directores de Tesis

Dr. Moisés Salinas Rosales.
Dr. Raúl Acosta Bermejo.

México D.F. Julio 2014

SIP-14 bis

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

ACTA DE REVISIÓN DE TESIS

En la Ciudad de México, D.F. siendo las 11:00 horas del día 9 del mes de junio de 2014 se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

Centro de Investigación en Computación

para examinar la tesis titulada:

"Esquema de cifrado para la transmisión de video en sistema operativo LINUX"

Presentada por el alumno(a):

García	**Cortés**	**Mónica**
Apellido paterno	Apellido materno	Nombre(s)

Con registro: | **A** | **1** | **2** | **0** | **4** | **7** | **5** |

aspirante de: **MAESTRÍA EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON OPCIÓN EN SISTEMAS DIGITALES**

Después de intercambiar opiniones los miembros de la Comisión manifestaron ***APROBAR LA TESIS***, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

LA COMISIÓN REVISORA

Directores de Tesis

Dr. Moisés Salinas Rosales

Dr. Raúl Acosta Bermejo

Dr. Ponciano Jorge Escamilla Ambrosio

M. en C. Sergio Sandoval Reyes

Dr. Eleazar Aguirre Anaya

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas

INSTITUTO POLITECNICO NACIONAL
CENTRO DE INVESTIGACION EN COMPUTACION
DIRECCION

INSTITUTO POLITÉCNICO NACIONAL

SECRETARÍA DE INVESTIGACIÓN Y POSGRADO

CARTA CESIÓN DE DERECHOS

En la Ciudad de México, D.F. el día 9 del mes Junio del año 2014, el (la) que suscribe García Cortés Mónica alumno (a) del Programa de Maestría en Ciencias en Ingeniería de Cómputo con opción en Sistemas Digitales con número de registro A120475, adscrito al Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Moisés Salinas Rosales y Dr. Raúl Acosta Bermejo y cede los derechos del trabajo intitulado Esquema de cifrado para la transmisión de video en sistema operativo LINUX, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección imonikx13@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Mónica García Cortés
Nombre y firma

Resumen

Aplicaciones multimedia, como lo es la transmisión de video sobre las redes de comunicaciones, gracias a la integración de cámaras de vídeo en dispositivos móviles, se han hecho cada vez mas comunes. Estos dispositivos cuentan con considerables capacidades de procesamiento por lo que se facilita el acceso a Internet. Las comunicaciones de video a través de la red están basadas en la suite de protocolos TCP/IP por lo cual heredan sus vulnerabilidades y limitantes, lo que hace posible la intercepción y escucha no autorizada, esto nos lleva a la necesidad de proteger los datos del video.

Con el fin de proteger los datos del video en una transmisión, se diseño un esquema para el Sistema Operativo Linux para la transmisión de video con servicio de confidencialidad, utilizando el algoritmo de cifrado simétrico AES (*Advanced Encryption Standard*). Para ello se tomo en cuenta la estructura del video y se decidió cifrar a partir de los macrobloques del video. Con fines de investigación se utilizo el códec H.263-1998 (RFC 4629) durante el proyecto, debido a que es uno de los codecs mas utilizados en la transmisión de video.

Se trabajo con la plataforma de desarrollo BeagleBoard-xM, como dispositivo móvil, que es un sistema embebido basado en el procesador DM3730 de TexasInstruments. Este procesador a su vez contiene integrados un procesador de propósito general (ARM Cortex-A8) y un procesador digital de señal (DSP C64P). Debido a la alta carga computacional de la codificación de vídeos en la transmisión y la escasa optimización del ARM para procesamiento de datos, se propuso llevar a cabo la ejecución de la aplicación en el ARM y encargar la tarea de generación de llaves con las que se va a cifrar el video al DSP. Lo anterior con la finalidad de reducir el consumo de energía y por tanto aumentar la vida útil del sistema embebido sobre el cual se ejecutará la aplicación desarrollada.

Los resultados obtenidos en el trabajo, demostraron que el esquema de cifrado en términos de desempeño, al utilizar al DSP, mejoro la eficiencia en el uso de CPU. En términos de seguridad, los resultados de percepción del video cifrado fueron similares a los resultados de trabajos relacionados. El esquema de cifrado cumple también con la característica de no aumentar datos para el cifrado y así la transmisión de video tenga retrasos no perceptibles al usuario.

Abstract

Multimedia applications such as the video transmission over communication networks have increased their popularity since mobile devices provide access to the Internet and have significant processing capabilities, because the video communications through the network are based on the suite of TCP/IP protocols inherit their vulnerabilities and limitations , making possible to intercept and eavesdropping.

In order to protect the people's information, it was looked to design a scheme for the Linux Operating System for transmitting video with Confidentiality service using symmetric encryption algorithm as AES (*Advanced Encryption Standard*). Taking account the structure of the video and decided to encrypt the macroblocks from the video, for research purposes, the codec H.263-1998 (RFC 4629) was used during the project , because it is one of the more codecs used in video transmission.

As a portable device, it will work with the development platform BeagleBoard-xM an embedded system based on DM3730 processor. This processor itself contains a embedded general purpose processor (ARM Cortex-A8) and a digital signal processor (DSP C64P).

Due to the high computational load of encoding videos in its transmission and low ARM optimization for data processing , it is proposed to carry out the execution of the application on the ARM and commission the task of generating the keys with which will encrypt the video in the DSP, in order to reduce power consumption and increase the lifetime of the embedded system, on which the developed application will run.

The results obtained in the study showed that the encryption scheme in terms of performance, using the DSP, improved the efficiency of CPU. In terms of security , the encryption results of video perception was similar to the results of related work. The encryption scheme also meets the characteristic of not increasing data for encryption and transmission of video and have no noticeable delay to the user .

Agradecimientos

En primer lugar quiero agradecer a mis padres Alejandro García y Enedina Cortés por su apoyo incondicional y su comprensión. Gracias por el amor que me han brindado, sus consejos y sus palabras de aliento, ya que sin todo eso no hubiera logrado este objetivo.

A mi hermana Rocío, gracias por motivarme a escalar un peldaño más en mi preparación profesional, sin ti no lo hubiera logrado, eres un gran ejemplo para mi.

Gracias a mis directores de tesis, el Dr. Moisés Salinas Rosales y el Dr. Raúl Acosta Bermejo por compartirme su conocimiento y su tiempo para la realización de este trabajo.

Al Instituto Politécnico Nacional por haberme proporcionado las herramientas necesarias para enfrentarme a las adversidades que se presenten día a día en la vida profesional.

Al Centro de Investigación en Computación por otorgarme la oportunidad de obtener el grado de Maestría en Ciencias en una institución competente a nivel internacional. Y a CONACYT, por otorgarme apoyo económico para que pudiera enfocarme en la realización de la investigación.

A mis amigos y compañeros gracias por haberme brindado su amistad durante este tiempo, por su apoyo y sus consejos.

Gracias.

Índice general

Índice de figuras

Índice de tablas

Glosario

Frame Es una imágen particular dentro de una sucesión de imágenes que componen un video. La continua sucesión de estos frames producen a la vista la sensación de movimiento, fenómeno dado por las pequeñas diferencias que hay entre cada uno de ellos.

Frame B (Imágen Bi-predicted/Bi-directional)Las imágenes B requieren decodificación previa de otras imágenes de la secuencia para ser decodificadas correctamente. Pueden contener tanto datos de imagen cómo vectores de desplazamiento, o también combinaciones de los dos elementos. Incluyen algunos modos de predicción que obtienen la predicción de una región en movimiento (por ejemplo, un macrobloque o una región de área menor) llevando a cabo un promediado de las predicciones obtenidas usando dos regiones de referencia previamente decodificadas.

Frame I (Imágen Intra) Es una imagen codificada sin referencia a ninguna imagen anterior, sino referida exclusivamente a ella misma.Típicamente, requieren mayor número de bits para su codificación que el otro tipo de imágenes (B,P).Éstas imágenes son frecuentemente utilizadas como punto de referencia para la decodifcación de otras imágenes.

Frame P (Imágen Predicted), Realizan la estimación de movimiento y deciden cuál es la forma más eficiente de codificar un macrobloque en función de los resultados obtenidos.La codificación de las imágenes P es algo más compleja que las imágenes intra(I).

MacroBloque Son la unidad básica sobre la cual se realiza la compensación de movimiento. Estos bloques contienen datos de luminancia y crominancia, posteriormente, en el momento de codificación, a cada uno de éstos bloques se le aplicará la Transformada Discreta del Coseno (DCT) y se obtendrán unos coeficientes cuantificados uniformemente listos para transmitir. La cabecera de los macrobloques proporciona información sobre el tipo de codificación usado en el macrobloque, la escala del cuantificador y los vectores de movimiento, y más valores que podemos ver en el siguiente apartado.

Video es la tecnología de la grabación, procesamiento, almacenamiento, transmisión de imágenes y reconstrucción por medios electrónicos digitales o analógicos de una secuencia de imágenes que representan escenas en movimiento.

Glosario de Términos

AC En la transformada de coseno Discreta, el resto de coeficientes que no sea la función base ($K = 0$)

AES Advanced Encryption Standard

CIF Formato Intermedio Común, tiene 352 píxels por línea, 288 líneas, una relación de 12:11.

CODEC Codificador-Decodificador

CPU Unidad Central de Procesamiento

DC En la transformada de coseno Discreta, es el coeficiente que pondera la función base ($K = 0$) constante

DCT Transformada de Coseno Discreta

DES Data Encryption Standard

DFD Dsp For Dummies

DSP Procesador Digital de Señales

ENG Electronic News Gathering

GOB Group of Blocks

GOP Group Of Pictures

GPP Procesador de Propósito General

ISDN Red Digital de Servicios Integrados: Red que procede por evolución de la Red Digital Integrada (RDI) y que facilita conexiones digitales extremo a extremo para proporcionar una amplia gama de servicios, tanto de voz como de otros tipos, y a la que los usuarios acceden a través de un conjunto de interfaces normalizados.

ISO International Standards Organization

ITU-T Sector de Normalización de las Telecomunicaciones de la UIT

MPEG Moving Pictures Experts Group

NIST National Institute of Standards and Technology

PCF Picture Clock Frequency

PSTN Red Telefónica Pública conmutada

UIT International Telecommunication Union

Capítulo 1

Diseño de la Investigación

1.1. Antecedentes

El video ha sido un medio importante para las comunicaciones y el entretenimiento durante décadas, inicialmente el video era capturado y transmitido en forma analógica, pero después con el desarrollo de los circuitos digitales se llevó a la digitalización de vídeo, y el video digital dio pie a la compresión y comunicación de video. La compresión de vídeo se convirtió en un área importante de investigación en la décadas de los 80's y 90's y permitió implementar una gran variedad de aplicaciones, incluyendo el almacenamiento de vídeo en DVD, transmisión de video a través de cable digital, satélite y televisión digital terrestre, videoconferencia y videoteléfonia a través de redes de conmutación de circuitos. El crecimiento y la popularidad de Internet fue la motivación de la comunicación de video a mediados de los años 90 sobre redes de paquetes.

Esto dio paso a que el video pudiera ser transmitido en vivo y/o bajo demanda. La transmisión en vivo es cuando se reproduce en la computadora del usuario el audio y video de un evento a medida que éste se desarrolla en el sitio de origen. La transmisión bajo demanda es la reproducción de contenido pre-grabado, almacenado y disponible para consultarse en cualquier momento.

La transmisión en vivo trabaja a través de la transferencia simultánea de medios digitales: datos de vídeo y voz que se reciben en un *stream* o flujo de datos continuo y en *"vivo"*. Los datos se transmiten por un servidor, se reciben y exhiben en tiempo real por una aplicación cliente en la estación de trabajo del usuario que suele ser un reproductor multimedia. Estas aplicaciones pueden empezar a mostrar el streaming de video o audio en cuanto se hayan recibido y guardado suficientes datos en el búffer de la estación receptora, así como se muestra la Figura 1.1

Debido a las limitaciones propias de los dispositivos móviles, el streaming es la estrategia que se usa tanto para la recepción de vídeos como de televisión.

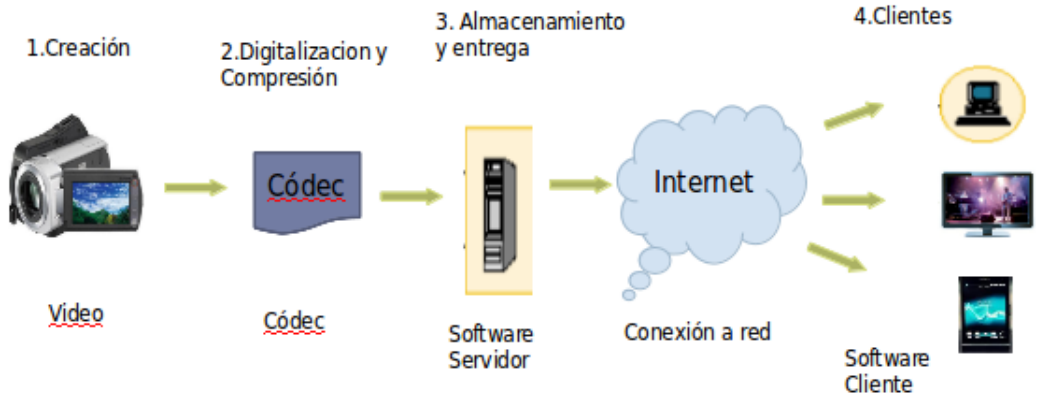


Figura 1.1: Videostreaming

Como consecuencia de que el video se puede transmitir por medio de Internet la seguridad de la información es necesaria porque la tecnología aplicada a la información genera riesgos. En términos generales, la información podría ser divulgada incorrectamente (es decir, su confidencialidad podría estar comprometida), modificada en forma inapropiada (es decir, su integridad podría verse comprometida), o se destruye o se pierde (es decir, su disponibilidad podría verse comprometida).

Actualmente el procesamiento digital dejo de ser realizado, en su mayoría por computadoras de uso personal, y paso a ser tarea de sistemas electrónicos integrados en dispositivos comunes de uso diario. Cabe mencionar entre ellos automóviles, teléfonos celulares y electrodomésticos. Este procesamiento otorga al dispositivo valores agregados ya que lo complementa con funciones como manejo eficiente de energía, seguridad, uso amigable y demás. Dichos sistemas electrónicos se conocen como sistemas embebidos. Por otro lado, los mismos factores indican más limitantes en el desarrollo de aplicaciones para sistemas embebidos que en el desarrollo de software convencional. Es por esta razón que se crean las plataformas de desarrollo, las cuales son sistemas embebidos de propósito general, con diversos periféricos y módulos incorporados. En ellas, los desarrolladores pueden concentrarse en probar algoritmos y aplicaciones sin tener que preocuparse por problemas de hardware.

1.2. Planteamiento del Problema

Las aplicaciones para transmitir video requieren que su flujo sea continuo siempre que el receptor lo solicite. Por lo tanto, no hay conceptos de playback de la secuencia de vídeo (es decir, que se ejecuta en directo). Un cifrado convencional está diseñado para datos genéricos, los cuales no soportan requerimientos específicos para aplicaciones de video, por lo tanto hay muchos desafíos para la seguridad en multimedia, tales como:

- El tamaño natural de los datos multimedia después de la compresión suele ser muy grande, incluso si se utilizan las mejores técnicas de compresión disponibles. Por ejemplo, el tamaño de un vídeo con compresión MPEG-1 de dos horas es de aproximadamente 1 GB. Por lo cual no se puede cifrar en su totalidad debido a que afectaría la percepción del usuario.
- Las futuras aplicaciones de multimedia deben ser ejecutadas en directo en los procesos, como el videostreaming.
- El rendimiento del procesamiento de flujos multimedia debe ser aceptable (es decir, limitado por un determinado valor de retardo).
- Las técnicas de cifrado deben tomar un tiempo corto y que solo presenten pequeña sobrecarga en comparación con las técnicas de compresión.

Además, el cifrado para aplicaciones multimedia en los dispositivos móviles cuenta con recursos limitados los cuales requieren de algoritmos eficientes con un mínimo procesamiento que permita el cifrado y la transmisión de video sin afectar la inteligibilidad del mismo.

Actualmente no hay un esquema de cifrado para la transmisión de video en directo que cumpla con las características anteriormente mencionadas y que se acople con los recursos de un dispositivo móvil.

1.3. Justificación

El uso de transmisión de video posibilita reuniones y trabajo cooperativo entre especialistas en diferentes áreas, a pesar de las posibles restricciones geográficas. Aunado a esto, algunas agencias gubernamentales, organismos de defensa nacional utilizan servicios de transmisión de video. Estas transmisiones están relacionadas con el uso de información privada, por lo que la confidencialidad de los datos que se transmite a través de la red puede verse comprometida, debido a que los datos se propagan en Internet y es posible el robo de datos. Sí un tercero tiene éxito en el robo de dicha información, entonces esa persona puede supervisar todas las actividades que puedan incluir información importante en ella. De ahí que se hace necesaria la implementación de sistemas criptográficos en video y la confidencialidad de los datos debe tenerse en cuenta.

El constante surgimiento de nuevas tecnologías de procesamiento de datos, como son los dispositivos móviles, impone también de manera continua, rigurosos requisitos hacia los procesos e infraestructura de seguridad. Debido a que la capacidad en el nivel de procesamiento es limitada, queda disponible la creación de esquemas cuya única función sea asegurar los datos del video transmitido. En razón de que actualmente no existen esquemas de cifrado para transmisión de video que se acoplen a dispositivos móviles, es necesario realizar la investigación en este tema

1.4. Objetivos

1.4.1. Objetivo General

Diseñar un esquema para la transmisión de video con servicio de confidencialidad para el S.O. LINUX, utilizando algoritmos de cifrado simétrico, para la protección contra la posible escucha de información por terceros.

1.4.2. Objetivos Particulares

- Analizar las estructuras de codificación de video para identificar una estrategia para la protección de la confidencialidad de éste.
- Analizar y proponer una plataforma de desarrollo que brinde similares prestaciones que un dispositivo móvil.
- Diseñar un esquema de cifrado que opere bajo el algoritmo AES para cifrar y descifrar información de video.
- Construir una plataforma el esquema de cifrado que eficientice el uso de los recursos de procesamiento disponibles en la plataforma seleccionada.
- Evaluar el esquema de cifrado en términos de desempeño y seguridad.

1.5. Alcance de la tesis

En este trabajo se desarrolla un componente de software para la transmisión de video con servicios de confidencialidad haciendo uso de un sistema embebido con características similares a las de un dispositivo móvil.

Este componente solo se enfocará en la confidencialidad de los datos del video que hayan sido codificados con el códec H.263+, debido a que este códec es uno de lo más empleados para la transmisión de video.

1.6. Contribuciones

Se desarrolló un esquema de cifrado que provee la confidencialidad de los datos que se obtienen en una transmisión de video, explorando la posibilidad de que sea eficiente al ejecutarlo en el DSP del sistema embebido y así reducir la carga de procesamiento en el GPP.

1.7. Método de investigación y desarrollo utilizado

El método de investigación del presente trabajo se estableció de la siguiente forma:

- Establecimiento de los problemas a resolver.
- Definición del objetivo principal y particulares.
- Investigación sobre los esquemas de cifrado de video que se han realizado anteriormente al igual que aplicaciones con estos.
- Diseño del esquema de cifrado a utilizar.
- Construcción de la cama de pruebas para el esquema.
- Análisis de los resultados obtenidos.
- Reporte de los resultados obtenidos del esquema de cifrado.

1.8. Organización del trabajo

Este proyecto se divide básicamente en cinco capítulos: Introducción, Estado del Arte, Marco de Referencia, Desarrollo de la Investigación y Pruebas o Resultados. A continuación se describe cada uno de ellos.

- En el primer capítulo se describe la presentación de la problemática contextualizada, la justificación del porqué es un problema, así como los objetivos propuestos, la estructura de la tesis y la posible solución al problema.
- En el segundo capítulo se dan a conocer los trabajos que se han realizado en el pasado con el fin de involucrar al lector en el tema de cifrado en la transmisión de video. Como primera parte se explican algunos algoritmos de cifrado en video que se han reportado en la literatura. Como segunda parte se mostraran algunos artículos que tratan de implementar los algoritmos anteriormente mencionados y los cuales pueden ser base de nuestra investigación.
- En el tercer capítulo se muestra el marco de referencia donde se explican los protocolos de comunicación utilizados para la transmisión de video. Se explican los elementos necesarios para que el lector entienda la composición del video, además se habla de los códecs mas utilizados y se explican las herramientas necesarias para asegurar la confidencialidad de los datos en una transmisión de video, como son los estándares de cifrado simétrico.
- En el cuarto capítulo se describe el desarrollo del proyecto, los pasos que se realizaron para resolver el problema planteado, la metodología utilizada, entre otros.

- En el quinto capítulo se presentan las pruebas y los resultados que se obtuvieron durante la realización del proyecto para así llevar a cabo la obtención de las conclusiones.
- En la bibliografía se lista una relación de referencias consultadas en este trabajo
- Por último, en los anexos se presenta como se realizó la instalación del sistema operativo LINUX, la instalación del DSP For Dummies y además las gráficas de los histogramas obtenidos de las 20 pruebas realizadas y las gráficas comparativas del uso de CPU.

Capítulo 2

Esquemas para el Cifrado de Video

Metodologías de cifrado se necesitan para poder proteger el video digital de los ataques durante su transmisión. Debido al enorme tamaño de videos digitales, que se transmiten por lo general en formatos comprimidos, como MPEG [10] o H.263, los algoritmos de cifrado para video digital generalmente trabajan en el dominio comprimido.

Se han propuesto varios esquemas de cifrado para asegurar la transmisión de vídeo, la mayoría de ellos trata de optimizar el proceso de cifrado con respecto a la velocidad de cifrado y proceso de visualización.

Antes de 1990 algunos métodos de codificación fueron estandarizados y la mayoría de los datos multimedia se transmitían en su forma cruda.

En este capítulo se describen los esquemas encontrados en la literatura que cumplen con el cifrado de video, concluyendo con una comparación entre ellos.

2.1. Esquema Naive

Es el método más directo para cifrar cada byte de la codificación de Moving Picture Experts Group (MPEG), utilizando estándares de cifrado como DES o AES. La idea del esquema Naive, para el tratamiento del bit-stream MPEG, es transformar el bit-stream como si fueran datos de texto y no utiliza ningún tipo de estructura especial [1].

El esquema Naive podría decirse que asegura el nivel de seguridad por completo en el flujo MPEG. Esto es porque ningún algoritmo eficaz puede romper los esquemas de cifrado AES especialmente, hasta ahora. Además este esquema cifra todo el video incluyendo todo tipo de imagen y los headers de MPEG.

Sin embargo, este algoritmo no es aplicable para la solución de vídeo grande, ya que es muy lento especialmente cuando se utiliza triple DES. Adicionalmente

debido a la operación de cifrado que aumenta el retraso y la sobrecarga de este esquema es inaceptable para el cifrado en una transmisión de vídeo.

2.2. Esquema de Permutación *Zig-Zag*

La principal idea de este algoritmo es que en lugar de mapear un bloque de 8x8 será en un vector de 1x64 en orden de *"zig-zag"*. El bloque de 8x8 es mapeado individualmente al vector 1x64 usando una lista de permutación al azar (llave secreta). Hay muchas maneras de producir la lista de permutación la cual tiene una distribución uniforme de todas las posibles permutaciones [26]. Este esquema consiste en los siguientes 3 pasos principales:

1. Genera una lista de 64 permutaciones.
2. Se completa el procedimiento de separación del bloque 8x8 y es cuantizado.
3. Se aplica la lista de permutación aleatoriamente para el bloque dividido y pasa el resultado al proceso de entropía de codificación.

Este método decrementa la velocidad de compresión porque la permutación al azar distorsiona la probabilidad de distribución de los coeficientes de la Transformada Discreta del Coseno (DCT) y hace que la tabla de Huffman sea menos utilizada.

El algoritmo de Permutación *"Zig-Zag"* no puede resistir al conocido ataque de texto plano. Asumiendo que anticipadamente conocemos ciertas partes del video, la llave secreta puede ser encontrada fácilmente comparado con el ya conocido ataque de texto plano de un *frame* cifrado del video. Este problema se puede enfrentar utilizando el método del *"lanzamiento de moneda"*. Esto es, si cae cruz se aplicará la permutación con una lista 1 (llave uno) al bloque y si cae cara se aplicará la permutación con una lista 2 (llave dos) al bloque. Este método es vulnerable solamente al ataque de cifrado de texto por que el coeficiente AC no cero tiene tendencia a acumularse en la parte superior izquierda de la esquina del bloque y esto hace que fácilmente un adversario pueda llegar a conocer la llave [19].

2.3. Esquema de cifrado de video (VEA) -Qiao y Nahrstedt

Qiao y Narhrstedt propusieron un nuevo método para el el cifrado del video que utiliza el comportamiento estadístico de la compresión de éste.

El análisis estadístico de la compresión del flujo de video muestra que su texto plano es diferente debido a que contiene características únicas a comparación del texto u otros datos.

Qiao y Narhrsted entre las características que analizaron en la compresión de video fueron la frecuencia de aparición de los valores de bytes y comparando entre diferentes flujos de video MPEG notaron que tienen una frecuencia similar de distribución. Otra característica fue la frecuencia de distribución de *"digrams"* entre 256x256 posibles pares, en este caso notaron que la mayor frecuencia en pares varia de *stream* a *stream*.

El esquema VEA actualmente consiste en dividir la entrada del flujo de video en pedazos (a1,a2,a3,...a2n-1,a2n). El pedazo es dividido en dos segmentos de datos, en listas impar (a1,a3,a5,...,a2n-1) y par (a2,a4,a6,...,a2n). Después de eso la llave cifradora se aplica en la lista de los pares: E(a2,a4,a6,...,a2n), donde E es la función de cifrado.

Al final el cifrado es una concatenación de la salida del esquema de cifrado XOR con la lista impar del flujo de video [15].

2.4. Esquema de cifrado de video (VEA) -Bharagava, Shi y Wang

Los autores propusieron un esquema de cifrado de video que utiliza una llave secreta aleatoriamente y cambia las señales de los coeficientes de la transformada discreta del coseno (DCT)en un *stream* de MPEG. Introducieron cuatro diferentes tipos de esquemas de cifrado de video:

1. Esquema I

   Esquema que utiliza la permutación de las claves de Huffman en los "*frames*"tipo I. Este método incorpora el cifrado y compresión en un solo paso.La parte secreta del esquema es una permutacion P la cual es usada para la permutación del estándar de MPEG con la lista de claves de Huffman.

   Para proteger la relación de compresión, la permutación P debe ser tal que permute las claves con el mismo número de bits.

   Este esquema es muy vulnerable a los ataques de texto plano y el ataque de solamente cifrar el texto. Si algunos *"frames"* del video se llegan a conocer por anticipado, entonces el atacante podría imaginarse y reconstruir la permutación secreta P por medio de comparar los *frames* ya conocidos con los *frames* cifrados [20].

2. Esquema II (VEA)

   Este esquema fue propuesto en el articulo "*A Fast MPEG Video Encryption Algorithm*" de Shi y Bharbaga [21], ya que los bloques de *frames* I llevan la

información más importante que contiene el video basta cifrar sólo la señal de los coeficientes DC de los bloques de *frames* I con sólo XORs con las señales de los coeficientes DC con una llave secreta.

El nivel de seguridad de este sistema depende de la longitud de la llave secreta. Si la llave es de un tamaño muy grande puede ser inviable y poco práctica.Por otra parte, si la llave es demasiado pequeña puede llegar a ser fácil de decifrar.

3. Esquema III (MVEA)

   En lugar de cifrar sólo la señal del coeficiente DC en los bloques de *frames* I, la señal de los valores diferenciales de los vectores de coeficientes DC y el movimiento en los *frames* P y B pueden ser cifrados con XORs con la llave secreta. Sin embargo, este tipo de mejora hace que la reproducción de vídeo sea más aleatoria. Al igual que el esquema II (VEA), el esquema III (MVEA) se basa en el tamaño de la clave secreta [22].

4. Esquema IV(RVEA)

   La diferencia entre el esquema IV (RVEA) y el esquema III (MVEA) es que el primero utiliza una criptografía tradicional de clave simétrica para cifrar las señales de coeficientes DCT y las señales de vectores de movimiento. El esquema acelera el proceso de cifrado al cifrar unicamente una parte de la señal en el flujo MPEG. Por lo tanto, en términos de seguridad, este esquema es mucho mejor que los tres esquemas esquema I, II (VEA) y III (MVEA). Además, se ahorra hasta el 90 porciento del tiempo de cálculo comparado con el esquema Naive [23].

2.5. AEGIS

El método AEGIS cifra solamente los *frames* I de todos los grupos de *frames* MPEG en una secuencia de vídeo MPEG, mientras que los *frames* B y P se quedan sin cifrar. Adicionalmente, para que el flujo de vídeo MPEG sea más seguro AEGIS también cifra la cabecera de la secuencia que contiene todos los parámetros de inicialización de decodificación como su anchura, altura, velocidad de *frames*, velocidad de bits y el tamaño del buffer [24].

El cifrado de la cabecera de secuencia hace que la identidad de secuencia del MPEG sea oculta y hace que el flujo de vídeo MPEG sea irreconocible.

Finalmente, AEGIS también cifra el código de los últimos 32 bits del flujo de MPEG, como resultado de ocultar aún más la identidad del flujo de bits de MPEG. AEGIS utiliza el motor de cifrado DES para el proceso de cifrado.

Iskender Agi y Gong Li [2] han demostrado que el cifrado de trama I puede no ser suficientemente seguro para algún tipo de vídeo. Su experimento demostró que es posible conocer algunas escenas de la decodificación de los *frames* P y B. Iskender Agi y Gong experimentaron con el cifrado de todos los bloques I en los *frames* P y B, además de cifrar todo el cuadro I el nivel de seguridad aumenta considerablemente. También han sugerido aumentar la frecuencia de las tramas I para mejorar la seguridad.

