4. MARCO REFERENCIAL

4.1 COMPUTACIÓN FORENSE

El término computación forense, se originó a finales de los 80's con los profesionales de las leyes; se usaba para denotar el examen de computadores personales en busca de evidencia digital del crimen (algunos prefieren llamar este aspecto de la computación forense, análisis).

A medida que la cantidad de computadores creció en la red, la informática forense evolucionó hasta convertirse en un término para el análisis después del incidente, de los computadores que han sido víctimas de intrusos o códigos maliciosos. Las personas a menudo describen el primer caso, en el que el tráfico de la red es capturado y analizado, como red forense. Algunos han argumentado que la informática forense es un término más preciso; cada vez hay más pruebas digitales que son capturadas de objetos comúnmente no pensados, como el computador[3], cámaras digitales o celulares.

A continuación se listan algunas definiciones de computación forense:

Según el FBI, la informática o computación forense es la ciencia de adquirir, preservar, obtener y presentar datos que han sido procesados electrónicamente y guardados en un medio computacional.

[3] YASINSAC, Alec."Computer Forensics Education". IEEE SECURITY & PRIVACY. 2003. pág 1. http://www.cs.albany.edu/~erbacher/publications/ForensicsEducationPaper.pdf

Judd Robbins, un prominente investigador en computación forense, define computación forense como “la aplicación de técnicas de investigación y análisis computacional en el interés de determinar evidencia legal potencial”. Otros expertos han llevado la definición un paso adelante, creyendo que la computación forense ha evolucionado y se ha convertido en una ciencia. Básicamente, computación forense es el trabajo de un detective digital. Es buscar en una escena del crimen evidencia digital, procesarla en un ambiente de laboratorio certificado y finalmente presentando los resultados de procedimientos legales. En otras palabras, es similar a realizar una autopsia, excepto que en este caso se realiza sobre un dispositivo digital y no en un cuerpo humano[4].

“Computación forense es la captura, procesamiento, preservación y análisis de la información obtenida de un sistema, red, aplicación u otro recurso computacional para determinar la fuente de un ataque sobre estos recursos”[5].

Es, también, el uso de técnicas especializadas para recuperación, autenticación y análisis de datos electrónicos, cuando un caso involucra cuestiones relacionadas con la reconstrucción de la forma como fue usado un computador, el examen de datos residuales y la autenticación de datos por medio de técnicas de análisis. La computación forense requiere expertos especializados que van más allá de las técnicas disponibles de recolección de datos y preservación para usuarios finales o sistemas de apoyo[6].

---

[4] OSELES, Lisa. “Computer Forensics: The Key to Solving the Crime”. 2001. pág. 4. http://faculty.ed.umuc.edu/~meinkej/inss690/oseles_2.pdf

[5] WEISE, Joel; POWELL, Brad. “Using Computer Forensics When Investigating System Attacks” Sun BluePrints™ OnLine. 2005. pág. 3. http://www.sun.com/blueprints/0405/819-2262.pdf

[6] “Glossary for Computer Forensics from Precise Cyber Forensics”. 2008. http://precisecyberforensics.com/glossary.html

4.2 EVIDENCIA DIGITAL

La evidencia digital es información de valor probatorio constituida por campos magnéticos y pulsos electrónicos que pueden ser recolectados y analizados con herramientas y técnicas especiales. Abarca cualquier información en formato digital que pueda establecer una relación entre un delito y su autor.

Para ser valorada como tal, el proceso de recolección de la misma debe ser avalado por las leyes propias del lugar y debe ser reconocida como evidencia por una entidad oficial.

La evidencia digital hace parte de la evidencia física, pero tiene como características propias que puede ser duplicada y copiada sin alteraciones respecto a la original. Desde el punto de vista del derecho probatorio, puede ser comparable con “un documento” como prueba legal. Con el fin de garantizar su validez probatoria, debe reunir las siguientes características:

Autenticidad: Garantizar que sus contenidos no han sido modificados; que la información proviene de la fuente identificada y que la información externa a ella es precisa.

Precisión: Debe ser posible relacionarla positivamente con el incidente. Los procedimientos seguidos y las herramientas utilizadas para su recolección, manejo, análisis y posterior presentación en una corte deben ser confiables. Adicionalmente, debe haber alguien que pueda explicar cómo fueron realizados los procedimientos y con qué tipo de herramientas se llevaron a cabo.

Suficiencia: Debe, por sí misma y en sus propios términos, mostrar el escenario completo, y no una perspectiva de un conjunto particular de circunstancias o eventos.

Con el fin de garantizar la validez de la evidencia digital manejada en una investigación judicial, la IOCE[7] definió cinco principios[8] para la recuperación, preservación y examinación de la misma:

- En la incautación de la evidencia digital, las acciones tomadas no deben cambiar la evidencia.
- Cuando es necesario para una persona acceder a la evidencia digital original, esa persona debe ser competente en ciencias forenses.
- Toda actividad relacionada con la incautación, acceso, almacenamiento o transferencia de evidencia digital tiene que estar completamente documentada, preservada y disponible para revisión.
- Todo individuo es responsable de todas las acciones tomadas con respecto a la evidencia digital mientras ésta esté en su posesión.
- Cualquier organización que es responsable de incautar, acceder, almacenar o transferir evidencia digital es responsable de actuar conforme a estos principios.