La desventaja es que tiene el inconveniente principal de aumentar la longitud de la cadena y en consecuencia el tiempo de cifrado. Por último, es importante señalar que este tipo de seguridad no es especialmente bueno para aplicaciones donde la seguridad es una de las principales prioridades (como el militar o teleconferencias de negocio), pero podría ser suficiente como una degradación de la calidad de los videos de entretenimiento, tales como la emisión de TV de pago.

2.6. Sign-Bit de los coeficientes de DCT

En este método se utiliza una llave secreta para transformar los bits de signo de los coeficientes DCT de los datos de vídeo MPEG. La llave secreta (k1, k2, k3, ..., K2M) se genera aleatoriamente con 2m de longitud, donde el número de llave y la longitud de la llave no está limitado.

Si los bits de signo de los coeficientes DC y AC son representadas por $S, (s1, s2, s3, ..., s2m)$, y luego los datos cifrados se representa como $Ek(Si) = bixorSi$ de longitud $2m$. La operación de cifrado cambia aleatoriamente los bits de signo de los coeficientes DCT.

La función de descifrado $Ek-1$ es la misma que la función de cifrado: $Ek(Ek) = S$. Para una llave de longitud m, un adversario tiene que tratar veces $2m$ con el fin de encontrar una llave.

En este esquema, varias llaves se pueden utilizar para mejorar la seguridad. Por ejemplo, en el esquema de 2 llaves, una llave es para los bloques de Y, y la otra para los bloques Cb y Cr. En el esquema de 3 llaves, una es para las tramas I, una para las tramas B, y una para tramas P.

2.7. Cifrado Byte

En este método se ha propuesto cifrar los bytes al azar en un flujo MPEG para su distribución gratuita, mientras que los bytes originales en las posiciones correspondientes se transmiten de forma cifrada para los usuarios legítimos. Esto es realmente equivalente a cifrar bytes en posiciones aleatorias.

Griswold encontró que el cifrado del 1 porciento de los datos son suficientes para hacer un video al menos un poco invisible. Sin embargo, la propuesta del criptoanálisis es totalmente insuficiente. Considerando el peor de los casos donde los datos de la cabecera del MPEG, sólo se cifra por casualidad utilizando este enfoque. Es bien sabido que los datos de cabecera se pueden reconstruir fácilmente siempre que el codificador en uso sea conocido [10].

Además, ningún escenario de ataque es considerado, sólo el caso de la reproducción del vídeo protegido en un decodificador estándar es descubierto. Con el fin de garantizar un cierto nivel de seguridad, una mayor cantidad de bytes deben ser cifrados y la necesidad de atención que se deben tomar sobre qué bytes están cifrados. J. Wen [31] describe un enfoque más general, como parte del estándar MPEG 4 IPMP, llamado *Syntax Unaware Runlength-based Selective Encryption* (SURSLE). Este esquema cifra X bits, los siguientes bits Y quedan en texto plano; los próximos bits Z se cifran de nuevo, y así sucesivamente.

Además de los problemas de seguridad mencionados anteriormente, ambos esquemas de destruyen parcialmente la sintaxis de flujo de bits MPEG y MPEG emulando marcadores importantes, lo que causa que un decodificador MPEG se bloquee.

2.8. Comparación entre los esquemas mencionados

Para comparar los esquemas que se han descrito anteriormente, se tomaron en cuenta los parámetros que se describen a continuación:

- Degradación Visual

  Esto es, se mide la distorsión de la percepción de los datos del video con respecto al video normal. Para datos sensibles, una alta degradación visual podria ser deseable para disfrazar completamente el contenido visual.

- Proporción del cifrado

  En este punto se mide la proporción entre el tamaño de la parte cifrada con respecto al tamaño completo de los datos. El tamaño de los datos cifrados deben ser menor para reducir la complejidad computacional.

- Velocidad

  En muchas de las aplicaciones en *tiempo real* es importante que los esquemas de cifrado y descifrado sean suficientemente rápidos para no atrasar la transmisión.

- Seguridad Criptográfica

  Define si el esquema de cifrado es seguro contra los ataques de "fuerza bruta" y ataques de texto plano-texto cifrado diferentes.

- Formato adecuado
  Se analiza que el formato del cifrado sea aceptado por el codificador.

- Compresión Amigable

  Es considerado un esquema de cifrado con compresión amigable si no tiene mucho impacto en la eficiencia de la compresión de datos. Es deseable que el tamaño de los datos cifrado no aumenten.

A continuación se muestra en la tabla 2.1 los parámetros que contiene cada esquema:

Tabla 2.1: Comparación de Algoritmos de Video existentes.

Algoritmos de Cifrado de Video	Degradación Visual	Proporción del cifrado	Velocidad	Seguridad Criptografica	Formato adecuado	Compresión Amigable
Naive	Alta	100%	Lento	Alto	?	No cambia
Permutación Zig-Zag	?	100%	Muy Rápido	Muy Bajo	?	Un gran incremento
VEA	Alta	50%	Rápido	Alto	No aceptable	No cambia
VEA I	Alta	?	Rápido	No aceptable	Adecuado	Satisfactorio (MPEG)
VEA II	Alta	Variable	?	No aceptable	No aceptable	Satisfactorio (MPEG)
MVEA	Alta	Baja	?	Alto	Adecuado	No Satisfactorio
RVEA	Alta	<15%	?	?	?	No Satisfatorio
AEGIS	?	30-60%	?	Muy Bajo	No aceptable	No Satisfactorio
Sign-Bit	?	?	Rápido	Aceptable	Adecuado	No
Cifrado Byte	Baja	?	?	Aceptable	No aceptable	No Satisfactorio

De acuerdo a la tabla 2.1, en términos de seguridad, el esquema de cifrado Naive es el más recomendable debido a que cumple con la mayoría de los parámetros mencionados anteriormente. En términos de eficacia, el esquema de cifrado Sign-Bit, es el más adecuado.

2.9. Real-Time Video Encryption for H.263 Videos

De Ujwala Potdar, Dayanand Ambawade (Miembros Profesionales de IEEE) presentado en la Conferencia Internacional sobre Frameworks Distribuidos para Aplicaciones Multimedia 2010 [14].

En este articulo se analizó dos tipos de esquemas de cifrado en H.263, uno fue el cifrado de todos los bits en el vídeo completo donde el esquema extrae todos los bits de signo coeficiente de DC y AC de todo el vídeo . Estos se seleccionan para que sean múltiplo de 8 al que se aplica una transformación invertible que produce el vídeo cifrado, utiliza el esquema DES en modo CFB y solo el receptor que conoce la llave secreta puede descifrar los datos del vídeo. El otro es un cifrado Framewise de bits, el esquema extrae toda la DC y AC bits de signo coeficiente framewise . El número de bits seleccionado se redondea a un múltiplo de 8 y se cifra. Dado que el cifrado y descifrado se realizan para cada frame,la técnica permite que sea en tiempo real.

Las pruebas que se realizaron fueron:

- Relación de cifrado: Es la relación entre el tamaño de la parte cifrada a todo el tamaño del archivo de vídeo de entrada. Entre más bajo sea la relación de cifrado, menor será la carga de procesamiento y mejor será el rendimiento. Sus resultados se muestran en la Tabla 2.2

Tabla 2.2: Relación de cifrado.

Parámetro	Para Entrada de video car16.263	
	Cifrado de bits de los coeficientes AC y DC	Cifrado Framewise de bits de los coeficientes AC y DC
Radio de Cifrado	0.0839	0.06739
Numero de bytes cifrados	1414.5	1136

- Tiempo de Cifrado y descifrado: Obtuvieron que el tiempo necesario para descifrar el esquema de vídeo mediante el cifrado framewise de bits es mucho más alto que el otro esquema. Ver Tabla 2.3

Tabla 2.3: Tiempo de Cifrado (A) y Descifrado (B).

Parámetro	Para Entrada de video car16.263	
	Cifrado de bits de los coeficientes AC y DC	Cifrado Framewise de bits de los coeficientes AC y DC
Tiempo de Cifrado (ms)	11.56	11.09

(A)

Parámetro	Para Entrada de video car16.263	
	Cifrado de bits de los coeficientes AC y DC	Cifrado Framewise de bits de los coeficientes AC y DC
Tiempo de Descifrado (ms)	11.609	305.31

(B)

- Degradación Visual: Este criterio mide la distorsión de la percepción del vídeo cifrado con respecto al vídeo original. Se supone que el sistema de cifrado de vídeo se puede descodificar y visualizar sin descifrado. Peak signal-to-noise ratio (PSNR) y Mean Square Error (MSE) son los principales indicadores utilizados para medir la degradación visual. La degradación visual es un criterio subjetivo, debido a que es difícil de definir un umbral de distorsión visual aceptable. Sus resultados fueron: (ver Tabla 2.4)

Tabla 2.4: Degradación Visual.

Parámetro	Para Entrada de video car16.263	
	Cifrado de bits de los coeficientes AC y DC	Cifrado Framewise de bits de los coeficientes AC y DC
PSNR	10.006	10.074
MSE	6492.963	6391.655

Como resultado, comprobaron que el esquema que cifra los bits de los coeficientes DC y AC de todo el vídeo se puede implementar sólo sí el video completo está disponible. Así, a pesar de que ofrece una mejor seguridad, no permite la operación en tiempo real. Por otro lado, el cifrado framewise de bit implica que el cifrado se puede iniciar inmediatamente después de un solo frame está disponible. Este esquema proporciona un funcionamiento en tiempo real con una disminución muy pequeña en la degradación visual.

2.10. Selective Encryption Algorithm Implementation for Video Call on Skype Client

De Alwi Alfiansyah Ramdan y Rinaldi Munir, presentado en la Séptima Conferencia Internacional sobre Sistemas de Telecomunicaciones, Servicios y Aplicaciones (TSSA)en el 2012 [16].

En este trabajo se propuso una aplicación cliente de Skype hecha especialmente para las videollamadas. Su principal objetivo es implementar un esquema de cifrado selectiva en videollamada, sobre todo a través de Skype. Este cliente de Skype está implementado en el lenguaje de programación C++ utilizando la API de Skype SkypeKit. Los dos esquemas que seleccionaron para este trabajo fueron RVEA y CbVEA. Sus pruebas fueron medir:

- Prueba de selección del esquema, probaron 8 archivos de video utilizando RVEA y 8 archivos utilizando CbVEA y midieron el tiempo de procesamiento de cada uno.

Tabla 2.5: Tiempo RVEA.

Nombre del Archivo de entrada	Número de Frames	Tiempo de Procesamiento (segundos)
bus.y4m	150	1.085
carphone.y4m	382	5.465
coastguard.y4m	300	4.314
container.y4m	300	4.254
foreman.y4m	300	4.294
grandma.y4m	870	12.303
miss_am.y4m	150	2.175
container.y4m	300	4.264

Tabla 2.6: Tiempo CbVEA.

Nombre del Archivo de entrada	Número de Frames	Tiempo de Procesamiento (segundos)
bus.y4m	150	1.137
carphone.y4m	382	5.73
coastguard.y4m	300	4.469
container.y4m	300	4.473
foreman.y4m	300	4.48
grandma.y4m	870	13.075
miss_am.y4m	150	2.261
container.y4m	300	4.523

- Se midió el tiempo de Cifrado y Descifrado utilizando el Skype-SeVid

Tabla 2.7: Tiempo Cifrado (A) y Descifrado (B) en Skype.

Número de Frames Cifrados	Tiempo de Procesamiento (segundos)	Tiempo de Procesamiento por Frame (segundos)
852	13.564	0.01592
892	15.985	0.01792
716	9.239	0.0129
991	16.103	0.01625
720	10.567	0.01468

(A) Cifrado

Número de Frames Cifrados	Tiempo de Procesamiento (segundos)	Tiempo de Procesamiento por Frame (segundos)
805	13.564	0.01685
871	15.985	0.01835
706	9.239	0.01309
982	16.103	0.0164
713	10.567	0.01482

(B)Descifrado

Como conclusiones obtuvieron que hay una posibilidad de que el número de bits seleccionados para ser cifrados en CbVEA es mayor que el número de bits seleccionados para ser cifrados en RVEA, de modo que el tiempo de procesamiento CbVEA es más largo que el tiempo de procesamiento RVEA. Basado en la prueba con Skype-SEVID, se encontró que la secuencia de datos enviados a través de la red no tiene similitud con la secuencia de datos de vídeo codificada o flujo de datos de vídeo antes del cifrado. Se menciona que Skype tiene canales de comunicación cifrados, aunque no se puede verificar debido al sistema cerrado del protocolo de Skype. Es posible ver la diferencia entre el flujo de datos de vídeo cifrado y el flujo de datos enviados sobre la red, porque Skype cifra el flujo de datos de vídeo cifrado en sus canales de comunicación. Sin embargo, esto no significa que en el canal de comunicación no puede haber espías por parte de Skype. También concluyeron que el proceso de descifrado es más tardado que el proceso de cifrado, esto es posible, debido al número de tramas descifrados, en algunos intervalos, es menor que el número de frames cifrados, a los mismos intervalos. Este número más pequeño de frames descifrados es posible debido a que el número de tramas recibidas es menor que el número de tramas enviadas ya que pudo haber pérdida de paquetes de datos en la red.

2.11. Resumen de contenido

Una gran parte de los esquemas de cifrado de video mencionados anteriormente, no se consideran seguros contra ataques de "fuerza bruta" y en general es necesario realizar más trabajo de investigación especializados en transmisión de video.

Se destacan esquemas como VEA [21] y MVEA [22], que para agilizar la velocidad del cifrado, sólo trabajan con ciertas partes del frame de video, como los coeficientes de DC de la Transformada Discreta de Fourier.

Uno de los aspectos más importantes en el diseño de los esquemas de cifrado de video es su implementación práctica. En el caso de los dispositivos móviles el procesamiento de cifrado y descifrado debe ser mínimo, este problema no ha sido considerado formalmente por los diseñadores de los esquemas de cifrado para video mencionados.

Hasta este momento solamente se ha descrito los esquemas de cifrado que funcionan para la transmisión de video en "*directo*", en el siguiente capítulo se habla de los protocolos de comunicación y los codecs que permiten la transmisión de video, además de los algoritmos de cifrado simétrico que se utilizaron para el diseño de los esquemas de cifrado que se mencionaron en este capítulo.

Capítulo 3

Marco de Referencia

3.1. Protocolos de Comunicación para la Transmisión de Video

Los protocolos de comunicaciones definen las reglas para la transmisión y recepción de la información entre los nodos de la red, de modo que para que dos nodos se puedan comunicar entre sí, es necesario que ambos empleen la misma configuración de protocolos. En este capítulo se describen los protocolos de comunicación y los codecs de video que se emplean para la transmisión de video. También se menciona los algoritmos de cifrado simétrico que se han empleado en esquemas de cifrado para video.

3.1.1. Protocolo de Internet (IP)

El protocolo de Internet está diseñado para su uso en sistemas interconectados de redes de comunicación de ordenadores por intercambio de paquetes. Este protocolo proporciona los medios necesarios para la transmisión de bloques de datos llamados datagramas desde el origen al destino, donde el origen y destino son host identificados por direcciones de longitud fija, también se encarga, si es necesario, de la fragmentación y reesamblaje de grandes datagramas para su transmisión a través de redes de trama pequeña.

Los paquetes de IP se pueden perder, retrasar o ser recibidos fuera de secuencia, cada paquete se encamina por si misma, usando la cabecera IP se adjunta al paquete físico.

Quizás los aspectos más complejos de IP son el direccionamiento y el enrutamiento. El direccionamiento se refiere a la forma como se asigna una dirección IP y cómo se dividen y se agrupan subredes de equipos. Una dirección IP es un número que identifica de manera lógica y jerárquicamente a una interfaz de un dispositivo dentro de una red que utilice el protocolo de Internet (Internet Protocol), que corresponde al nivel de red o nivel 3 del modelo de referencia OSI. Dicho número no se ha de

confundir con la dirección MAC que es un número físico que es asignado a la tarjeta o dispositivo de red (viene impuesta por el fabricante), mientras que la dirección IP se puede cambiar. El usuario al conectarse desde su hogar a Internet utiliza una dirección IP. Esta dirección puede cambiar al reconectar. A la posibilidad de cambio de dirección de la IP se denomina dirección IP dinámica.

Los sitios de Internet que por su naturaleza necesitan estar permanentemente conectados, generalmente tienen una dirección IP fija (IP fija o IP estática); es decir, no cambia con el tiempo. Los servidores de correo, dns, ftp públicos, servidores web, conviene que tengan una dirección IP fija o estática, ya que de esta forma se facilita su ubicación.

El enrutamiento consiste en encontrar un camino que conecte una red con otra y, aunque es llevado a cabo por todos los equipos, es realizado principalmente por routeadores, que no son más que computadoras especializadas en recibir y enviar paquetes por diferentes interfaces de red, así como proporcionar opciones de seguridad, redundancia de caminos y eficiencia en la utilización de los recursos [7].

La figura 3.1 ilustra los protocolos de capas más altas y que posteriormente serán explicados.

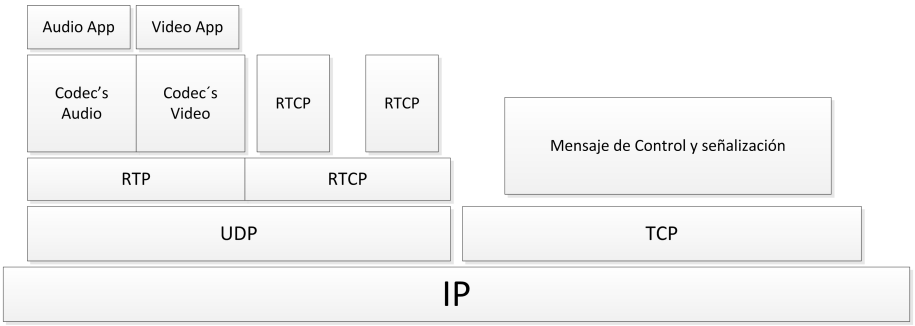


Figura 3.1: Protocolos de Red.

3.1.2. Protocolo de Control de Transmisión (TCP)

TCP es uno de los protocolos fundamentales en Internet. Fue creado entre los años 1973 y 1974 por Vint Cerf y Robert Kahn. TCP proporciona transporte fiable, orientado a la conexión a través de IP. TCP utiliza números de secuencia y acuses de recibo positivos para asegurar que cada bloque de datos, denominado segmento, se ha recibido. Segmentos perdidos son retransmitidos hasta que son recibidos con éxito.TCP da soporte a muchas de las aplicaciones más populares de Internet (navegadores, intercambio de ficheros, clientes FTP, etc.) y protocolos de aplicación HTTP, SMTP, SSH y FTP.

Funcionamiento

Las conexiones TCP se componen de tres etapas: establecimiento de conexión, transferencia de datos y fin de la conexión. Para establecer la conexión se usa el procedimiento llamado negociación en tres pasos (3-way handshake). Para la desconexión se usa una negociación en cuatro pasos (4-way handshake). Durante el establecimiento de la conexión, se configuran algunos parámetros tales como el número de secuencia con el fin de asegurar la entrega ordenada de los datos y la robustez de la comunicación.

Durante la etapa de transferencia de datos, una serie de mecanismos claves determinan la fiabilidad y robustez del protocolo. Entre ellos están incluidos el uso del número de secuencia para ordenar los segmentos TCP recibidos y detectar paquetes duplicados, sumas de comprobación para detectar errores, y asentimientos y temporizadores para detectar pérdidas y retrasos.

3.1.3. Protocolo de datagrama de usuario (UDP)

UDP es un protocolo del nivel de transporte basado en el intercambio de datagramas. Se ubica en la capa 4 del modelo OSI. Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera. No tiene confirmación ni control de flujo, por lo que los paquetes pueden adelantarse unos a otros. Tampoco ofrece un control de error, ya que no hay confirmación de entrega o recepción por lo tanto no garantiza si algunos paquetes se perdieron o duplicaron [5]

Su uso principal es para protocolos como DHCP, BOOTP, DNS y demás protocolos en los que el intercambio de paquetes de la conexión/desconexión son mayores, o no son rentables con respecto a la información transmitida, así como para la transmisión de audio y vídeo en tiempo real, donde no es posible realizar retransmisiones por los estrictos requisitos de retardo que se tiene en estos casos.

En la familia de protocolos de Internet UDP proporciona una sencilla interfaz entre la capa de red y la capa de aplicación, no otorga garantías para la entrega de sus mensajes y el origen UDP no retiene estados de los mensajes UDP que han sido enviados a la red. UDP sólo añade multiplexado de aplicación y suma de verificación de la cabecera y la carga útil. Cualquier tipo de garantías para la transmisión de la información deben ser implementadas en capas superiores.

UDP es generalmente el protocolo usado en la transmisión de vídeo y voz a través de una red. Esto es porque no hay tiempo para enviar de nuevo paquetes perdidos cuando se está escuchando a alguien o viendo un vídeo en tiempo real [13].

3.1.4. Protocolo de Transporte de Tiempo real (RTP)

Este protocolo está desarrollado por el grupo de trabajo de transporte de Audio y Video del IETF (Internet Engineering Task Force, en español Fuerza de Tareas de Ingeniería de Internet), publicado por primera vez como estándar en 1996 como la RFC 1889 [18], y fue actualizado posteriormente en 2003 en la RFC 3550.

Surgió con la idea de crear un protocolo específico para la gran demanda de recursos en tiempo real por parte de los usuarios.

Su objetivo es proporcionar el transporte de extremo a extremo para aplicaciones con requisitos de tiempo real en redes unicast o multicast tales como:

- Videoconferencia
- Difusión de Audio/Video
- Telefonía
- Archivos multimedia

Este protocolo proporciona servicios de una red extremo a extremo para transmisión de datos en tiempo real, aunque es un protocolo independiente de transporte y de red es a menudo utilizado sobre UDP.

Funcionamiento

RTP funciona sobre el protocolo de transporte de UDP, el emisor encapsula un trozo de datos dentro del paquete RTP y esté a su vez es encapsulado dentro de un datagrama UDP que viaja en un paquete IP. El receptor extrae los datos RTP del UDP y pasa los datos al reproductor para que descodifique el contenido y lo reproduzca [8].

RTP permite asociar a cualquier dispositivo su propio *stream* de paquetes. Por ejemplo si dos personas quieren llevar a cabo una videoconferencia, se puede abrir dos *streams*, uno de audio y otro de vídeo, uno en cada sentido para cada tipo de datos. Aunque la realidad es que los *streams* de audio y vídeo están entrelazados, se envían como un único stream y se establece una única conexión para enviar los dos *streams* (audio y video).

RTP sufre vulnerabilidades al igual que otros protocolos. Por ejemplo,un usuario atacante podría autenticar de forma falsa direcciones de red de origen o destino, cambiar el encabezado e incluso cambiar el algoritmo de codificación. Utilizando el protocolo RTP sin su protocolo de control RTCP, los campos CNAME y NAME podrían usarse para autenticar a otro usuario. Debido a estas vulnerabilidades entre otras, es importante saber algunos aspectos de seguridad para hacer un uso más responsable del protocolo.

RTP es usado actualmente en la telefonía VoIP, llamadas telefónicas a través de Internet y sistemas de videoconferencia. Por tanto, la captura de paquetes RTP es un problema para la integridad de la conversación debido a las vulnerabilidades en seguridad. El tema de vulnerabilidades y agujeros en seguridad está siendo un tema de actualidad debido a los problemas que plantean para los usuarios.

A continuación se describe el encabezado que conforma un paquete RTP: (Ver Figura 3.2)

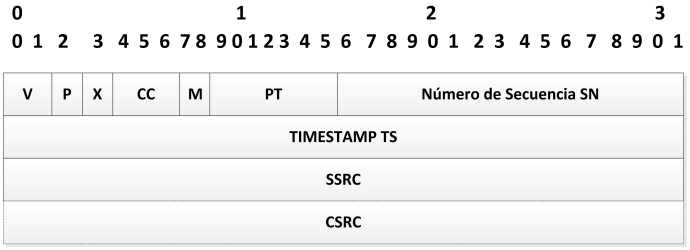


Figura 3.2: Encabezado RTP.

Los primeros 12 octetos están en todos los paquetes de RTP, mientras que la lista de identificadores CSRC (Contributing Source) solo está presente cuando hay un mezclador, el cual se encarga de cambiar el formato del stream con el fin de adecuarlo al ancho de banda del receptor. Cada uno de los campos se definen a continuación:

- Versión (V): Tamaño de 2 bits, los cuales identifican la versión RTP. Actualmente es la versión 2.
- Relleno (P): Es de 1 bit, sí este está activado, el paquete contiene uno o más octetos de relleno adicional a los datos. El último octeto indica cuántos octetos se han de descartar, incluyendo él mismo.
- Extensión (X): De 1 bit, sí está activado, la cabecera fija solo puede llevar una extensión.
- Número de CSRC (CC): Es de tamaño de 4 bits. Contiene el número de identificadores CSRC que hay en la cabecera fija.
- Marcador (M): De 1 bit. Es definido por el perfil particular de medios.
- Tipo de Carga útil: 7 bits. Un índice en una tabla del perfiles de medios que describe el formato de carga útil.
- Número de Secuencia: 16 bits. Un único número de paquete que identifica la posición de éste en la secuencia de paquetes. El número del paquete es incrementado en uno para cada paquete enviado.

- Sellado de tiempo: 32 bits. es la información más importante de las aplicaciones que no necesitan retardo. Refleja el instante de muestreo del primer byte en la carga útil. El emisor establece el TimeStamp según el instante en que se muestra el primer octeto en el paquete. El receptor después de recibir los paquetes de datos utiliza el TimeStamp para reconstruir el tiempo original. TimeStamp se utiliza también para sincronizar distintos flujos como información de audio y vídeo en MPEG. Varios paquetes consecutivos pueden tener el mismo sellado si son lógicamente generados en el mismo tiempo, por ejemplo, si son todo parte del mismo frame de vídeo. Sin embargo, RTP por si sólo no es responsable de la sincronización, ya que esta misión está destinada al nivel de aplicación.

- SSRC. Id. de origen de sincronización: 32 bits. Identifica la fuente de sincronización. Sí la cuenta CSRC es cero, entonces la fuente de carga útil es la fuente de sincronización. Sí la cuenta CSRC es distinta a cero, entonces el SSRC identifica el mixer (mezclador).

- CSRC. Id. de origen de contribución: 32 bits cada uno. Identifica las fuentes contribuyentes para la carga útil. El número de fuentes contribuyentes está indicado por el campo de la cuenta CSRC.

3.1.5. Protocolo de Transporte Seguro de Tiempo Real (SRTP)

Es un perfil del Protocolo de Transporte de Tiempo Real (RTP), el cual proveé confidencialidad, mensajes de autentificación y protección de repetición para el tráfico de RTP y para RTCP (Protocolo de Control RTP) [3].

SRTP puede obtener un alto rendimiento y baja expansión del paquete. Demuestra que puede ser una aceptable protección para ambientes heterogéneos (una combinación entre redes cableados e inalámbricos), para obtener tales características

Cada stream de SRTP requiere que el emisor y el receptor mantengan el estado de información del cifrado, a esta información se le denomina "contexto criptográfico". SRTP utiliza dos tipos de llaves: las llaves de sesión y llaves maestras, la primera se utiliza directamente en la transformación criptográfica por ejemplo, mensaje de autenticación o el cifrado y la segunda es una cadena de bits aleatorias (propuesta por el protocolo de gestión de llaves) de la que se derivan las llaves de sesión de manera segura.

3.2. Transmisión de Video

En esta sección se definen los conceptos de compresión y codec y se describen algunos de los codecs que existen para transmitir el video.

3.2.1. Compresión de video

La compresión, reducción de la tasa de bits, reducción de datos y codificación de la fuente son términos que significan básicamente lo mismo en este contexto. En esencia, la compresión se realiza utilizando una cantidad menor o tasa de datos. Cabe señalar que en la compresión de audio significa tradicionalmente un proceso en el que se reduce el rango dinámico del sonido, pero por ejemplo en el contexto de MPEG la misma palabra significa que la tasa de bits se reduce, idealmente dejando la señal sin cambios. Siempre y cuando el contexto sea claro, los dos significados pueden coexistir sin una gran confusión. Hay varias razones de por que las técnicas de compresión son populares:

- La compresión extiende el tiempo de reproducción de un dispositivo de almacenamiento determinado.
- La compresión permite la miniaturización. Con pocos datos a almacenar, el tiempo de reproducción se obtiene igual con menor hardware. Esto es útil en ENG (Electronic News Gathering) y dispositivos de consumo.
- Las tolerancias pueden ser relajados. Con menos de datos para registrar, densidad de almacenamiento se puede reducir haciendo equipo que es más resistente a entornos adversos y que requiere menos mantenimiento.
- En los sistemas de transmisión, la compresión permite una reducción en el ancho de banda que generalmente dará como resultado una reducción en el coste. Esto puede hacer posible un servicio que sería impracticable sin ella.
- Si un determinado ancho de banda está disponible para la señal no comprimida, la compresión permite más rápido que en tiempo real de la transmisión en el mismo ancho de banda.

Básicamente la compresión se divide en:

- Compresión sin pérdidas: Esta técnica se utiliza principalmente, cuando es necesario transmitir mensajes de emisor a receptor sin que se produzca ningún tipo de distorsión o de cambio significativo respecto al mensaje original. Se basan, principalmente, en utilizar más o menos bits según la frecuencia de utilización, es decir, se observa si se producen repeticiones para intentar codificarlas con menos bits de los que en principio serían necesarios.
- Compresión con pérdidas: Cuando es necesario llegar a unos flujos de información que no sobrepasen unos determinados limites de ancho de banda y esto no es posible utilizando técnicas de compresión sin pérdidas, será necesario comprimir aún más la señal aunque esto repercuta en su calidad.

3.2.2. Codificación de Video (CODEC)

Un CODEC codifica una imagen de origen o secuencia de video en un formato comprimido y decodifica esto para producir una copia o aproximación de la secuencia de origen [17]. Un codificador de vídeo se compone de tres unidades principales [9]

- Un modelo temporal.
- Un modelo espacial.
- Un codificador de entropía.