Además, la IOCE definió que los principios de la evidencia digital debían estar regidos por los siguientes atributos:

- Consistencia con todos los sistemas legales.
- Permitir el uso de un lenguaje común.
- Durabilidad.
- Habilidad para cruzar límites internacionales.

[7] International Organization on Computer Evidence.

[8] Los principios fueron presentados y aprobados en la Conferencia Forense Internacional de crimen de alta tecnología, en octubre de 1999.

- Habilidad para infundir confianza en la integridad de la evidencia.
- Aplicabilidad a toda la evidencia forense.
- Aplicabilidad en todos los niveles, incluido el individuo, la organización y el país.

4.3 FORMATO DE ARCHIVO

Un formato de archivo es una forma específica de codificar datos digitales para garantizar su persistencia en la forma de archivo. Un archivo es una colección nombrada de datos relacionados que pueden ser almacenados y accedidos por su nombre.

Los formatos de archivo son necesarios porque los computadores almacenan, manipulan y comunican los datos en binario, haciendo que sea necesario tener algún sistema para convertir los datos a su forma binaria y devolverlos luego a una forma que sea fácilmente comprensible por humanos.

El tipo de archivo más común es el de texto. De éste, el más simple y útil es el texto plano, que consiste únicamente en lo códigos binarios de los caracteres legibles (como las letras de un alfabeto, los números y los signos de puntuación) y los de un grupo de caracteres de control. Una de las ventajas más grandes del formato de archivo de texto plano es que es fácilmente legible, rápido de examinar y hacer búsqueda en sus contenidos, aún si no se cuenta con software especializado, y además, es fácil hacer conversiones desde otros formatos de archivo.

La identificación de los formatos de archivo varía de acuerdo al sistema operativo. Por ejemplo, en los sistemas operativos Windows, el formato del archivo se

determina con base en la extensión, mientras que en Linux y otros sistemas operativos basados en Unix, el formato se identifica con el número mágico, el cual es un número incrustado al inicio o cerca del inicio del archivo.

4.4 PROCEDIMIENTO PARA EL MANEJO DE INVESTIGACIONES FORENSES

En el artículo de la ACIS "Evidencia Digital en el Contexto Colombiano", Daniel A. Torres, Jeimy J. Cano y Sandra J. Rueda definen una aproximación metodológica que permite el manejo adecuado de la evidencia digital, minimiza la posibilidad de cometer errores en su manejo, garantizando la admisibilidad de la misma en situaciones jurídicas[9].

La aproximación propuesta incluye cinco etapas, que serán expuestas a continuación.

4.4.1 Planeación

En esta etapa se debe determinar el proceso que se seguirá para la recolección de la evidencia, identificando los presuntos actores involucrados y el grado de contaminación que pueda haber en la escena. La escena debe ser descrita detalladamente, considerando todo aquello que pueda constituir evidencia digital.

4.4.2 Recolección

Es la etapa más crítica, pues en ella se recoge y conserva la información relevante, garantizando que cumpla con la Ley 527 de la Legislación Colombiana, que dice: "Para valorar la fuerza probatoria de la información digital … habrá de

[9] TORRES, Daniel; CANO, Jeimy; RUEDA, Sandra. "Evidencia digital en el contexto colombiano: Consideraciones técnicas y jurídicas para su manejo". ACIS. 2006.
http://www.acis.org.co/index.php?id=856

tenerse en cuenta la confiabilidad en la forma en la que se haya generado, archivado o comunicado la información".

En los procedimientos de computación forense deberá trabajarse con una copia idéntica, bit a bit, desde el comienzo hasta el final del archivo, sin importar si hay espacios defectuosos o en blanco, usando una herramienta que permita copiar el contenido visible y el invisible de la evidencia digital.

4.4.3 Aseguramiento

En esta etapa se busca cumplir con los requisitos de la Ley 527 de la Legislación Colombiana: "Para valorar la fuerza probatoria de la información digital… habrá de tenerse en cuenta la confiabilidad en la forma en la que se haya conservado la integridad de la información y la forma en la que se identifique a su iniciador". El investigador debe contar con alguna estrategia para mantener y verificar la integridad de la evidencia.

4.4.4 Análisis

Para realizar las tareas de análisis se sugiere trabajar con dos respaldos (*backups*), de modo que si alguno de ellos resulta afectado, pueda extraerse una nueva copia sin perder la validez e integridad de la evidencia digital. En el análisis deberán llevarse a cabo tareas de reconstrucción y recuperación, pues puede existir información oculta o que no es fácilmente visible.