La entrada para el modelo temporal es una secuencia de vídeo sin comprimir . El modelo temporal intenta reducir la redundancia temporal mediante la explotación de las similitudes entre cuadros de vídeo vecinos, por lo general mediante la construcción de una predicción de la trama de vídeo actual.

El modelo espacial reduce la redundancia espacial entre los píxeles dentro de una imagen (similitud de píxeles, dentro de los frames).

El codificador de entropía elimina la redundancia estadística en los datos (por ejemplo, los datos que representan los vectores y los coeficientes de frecuencia de ocurrencia de los códigos binarios cortos) y produce un flujo de bits comprimido o archivo que puede ser transmitido y/o almacenado.

A continuación en la figura 3.3 se muestra y se explicarán los principales CODECs que existen.

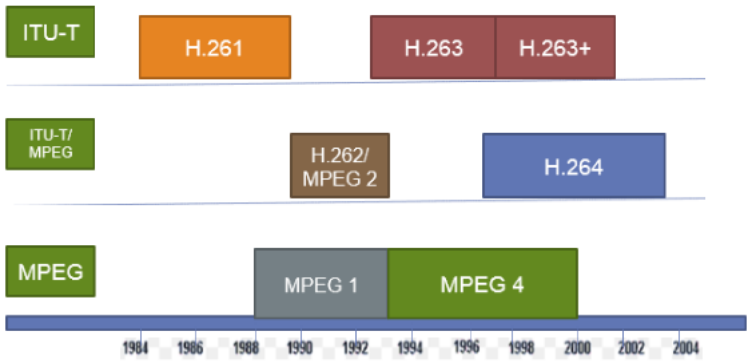


Figura 3.3: Principales CODEC´s.

3.2.3. MPEG

Se iniciará explicando los CODEC´s de la familia MPEG. MPEG (Moving Pictures Experts Group) fue formado por la ISO (International Standards Organization) para establecer un conjunto de estándares de compresión y transmisión video y audio [30].

Además de comprimir imágenes estáticas compara los fotogramas presentes con los anteriores y los futuros para almacenar sólo las partes que cambian.MPEG aplica la compresión temporal y la espacial.

1. **MPEG-1**

   Es la primera generación de los codecs de video propuesto por MPEG como un estándar que provee la codificación de video para los medios digitales de almacenamiento (DSM) tales como discos compactos (CD′s), cintas de audio digital (DAT), etc.

   Este desarrollo fue en respuesta de las necesidades de la industria para almacenar de una manera eficiente información visual en otro medio de almacenamiento en vez de las convencionales grabadoras de video analógicas (VCR′s). En la mayoría de las aplicaciones del video MPEG-1 su tasa de bits es de un rango de 1 a 1.5 Mbits/s.

   Es un estándar definido para la codificación de tasa relativamente baja de bits de baja resolución de las imágenes espaciales. MPEG-1 fue dirigido como un software orientado de procesamiento de imágenes, donde lo grande y variable de la longitud de los paquetes podría reducir la sobrecarga del software.

   Características del MPEG-1:

   - Utiliza una predicción de movimiento bi-direccional con una codificación de precisión de medio pixel.
   - Compatible con una gama de vectores de movimiento de -512 a 511.5 pixeles para vectores de medio pixel y de -1024 hasta 1023 para vectores de movimiento de pixel completo (entero).
   - Utiliza cuantización visual ponderada: define por defecto 64 elementos de matriz de cuantificación, además que permite matrices personalizadas apropiadas para diferentes aplicaciones.
   - El tamaño de la imagen puede ser tan grande como 4kx 4k pixeles.

   Estándar MPEG-1

   El estándar consta de un documento de cinco partes:

   1.Sistemas. En esta parte se especifica el diseño lógico y los métodos utilizados para almacenar el audio, el video y otros datos codificados en una secuencia de bits, y para matener la sincronización entre los diferentes contenidos. Está diseñado específicamente para el almacenamiento de la

información y la transmisión por canales de datos [4].

2.Vídeo. Específica cómo está compuesto el video y es definido en ISO/IEC-11172-2.

3.Audio. Formatos MPG y MP3.

4.Prueba de conformidad, comprobando el cumplimiento del estándar por parte de las implementaciones.

5.Software de referencia. Ejemplo de software mostrando como codificar y descodificar de acuerdo con el estándar.

Trama de MPEG-1

Hay 6 capas en MPEG-1: La secuencia de video, grupo de imágenes, imagen, rebanada, macrobloques y capas de bloques [30].(Ver Figura 3.4)

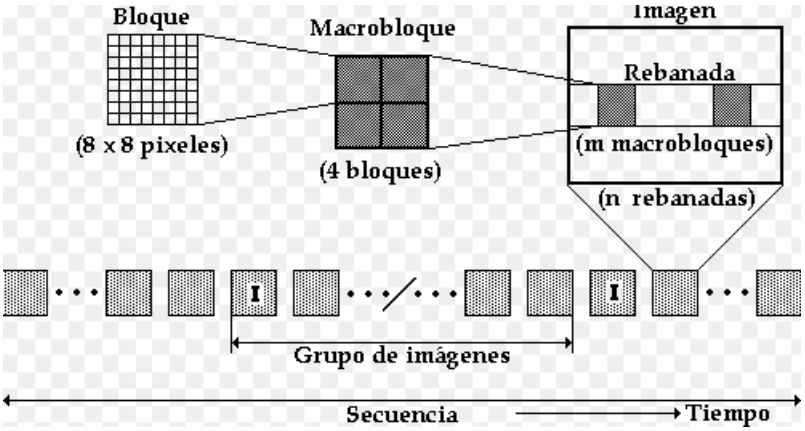


Figura 3.4: Capas de MPEG.

- Una capa de secuencia de video consiste en: la cabecera de la secuencia, uno o más grupos de imágenes y un código de final de secuencia. Contiene la configuración del tamaño de la imagen (tamaño horizontal y vertical), la tasa de imagen, velocidad de bits, el mínimo tamaño del buffer del decodificador y los datos de usuario.
- La capa GOP (Group Of Pictures) consiste en un conjunto de imágenes que están en un orden continuo. Esta capa contiene los ajustes de los siguientes parámetros:
  - Código de tiempo: da el intervalo de tiempo desde que inicia la secuencia en horas-minutos-segundos.
  - La bandera de cerrado de GOP: el cual indica si la operación de decodificación necesita imágenes previas del GOP para la compensación

del movimiento.
- Bandera del enlace roto: indica si el previo GOP puede ser utilizado para decodificar el actual GOP.
- Datos de Usuario.

- La capa de Imagen actúa como una unidad primaria de codificación. Contiene los ajustes de los siguientes parámetros:
- La referencia temporal de cual es el número de imagen en la secuencia, el cual se utiliza para determinar el orden de la visualización.
- Los tipos de Imágenes (I/P/B/D)
- La saturación inicial del buffer del decodificador el cuál entrega el número de bits que debe estar en el buffer de video comprimido (se utiliza para evitar el desbordamiento del buffer del decodificador).
- La resolución de los vectores del movimiento hacia adelante y el rango de las imágenes P y B.
- La resolución de los vectores del movimiento hacia atrás y el rango de las imágenes.

2. **MPEG-2**

MPEG-2 es por lo general usado para codificar audio y video para señales de transmisión, que incluyen televisión digital terrestre, por satélite o cable.

MPEG-2, con algunas modificaciones, es también el formato de codificación usado por los discos SVCD′s y DVD′s comerciales de películas. MPEG-2 llegó ha ser muy importante debido a que fue escogido como el esquema de compresión para DVB (Digital Video Broadcasting) y DVD (Digital Video Disk/Digital Versatil Disk).

Se esperaba que con MPEG-2 se facilitaría el intercambio de flujos de bits entre diferentes aplicaciones, sin embargo, teniendo en cuenta la factibilidad de la implementación de la sintaxis completa de la cadena de bits, un número limitado de grupos de la sintaxis se estipulaba por medio de perfiles y niveles [12].

Un perfil es un subconjunto de toda la sintaxis de flujo de bits que se define por la especificación de MPEG-2. Dentro de los límites impuestos por la sintaxis de un perfil dado, todavía es posible abarcar grandes variaciones en el rendimiento de los codificadores y decodificadores, dependiendo de los valores tomados por los parámetros en el flujo de bits. Por ejemplo, es posible especificar tamaños de trama tan grandes como (aproximadamente) 214 muestras de ancho por 214 líneas de alto. En la actualidad no es ni práctico ni económico de implementar un decodificador capaz de tratar con todos los posibles tamaños de bastidor. Para hacer frente a este problema, los niveles se definen dentro de cada perfil.

Un nivel es un conjunto definido de restricciones impuestas a los parámetros en el flujo de bits. Estas renstricciones pueden ser límites simples en números. Alternativamente, pueden adoptar la forma de restricciones en las combinaciones aritméticas de los parámetros (por ejemplo enmarcar altura, velocidad y ancho del frame). Ambos perfiles y niveles tienen una relación jerárquica, y la sintaxis soportada por un perfil más alto o nivel también deben ser compatibles con todos los elementos sintácticos de los perfiles más bajos o niveles [9].

3. **MPEG-4**

   MPEG-4 fue lanzado para tratar una nueva generación de aplicaciones y servicios multimedia. La base del estándar MPEG-4 fue desarrollado durante un período de 5 años (1995-1999), sin embargo al estandar MPEG-4 se le añaden nuevas piezas continuamente a medida que la tecnología evoluciona para tratar las nuevas aplicaciones.

   MPEG-4 tiene como objetivo proporcionar herramientas y algoritmos para el almacenamiento eficiente, transmisión y manipulación de los datos de vídeo en entornos multimedia. Las principales motivaciones detrás de esta tarea son el éxito probado de vídeo digital en tres ámbitos de la televisión digital, las aplicaciones gráficas interactivas y la multimedia interactiva (World Wide Web, la distribución y el acceso a contenido de la imagen).

   Al igual que MPEG-2, MPEG-4 añade varias funcionalidades, las cuales los usuarios solo pueden estar interesados en un subconjunto de ellos, éstos se definen como perfiles y también añade niveles. Pero a diferencia de los codecs anteriores, la codificación está basada en objetos, las tramas de vídeo se definen en términos de capas de planos de objetos de vídeo (VOP). Cada VOP es entonces un cuadro de video de un objeto específico de interés para ser codificados ó para interactuar con él.

3.2.4. H.261

Este estándar define métodos para la codificación y decodificación del video para la transmisión digital sobre los Servicios Integrados Digitales de la Red-*Integrated Services Digital Network*- (ISDN) a tasas de p x 64 kbit/s, donde p está en un rango de 1-30. Principalmente su uso se enfoca en la videotelefonía, videoconferencia y otros servicios audiovisuales. Fue ratificado en diciembre de 1990.

El estándar H.261 permite hasta tres imágenes a interpolar entre las imágenes transmitidas, lo que reduce la velocidad de frames a 15, 10 y 7.5, respectivamente. Usa el formato de resolución intermedio (QCIF) y reduce la frecuencia de muestreo aún más para adaptarse a los canales de baja tasa de bits. En CIF y QCIF,

los bloques DCT se agrupan en macrobloques (MB) de cuatro luminancia y dos bloques de crominancia Cb y Cr correspondientes. El MB a su vez, se agrupan en capas denominadas grupos de bloques (GOB). Por lo que a partir de este codec, se usaron los grupos de bloques.

3.2.5. H.263

El principal objetivo de este estándar fue codificar el video a baja o muy baja tasa de bit para aplicaciones tales como redes en móviles, en la red telefónica pública conmutada (PSTN) y en la banda estrecha de la Red Digital de Servicios Integrados (ISDN).

Proporciona una mejor compresión que el H.261, soporta la calidad de vídeo básica a velocidades de bits de a 30 kbit/s, y es parte de un conjunto de normas diseñadas para operar en un amplio rango de redes de circuitos y de conmutación de paquetes.

La tecnología H.263 se convirtió posteriormente en la base para MPEG-4 Parte 2, destinado a desarrollar una tecnología optimizada para aplicaciones de muy baja velocidad de bits en la primera etapa. Como resultado de ello, las normas de teléfonos móviles tales como 3GPP incluyen H.263 como estándar de compresión de subsistema de video.

La "línea de base" del modelo de codificación de H.263 fue adoptado como el núcleo del MPEG-4 Perfil visual simple (Simple Profile Visual).

La versión original de H.263 incluye cuatro modos opcionales de codificación (cada una se describe en el anexo de la norma) y en la segunda edición se añadió una serie de nuevos modos opcionales para admitir características como la eficiencia de compresión mejorada y la transmisión a través de redes sólidas con pérdida.

La sintaxis de vídeo tiene una estructura jerárquica de cuatro capas primarias. De arriba hacia abajo, esas capas son [6]:

- Imagen
- Grupo de Bloques o rebanadas o segmento de imagen de vídeo (GOB)
- Macrobloque
- Bloque

Cada imagen se divide en grupos de bloques (GOB) o bien en rebanadas.Un grupo de bloques (GOB, group of blocks) comprende hasta k∗16 líneas, donde k depende del número de líneas en el formato de imagen. Cada GOB se divide en macrobloques. Un macrobloque consiste en 4 bloques de luminancia y los dos bloques

de diferencia de color correspondientes espacialmente, como se muestra en la figura 3.5. Cada bloque de luminancia o de crominancia se relaciona así con 8 píxels por 8 líneas de Y, CB o CR. Además contiene otros factores como: la predicción de las imágenes tipo P se analiza a nivel de los macrobloques y que en la imagen B se puede transmitir el vector de movimiento junto al macrobloque.

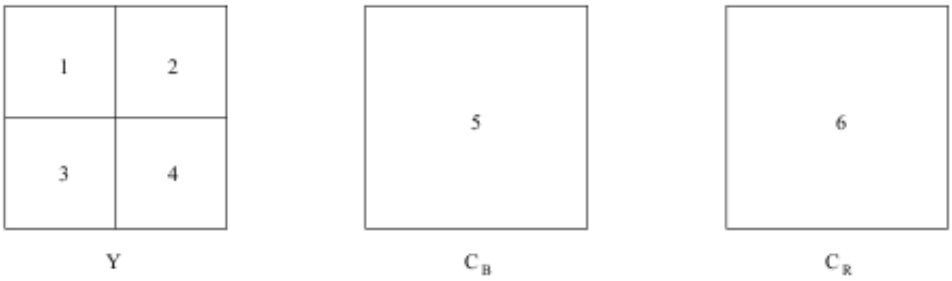


Figura 3.5: Estructura de un Macrobloque.

En la siguiente figura (3.6) se muestra los datos del *header* de cada imagen, seguido de los datos del grupo de macrobloques y de los macrobloques.

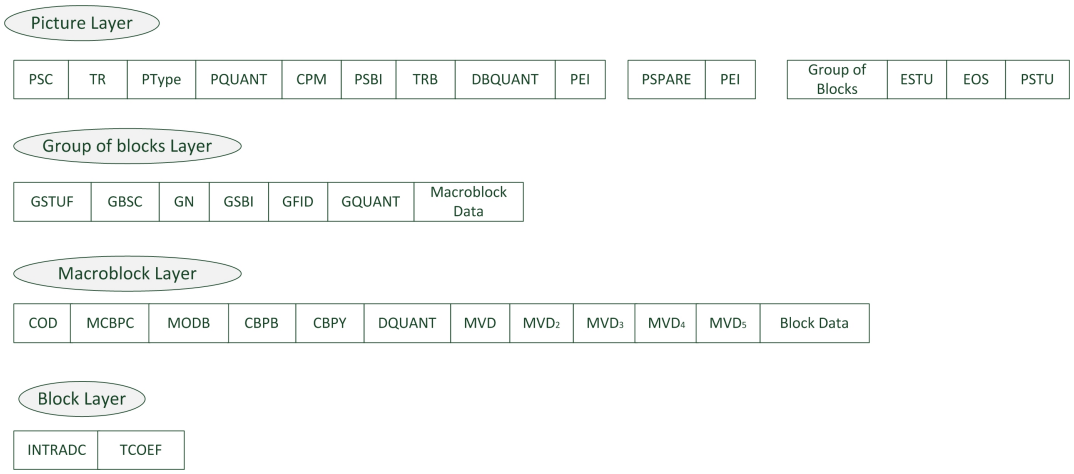


Figura 3.6: Formato H.263.

Los principales campos del *header* de la imagen son:

- Código de comienzo de imagen (PSC, picture start code) (22 bits), el código de comienzo de imagen (PSC) es una palabra de 22 bits. Su valor es 0000 0000 0000 0000 1 00000. Todos los códigos de comienzo de imagen estarán alineados en bytes. Esto se consigue insertando los bits PSTUF que sean necesarios antes del código de comienzo, de manera que el primer bit del código de comienzo es el primer bit (más significativo) de un byte.
- Referencia temporal (TR) (8bits), se forma incrementando en 1 el valor que tenía en el encabezamiento de imagen de referencia temporalmente previo más

el número de imágenes saltadas o no referenciadas en la frecuencia de reloj de imagen desde la transmitida anteriormente. La interpretación de la TR depende de la frecuencia de reloj de imagen activa. En la frecuencia de reloj de imagen de formato CIF normalizado, TR es un número de 8 bits que puede tener 256 valores posibles. La aritmética se ejecuta con sólo los 8 bits menos significativos (LBS, least significant bit). Si se señaliza la utilización de una frecuencia de reloj de imagen personalizada, la referencia temporal ampliada (ETR, extended temporal reference) y la referencia temporal(TR) forman un número de 10 bits de los que TR almacena los 8 bits menos significativos (LBS,least significant bit) y ETR almacena los 2 bits más significativos (MSB, most significant bit). La aritmética en este caso se ejecuta con los 10 bits menos significativos. En el modo tramas PB opcional o en el modo tramas PB mejoradas, TR se refiere únicamente a las imágenes P.

- Tipo de información (PTYPE, type information) (Longitud variable), información sobre la imagen completa, este valor informa sobre el formato de la fuente (CIF, 4CIF, 16CIF, etc.), sobre el tipo de imagenes (INTRA o INTER), etc.

Los principales campos del *header* del GOB son:

- Código de comienzo de grupo de bloques (GBSC, group of block start code) (17 bits), Palabra de 17 bits. Su valor es 0000 0000 0000 0000 1. Los códigos de comienzo GOB pueden estar alineados en bytes. Esto se consigue insertando GSTUF antes del código de comienzo, de manera que el primer bit del código de comienzo es el primer bit (el más significativo) de un byte.

- Número de grupo (GN, group number) (5 bits), Palabra de código de longitud fija de 5 bits. Los bits son la representación binaria del número del grupo de bloques. En el caso del GOB numerado 0, el encabezamiento GOB que incluye GSTUF, GBSC, GN, GSBI, GFID y GQUANT está vacío; en el PSC se utiliza 0 como número de grupo.

Los principales campos del *header* del Macrobloque son:

- Indicación de macrobloque codificado (COD, coded macroblock indication) (1 bit), Un bit que, cuando está puesto a "0", señala que el bloque está codificado. Cuando está puesto a "1", no se transmite más información para este macrobloque en este caso, el decodificador tratará el macrobloque como un bloque INTER con vector de movimiento para todo el bloque igual a cero y sin datos de coeficiente. COD sólo está presente en las imágenes que no son de tipo INTRA, en cada macrobloque de esas imágenes.

- Tipo de macrobloque y patrón de bloque codificado para la crominancia (MCBPC, macroblock type y coded block pattern for chrominance) (Longitud variable), MCBPC es una palabra de código de longitud variable que dé información sobre el tipo de macrobloque y el patrón de grupo codificado para la crominancia

3.3. Herramientas Criptográficas para la Confidencialidad

3.3.1. Criptografía

Es el estudio de técnicas matemáticas en relación con los aspectos de seguridad de la información, tales como la confidencialidad, la integridad de los datos, la autenticación de la entidad, y la autenticación del origen de datos [27].

La Criptografía se puede dividir en:

- Criptografía Simétrica: Usa una clave secreta K para el proceso de cifrado y para descifrado. Esta clave secreta se debe acordar previamente a través de un canal seguro o por medio de un método seguro y una vez acordada se debe de mantener en secreto [11]. (Ver Figura 3.7)

Figura 3.7: Criptografía Simétrica.

- Criptografía Asimétrica: Se utilizan dos llaves, cualquiera que tenga la llave pública puede cifrar datos pero no puede descifrar, en este caso solo la persona que tenga la llave privada puede descifrar los datos.(Ver figura 3.8)

Figura 3.8: Criptografía Asimétrica.

Una característica que diferencia a las dos tipos mencionadas es el costo de procesamiento asociado a cada una de ella; por un lado el cifrado simétrico presenta un costo relativamente bajo y asequible para muchas aplicaciones incluso con necesidades de procesamiento de altos volúmenes de información. Por otro lado el cifrado asimétrico presenta altos costos de procesamiento y por ende su uso es restringido por muchos escenarios.

3.3.2. Algoritmos de Cifrado Simétrico

DES (Data Encryption Standard)

Es un algoritmo desarrollado originalmente por IBM a requerimiento del NBS (National Bureau of Standards, Oficina Nacional de Estandarización, en la actualidad denominado NIST, National Institute of Standards and Technology, Instituto Nacional de Estandarización y Tecnología) de EE.UU. y posteriormente modificado y adoptado por el gobierno de EE.UU.

En 1977, DES se empleó como estándar de cifrado de todas las informaciones sensibles no clasificadas. Posteriormente, en 1980, el NIST estandarizó los diferentes modos de operación del algoritmo. Es el más estudiado y utilizado de los algoritmos de clave simétrica.

El nombre original del algoritmo, tal como lo denominó IBM, era Lucifer. Trabajaba sobre bloques de 128 bits, teniendo la clave igual longitud. Se basaba en operaciones lógicas booleanas y podía ser implementado fácilmente, tanto en software como en hardware.

Tras las modificaciones introducidas por el NBS, consistentes básicamente en la reducción de la longitud de clave y de los bloques, DES cifra bloques de 64 bits, mediante permutación y sustitución y usando una clave de 64 bits, de los que 8 son de paridad (esto es, en realidad usa 56 bits), produciendo así 64 bits cifrados [28].

- Algoritmo DES

  El algoritmo general de DES se muestra en la Figura 3.9, donde se muestra que DES tiene 19 Etapas diferentes, la primer etapa es un permutación inicial (IP) del texto plano de 64 bits independiente de la llave, la última etapa es otra permutación como la inicial pero inversamente. La penúltima etapa intercambia los 32 bits de la izquierda y los 32 bits de la derecha [29].

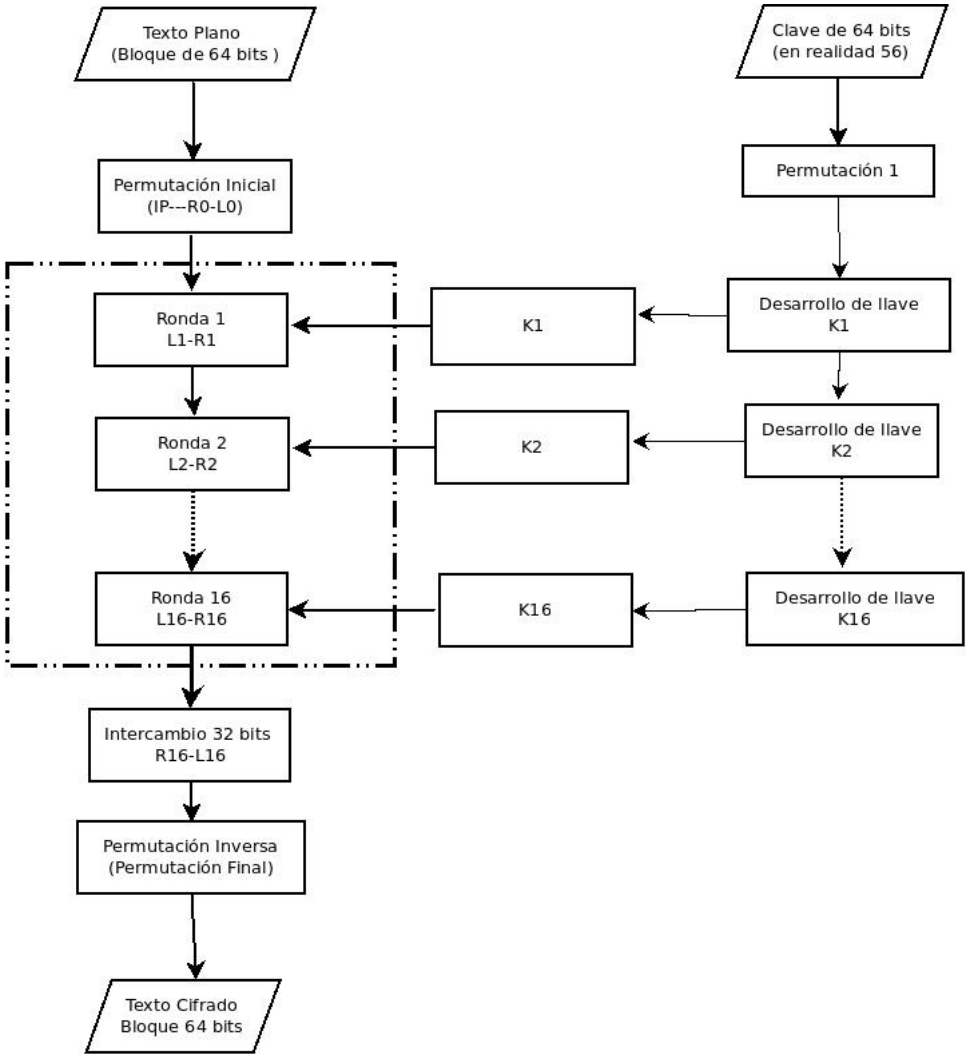


Figura 3.9: Algoritmo General DES

Las 16 etapas faltantes son 16 rondas que siguen el mismo desarrollo, el cual se muestra en la Figura 3.10. Donde después de la Permutación Inicial, el bloque de 64 bits se divide en 2, $L_{i-1} = 32$ bits y $R_{i-1} = 32$ bits. Como siguiente paso R_{i-1} se expande a 48 bits y se aplica una XOR con la llave correspondiente K_i, el resultado se parte en 8 bloques de 6 bits cada uno, los cuales nos ayudan a sustituir los datos por las *S-Cajas*, se concatena cada bloque que nos dio como resultado y se aplica una permutación, el siguiente paso es aplicar un XOR con L_{i-1}.

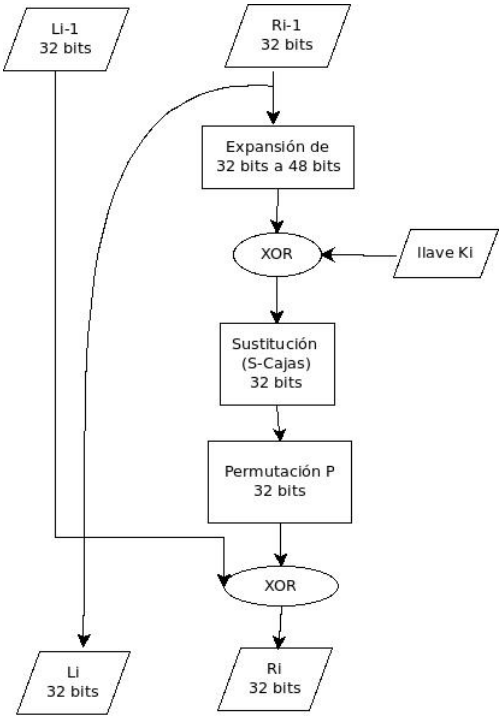


Figura 3.10: Rondas DES.

En cada una de las 16 iteraciones se aplica una llave K_i, la cual se obtiene a partir de la clave de 56 bits y es distinta en cada iteración. La generación de llaves se explica en la figura 3.11.

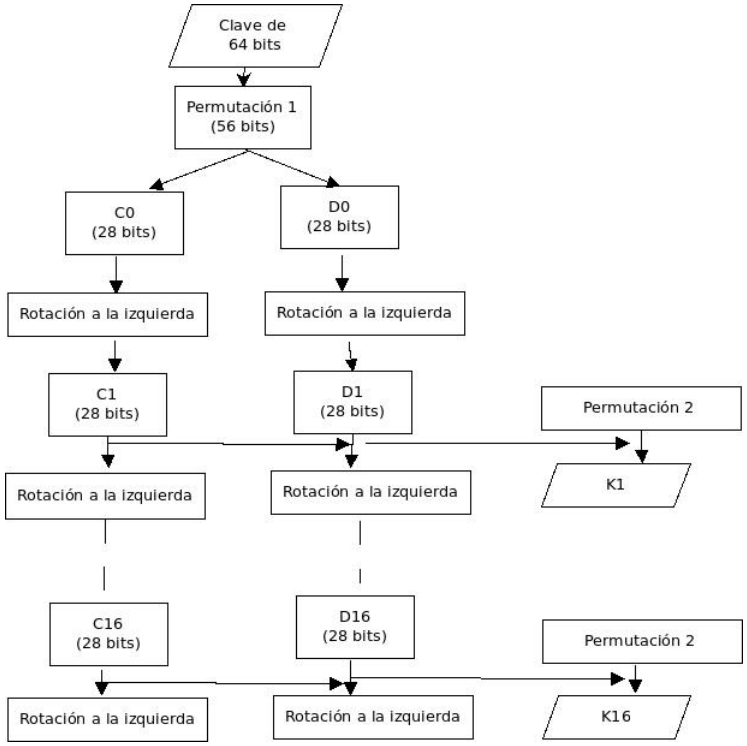


Figura 3.11: Generador de llaves DES.

AES (Advance Encryption Standard)

En el año 1997, el Instituto Nacional de Estándares y Tecnología de EEUU (NIST), emprende un proceso abierto para la selección de un nuevo algoritmo de cifrado que sustituya al antiguo estándar de cifrado (DES). Rijndael fue el algoritmo elegido después de pasar un periodo de análisis durante aproximadamente 3 años, Rijndael fue elegido como la mejor opción dentro de 15 candidatos, sus principales características fueron su fácil diseño y su versatilidad en ser implementado en diferentes dispositivos [29].

El bloque de cifrado de AES tiene una longitud de 128 bits, la longitud de la clave K varía de 128, 192 y 256 bits, en cada caso AES tiene 10,12, y 14 rondas respectivamente. Cada ronda consiste de 4 transformaciones básicas, la última ronda es especial y consiste de 3 operaciones básicas, añadiendo siempre una ronda inicial [25]. Por otro lado tenemos el programa de claves o extensión de la clave. Por lo anterior nuestra descripción consiste en describir las transformaciones básicas *AddRoundKey*, *SubByte*, *ShiftRows*, *MixColumns*, y por último el de *Key Schedule*.

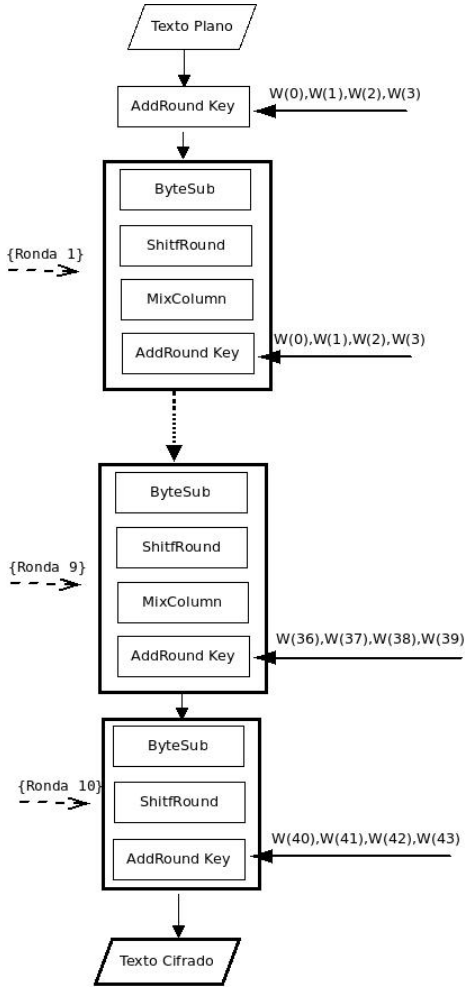


Figura 3.12: Algoritmo general de AES.

En la Figura 3.12, se muestra el caso donde el algoritmo AES tiene una entrada de texto plano de 128 bits, el cual es agrupado en 16 bytes de 8 bits cada uno, y estos se juntan en una matriz de 4x4.

Como primer paso se aplica una XOR con la matriz de llaves correspondientes dependiendo de la ronda en la que se encuentra, como primera ronda se aplica XOR con la matriz de llaves iniciales. A este proceso se le llama *Add RoundKey*.

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{pmatrix} \oplus \begin{pmatrix} k_{00} & a_{01} & k_{02} & k_{03} \\ k_{10} & a_{11} & k_{12} & k_{13} \\ k_{20} & a_{21} & k_{22} & k_{23} \\ k_{30} & a_{31} & k_{32} & k_{33} \end{pmatrix} \tag{3.1}$$

La matriz estado tiene 4 columnas y toma de la extensión de la clave también 4 columnas. El programa de claves genera las matrices de claves necesarias para todas

las rondas.

Como siguiente paso es la transformación *ByteSub*, donde cada uno de los bytes en la matriz es intercambiado por otro que pertenece a una tabla llamada *S-Box*. Se escribe cada byte como 8 bits: $b_0, b_1, b_2, b_3, b_4, b_5, b_6, b_7$, tomando como entrada de la fila b_0, b_1, b_2, b_3 y la entrada de la columna b_4, b_5, b_6, b_7 y la intersección seria el resultado de la salida.

Después es la transformación *ShiftRow* donde se aplica a la matriz estado, corrimientos a sus columnas. Sea aplica a la matriz aij shifts (corrimientos izquierdos circulares de bytes) a las renglones, de la siguiente manera, recorre 0 bytes a el primer renglón, 1 byte al segundo renglón, 2 bytes al tercer renglón, y 3 bytes recorridos al cuarto renglón (Ver Figura 3.13).

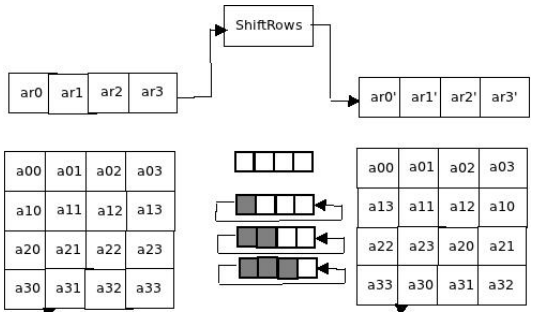


Figura 3.13: ShiftRow.

Posteriormente *MixColumns* opera columna por columna de la matriz, tomando a cada una como un polinomio de grado tres. Es decir, que *MixColumns* toma cada columna A, y la manda a otra columna A',que se obtiene al multiplicar A por un polinomio constante $c(x) = 3x_3 + 1x_2 + 1x + 2$. (Ver Figura 3.14)

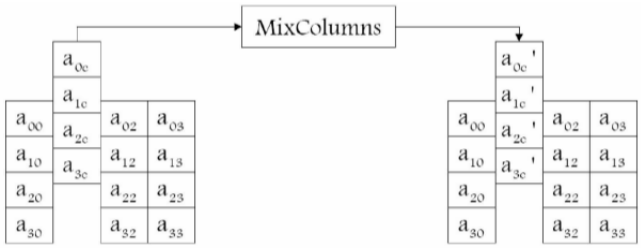


Figura 3.14: MixColumns.

Y se finaliza la primer ronda aplicando otra vez *AddRoundKey*, ahora con la

matriz resultado del paso *MixColumns* y haciendolo XOR con el conjunto de llaves derivado de la original. Estos pasos se repiten hasta llegar a las 10 rondas, 12 o 14, dependiendo de los casos.

Ahora en cada ronda se ocupan un conjunto de llaves que se calculan expandiendo la llave original, a este proceso se le llama *KeySchedule*. La llave original consta de 128 bits, por ejemplo, los cuales son acomodados en una matriz de 4x4 de bytes. A esta matriz se le añade 40 columnas más, con los siguientes pasos: Primero se colocan las primeras 4 columnas con la llave original , las nuevas columnas son generadas recursivamente. Supongamos que las columnas están definidas por $W(i-1)$, entonces si i no es múltiplo de 4:

$$W(i) = W(i-4) \oplus W(i-1) \tag{3.2}$$

Pero si i es múltiplo de 4 entonces:

$$W(i) = W(i-4) \oplus T(W(i-1)) \tag{3.3}$$

Donde *T(W(i-1)* es la transformación de *W(i-1)* obtenido de los siguientes pasos:

Por ejemplo si es W(4),tomaremos la columna W(3) e intercambiaremos el primer elemento con el último elemento como lo muestra la Figura 3.15.

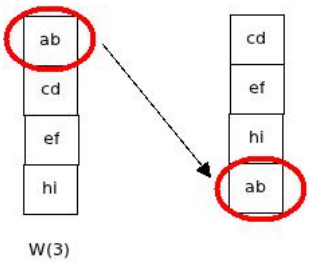


Figura 3.15: Rotación de elementos para llaves.

Como siguiente paso se intercambian los bytes por los bytes de la *S-box* del paso *ByteSub* y finalmente se aplica XOR con una *round constant*, la cual se calcula con la siguiente formula:

$$r(i) = 00000010^{\frac{(i-4)}{4}} \tag{3.4}$$

3.3.3. Modo de Operación: Modo Contador

Un bloque cifrado toma un bloque de longitud fija de texto de b bits de longitud y una llave como entrada y produce un bloque de b bits de texto cifrado . Si la cantidad de texto claro a cifrar es mayor que b bits, entonces el bloque cifrado puede todavía ser utilizado para romper el texto claro en bloques de B bits. Cuando hay varios bloques de texto en claro y se cifran utilizando la misma clave, una serie

de problemas de seguridad surgen. Para aplicar un bloque cifrado en una variedad de aplicaciones, cinco modos de funcionamiento se han definido por el NIST. En esencia, un modo de operación es una técnica para mejorar el efecto de un algoritmo criptográfico o adaptar el algoritmo para una aplicación, tales como la aplicación de un cifrado en bloque a una secuencia de bloques de datos o un flujo de datos. Estos modos están diseñados para utilizarse con cualquier sistema de cifrado simétrico de bloques, incluyendo triple DES y AES. Principalmente discutiremos del modo Contador (CTR).

Para el modo de operación contador se utiliza un contador igual al tamaño de bloque de texto claro. El único requisito es que el valor del contador debe ser diferente para cada bloque de texto claro que se cifra. Típicamente, el contador se inicializa a un valor y luego incrementa en 1 para cada bloque posterior. Para el cifrado, el contador se cifra y luego XOR con el bloque de texto plano para producir el bloque de texto cifrado, no hay encadenamiento. Para el descifrado, se utiliza la misma secuencia de valores de contador, con cada contador de cifrado XOR con un bloque de texto cifrado para recuperar el bloque de texto claro correspondiente. Por lo tanto, el valor inicial contador debe estar disponible para el descifrado [32]. En la Figura 3.16 se muestra el esquema completo del Modo de operación contador.

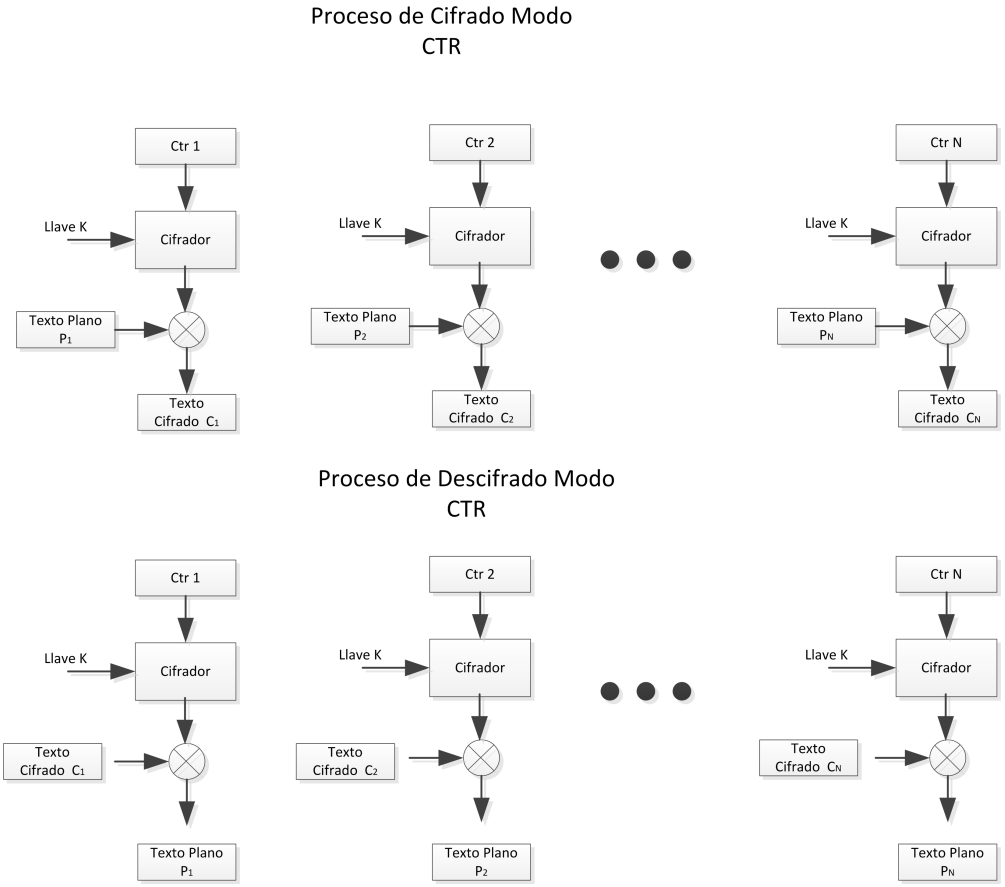


Figura 3.16: Modo CTR.

Ventajas

- Eficiencia del Hardware: A diferencia de los otros modos de operación, en el modo CTR se puede hacer en paralelo, múltiples bloques de texto plano. El rendimiento sólo está limitado por la cantidad de paralelismo que se pueda lograr.
- Eficiencia del Software: Debido a las oportunidades para la ejecución en paralelo en el modo CTR, los procesadores que soportan funciones paralelas, entregan instrucciones por ciclo de reloj por lo tanto se puede lograr eficiencia.
- Simplicidad: A diferencia de BCE y modos CBC, el modo CTR requiere sólo la aplicación del algoritmo de cifrado y no el algoritmo de descifrado . Esto es lo más importante cuando el algoritmo de descifrado difiere sustancialmente del algoritmo de cifrado, como lo hace para AES .

3.4. Resumen de contenido

Como primer punto se describió los protocolos de comunicación que ayudan a la transmisión de video. Se estableció la diferencia de los protocolos TCP y UDP,

siendo UDP el ques utilizado para transmitir video, debido a que no introduce retardos para establecer una conexión, no mantiene estado de conexión alguno y no realiza seguimiento de estos parámetros. También se hablo de las características del Protocolo de Transporte de Tiempo-Real el cual se utilizó para realizar éste trabajo y se explicará en el siguiente capitulo.

Por otro lado se explicaron los codecs más utilizados, donde destacó el codec H.263, por que es el codec que se emplea para la transmisión de video en directo, por lo que se describió los componentes que contiene ese codec.

El último punto de este capítulo, fue la descripción de los algoritmos de cifrados simétricos. Tomando en cuenta solo el algoritmo DES y AES debido a que los esquemas encontrados en la literatura utilizaban estos algoritmos. También se explicó el modo de operación contador, el cual convierte una unidad de cifrado por bloques en una unidad de flujo de cifrado. Genera el siguiente bloque en el flujo de claves cifrando valores sucesivos de un contador.

En el capítulo se explica el desarrollo del diseño del esquema de cifrado, utilizando los conceptos que se vieron en este capítulo.

Capítulo 4

Desarrollo del Esquema de cifrado

En este capítulo se describe detalladamente todo lo relativo al proceso de diseño y desarrollo del esquema de cifrado, al igual el establecimiento de los requisitos y se discutirán las distintas decisiones para realizar las pruebas al esquema.

4.1. Requisitos tomados en cuenta para el Esquema de Cifrado

A continuación se muestran los requisitos, obtenidos teniendo en cuenta las necesidades del proyecto y las limitaciones:

- **Cifrado baja latencia**: Al querer cifrar la información en una transmisión de video, es necesario contar con un esquema de cifrado que no afecta la percepción del usuario, es decir, no afecte el tiempo de transmisión y recepción.
- **Costo (Hardware)**:Se requiere aprovechar todos los recursos que se tienen en nuestra plataforma Beagleboar-xM, sin necesidad de aumentar hardware adicional a ésta, debido a que se pretende simular un dispositivo móvil.
- **Algoritmo de Cifrado AES**: Es un estándar de cifrado por el gobierno de los Estados Unidos y actualmente es seguro de modo que un ataque contra el AES de llave de 128 bits, requiera "sólo" de 2^{120} operaciones sería considerado como un ataque que "rompe" el AES por el momento aún es imposible.
- **Códec de video H.263+**: Se usará el códec H.263+ debido a que es el códec mayor empleado en transferencias de video en dispositivos móviles, además que es de código abierto bajo la biblioteca *libavcodec*.

4.2. Arquitectura del Esquema de Cifrado

La Figura 5.1 muestra la arquitectura del sistema, la cual se divide en dos partes:

- Usuario A

  Inicia la grabación del video por medio de una cámara web, que está conectada a un sistema embebido corriendo con una distribución de Linux, este video es capturado y procesado por el cifrador y enviado al usuario B por medio de la conexión Ethernet.

- Usuario B

  Recibe el stream de video, decodifica y descifra los datos para que el Usuario B pueda visualizar el video.

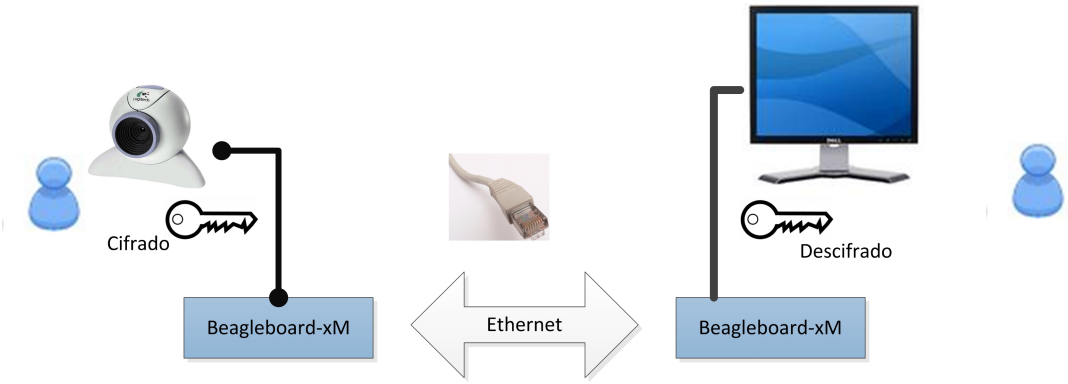


Figura 4.1: Arquitectura del Esquema de cifrado

4.3. Diseño del Esquema de cifrado

- Detalles de Hardware

  Se decidió utilizar la tarjeta Beagleboard-xM rev. C, la cual es una plataforma de desarrollo de bajo costo y es ampliamente usada en el área de los sistemas embebidos a nivel mundial. Contiene numerosas funcionalidades dado el gran número de periféricos que posee. El sistema consta de un procesador DM3730 (ARM cortex A8 de 1GHz de frecuencia como procesador de propósito general y un DSP C64P de 800 MHz de frecuencia) y de una memoria extra DDR RAM de 512MB. (Ver Figura 4.2)

  Entre los periféricos disponibles en la Beagleboard-xM se tiene:

  - DVI-D
  - S-Video
  - USB (host y OTG)
  - Ethernet

- Entrada y Salida de Audio
- JTAG
- R232
- Puerto para microSD

El Procesador DM3730 puede soportar varios sistemas operativos tales como distribuciones Linux (Angstrom y Ubuntu), Windows CE y Android, además esta tarjeta de desarrollo trabaja con una fuente de poder de 5 volts DC.

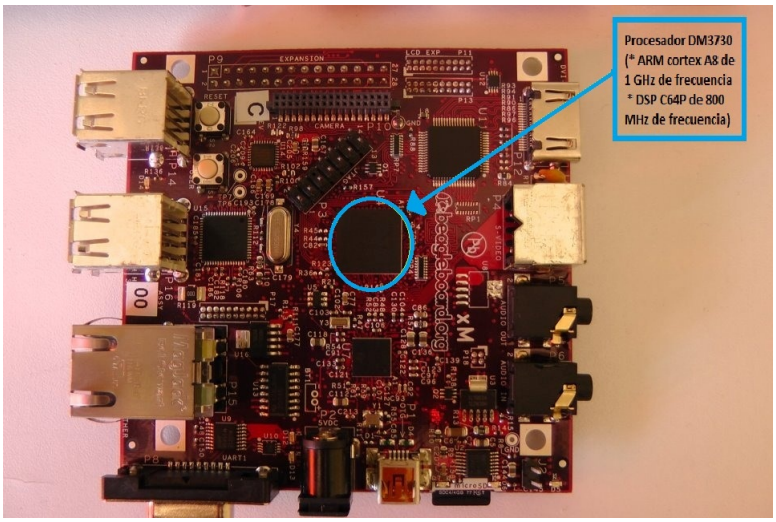


Figura 4.2: Beagleboard-XM rev. C

- Software

  Δ Sistema Operativo

  A fin de utilizar al máximo el potencial de la tarjeta Beagleboard-xM, se puede instalar el sistema operativo Linux o Windows CE, los cuales incluyen los drivers para ser capaz de utilizar la mayoría de sus periféricos. Como parte del diseño se escogió para el procesador DM3730 el Sistema Operativo Linux debido a que la comunidad Beagleboard Linux es muy activa, es decir, es un poco más fácil de obtener ayuda en algún problema y además la mayoría de los programas que se pueden instalar en él son de código abierto y permiten modificarlos.

  Δ Aplicación

  Teniendo elegido el Sistema Operativo Linux, se buscó una aplicación con las características de transmisión de video, por lo que se eligió una biblioteca multimedia de comunicación libre y de código abierto llamada

PJSIP[1]. Está escrita en lenguaje C e implementa protocolos como SIP, SDP, RTP, STUN e ICE. Es compatible con audio, video y mensajería instantánea y cuenta con una extensa documentación.

PJSIP consiste de varias bibliotecas que interactúan como se muestra en la Figura 4.3. Además que contiene varias aplicaciones como: vid_streamutil para transmitir video, streamutil para transmitir audio ó playfile para reproducir un archivo multimedia. Para este trabajo se utilizó la aplicación vid_streamutil.

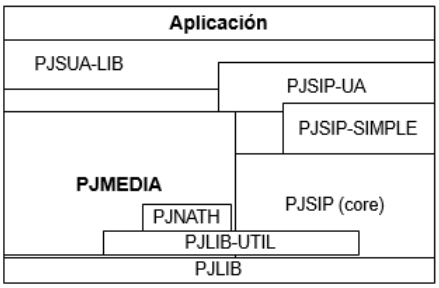


Figura 4.3: PJSIP

- Diseño del diagrama de Cifrado

Por otro lado para cifrar los datos, en este proyecto se decidió utilizar el estándar AES en modo contador. Como se ha mencionado en la sección 3.3.3, AES puede trabajar con tres longitudes de llave diferentes: 128, 192 y 256 bits, pero en esta implementación se ha optado por utilizar una llave de 128 bits. Debido a que sí se utiliza llaves de mayor longitud requiere más procesamiento en el cifrado y esto podría afectar la transmisión de video por los retrasos que tendría. Además se utilizo el AES en modo contador, porque a diferencia de otros modos de operación, éste no necesita de los bloques anteriores y por lo tanto sí se pierde un paquete puede seguir funcionando. A continuación se explica cómo se diseño del esquema de cifrado.

El proceso de la transmisión de datos del video es como se muestra en la figura 4.4, el programa se divide en el emisor y el receptor, la parte del remitente recoge información de vídeo y envía datos al codificador de H.263+ para codificar los datos, cifra los datos del video, empaqueta a RTP, y el paquete es transmitido por la red. El receptor recibe datos de vídeo RTP, desempaqueta RTP, descifra los datos del video y los envía al decodificador H.263+ para la decodificación de nuevo.

[1]http://www.pjsip.org/

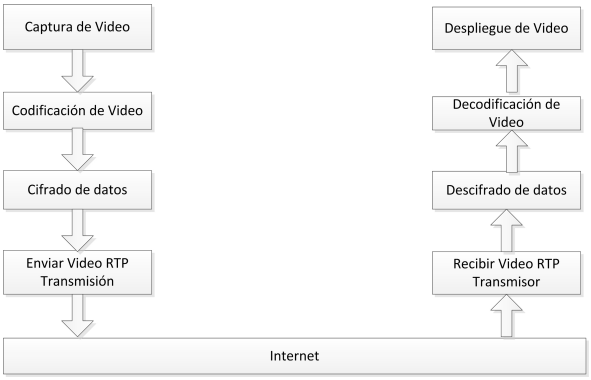


Figura 4.4: Proceso de transmisión de datos de video.

Dado la explicación anterior, en este proyecto se trabaja con el paquete de RTP, el cual es conformado por un header con los campos que se describieron en la sección 3.1.4, siguiendo con el payload de RTP, que en este caso es conformado por el header H.263+ y su payload del codec H.263+ (ver Figura 4.5).

Figura 4.5: Paquete RTP

Como se describió en la 3.2.5, el header de H.263+ cuenta con un campo llamado Referencia Temporal, el cual es de 8 bits y es el único campo que al ser transmitido el receptor lo recibe con el mismo valor[2]. Debido a que el codificador entrega los datos en bytes y los bits correspondientes a la referencia Temporal no correspondía al inicio del byte, se decidió tomar los últimos 6 bits del StartCode de la cabecera de video y 2 bits no utilizados después de la Referencia Temporal obteniendo el total de 2 bytes, comprobando que al ser transmitidos se reciban sin ningún cambio. Por otra parte, en el header de RTP también existe otro campo con las mismas características, el TimeStamp, este es de 32 bits, los dos campos son utilizados para crear nuestro vector T del cifrador y así no perder la sincronía entre paquetes. Como se muestra en la figura 4.6.

[2]Por cada paquete transmitido el campo Referencia Temporal es diferente

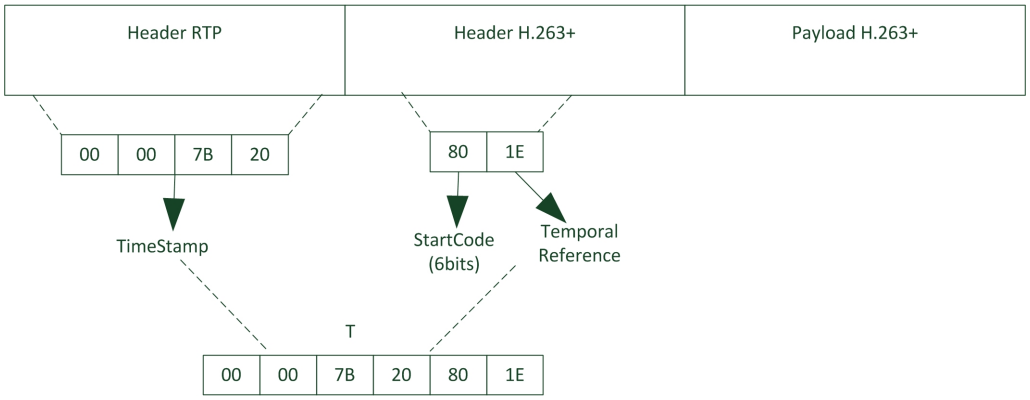


Figura 4.6: Vector compuesto por TimeStamp y la Referencia Temporal

En el cifrado AES modo contador, la entrada de datos es de 128 bits por lo que se decidió concatenar el vector T (mencionado anteriormente de 48 bits) con un vector inicial de 64 bits y un contador de 16 bits y en la salida, entrega un vector también de 128 bits. Sin embargo en este caso, cada frame transmitido es de 1280 bytes, por lo que, se necesita un vector de llaves de 1280 para poder aplicar la operación XOR, por lo que este proceso se tiene que repetir 80 veces para que se obtengan 80 vectores de 16 bytes de longitud y así tener un vector final de 1280 bytes. Entonces por cada frame que se reciba se obtendrá el vector T y se concatenará con el vector inicial y el contador, este último se aumentará a uno, para generar otro bloque a cifrar, hasta llegar a generar 80 bloques cifrados y así poder concatenarlos para formar el vector de 1280 bytes y por último aplicar la función XOR con el frame que haya sido transmitido, este proceso se realizara cada vez que un frame sea transmitido. La figura 4.7 muestra el esquema de cifrado completo.

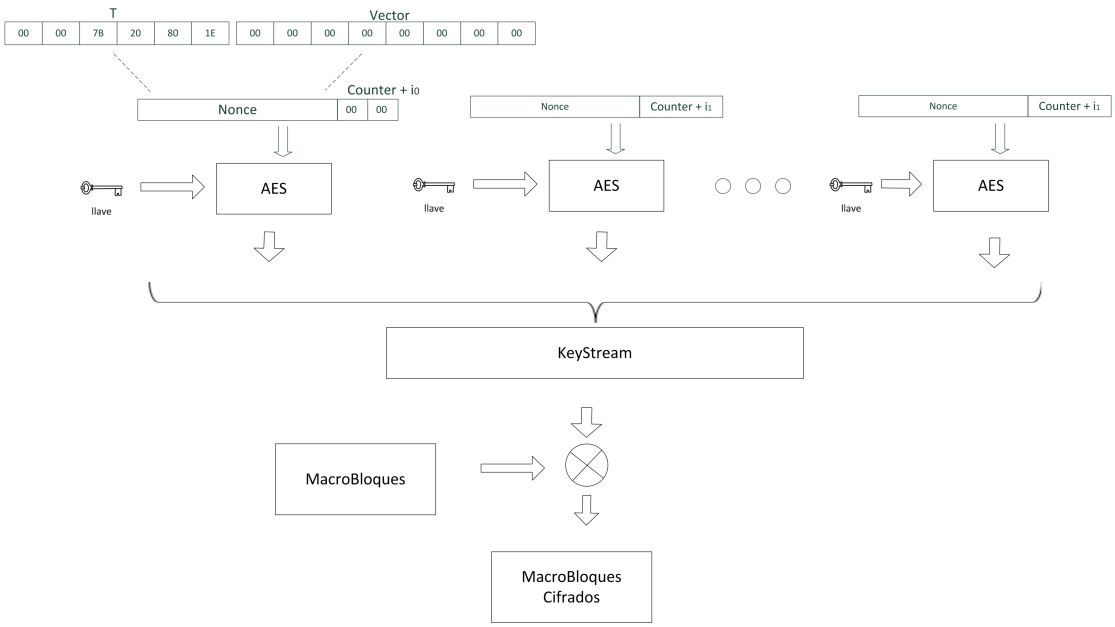


Figura 4.7: Cifrado

El esquema anterior muestra que se aplica la operación XOR con el vector generado de las llaves con un vector llamado Macrobloques. Este vector se pretende llenar con los valores de los macrobloques de cada frame transmitido de H.263+, la cual su estructura se explicó anteriormente en la sección 4.2.3 de este trabajo, debido a eso se decidió solo cifrar está parte (macrobloques) del códec H.263+.

El el caso del descifrado, se realizó las mismas operaciones que cuando se cifro el video. Primero se recibió el paquete RTP que envió el transmisor. Del paquete que llegue se obtiene el TimeStamp y la Referencia Temporal de los encabezados correspondientes. Se procederá a obtener el vector de 1280 bytes con el AES modo contador y este vector se le aplica la operación XOR con los datos recibidos como macrobloques (los cuales se reciben cifrados).

4.4. Construcción del ambiente de pruebas para el esquema de Cifrado

Presentado el diseño del esquema de cifrado, se prosigue con la construcción del ambiente de pruebas para este esquema. Como se utilizará la tarjeta Beagleboard-xM y como se vieron en las especificaciones de ésta, cuenta con un DSP, el cual se tomó la decisión de usarlo como generador de llaves para nuestro cifrado.

En la Figura 4.8 se encuentran las principales etapas y el orden que se seguirá para explicar la construcción del ambiente de pruebas.

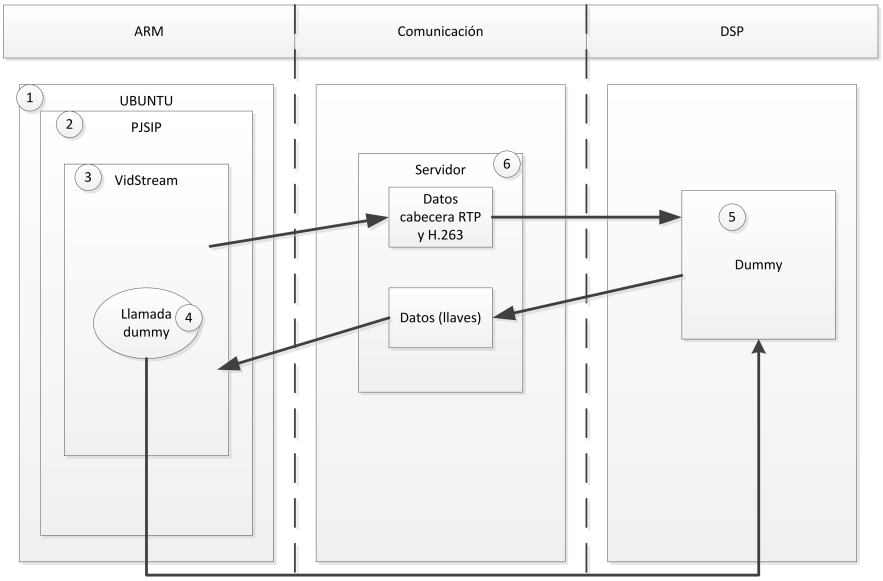


Figura 4.8: Diagrama de Pruebas

Como primer paso se explicará el software que se instaló en la tarjeta para que este proyecto funcionará, iniciando con la instalación del Sistema Operativo LINUX, la figura 4.9 muestra su estructura.

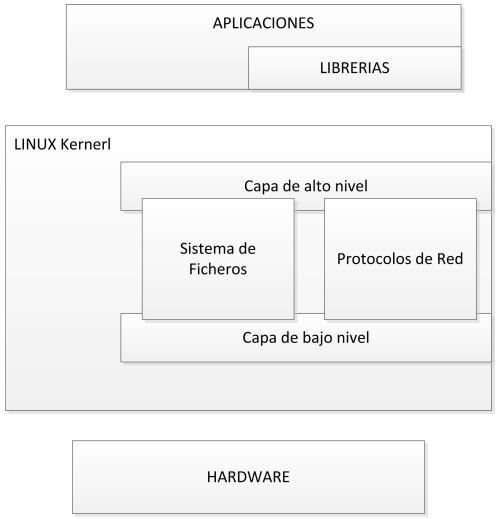


Figura 4.9: Estructura de Linux.

Como se puede observar inmediatamente después del hardware se encuentra el núcleo (kernel) del sistema. El kernel es el componente básico del sistema operativo y su principal función es gestionar el hardware de una manera coherente proporcionando facilidad para el usuario manejarlo desde software. Los módulos del kernel son partes de código que pueden ser enlazados dinámicamente con el núcleo del sistema. La mayor parte de los módulos del kernel Linux son drivers

de dispositivos o manejadores de sistemas de ficheros, en este caso se insertó un módulo que ofrece soporte para sincronizar y comunicar el ARM y el DSP llamado DSP-Bridge.

DSP-Bridge fue desarrollado por Texas Instruments, el código fuente de parte del ARM es abierto mientras el código para el lado del DSP es completamente cerrado. Por esta razón se decidió utilizar un script *ti-tidspbridge.sh* el cual se encarga de habilitar el DSP-Brigde en el kernel. Además este script carga una imagen Bios que permite la ejecución de instrucciones base. (Ver Figura 4.10)

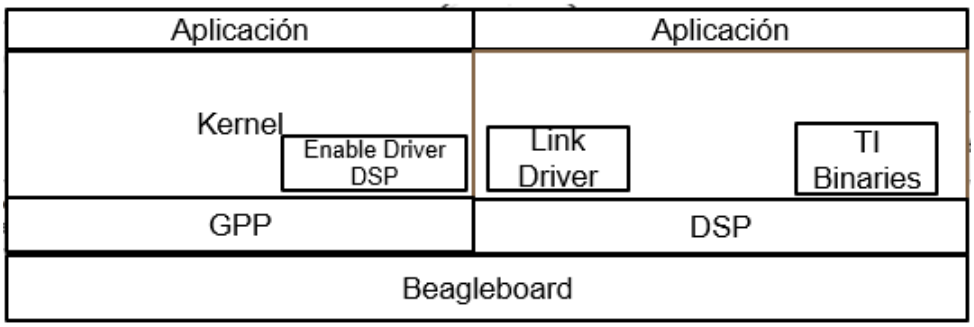


Figura 4.10: Dspbrigde en kernel.

Además es importante resaltar la dependencia del módulo DSP_Bridge con respecto a la versión del kernel sobre el que se pretende insertar, generalmente cuando se utiliza una herramienta de software, el fabricante proporciona el código fuente de la misma y ficheros *makefile* que contienen las órdenes para generar el módulo, una de esas órdenes hace refencia al código fuente pertenenciente al kernel que se está ejecutándose en el sistema embebido, lo que obliga que al compilar los módulos del kernel para la versión exacta del kernel empleado en la tarjeta ya que de lo contrario no será posible emplear las funcionalidades de éste.Debido a esto el kernel que se utilizó fue la versión 2.6.37, el cual permite que se pueda utilizar el driver del DSP.

Como segundo paso en el Procesador de Propósito General (GPP) es la instalación de la aplicación PJSIP. Para que esta funcione sin ningún problema, primero es necesario instalar dos bibliotecas: FFmpeg y SDL.

1. FFmpeg
   Es una solución completa, multi-plataforma para grabar, convertir y transmitir audio y vídeo. Incluye la biblioteca libavcodec la cual es la biblioteca principal para codificador de audio/video.

2. SDL
   Simple DirectMedia Layer es una librería de desarrollo multiplataforma diseñado para proporcionar acceso a bajo nivel de audio, teclado, ratón,

joystick, y el hardware de gráficos a través de OpenGL y Direct3D. Es utilizado por el software de reproducción de vídeo, emuladores y juegos populares.

SDL soporta oficialmente Windows, Mac OS X, Linux, iOS y Android. Además la documentación para otras plataformas se pueden encontrar en el código fuente.

SDL está escrito en C, funciona de forma nativa con C++, y hay enlaces disponibles para varios otros lenguajes, incluyendo C# y Python.

Se instaló PJSIP en la tarjeta Beagleboard-XM y se compila dentro de ésta. Como se apreció en la fig. 4.3 PJSIP está compuesta de varias bibliotecas internamente, una de ellas es PJMEDIA, que como se mencionó anteriormente, interactúa con el stream de video. A continuación se muestra en la figura 4.23 el flujo del stream de video en PJSIP.

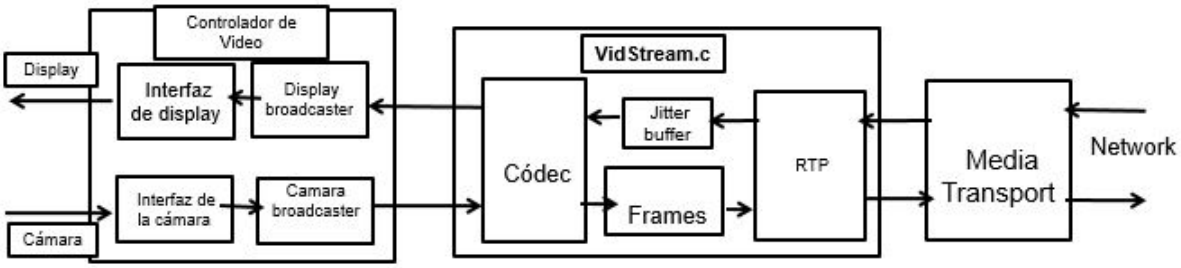


Figura 4.11: Flujo de stream de video en pjsip.

El archivo VidStream.c contiene los elementos necesarios para la transmisión de video, principalmente es donde se codifica y se manda al paquete RTP. Por lo que se decidió identificar en ese archivo dónde se obtenía el stream del video, fueron dos opciones las que se encontraron que son las siguientes funciones: (Figuras 4.12 y 4.23)

- Codec

```
status = pjmedia_vid_codec_encode_begin(stream->codec, &enc_opt, frame,
                                        channel->buf_size -
                                           sizeof(pjmedia_rtp_hdr),
                                        &frame_out,
                                        &has_more_data);
```

Figura 4.12: Función videocodec.

- RTP

```
status = pjmedia_transport_send_rtp(stream->transport,
                                    (char*)channel->buf,
                                    frame_out.size +
                                        sizeof(pjmedia_rtp_hdr));
```

Figura 4.13: Función sendRTP.

La primera opción se descartó debido a que si se accedía al video directamente del codec y se aplicaba el cifrado, al recibirlo (el video), el decodificador no entendía todos los elementos que llegaban por lo que la imagen resultaba ser borrosa.

Por lo cual se eligió la segunda opción en donde como se indicó el diseño anteriormente, se obtenía el TimeStamp y la Referencia Temporal[3] , el paquete RTP se encuentra dentro de la variable *channel->buf* la cual es el paquete a enviar. De esta variable se obtiene los datos necesarios para trabajar con el esquema de cifrado, a partir del byte 4 al 7 es el TimeStamp y del byte 14 al 15 se obtiene la Referencia Temporal, estos datos se guardan en una variable del tipo unsigned char llamada Vector de tamaño de 6 bytes.

Por otro lado de la misma variable *channel->buf* se obtienen los macrobloques, que son a partir del byte 80. Estos son guardados en otra variable del tipo unsigned char llamada Buffer para la facilidad del cifrado.Después de obtener todos los datos necesarios para cifrar, se necesitaba llamar al DSP y enviarle estos datos, por lo tanto nos auxiliamos de la aplicación *Dsp for Dummy* la cual más adelante se explicará con detalle.

En este proyecto lo que se necesitaba era que la aplicación fuera dinámica, es decir, cada vez que un frame llegara el Dsp for Dummy se activaría y generaría las llaves para el cifrado.

Para que ésta fuera dinámica se mandaba a llamar la aplicación dentro de la clase VidStream.c mediante la llamada de procesos System.

La función System es la forma más sencilla de invocar una orden UNIX desde un programa en C, la cual toma como único parámetro la orden que se quiere ejecutar. Esta función ejecuta la cadena como parámetro tal y como si la hubiéramos tecleado desde la consola (Ver Figura 4.14).

```
system ("cd /home/ubuntu/DSP/; ./dummy");
```

Figura 4.14: Llamada a la aplicación DSP For Dummy

[3]La Referencia Temporal se obtiene del encabezado del codec H.263 (Fig. 3.6)

Al terminar de ejecutarse la aplicación Dsp for Dummy, ésta entregará el vector de llaves de tamaño 1280 bytes el cual se le aplica la operación XOR con el vector de macrobloques del frame. Después el resultado de esta operación se vuelve a almacenar en la variable *channel->buf*, para así enviar el paquete.

Siguiendo con la secuencia de la Figura 4.8, se explicará la aplicación llamada Dsp for Dummy, con la cual ciframos en el DSP. Para poder utilizar esa aplicación y compilarla fue necesario instalar las herramientas para el DSP (C64x), se emplearon las denominadas "Code Generation Tools" en su versión C6000, del fabricante Texas Instruments, que contiene las utilidades típicas para la construcción del ejecutable en este caso la aplicación de Dsp for dummy.

DSP for Dummies (DFD) es un Framework diseñado para facilitar a los desarrolladores la interacción entre sus programas y el DSP. La codificación del DSP se presenta en dos partes: el binario que se ejecuta en el DSP y el binario que se ejecuta en el procesador ARM.

- El binario que se ejecuta en el DSP se encarga de la coherencia de la caché, del análisis de mensajes, etc, y presenta una interfaz sencilla para el programador. Lo que se hace para que funcione es que se edite el archivo dummy_dsp.h (que viene en la carpeta de archivos del proyecto) y se añade la función que se desea correr en el DSP (en este caso la función que llama el AES). Cada función está asociada con un código de operación de mensaje, que también se define en el archivo. La función toma un buffer DMA (Direct Memory Access) de entrada, buffer DMA de salida, una estructura variable global que se comparte entre todas las funciones, y dos argumentos de 32 bits que se transmiten desde el software. DFD también proporciona una función opcional "ociosa" que se ejecuta cuando el DSP está a la espera para que los mensajes lleguen.

- Para el binario del ARM, es necesario incluir algunos archivos y llamar el código de inicialización del DSP. Para todo lo demás se escribe el código como de costumbre, la función main() se encuentra en dummy_arm.c.

En la siguiente figura 4.15 se muestra la interacción entre el DSP-ARM mediante la aplicación DFD, cuando se desea realizar el trabajo en el DSP, se debe de escribir los datos al buffer DMA de entrada y se escribe el "*mensaje*" para el DSP. Este mensaje es una estructura (struct dsp_*msg*) que consta de un comando y dos argumentos (arg1,arg2) para que la función *DSP_función* (especificado en dummy_dsp.h) realice las operaciones corresspondientes. Luego se llama la función *dsp_send()* para enviar los datos y mensaje al DSP.

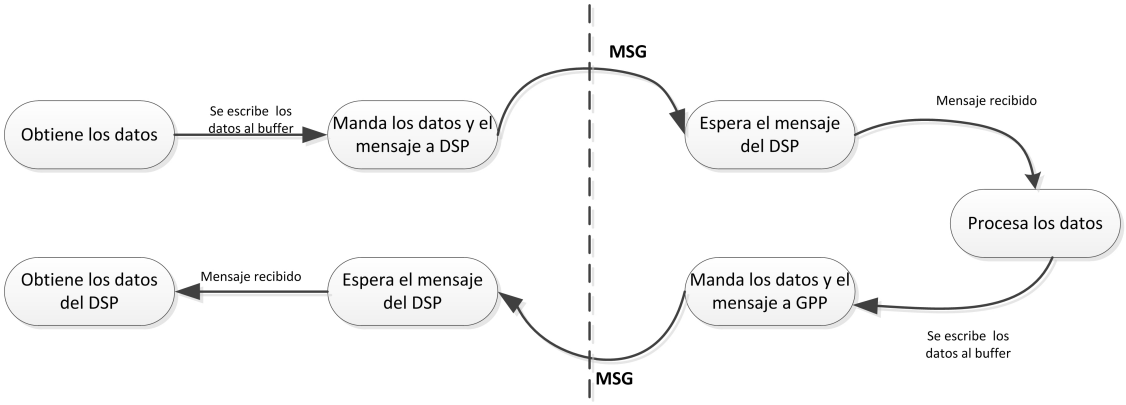


Figura 4.15: Arquitectura DSP-GPP

Además el DFD implementa dos tipos de funciones de recepción: una función de bloqueo de recepción y un no-bloqueo de recepción. El primero espera hasta que el DSP complete la operación y regrese antes de continuar la ejecución y el segundo recibe retornos instantáneamente junto con un valor booleano. El valor booleano indica si el mensaje ya ha sido recibido, si no es así, el procesador ARM puede seguir haciendo su trabajo hasta que el DSP pueda completar su tarea, esto permite ejecutar fácilmente el procesador ARM y DSP en paralelo por lo que maximiza su rendimiento.

Ahora para la ejecución del AES en el DSP como se mencionó anteriormente se modificó el archivo dummy_dsp.h. Este archivo está estructurado de la siguiente manera (Fig.4.16):

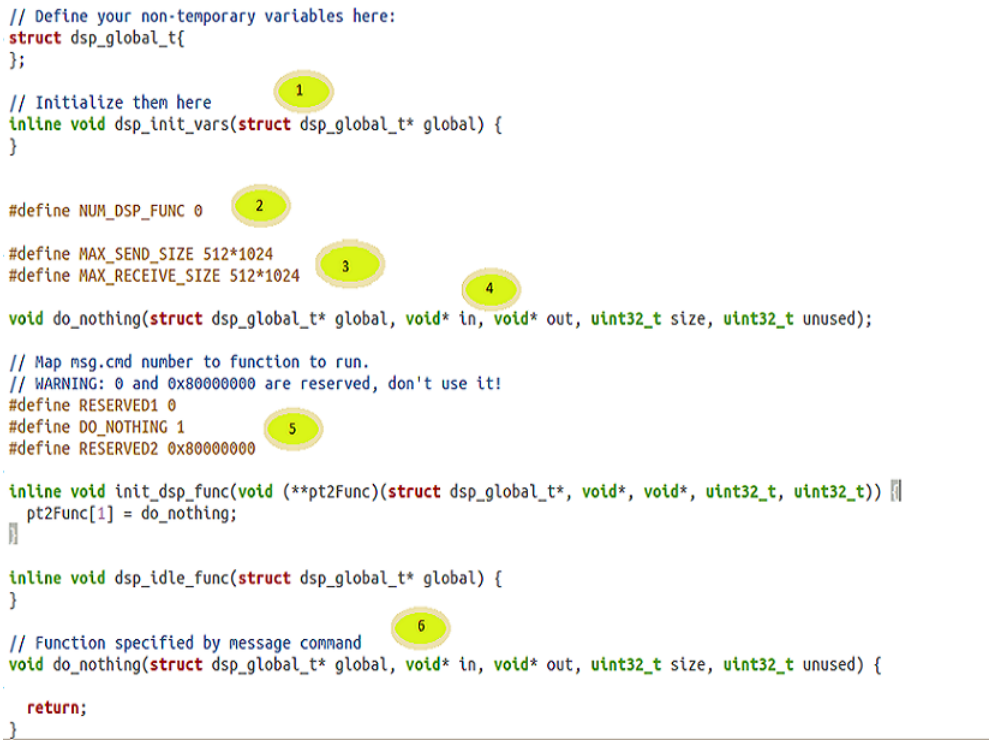


Figura 4.16: Archivo dummy_dsp.h

1. Se inicializan las variables en este espacio, en este caso para evitar más procesamiento en el DSP, se utilizó el precálculo de la S-box para el cifrado del AES.

2. Se define el número de funciones que se quieren correr en el DSP, en este caso son dos funciones debido a que una es la función general que manda a llamar a las funciones del AES y el otro es la función donde espera la entrada de más datos en el DSP.

3. Se declará el valor máximo del tamaño de los buffers DMA de entrada y salida, el cual es de 512KB.

4. Declaración de las funciones, dependiendo del número que se mande en el mensaje y el que tenga en sus argumentos declarados, el buffer DMA de entrada y salida, es la función a ejecutar. En este caso se declara como principal la función del cifrado AES, tal como lo muestra la Figura 4.17

```
void aes_cifrado(struct dsp_global_t* global, void* in, void* out, uint32_t size, uint32_t unused);
```

Figura 4.17: Declaración de la Función AES

Además en esta parte también fueron declaradas las funciones auxiliares que nos ayudan para implementar el cifrado AES, en nuestro caso por

cada transformación básica del AES se realizó una función, pero no solo se declararon, sino que como son funciones auxiliares y llamadas por la función general, en esta parte también son desarrolladas. Las funciones son (Fig. 4.18):

```
void AddRoundKey(unsigned char state[][4], unsigned int  w[])   //ADD ROUND KEY
void SubBytes(unsigned char state[][4]) //SubBytes
void ShiftRows(unsigned char state[][4]) // ShiftRows
void MixColumns(unsigned char state[][4])  //MixColumns
unsigned int SubWord(unsigned int  word)
void KeyExpansion(unsigned char key[], unsigned int  w[], int keysize) //KEY EXPANSION
void a_encrypt(unsigned char in[], unsigned char out[], unsigned int  key[], int keysize) //AES CIFRADO
```

Figura 4.18: Declaración de las Funciones Auxiliares para realizar el AES

Cada función realiza las operaciones correspondientes del AES de acuerdo a su nombre mientras que la función *a_ encypt()* recibe el vector a cifrar y lo transforma en una matriz de 4x4 bytes y manda a llamar cada una de las funciones anteriores pasando esta matriz llamada *state* en la primera etapa del AES, a partir de esta, las funciones son llamadas hasta completar las 10 rondas del AES de 128 bits (ver figura 4.19). En la función *KeyExpansion()* si recibe el vector de 16 bytes que es la llave que se entrega y se crea una matriz de 4x4 bytes y como salida entrega una matriz de 4x4 bytes.

```
AddRoundKey(state,&key[0]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[4]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[8]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[12]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[16]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[20]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[24]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[28]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[32]);
SubBytes(state); ShiftRows(state); MixColumns(state); AddRoundKey(state,&key[36]);
SubBytes(state); ShiftRows(state); AddRoundKey(state,&key[40]);
```

Figura 4.19: Rondas del AES

5. Se define las localidades de memoria que son reservadas y el nombre de la función a ejecutar con el número correspondiente que contiene el mensaje que se enviará.

6. Se declaran y desarrollan las funciones importantes, tales como *aes_ cifrado()*, donde como entrada tiene el vector T (que se había mencionado en el diseño del Esquema de cifrado) para generar el nonce e invoca la función *a_ encrypt* 80 veces para generar el stream de llaves de 1280, ademas la llave es un vector fijo que se encuentra en esa función y es la entrada de la función *KeyExpansion()*.

Por último se explicará cómo se realizó la comunicación entre el DSP for dummy y la aplicación PJSIP, para esto se buscó la forma más adecuada para la transmisión de datos entre aplicaciones y se probó por tres caminos:

- Archivos
- Memoria Compartida
- Socket

La primera opción, Archivos, se creó un Archivo desde la clase VidStream.c donde se escribía los datos y finalmente se cerraba, del lado del DSP se abría el Archivo para obtener los datos y volver escribir en él, la desventaja de este método es que era muy lento el proceso de abrir y cerrar archivos y esto provocaba que se parará la transmisión del video, por lo que se descartó la idea.

La segunda opción, Memoria Compartida, se intentó utilizar, pero en las funciones que corren en el DSP son muy limitados, entonces funciones como *int shmget()* no existían en el lado del DSP y no habia bibliotecas que nos ayudaran a correr estas funciones.
Por último se decidió utilizar las funciones de Sockets para realizar una aplicación de Servidor-Cliente, es decir, se creó primero una aplicación Servidor que siempre está escuchando sí hay alguna conexión y si algún cliente envía petición, éste responde. Y después como clientes la clase Videstram.c y el DFD. Las características del Socket son:

- El dominio que se utilizó fue AF_UNIX debido a que será un socket interno de UNIX (Sockets del sistema de archivos).
- El Tipo es SOCK_STREAM ya que es implementado por conexiones TCP/IP
- Puerto a utilizar 5001.

```
sockfd = socket(AF_UNIX, SOCK_STREAM, 0);
if (sockfd < 0)
{
    perror("ERROR abriendo socket");
    exit(1);
}
/* Inicializar la estructura del socket */
bzero((char *) &serv_addr, sizeof(serv_addr));
portno = 5001;
serv_addr.sin_family = AF_UNIX;
serv_addr.sin_addr.s_addr = INADDR_ANY;
serv_addr.sin_port = htons(portno);
```

Figura 4.20: Parte del código de implementación del Servidor

El funcionamiento es: primero el Servidor es ejecutado antes de que inicie la aplicación de PJSIP y se queda escuchando, la aplicación inicia y al momento de obtener el TimeStamp y la Referencia Temporal, en el VidStream.c inicia el proceso

del cliente, éste se conecta al Servidor se envía las dos variables anteriores y cierra la conexión y llama la aplicación de DFD. Al iniciar la aplicación DFD ésta inicia la conexión con el servidor y éste le envia los datos al DFD, se hace el proceso de generación de llaves y se lo envía al servidor y cierra conexión. Enseguida finaliza la aplicación DFD y continua la aplicación PJSIP, y se vuelve a conectar el cliente de la clase VidStream y el servidor le envía el vector de llaves y cierra la conexión y continua con el proceso de cifrado. Esto se realiza cada vez que se transmite un frame. En la figura 4.21 se muestra el flujo de datos entre Servidor y cliente.

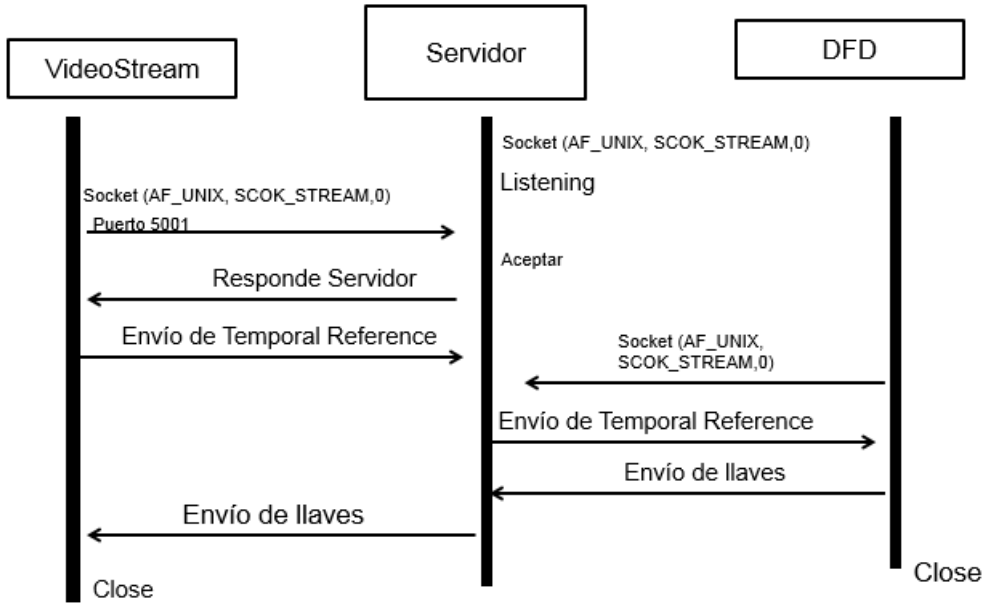


Figura 4.21: Diagrama de la comunicación entre Dummy y VidStream

Por otro lado para obtener los ejecutables para el ARM del Beagleboard-xM fue necesario disponer de un conjunto de herramientas que permitan obtener el ejecutable ARM desde una arquitectura con procesador x64 Intel, esto es, un entorno de compilación cruzada a través de la cadena de herramientas empleada. Para este proyecto se ha elegido la ***toolchain*** del fabricante **Codesourcery** para arquitectura ARM. El esquema del entorno de compilación cruzada que se utiliza se muestra a continuación: (Fig.4.22)

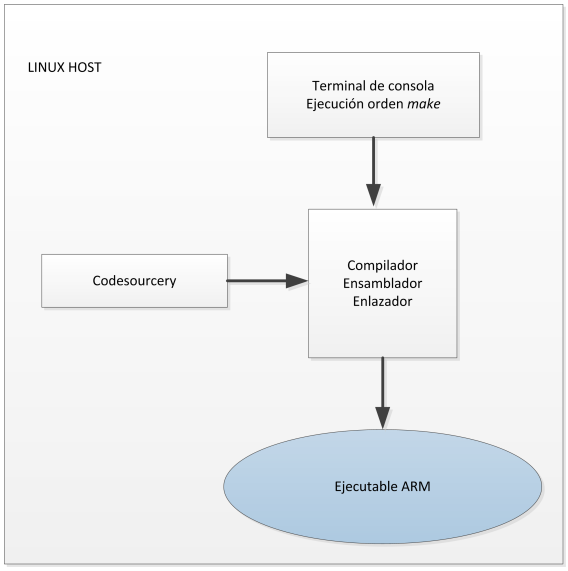


Figura 4.22: Entorno de compilación cruzada

Después de que se instaló el *toolchain* Codesourcery para que se pueda compilar una aplicación solo se debe de ejecutar el siguiente comando:

$\$arm-none-linux-gnueabi-gcc\ -o\ servidor\ servidor.c$

Entonces una vez que se terminó el programa Servidor que se utiliza como la comunicación entre VidStream y DFD, se compila con esta herramienta en una arquitectura x64 Intel, y se agrega en la raíz donde se encuentra el PJSIP en la tarjeta SD.

Por último se muestra un diagrama general de cómo funciona el esquema de cifrado. (ver Figura 4.23)

4.5. Resumen de contenido

Se dieron los pasos que se siguió para el desarrollo del diseño del esquema de cifrado para video, tomando en cuenta los requisitos que se establecieron para que el esquema de cifrado pudiera ser probado en plataformas móviles.

El problema que tenían algunos esquemas existentes era el no poder sincronizarse. De acuerdo con lo estudiado en la sección 3.2.5, se decidió tomar los datos de Temporal Reference y StartCode, ademas del TimeStamp para poder sincronizar el cifrado.

En el siguiente capítulo se realizaran las pruebas al esquema de cifrado realizado.

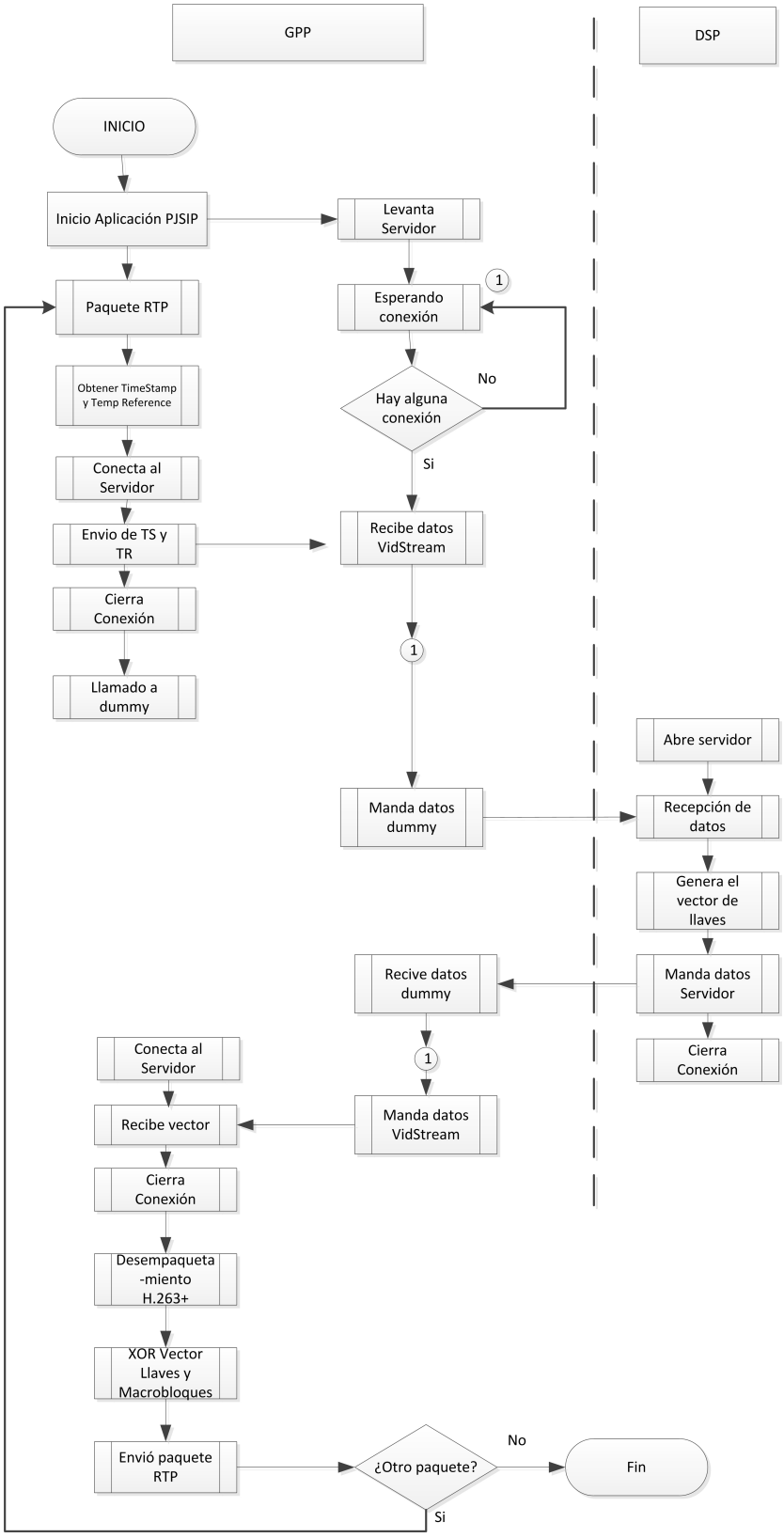


Figura 4.23: Diagrama general del cifrado

Capítulo 5

Pruebas y Resultados

En este capítulo se presentan las pruebas realizadas al esquema de cifrado. También se presentará un análisis de los resultados obtenidos y valoración de los mismos.

5.1. Diseño de pruebas

Se realizaron 20 transmisiones de video con duración aproximada de 5 minutos cada uno, sus características se muestran en la tabla 5.1

Tabla 5.1: Caracteristicas de los videos transmitidos

Nombre	Frames transmitdos	Resolución	Tiempo transmitido
Video1	1408	CIF (352x288)	00:05:25
Video2	1309	CIF (352x288)	00:05:30
Video3	1275	CIF (352x288)	00:05:17
Video4	1285	CIF (352x288)	00:05:18
Video5	1361	CIF (352x288)	00:05:59
Video6	1375	CIF (352x288)	00:05:38
Video7	1238	CIF (352x288)	00:05:15
Video8	1229	CIF (352x288)	00:05:04
Video9	1285	CIF (352x288)	00:05:12
Video10	1289	CIF (352x288)	00:05:12
Video11	1356	CIF (352x288)	00:05:55
Video12	1245	CIF (352x288)	00:04:45
Video13	1375	CIF (352x288)	00:05:35
Video14	1362	CIF (352x288)	00:05:27
Video15	1284	CIF (352x288)	00:05:34
Video16	1401	CIF (352x288)	00:05:52
Video17	1387	CIF (352x288)	00:05:24
Video18	1254	CIF (352x288)	00:04:59
Video19	1393	CIF (352x288)	00:05:27
Video20	1324	CIF (352x288)	00:05:03

A las 20 transmisiones de video se les realizaron dos pruebas:

- La primera fue para obtener el rendimiento y eficiencia del esquema de cifrado, comparando el uso del CPU cuando se implementa el esquema en el DSP y en el GPP. Debido a que este esquema se tiene pensado utilizarlo en dispositivos móviles, el rendimiento y la eficiencia debe ser mínima.
- La segunda prueba es para saber el nivel de seguridad que comprende nuestro esquema de cifrado. Para comprobar que nuestro esquema de cifrado realmente ofrezca el servicio de confidencialidad se realizaron pruebas donde se obtenía el MSE, PSNR y el Coeficiente de Correlación, las cuales nos ayudan a saber si el video cifrado obtenido difiere con respecto al video original y así no obtener datos del video.

5.2. Entorno de pruebas

Para realizar las pruebas fue necesario tener un esquema como el que se muestra a continuación (Fig 5.1):

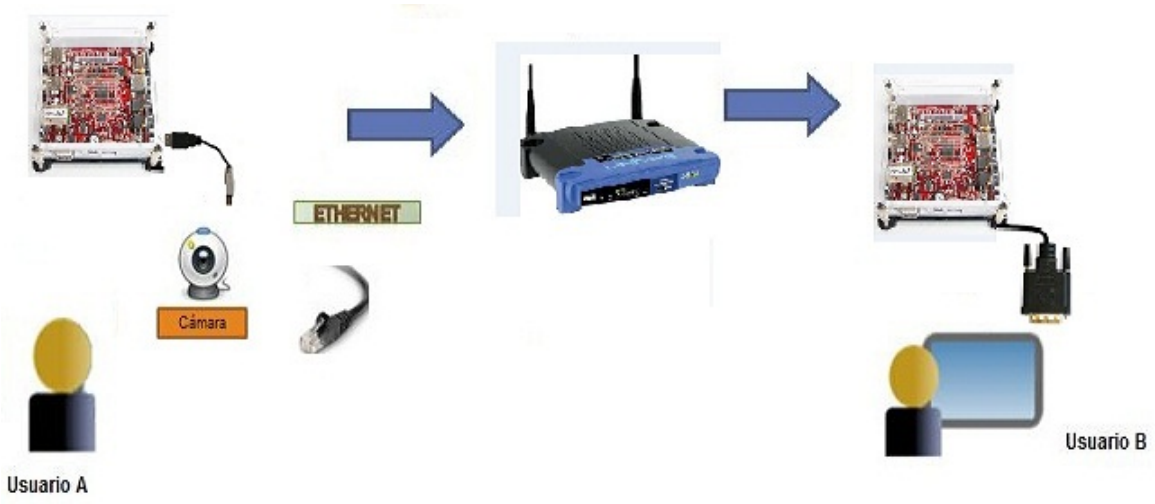


Figura 5.1: Arquitectura del Esquema de cifrado

Se tendrá una cámara conectada a una tarjeta Beagleboard-xM, por la cual se transmitirá el video, conectados a un router. En este caso el router ejerce el punto de acceso hacia el receptor.

Por otro lado para iniciar las pruebas se tuvo que acceder a la aplicación para transmitir video, llamada *vid_streamutil*. Esta viene incluida en la carpeta de *Samples* de PJSIP. De la parte del transmisor se configura con los siguientes parámetros (Figura 5.2):

- send-only: nuestra aplicación solamente será unidireccional, por lo que el transmisor solo tendrá permitido mandar la información (stream).
- remote: indica tanto la dirección IP como el puerto donde se recibirá el video

Figura 5.2: Ejecución del programa transmisión

Del lado del receptor sólo se indicará que tiene permitido únicamente recibir el video. (Figura 5.3)

Figura 5.3: Ejecución del programa Recepción

5.3. Ejecución de las Pruebas de Rendimiento y Eficiencia

El esquema de cifrado realiza el procesamiento de las llaves dentro del DSP con el fin de mejorar el rendimiento y eficiencia del sistema. El rendimiento en el núcleo ARM fue medido a través de la herramienta *top* de Linux (Figura 5.4), esto es, un comando ejecutado en un terminal de consola del sistema operativo embebido en el ARM que muestra en tiempo de ejecución, un listado de los procesos que se están ejecutando en el sistema, especificando además el porcentaje de CPU y memoria empleados, sus IDs, usuarios que lo están ejecutando, etc.

```
top - 14:58:46 up  3:29,  1 user,  load average: 0.43, 0.80, 1.00
Tasks: 134 total,   2 running, 130 sleeping,   0 stopped,   2 zombie
Cpu(s): 31.0%us, 24.8%sy,  0.0%ni, 31.9%id, 11.3%wa,  0.0%hi,  1.0%si,  0.0%st
Mem:    482244k total,   332964k used,   149280k free,    21004k buffers
Swap:  1048572k total,     4184k used,  1044388k free,   124512k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 1912 ubuntu    20   0 73764 7868 5084 S 17.2  1.6   0:00.57 vid_streamutil
 1305 ubuntu    20   0 54996  11m 8836 R 10.9  2.4   0:07.26 xfdesktop
  708 root      20   0  133m  62m 5524 S  6.0 13.3  58:26.72 Xorg
    3 root      20   0     0    0    0 S  3.6  0.0   0:01.12 ksoftirqd/0
 1911 ubuntu    20   0  2164 1040  756 R  3.0  0.2   0:00.52 top
   44 root      20   0     0    0    0 S  2.1  0.0   0:05.12 mmcqd/0
 1866 ubuntu    20   0 70280  11m 8508 S  1.5  2.4   0:02.21 xfce4-terminal
 1865 ubuntu    20   0 70536  11m 8496 S  1.2  2.4   0:02.74 xfce4-terminal
 1299 ubuntu    20   0 18064 8244 6204 S  0.6  1.7   0:02.54 xfwm4
```

Figura 5.4: Comando TOP.

La tabla 5.2 muestra el uso de CPU(%), tanto promedio, valor máximo y valor mínimo de las 20 transmisiones de video, usando el cifrado en el GPP.

Tabla 5.2: Uso del CPU con el esquema de cifrado en el GPP.

CPU	
Propiedad	**Valor (%)**
Promedio	62.06185567
Valor Mínimo	13.2
Valor Máximo	74.4

La tabla 5.3 muestra el uso de CPU, tanto promedio, valor máximo y valor mínimo de las 20 transmisiones de video, usando el cifrado en el DSP, donde ademas de mostrar el uso de la aplicación Vid_stream, se muestra también el análisis de la aplicación que nos ayuda a comunicarnos con la aplicación DFD, Server.

Tabla 5.3: Uso del CPU utilizando el DSP.

CPU		
Propiedad	**Valor Vid_Stream (%)**	**Valor Server**
Promedio	9.62	0.717475728
Valor Mínimo	0.00	0
Valor Máximo	17.20	1.3

Al obtener los datos anteriores se puede demostrar que el valor promedio de uso del CPU, cuando el esquema de cifrado es implementado en su totalidad en el GPP es mucho mayor que cuando se encuentra en el DSP.

Se realizó la gráfica de comparación (Fig. 5.5) entre los dos casos de cifrado, además que se agregó cuando la aplicación no contiene el componente de cifrado. Se puede apreciar que el uso del CPU cuando se aloja el esquema de cifrado en éste, es mucho menor que en los casos anteriores.

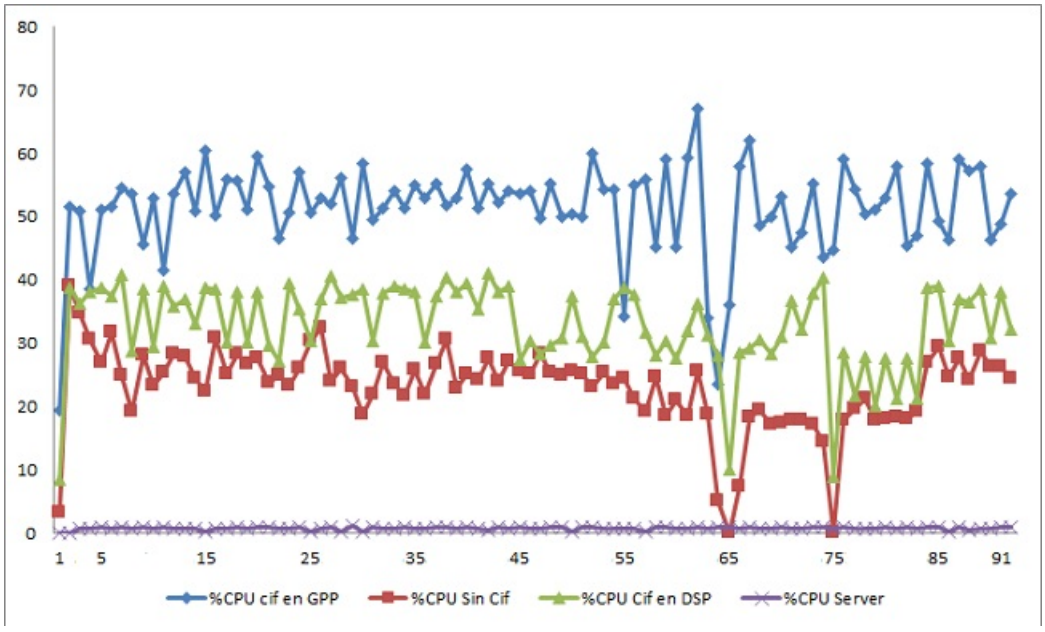


Figura 5.5: Comparacion del uso del CPU.

Por otro lado se midió el tiempo de cifrado y descifrado que se obtuvo al transmitir los videos. La tabla 5.4 muestra el tiempo que tarda en transmitir el video sin cifrar y la tabla 5.5 muestra el tiempo que tarda en transmitir el video ya cifrado. Se puede considerar que el esquema de cifrado no aumenta demasiado el tiempo para la transmisión de cada frame.

Tabla 5.4: Promedio del Tiempo de transmisión de un frame sin cifrar.

Parámetro	Tiempo sin Cifrado (ms)
Promedio	11.45
Valor Mínimo	20.32
Valor Máximo	10.23

Tabla 5.5: Promedio del Tiempo de transmisión de un frame cifrado y descifrado.

Parámetro	Tiempo de Cifrado (ms)	Tiempo de Descifrado (ms)
Promedio	16.46	13.89
Valor Mínimo	10.98	9.55
Valor Máximo	10.57	8.29

5.4. Pruebas de Seguridad

A diferencia del cifrado en texto, el cifrado multimedia requiere tanto de seguridad criptográfica y seguridad perceptual. El primero se refiere a la seguridad contra los ataques criptográficos, y el segundo significa que el contenido multimedia cifrado es ininteligible para la percepción humana. En el contexto de la seguridad perceptual, sabiendo sólo partes de los datos multimedia pueden ser de poca ayuda en la comprensión de los contenidos multimedia. Por lo tanto, el cifrado de partes de los datos multimedia puede ser razonable si se confirma la seguridad criptográfica.

Para realizar las siguientes pruebas se decidió utilizar el software de Wireshark, para obtener la capturas de los datos del video transmitido. Como se muestra en la figura 5.6, se eligió el paquete de RTP que contenía el codec H.263 y se guardó como datos crudos para así poder procesarlos.

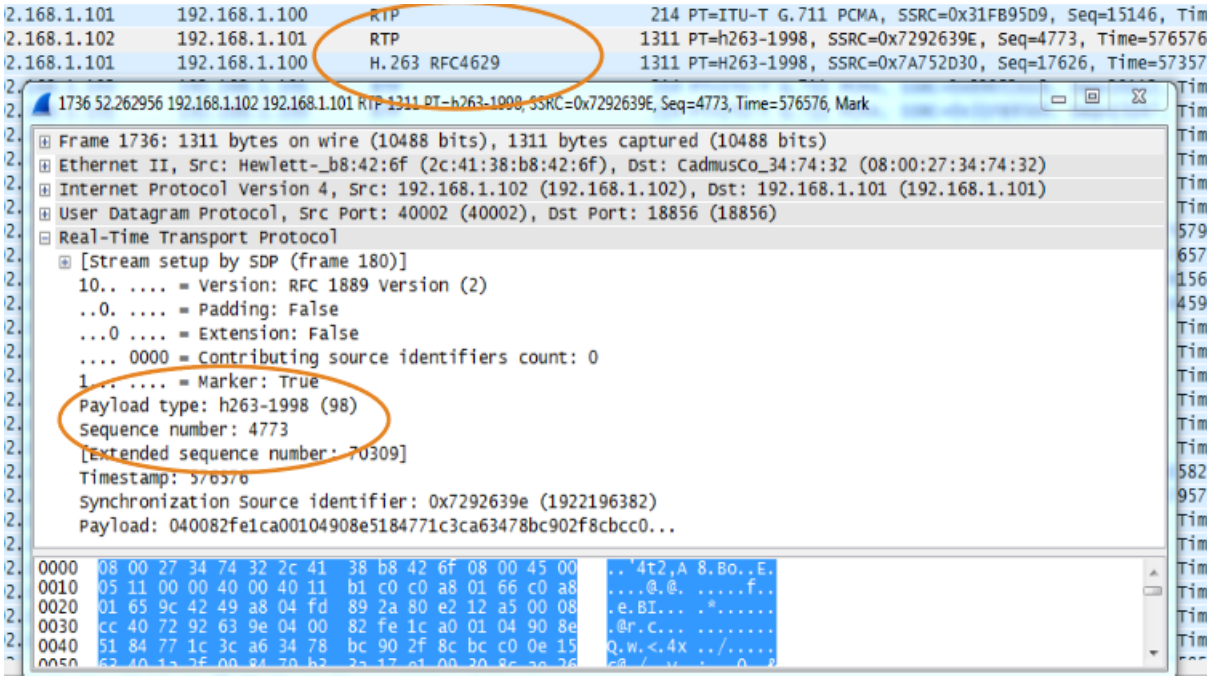


Figura 5.6: Datos obtenidos de captura de Wireshark

5.4.1. Error Cuadrático Medio (Mean Square Error MSE)

Es el error cuadrático acumulativo entre dos imágenes digitales y puede utilizarse para verificar el *efecto avalancha* (un pequeño cambio a la imagen original puede causar un cambio significativo en la imagen cifrada).

El MSE se refiere a la diferencia entre los puntos originales y los puntos calculados en el proceso de cifrado, siendo el error el valor que difiere. Un valor alto de MSE significa que hay un valor de error alto entre las imágenes, por lo que el video cifrado será diferente al video original.

Matemáticamente, el MSE se calcula con la ecuación 5.1.

$$MSE = \frac{1}{n}\sum_{i=0}^{n-1}(c_i - p_i)^2 \tag{5.1}$$

De acuerdo a estas pruebas se obtuvieron los siguientes resultados: (Tabla 5.6)

Video	MSE
Video1	10829
Video2	10765
Video3	83429
Video4	13892
Video5	11561
Video6	94521
Video7	14120
Video8	10942
Video9	10129
Video10	73910
Video11	10745
Video12	10112
Video13	10572
Video14	92782
Video15	79321
Video16	68249
Video17	10027
Video18	93866
Video19	63029
Video20	10532

Tabla 5.6: Resultados de MSE.

El promedio de los resultados se muestra en la tabla 5.7

MSE	
Promedio	10909.25
Valor Mínimo	10636
Valor Máximo	11643

Tabla 5.7: Promedio de MSE de las 20 transmisiones.

En términos de MSE, al obtener valores mucho mayores, nos indica que la diferencia entre los frames de cifrado y frames originales del video es mayor por lo cual la calidad es diferente.

5.4.2. Señal de Ruido Pico (Peak Sign-to-Noise PSNR)

Se utiliza inicialmente para medir las pérdidas de calidad en imágenes generadas por operaciones tales como la compresión, errores de transmisión, etc. Es medida en una escala logarítmica y se basa en el error cuadrático medio (MSE) entre la comparación de la imagen original y la imagen transformada. Matemáticamente se calcula como:

$$PSNR = 10 \lg_{10} \frac{L^2}{MSE} \tag{5.2}$$

Donde L son los 255 valores de colores correspondientes. Los valores típicos que adopta este parámetro están entre 30 y 50 dB, este nos indica que entre mayor sea el valor, cuanto mejor es la codificación.

De acuerdo a estas pruebas se obtuvieron los siguientes resultados: (Tabla 5.8 y Tabla 5.9)

Donde se puede demostrar que el valor de PSNR obtenido en las 20 pruebas que se realizaron es menor a 13 dB, tomando en cuenta que entre mayor sea el valor de PSNR la calidad de la imagen es mejor, esto nos indica que para una imagen cifrada el valor de PSNR debería ser un valor pequeño para la protección del contexto, por lo que nuestro esquema protege los datos de la imagen original.

Tabla 5.8: Resultados de PSNR.

Video	PSNR (dB)
Video1	7.8191
Video2	7.6322
Video3	7.7052
Video4	7.7634
Video5	7.8022
Video6	7.8781
Video7	7.8942
Video8	7.8591
Video9	7.8521
Video10	7.6166
Video11	7.6863
Video12	7.8613
Video13	7.7819
Video14	7.7614
Video15	7.847
Video16	7.5043
Video17	7.8579
Video18	7.831
Video19	7.8229
Video20	7.7801

Tabla 5.9: Promedio de los resultados de PSNR.

PSNR (dB)	
Promedio	7.777815
Valor Mínimo	7.5043
Valor Máximo	7.8942

5.4.3. Amabilidad de Compresión

Su compresión es *amigable* del algoritmo de cifrado, sí tiene un pequeño impacto en la eficiencia de los datos comprimidos cuando éste se cifre y se descifre, es decir no aumente el tamaño de los datos debido a que es suficiente con la redundancia de datos que éste tiene. Respecto a este concepto nuestro esquema de cifrado cumple con la característica de no aumentar más datos al video, es decir el tamaño del video original es el mismo tamaño del video cifrado.

5.4.4. Coeficiente de Correlación

Determina la relación que existe entre dos variables, es decir calcula el grado de similitud de dos variables. En el caso de las imágenes cifradas, el valor del coeficiente de correlación debe de ser muy pequeño o muy cerca a cero, en caso de que resulte igual a uno quiere decir que la imagen cifrada es idéntica a la original, por otro lado si es igual a -1 entonces la imagen cifrada es el negativo de la imagen original.

La tabla 5.10 muestra todos los resultados de los 20 videos que se obtuvieron. De ahí se analizó dos casos: el primer caso resultó ser negativo el valor del coeficiente de correlación pero como tiende al valor cero no puede ser el negativo del frame original, por lo que es un valor aceptable para que se pueda concluir que el video cifrado es diferente al video original. En el segundo caso el valor es positivo y muy cercano a cero, en este caso los videos son también diferentes. La tabla 5.11 muestra el promedio de los datos obtenidos.

Tabla 5.10: Resultados de los Coeficientes de Correlación.

Nombre	Coeficiente de Correlación
Video 1	-0.000369
Video2	0.000425
Video3	-0.000662
Video4	0.001300
Video5	0.000444
Video6	0.000036
Video7	0.000959
Video8	-0.000285
Video9	0.002
Video10	0.000281
Video11	0.000248
Video12	0.000489
Video13	-0.000762
Video14	0.000425
Video15	-0.00062
Video16	-0.000946
Video17	-0.000296
Video18	-0.000142
Video19	-0.000215
Video20	0.0017

Tabla 5.11: Resultados promedio de los Coeficientes de Correlación.

Coeficiente de Correlación	
Promedio	0.00020048
Valor Minimo	-0.000946
Valor Maximo	0.002

Por otro lado, se calculó el coeficiente de correlación comparando las imágenes originales con las imágenes descifradas. Los resultados se muestran en la tabla 5.12 y el promedio de éstos se muestran en la tabla 5.13. Concluimos que las imágenes son muy similares, debido a que se obtiene como resultado de coeficiente de correlación que tiende a 1. Por lo que se demuestra que el método de descifrado también funciona.

Tabla 5.12: Resultados de los Coeficientes de Correlación (Descifrado).

Nombre	Coeficiente de Correlación
Video 1	0.999340
Video2	0.99982
Video3	0.999310
Video4	0.999340
Video5	0.9997
Video6	0.99953
Video7	0.99984
Video8	0.99996
Video9	0.99973
Video10	0.99982
Video11	0.99958
Video12	0.99347
Video13	0.99945
Video14	0.99932
Video15	0.99945
Video16	0.99939
Video17	0.99809
Video18	0.99979
Video19	0.99999
Video20	0.99963

Tabla 5.13: Resultados promedio de los Coeficientes de Correlación (Descifrado).

Coeficiente de Correlación	
Promedio	0.999228
Valor Minimo	0.993470
Valor Máximo	0.999990

5.4.5. Histograma

Muestra la frecuencia de los datos, en la que el eje horizontal representa unidades discretas, ciertos rangos, o intervalos, en tanto que el eje vertical representa la frecuencia. En nuestro caso se obtuvo la imagen 5.7 donde muestra que el histograma del video cifrado de nuestros datos tiene las mismas probabilidades de aparición, es decir, una aparición uniforme a comparación con el video original, en la que se puede notar que varía.

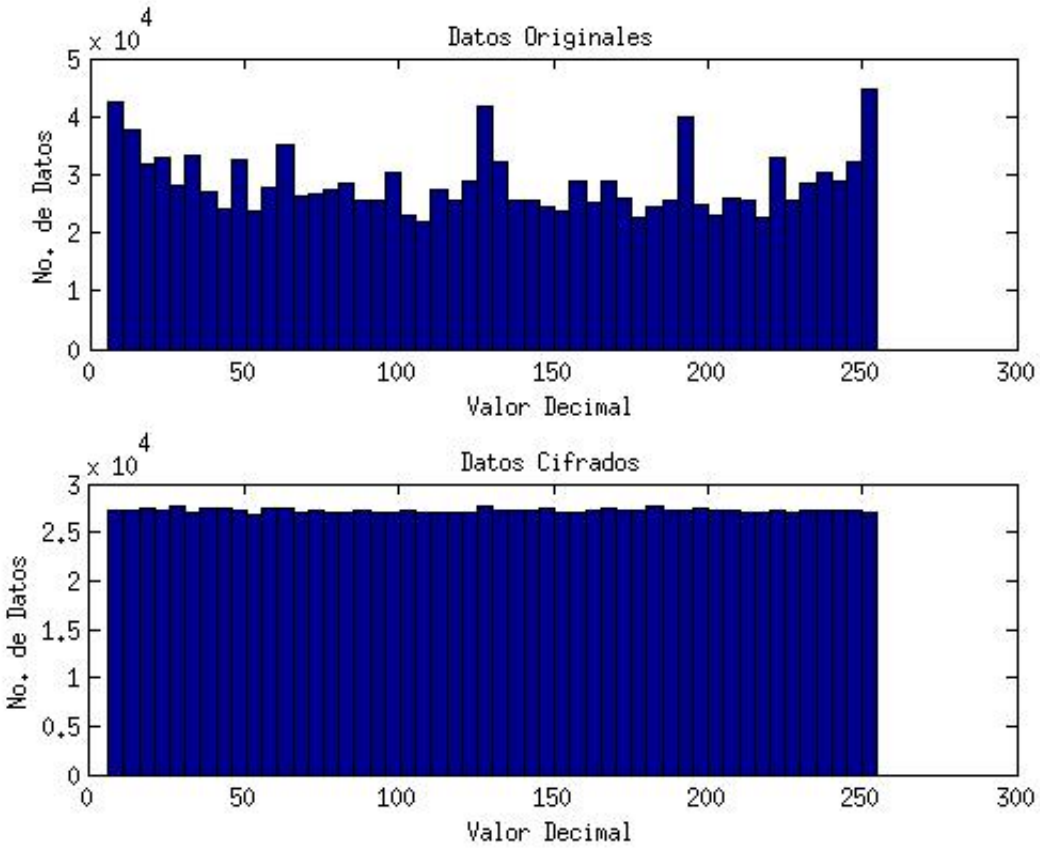


Figura 5.7: Histograma video1 sin cifrar y cifrado

5.4.6. Desviación Estándar

Es una medida de dispersión que indica qué tan distribuidos están los valores a lo largo de los rangos, en este caso, 0 a 255 valores.

Se calcula de acuerdo a la formula:

$$\sigma = \sqrt{VAR(x)} \tag{5.3}$$

$$VAR(x) = \frac{1}{N}\sum_{i=1}^{N}(x_i - E(x))^2 \tag{5.4}$$

Donde $VAR(X)$ es la varianza del valor de un pixel x en la imagen cifrada y σ es la desviación estándar.

Los datos que nos dio como resultados en la prueba se muestra en la tabla 5.14 y el promedio de esos datos se encuentran en la tabla 5.15.

Tabla 5.14: Resultados de la desviación estándar.

Nombre	Desviación Estandar Original	Desviación Estandar Cifrado
Video 1	17150	830.4056
Video2	18342	800.1046
Video3	17375	820.5932
Video4	17045	848.3912
Video5	17092	850.2347
Video6	17934	813.9722
Video7	17641	818.9351
Video8	17108	831.8932
Video9	17159	831.9899
Video10	18102	799.0012
Video11	18243	800.9263
Video12	17198	831.0302
Video13	17376	822.8933
Video14	17267	823.8931
Video15	17489	825.391
Video16	17297	832.8402
Video17	17523	824.8421
Video18	17634	817.9096
Video19	17623	818.0563
Video20	17789	817.9561

Tabla 5.15: Resultados promedio de la desviación estándar.

	Desviación Estandar Original	Desviación Estandar Cifrado
Promedio	17519.35	823.062955
Valor Minimo	17045	799.0012
Valor Maximo	18342	850.2347

La desviación estándar del video original sale muy alto por que los valores de todas las muestras están más dispersas debido a que sus valores son más cercanos a la media aritmética de todas las muestras. En el cifrado es más pequeño debido a que el valor de todas las muestras no se diferencia tanto del valor aritmético de todas las muestras, por lo que en el histograma se muestra más homogéneo.

5.4.7. Entropía

El concepto de la entropía es muy importante para el análisis de un esquema de cifrado. La entropía, es una medida de la incertidumbre de una variable aleatoria.

Sea X una variable aleatoria discreta y la función de masa de probabilidad $p(x) = \Pr X = x$ donde $x \epsilon \psi$. La entropía $H(X)$ se define de la siguiente manera:

$$H(X) = \sum_{i=0}^{2^N-1} p(m_i) \times \log_2 \frac{1}{p(m_i)} \tag{5.5}$$

Donde $p(m_i)$ representa la probabilidad de ocurrencia del símbolo m_i. Por ejemplo si se considera una fuente aleatoria que genera 28 símbolos con una probabilidad igual, es decir, m= M1...M28, y cada símbolo es representado por 8 bits, de acuerdo con la ecuación 5.5, la entropía obtenida es $H(m) = 8bits$, lo cual corresponde a una fuente aleatoria uniforme. Es decir cuando los mensajes están cifrados para una fuente que genera 28 símbolos con la misma probabilidad, su entropía debe ser de 8 bits idealmente. En caso de que si la entropía es mucho menor que 8 bits, entonces existe un cierto grado de previsibilidad.

De acuerdo con nuestras pruebas se obtuvieron los resultados de la tabla 5.16, mientras que el promedio de esos datos, lo muestra la tabla 5.17

Tabla 5.16: Resultados de Entropia

Nombre	Entropia ciphertext
Video 1	7.9656
Video2	7.9867
Video3	7.9725
Video4	7.9589
Video5	7.9346
Video6	7.9642
Video7	7.9986
Video8	7.9631
Video9	7.9135
Video10	7.9478
Video11	7.9357
Video12	7.9853
Video13	7.9256
Video14	7.9356
Video15	7.9814
Video16	7.9895
Video17	7.9912
Video18	7.9736
Video19	7.9834
Video20	7.9952

Tabla 5.17: Resultados promedio de Entropia

Entropía	
Promedio	7.997500
Valor Mínimo	7.963146
Valor Máximo	7.995200

Se muestra que durante las 20 pruebas se obtuvo un rango de 7.9256 a 7.9986 los cuales son muy cercanos al valor teórico de 8 bits (debido a que cada carácter es representado por 8 bits como en el ejemplo anterior), demuestra que es imposible predecir la información original.

5.4.8. Percepción del video cifrado

A continuación se muestra algunos frames del video original y cómo se ve cuando está cifrado. En la figura 5.8 se muestra un frame de cuando el video que se transmitió no había tanta luz y como se ve el cifrado de éste, ademas que se muestra el descifrado del video.

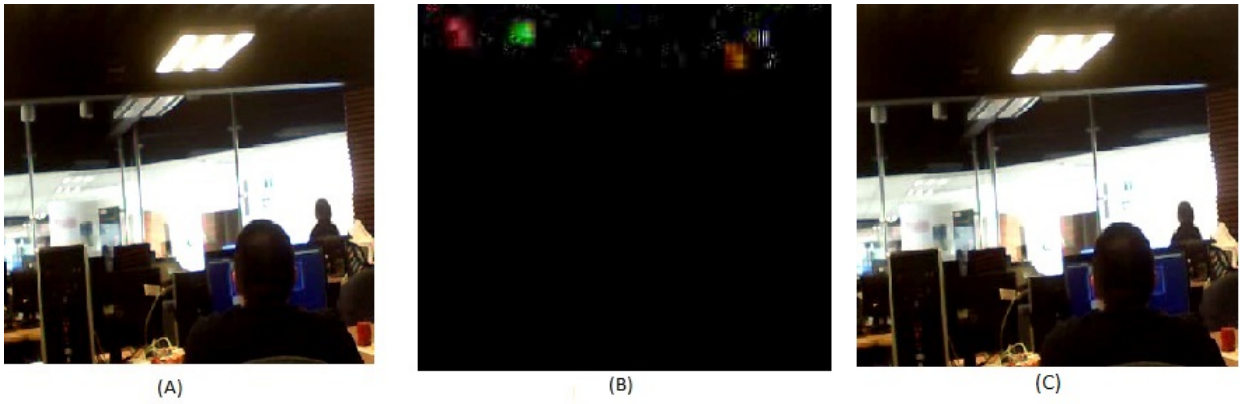


Figura 5.8: Captura de un frame de video estático (A)Original, (B)Cifrado, (C) Descifrado

En la figura 5.9 se muestra frames con movimiento, es decir, el video no solo enfocó un escenario, si no que se grabó movimientos de las personas u objetos. Se muestra la imagen original (A), como se ve el cifrado (B) y el descifrado del video (C).

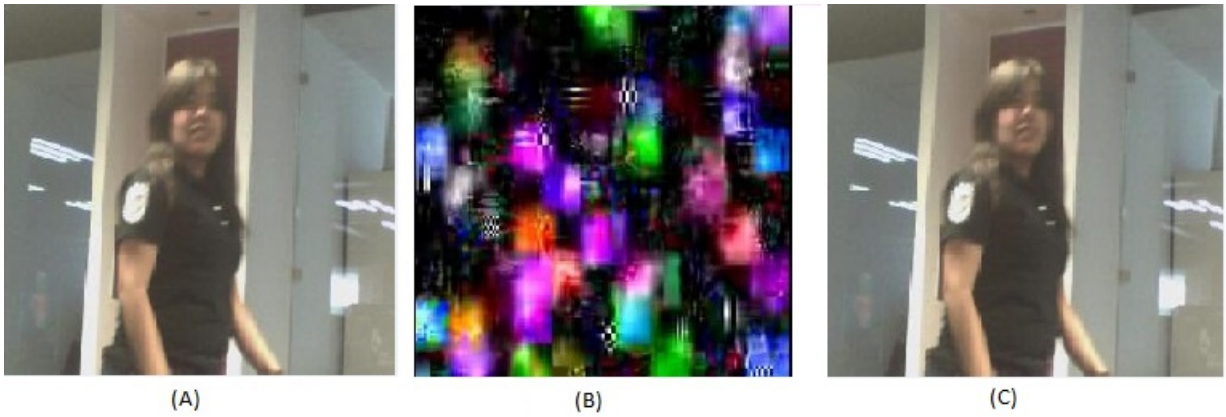


Figura 5.9: Captura de un frame de video con movimiento (A)Original, (B)Cifrado, (C)Descifrado

5.4.9. Resumen de contenido

De acuerdo a lo experimentado, el esquema de cifrado demuestra que en términos de desempeño, al utilizar el DSP de la tarjeta Beagleboard-xM como generador de llaves, mejora la eficiencia en el uso de CPU según lo mostrado en la sección de Pruebas y Resultados. En términos de seguridad, los resultados de percepción del video cifrado fueron similares a los resultados de trabajos relacionados, cumpliendo con los requisitos en conceptos como lo son PSNR y MSE. En el aspecto de confidencialidad de los datos se obtuvo una homogeneidad en el histograma (perceptualmente), demostrándolo con la desviación estándar de los datos, además de la entropía. El esquema de cifrado cumple también con la característica de no aumentar datos para el cifrado y así la transmisión de video no tenga retrasos. También se demuestra que el video original se recupera al realizar el descifrado de este.

Conclusiones, Trabajo Futuro y Productos de la Investigación

Teniendo en cuenta el objetivo general y los objetivos particulares que se plantearon con anterioridad se concluye que:

- Se presentó el proceso de diseño de un esquema de cifrado, para la transmisión de video utilizando sistemas embebidos BeagleBoard-xM como nuestro dispositivo móvil. Para ello se hizo uso de los diferentes periféricos y procesadores que esta plataforma de desarrollo que posee y la interfaz de programación de Ubuntu.
- Se analizó la estructura del video H.263+, para no cifrar todos los frames completos del video. Se decidió cifrar solamente los macrobloques del video, esto permitió que el procesamiento de cifrado fuera más rápido y se aseguro la confidencialidad de la información del video, debido a que los macrobloques contienen la información más importante del video.
- Se demostró que el algoritmo AES en modo contador funciona en aplicaciones para transmisión de video en vivo. Permitiendo que la transmisión fuera segura y sin tanto tiempo en retardo.
- Se comprobó que al hacer uso del DSP de la tarjeta Beagleboard-xM disminuiría el consumo del uso de su CPU, por lo que demostramos que es eficiente el esquema de cifrado propuesto.
- Se demostró que el esquema de cifrado para video propuesto es competitivo con esquemas presentados en el capítulo 2 y al funcionar en el sistema Beagleaboard-xm el cual presenta características similares a un dispositivo móvil, se comprobó que puede funcionar para dispositivos móviles que se encuentran en el mercado.

A continuación se presenta como trabajo futuro los siguientes puntos que tengan como parte de partida el esquema de cifrado en la transmisión de video:

- Poder implementar el esquema de cifrado en un dispositivo móvil, como son los teléfonos celulares, y así utilizar el DSP de estos.

- Probar el esquema en condiciones diferentes, es decir, otra resolución, utilizando otros códec's y disminuir el tamaño de datos cifrados.

Como producto final de la investigación se entrega un esquema de cifrado que se podrá utilizar en transmisiones de video, siendo que este funcionara en un sistema con las mismas características que se presentaron en el desarrollo de la investigación.

Anexos

Anexo A

Instalacion del sistema operativo LINUX e instalación del DFD

- Requisitos
  - Tarjeta Beagleboard-xM
  - microSD de minimo 8 Gbytes
  - Computadora Personal de 64 bits con sistema operativo 12.04 LTS
  - Teclado y mouse para la tarjeta Beagleaboard-xM
  - Eliminador de 5 volts a 2.5A
  - 2 conectores de ethernet
  - Monitor para la tarjeta Beagleaboard-xM
- Procedimiento

En la PC, se abre la aplicación terminal y se introduce lo siguiente:

```
monik@monik-Satellite-M645:~$ sudo apt-get install build-essential gcc-arm-linux
-gnueabihf uboot-mkimage ia32-libs
```

Figura A.1: Instalación de programas necesarios

Estos comandos ayudan a instalar los paquetes necesarios para poder compilar el S.O. Ubuntu para la tarjeta Beagleaboard-xm.

Del sitio web: *http://rcn-ee.net/deb/rootfs/*, se descarga el paquete de S.O Ubuntu. Una vez descargado se extraen los archivos. Se inserta en la PC la microSD y en la terminal se escriben los siguientes comandos: (Figura A.2), donde FILE es la ruta donde se extrajo los archivos de Ubuntu anteriormente descargados.

```
cd ~/Desktop/<FILE>
sudo ./setup_sdcard.sh --mmc /dev/<DRIVE> --uboot beagle_xm
```

Figura A.2: Comandos para montar el SO en la SDcard

Una vez que se acabe de ejecutar el script, si el sistema no detecta la tarjeta, la extraemos y la volvemos a insertar y se ejecutan los siguientes comandos: (Ver Figura A.3)

```
cd /media/root/home/ubuntu
wget https://raw.github.com/RobertCNelson/tools/master/x86/ti_omap/
create_dsp_package.sh
```

Figura A.3: Comando para descargar el programa DFD

Se extrae la tarjeta microSD y se inserta en la tarjeta Beagleaboard-xM, se conecta el cable USB-Serial y todos los componentes que se requieran para la tarjeta Beagleaboard-xM, (es decir, mouse, cable ethernet, teclado), mientras en la PC se introduce el siguiente comando en la terminal: $sudominicom - s$ y se configura con los parametros como se muestran en la figura A.4 para poder comunicarnos con la tarjeta por medio de la PC.

```
+-----------------------------------------------------------------------+
| A - Dispositivo Serial            : /dev/ttyUSB0                      |
| B - Localización del Archivo de Bloqueo : /var/lock                   |
| C - Programa de Acceso           :                                    |
| D - Programa de Salida             :                                  |
| E - Bps/Paridad/Bits             : 115200 8N1                         |
| F - Control de Flujo por Hardware: Sí                                 |
| G - Control de Flujo por Software: No                                 |
|                                                                       |
|    ¿Qué configuración alterar?                                        |
+-----------------------------------------------------------------------+
```

Figura A.4: Configuración de Minicom

Iniciamos la tarjeta con user como ubuntu y password como temppwd y en la terminal de la tarjeta introducimos los siguientes comandos: A.5

```
sudo apt-get update sudo apt-get upgrade
sudo apt-get install build-essential
sudo chmod a+x DSP_Install_libs.tar.gz
tar xf DSP_Install_libs.tar.gz
git clone git://github.com/RobertCNelson/tools.git
cd ~/tools/pkgs
./ti-tidspbridge.sh #Esperar hasta que termine el proceso
sudo /etc/init.d/dsp_init start
sudo dsp-test
```

Figura A.5: Setup DSP

Si es exitoso mostrará: *Copied 1000 times*. Por otro lado se necesitará descargar la aplicación DFD y la herramienta C600 Code Generation de los siguientes links:

$http://processors.wiki.ti.com/images/c/c3/Dspfordummies.tar.gz$
$http://softwaredl1.ti.com/dsps/forms/self\_cert\_export.html?prod\_no=ti\_cgt\_c6000\_7.4.2\_setup\_linux\_x86.bin\&ref\_url=$

$http://softwaredl.ti.com/dsps/dsps\_registered\_sw/sdo\_ccstudio/codegen/C6000/7.4.2$

Al terminar de descargar los paquetes se abrirá una terminal en la PC y se escribirá lo siguiente: (Ver Figura A.6)

```
cd ~/Desktop
sudo ./ti_cgt_c6000_7.4.2_setup_linux_x86.bin
```

Figura A.6: Ejecución del Instalador C6000

Al ejecutar los comandos el instalador de la herramienta se abrirá (Figura A.7) y seguimos los pasos que muestra.

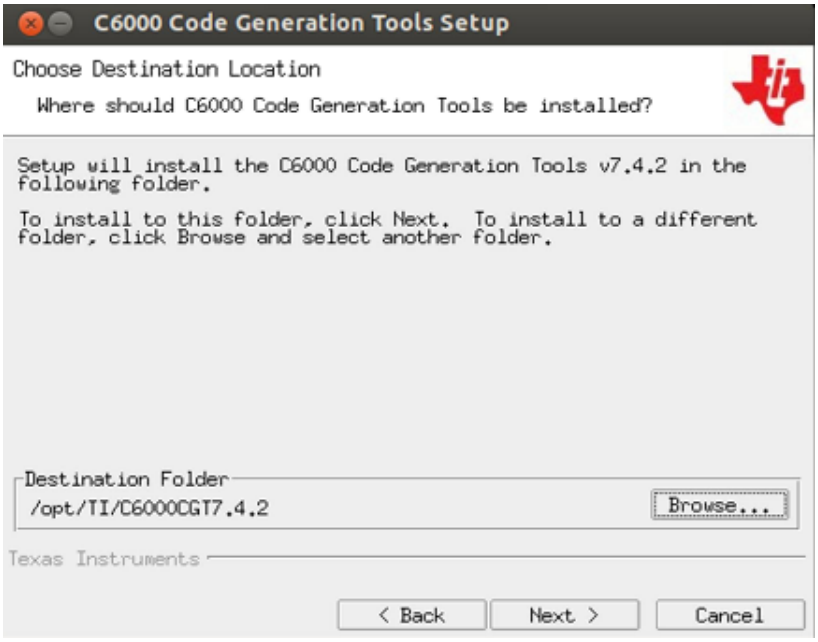


Figura A.7: Instalador de C6000

Una vez que termine el instalador, se ejecutan los siguientes comandos: A.8

```
sudo cp -avr ~/Desktop/ti/dspbridge/dsp/doffbuild/ /opt/TI/
cd dsp-for-dummies/
gedit Makefile
```

Figura A.8: Copiando folder doffbuild

Ésto abrirá el Makefile de la aplicación DFD y se tendrá que modificar las siguientes líneas:

1. Linea #2 CROSS_COMPILE ? = arm−linux−gnueabihf−
2. Linea #21 DSP_TOOLS := /opt/TI/C6000CGT7.4.2
3. Linea #22 DSP_DOFFBUILD := /opt/TI/doffbuild

Guardamos el archivo y ejecutamos los comandos de la figura A.9

```
make clean
make
sudo cp dummy.dll64P /media/rootfs/lib/dsp/
sudo cp dummy /media/rootfs/home/<username>/
```

Figura A.9: Copiando Libreria y Ejecutable DFD

Por último en la terminal de la tarjeta Beagleboard-xM se darán permisos para poder utilizar el DSP, por lo que ejecutamos la instrucción *sudo chmod 777 /dev/DSPBrigde* y luego escribimos en la terminal lo de la figura A.10.

```
sudo ./dummy 32 0 3
```

Figura A.10: Ejecución de DFD

Si todo concluyó con éxito, en la terminal se mostrar una serie de números que representa la matriz.

Anexo B

Resultados

Este anexo se dividirá en dos secciones: la primera mostrará todas las gráficas de los histogramas obtenidos de los 20 videos, en la segunda sección se encontrarán las gráficas del uso del CPU cuando el esquema de cifrado está en el GPP y cuando el esquema de cifrado esta en el DSP, así como el uso del CPU de la aplicación Vid_Stream sin Cifrado.

Histogramas

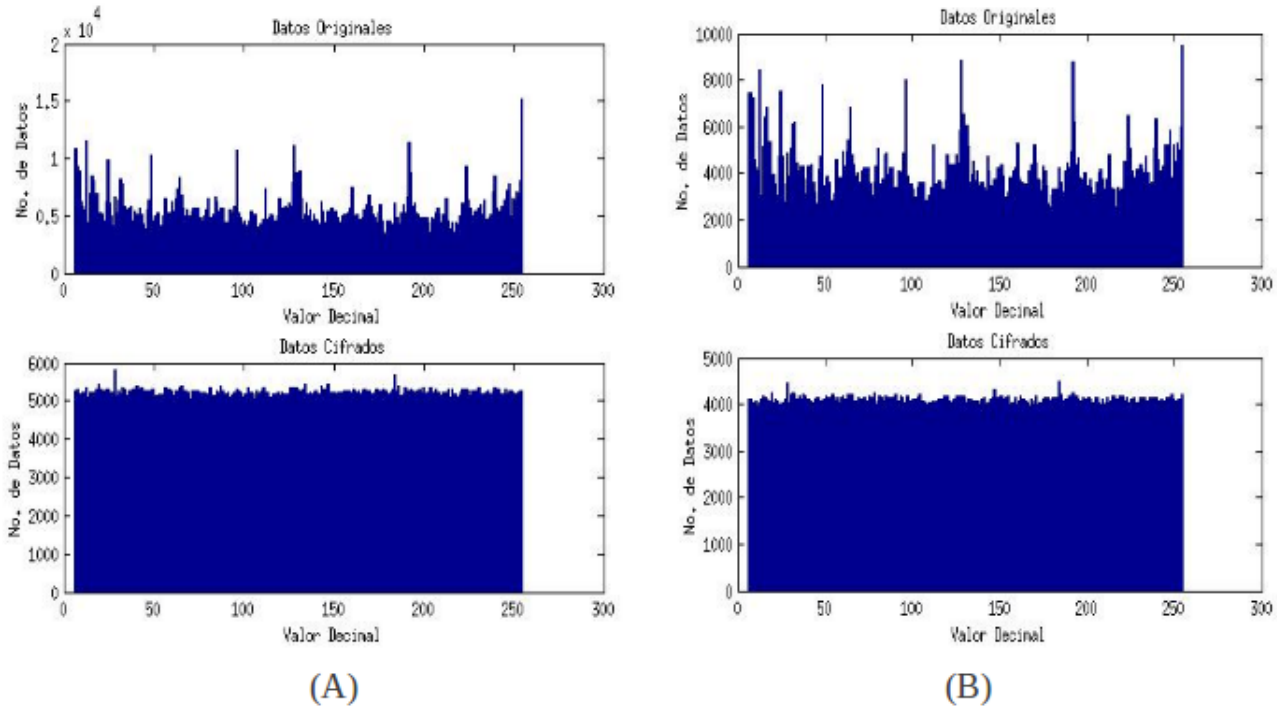


Figura B.1: Histograma cifrado-descifrado de video1 (A) y video2 (B)

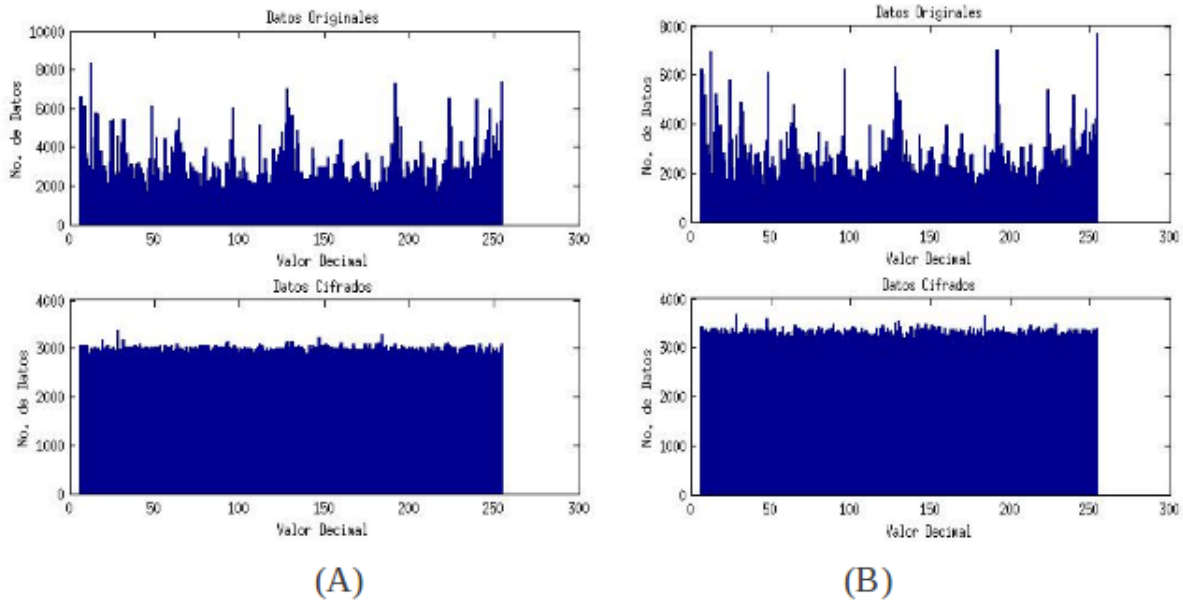


Figura B.2: Histograma cifrado-descifrado de video3 (A) y video4 (B)

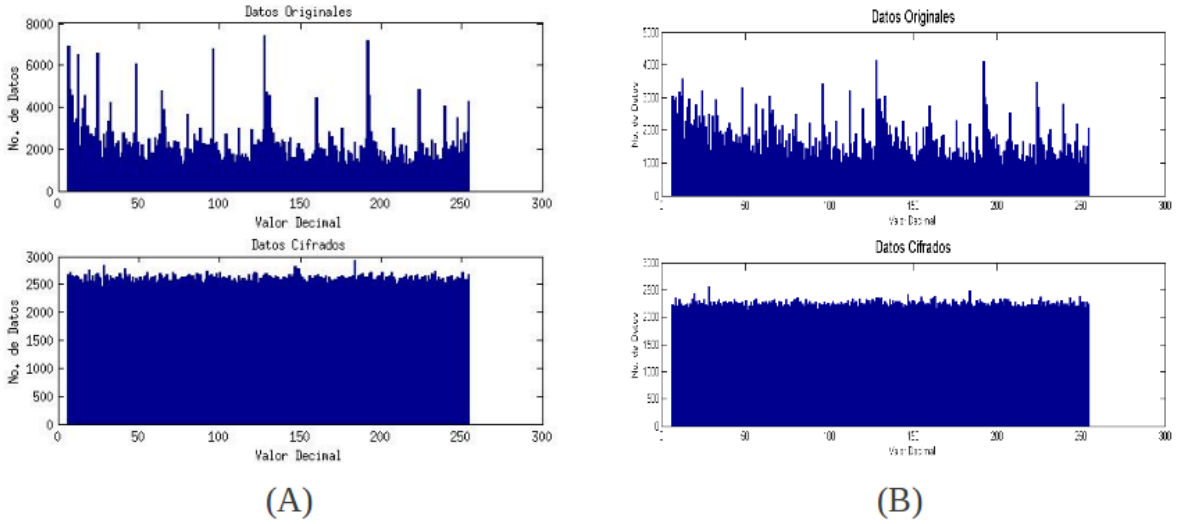


Figura B.3: Histograma cifrado-descifrado de video5 (A) y video6 (B)

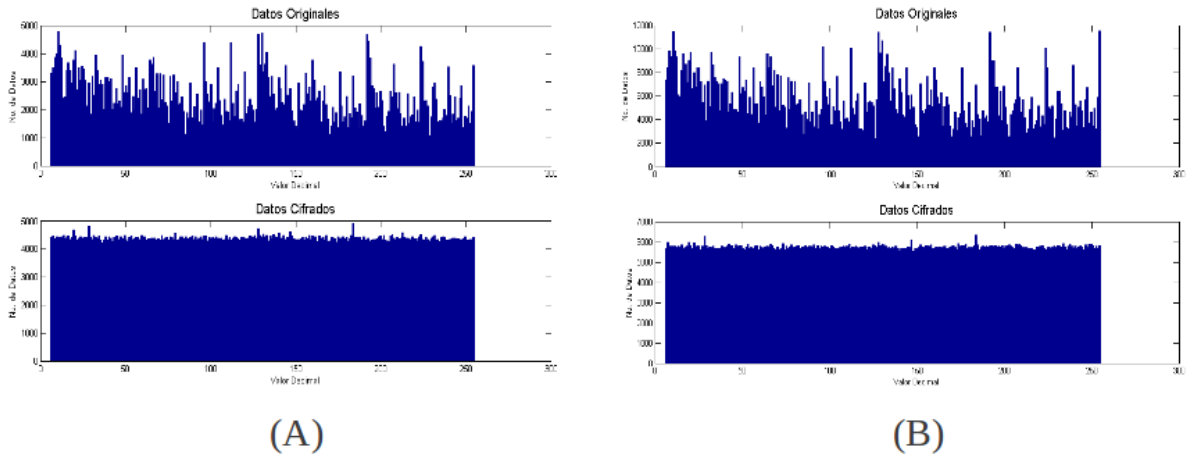


Figura B.4: Histograma cifrado-descifrado de video7 (A) y video8 (B)

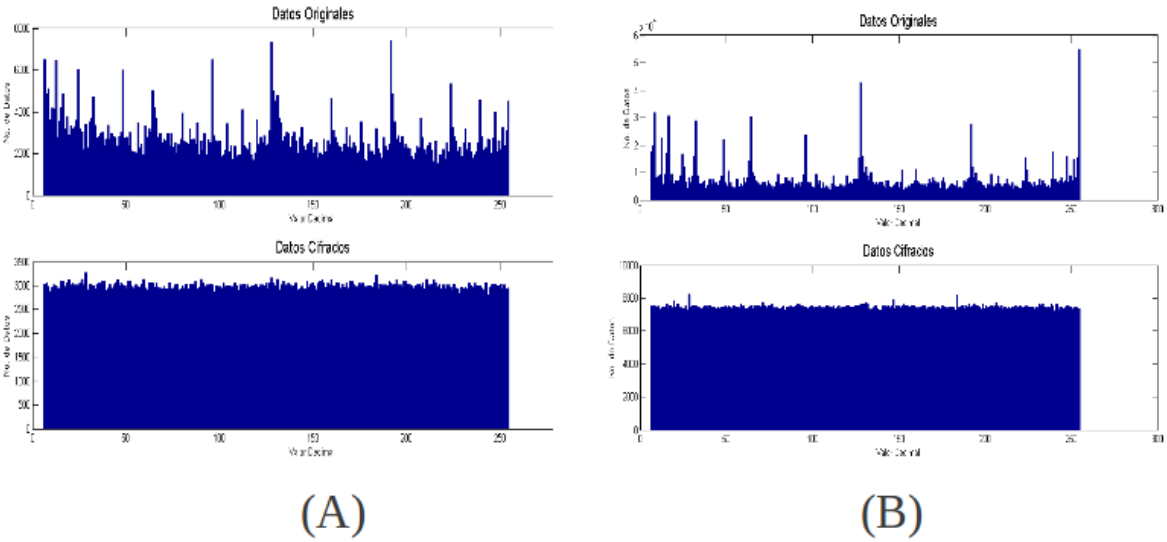


Figura B.5: Histograma cifrado-descifrado de video9 (A) y video10 (B)

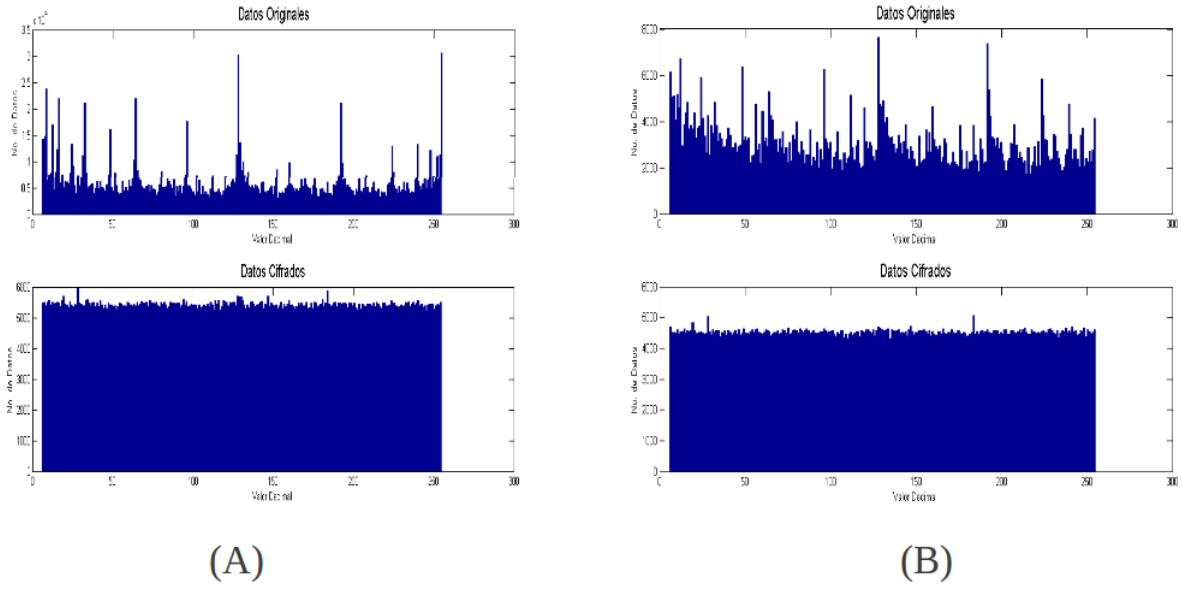


Figura B.6: Histograma cifrado-descifrado de video11 (A) y video12 (B)

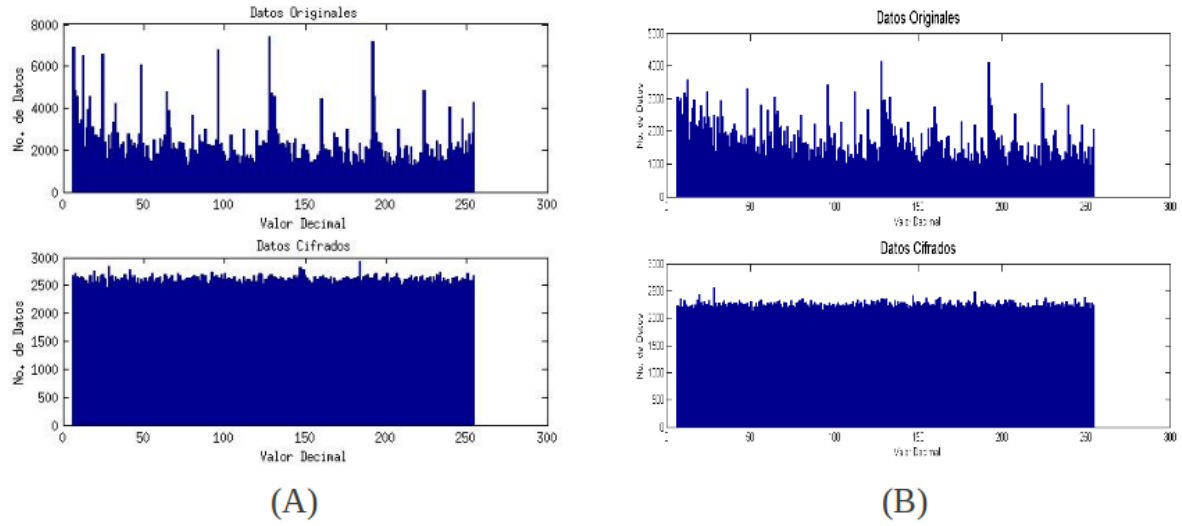


Figura B.7: Histograma cifrado-descifrado de video13 (A) y video14 (B)

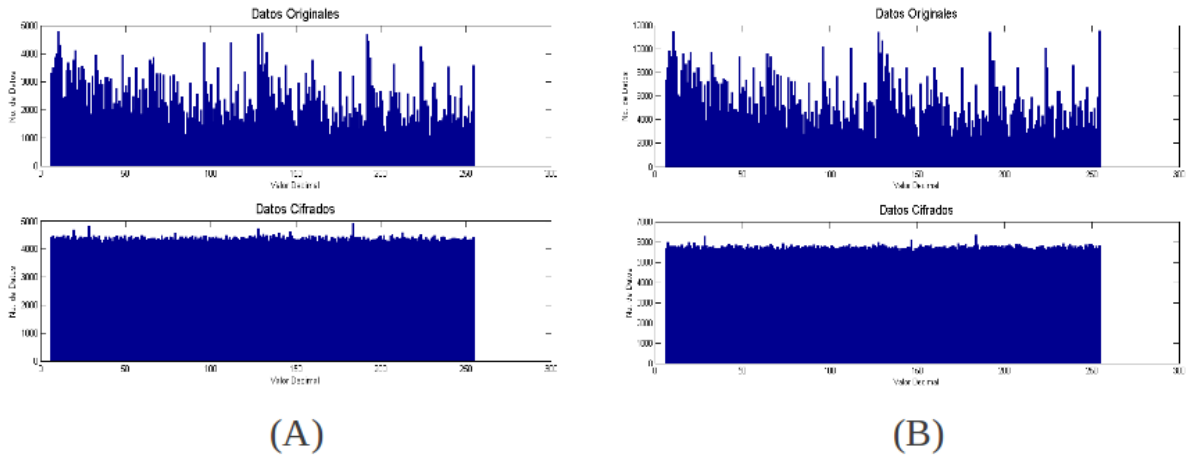


Figura B.8: Histograma cifrado-descifrado de video15 (A) y video16 (B)

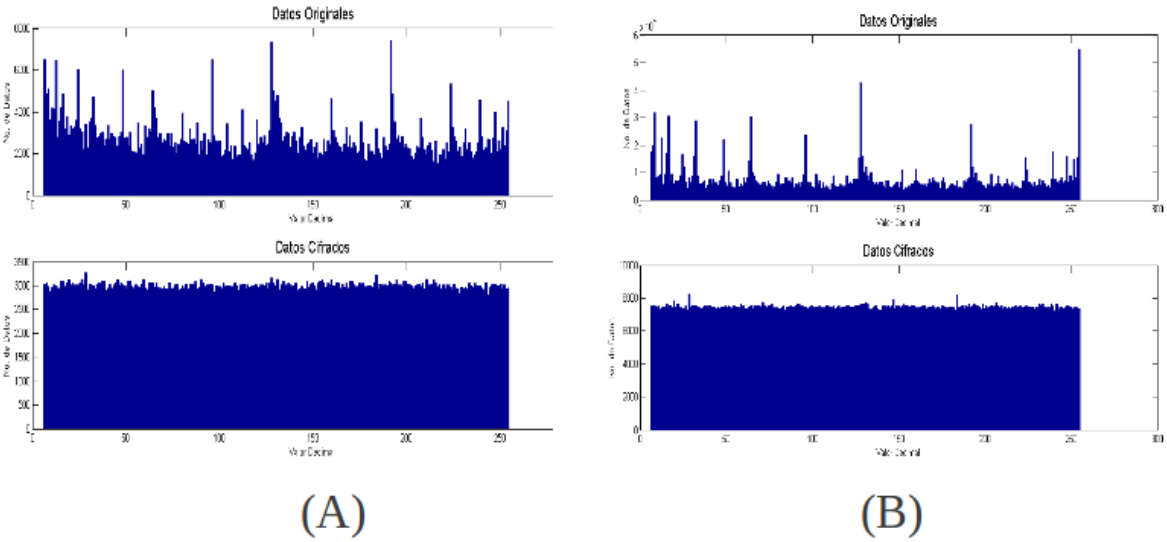
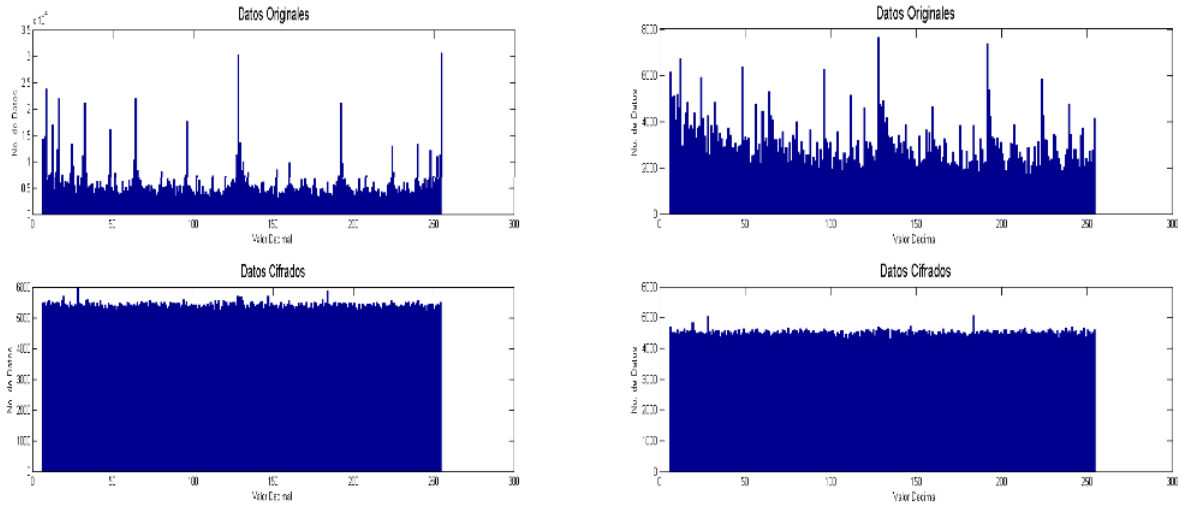


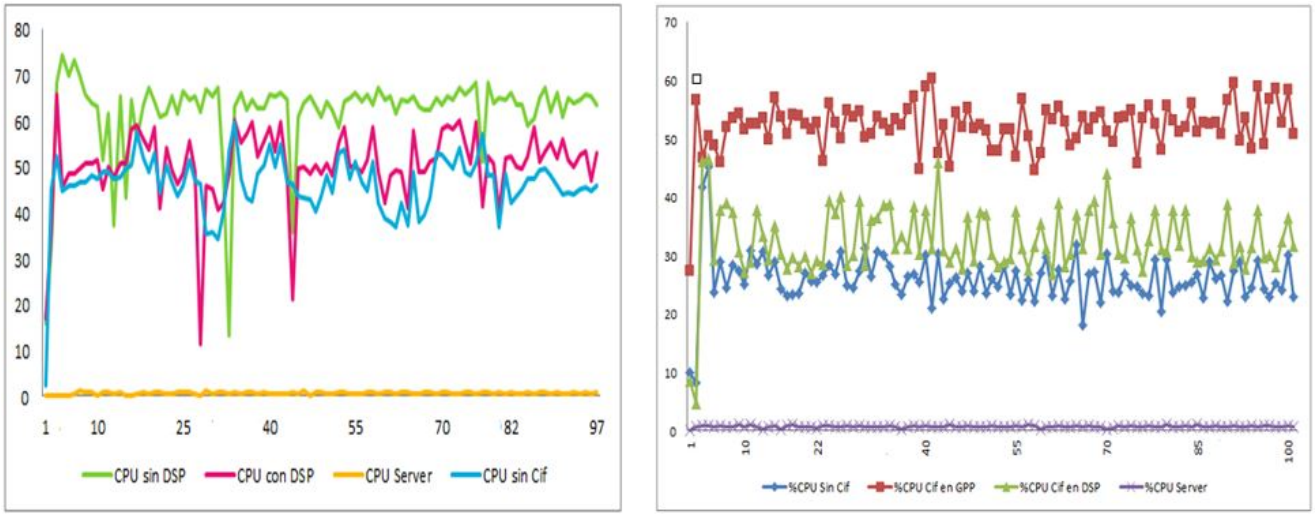
Figura B.9: Histograma cifrado-descifrado de video17 (A) y video18 (B)



(A) (B)

Figura B.10: Histograma cifrado-descifrado de video19 (A) y video20 (B)

Uso del CPU usando el GPP y DSP



(A) (B)

Figura B.11: Uso de CPU video1(A) y Uso de CPU video2 (B)

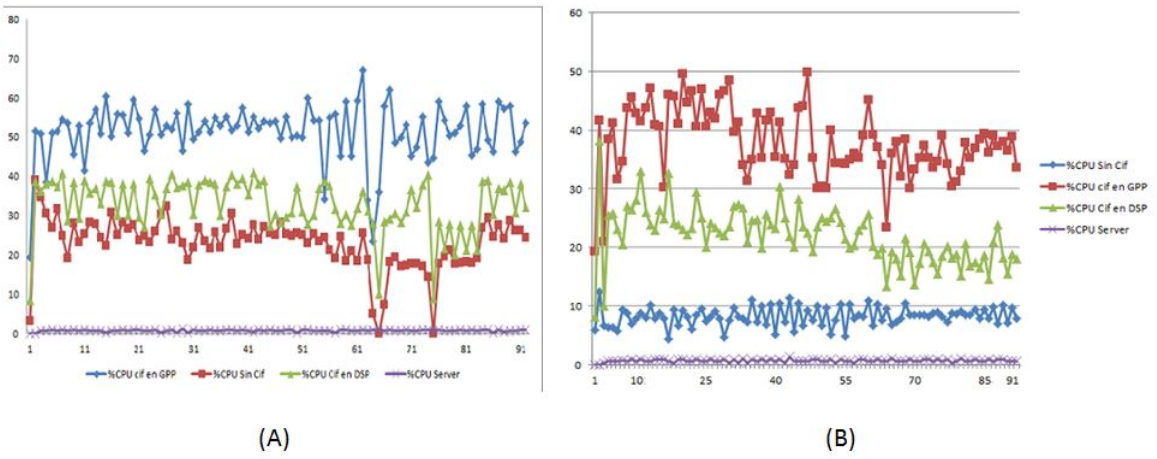


Figura B.12: Uso de CPU video3(A) y Uso de CPU video4 (B)

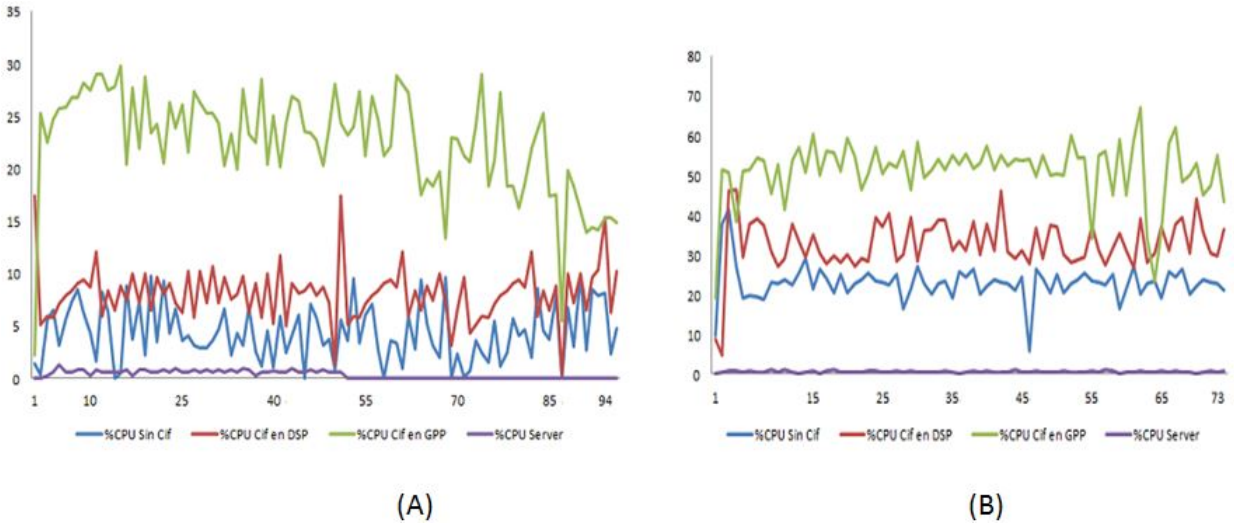
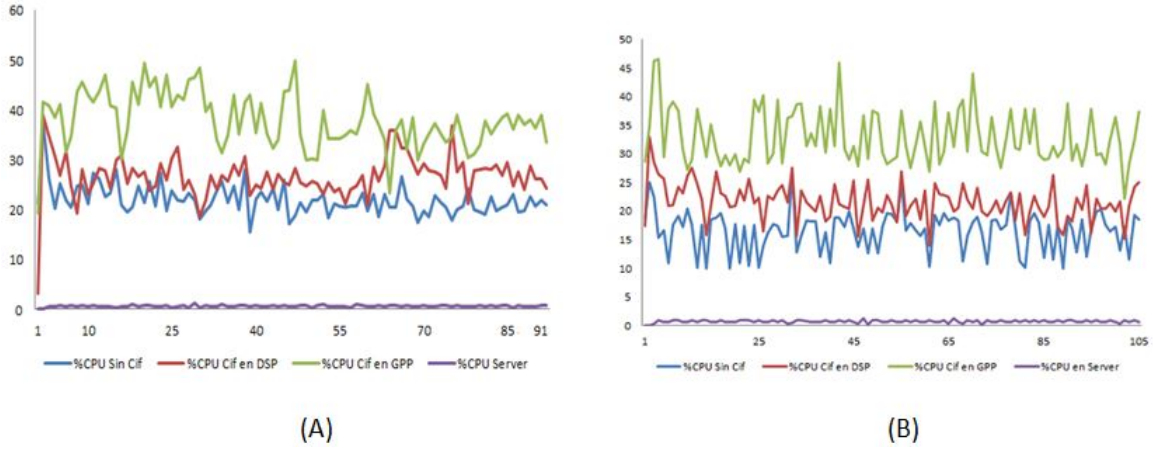
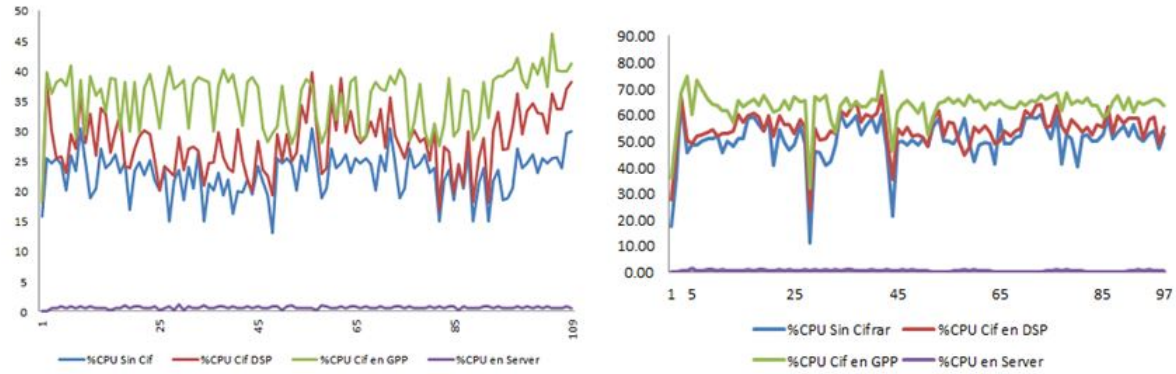


Figura B.13: Uso de CPU video5(A) y Uso de CPU video6 (B)



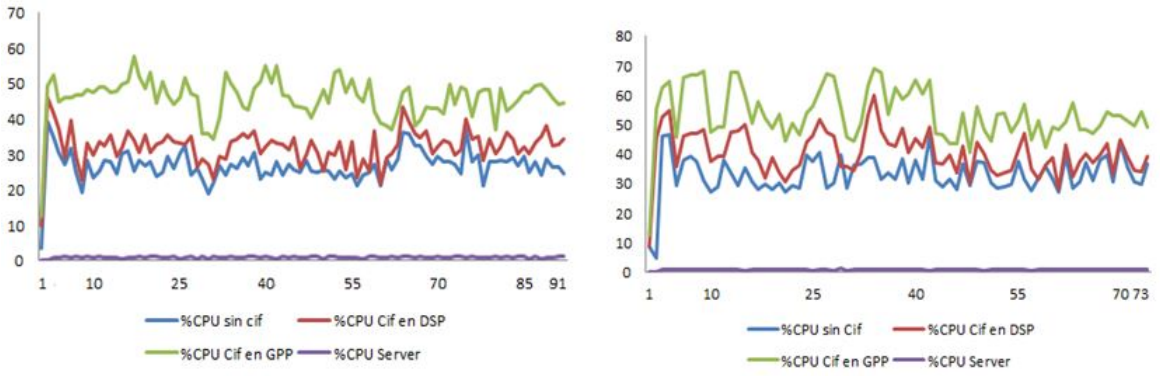
(A) (B)

Figura B.14: Uso de CPU video7(A) y Uso de CPU video8 (B)



(A) (B)

Figura B.15: Uso de CPU video9(A) y Uso de CPU video10(B)

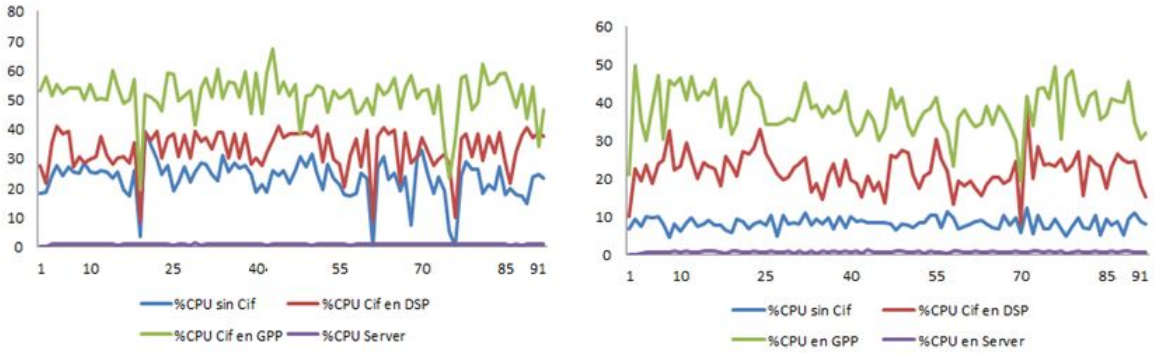


(A) (B)

Figura B.16: Uso de CPU video11(A) y Uso de CPU video12 (B)

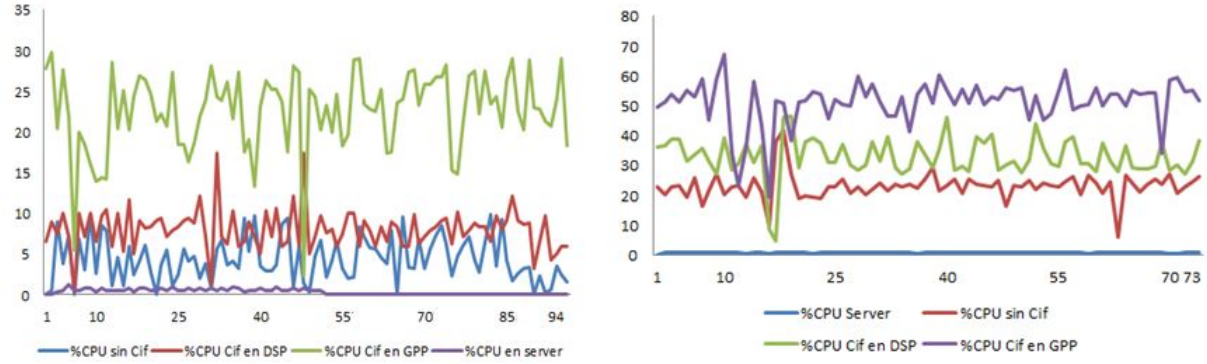
(A) (B)

Figura B.17: Uso de CPU video13(A) y Uso de CPU video14(B)



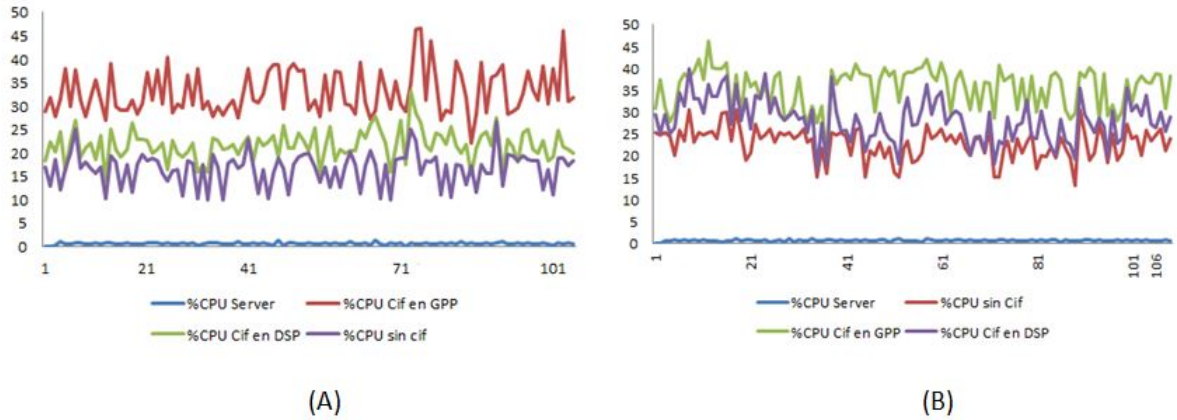
(A) (B)

Figura B.18: Uso de CPU video15(A) y Uso de CPU video16 (B)



(A) (B)

Figura B.19: Uso de CPU video17(A) y Uso de CPU video18(B)



(A) (B)

Figura B.20: Uso de CPU video19(A) y Uso de CPU video20 (B)

Bibliografía

[1] Abomhara, M., Zakaria, O., and Khalifa, O. O. An overview of video encryption techniques. *International Journal of Computer Theory and Engineering 2*, 1 (2010), 1793–8201.

[2] Agi, I., and Gong, L. An empirical study of secure mpeg video transmissions. In *Network and Distributed System Security, 1996., Proceedings of the Symposium on* (1996), IEEE, pp. 137–144.

[3] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and Norrman, K. Rfc 3711: The secure real-time transport protocol (srtp). *Dostupné z:<http://www. ietf. org/rfc/rfc3711. txt* (2004).

[4] Casamor, A. S. *Almacenamiento de información en los ordenadores.* www.bubok.es.

[5] Clark, M. P. *Data Networks, IP and the Internet: protocols, design and operation.* John Wiley & Sons, 2003.

[6] de Telecomunicaciones UIT, U. I. *Codificacion de video para comunicacion a baja velocidad binaria H.263*, primera edicion ed. Serie H: Sistemas audiovisuales y multimedios. ITU, 2005.

[7] Deering, S. E. Internet protocol, version 6 (ipv6) specification.

[8] Frederick, R., Jacobson, V., and Design, P. Rtp: A transport protocol for real-time applications. *IETF RFC3550* (2003).

[9] Ghanbari, M. *Standard codecs: Image compression to advanced video coding.* No. 49. Iet, 2003.

[10] Griwodz, C. Video protection by partial content corruption. In *Multimedia and Security Workshop at ACM Multimedia* (1998), vol. 98, Citeseer.

[11] Menezes, A. J., Van Oorschot, P. C., and Vanstone, S. A. *Handbook of applied cryptography.* CRC press, 1996.

[12] Okubo, S., McCann, K., and Lippmann, A. Mpeg-2 requirements, profiles and performance verification—framework for developing a generic video coding standard. *Signal Processing: Image Communication 7*, 3 (1995), 201–209.

[13] Postel, J. User datagram protocol. *Isi* (1980).

[14] Potdar, U., Ambawade, D., Apte, R. V., Chandok, G. S., Modi, R., and Sathe, B. M. Real-time video encryption for h. 263 videos. In *Distributed Framework and Applications (DFmA), 2010 International Conference on* (2010), IEEE, pp. 1–5.

[15] Qiao, L., and Nahrstedt, K. A new algorithm for mpeg video encryption. In *Proc. of First International Conference on Imaging Science System and Technology* (1997), Citeseer, pp. 21–29.

[16] Ramdan, A. A., and Munir, R. Selective encryption algorithm implementation for video call on skype client. In *Telecommunication Systems, Services, and Applications (TSSA), 2012 7th International Conference on* (2012), IEEE, pp. 120–124.

[17] Richardson, I. E. Video codec design, developing image and video compression system. *John Wiley & Sons, Ltd. ISBN 471485535* (2002), 9780471485537.

[18] Schulzrinne, H. Rtp: A transport protocol for real-time applications.

[19] Shah, J., Saxena, D., et al. Video encryption: A survey. *arXiv preprint arXiv:1104.0800* (2011).

[20] Shi, C., and Bhargava, B. An efficient mpeg video encryption algorithm. In *Reliable Distributed Systems, 1998. Proceedings. Seventeenth IEEE Symposium on* (1998), IEEE, pp. 381–386.

[21] Shi, C., and Bhargava, B. A fast mpeg video encryption algorithm. In *Proceedings of the sixth ACM international conference on Multimedia* (1998), ACM, pp. 81–88.

[22] Shi, C., and Bhargava, B. Light-weiimage and video encryption: from digital rights management to secured personal communicationght mpeg video encryption algorithm. In *Proc. Int. Conf. Multimedia* (1998), pp. 55–61.

[23] Shi, C., Wang, S.-Y., and Bhargava, B. Mpeg video encryption in real-time using secret key cryptography. In *in Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications* (1999), Citeseer.

[24] Spanos, G. A., and Maples, T. B. Performance study of a selective encryption scheme for the security of networked, real-time video. In *icccn* (1995), Published by the IEEE Computer Society, p. 0002.

[25] Stallings, W. *Fundamentos de seguridad en redes: aplicaciones y estándares.* Pearson Educación, 2003.

[26] Tang, L. Methods for encrypting and decrypting mpeg video data efficiently. In *Proceedings of the fourth ACM international conference on Multimedia* (1997), ACM, pp. 219–229.

[27] Uhl, A., and Pommer, A. *Image and video encryption: from digital rights management to secured personal communication*, vol. 15. Springer, 2005.

[28] Van Tilborg, H. C., and Jajodia, S. *Encyclopedia of cryptography and security*, vol. 1. Springer, 2011.

[29] Washington, L. C., and Trappe, W. *Introduction to cryptography: with coding theory*. Prentice Hall PTR, 2002.

[30] Watkinson, J. *The MPEG handbook*. Taylor & Francis, 2004.

[31] Wen, J., Severa, M., Zeng, W., Luttrell, M. H., and Jin, W. A format-compliant configurable encryption framework for access control of video. *Circuits and Systems for Video Technology, IEEE Transactions on 12*, 6 (2002), 545–557.

[32] William, S., and Stallings, W. *Cryptography and Network Security, 4/E*. Pearson Education India, 2006.