4.4.5 Presentación de la evidencia digital

Para presentar la evidencia digital Sommers, en el documento "*Downloads, Logs and Captures: Evidence from Cyberspace*", especifica que puede ser apropiado ofrecer dos posibilidades:

- Evidencia de bajo nivel: Mostrar la información tal y cómo es, sin ningún tipo de anotación y modificación.

- Evidencia “editada”: Sólo se muestra la información relevante, explicando que se hizo con ella y por qué.

“La evidencia digital debe ser cuidadosamente recopilada y manejada, para posteriormente cumplir con los requisitos de admisibilidad en una corte. Independiente de una legislación particular, es esencial garantizar la confiabilidad e integridad de la evidencia”[10].

4.5 TEORÍA DE LA INFORMACIÓN

El origen de la teoría de la información está ligado al hito histórico de la publicación del artículo de C.E. Shannon “*A Mathematical Theory of Communication*” en Julio y Octubre de 1948[11]. Su aparición estuvo motivada por la necesidad de simplificar la transmisión de información en forma más eficiente a través de distintos canales.

Shannon fue quien, por primera vez, definió el concepto de información en términos del tamaño del código binario necesario para representar la cantidad de información del mensaje, sin tener en cuenta el significado que éste conlleva. Según la teoría de la información, dos textos tienen la misma cantidad de información, siempre que ambos se puedan codificar con el mismo número de bits.

En forma precisa, Shannon definió información, en términos del número de bits requerido para transmitir un determinado mensaje. En cuanto tiene que ver con la transmisión de la información, ya sea alfanumérica (palabras, números), música,

[10] IBID
[11] SHANNON, *A Mathematical Theory of Communication*, http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

video o archivos en general, lo que importa en últimas, es el número de 0´s y 1´s necesarios para codificar la información y transmitirla.[12]

C.E. Shannon define entonces un modelo matemático para poder analizar los sistemas de comunicación por medio de variables aleatorias.[13]

En la teoría de la información, no se tiene en cuenta el contenido semántico del mensaje, objeto de la información. La información aquí se define como una medida, una cantidad discreta, igual que lo es la densidad o la masa, medible en el sentido físico.

Son cinco los elementos básicos en la descripción teórica de todo sistema general de información, de acuerdo al modelo propuesto por Shannon y Bush:

1 – Una *fuente* de información o un dispositivo que transforma la información o mensaje en algo apropiado para ser transmitido por un medio particular. Esta fuente, para el modelo que Shannon creó, codificará el mensaje en 0´s y 1´s.

2 – El *medio* o *canal* a través del cual se transmite el mensaje. En la tecnología moderna estos canales pueden ser los cableados de las redes telefónicas, las redes de microondas, las mismas ondas de radio y la tan útil red de la Internet.

3 – Un *dispositivo decodificador*, entendido como el proceso de convertir la serie de 1´s y 0´s nuevamente en el mensaje original, inteligible para quien lo recibe.

4 – El *destinatario* o *receptor* del mensaje.

5 – Una *fuente de ruido*, llamada también interferencia o distorsión, que podría afectar el mensaje, en forma impredecible durante la transmisión.

---

[12] PAREJA, Diego. Del bit a la revolución, un homenaje a Claude Shannon. www.matematicasyfilosofiaenelaula.info/conferencias/Shannon.pdf

[13] ROJAS, Jennifer. Modelo Matemático para la Tipificación en la Clasificación de los alelos HLA http://catarina.udlap.mx/u_dl_a/tales/documentos/mosl/rojas_b_js/

El enfoque de Shannon se encaminó hacia la codificación de la información como señales en forma completamente digital como sucesiones de 0´s y 1´s, a las que se referiría de allí en adelante como bits (binary digits), acogiendo la sugerencia de su colega de la Universidad de Princeton John Tukey. Después de Shanon, el problema del manejo de la información se encaminó a buscar la forma más eficiente de enviar secuencias discretas de pulsos eléctricos o electromagnéticos de un punto a otro.

Shannon pronto observó que, mientras pequeñas variaciones en la señal analógica, pueden distorsionar un mensaje y así la información contenida en él, la naturaleza discreta de una señal digital es menos propensa a la contaminación por el ruido en el sistema. Más aún, al adicionar extra bits a la señal, podía adicionarse un proceso automático de detección y corrección en el sistema. La codificación digital de la información y sus códigos auto correctores son hechos tan cotidianos, que ya no nos preguntamos, si podría ser de otra manera.

El enfoque de Shannon permite medir el tamaño de la información, en el sentido de saber cuánta información lleva una determinada señal. El proceso se reduce a contar el número mínimo de bits necesarios para codificar la información. Basado en esta simple idea, Shannon fue capaz de desarrollar una teoría cuantitativa del contenido de la información, muy útil sobre todo para los ingenieros quienes tienen que decidir sobre la capacidad de los canales, en las redes de comunicación.

En la teoría de Shannon lo que se mide es el tamaño, en el sentido binario, de la señal, no importa lo que esta señal pueda representar.

Todas las comunicaciones hoy se miden en bits por segundo, la misma noción que Shannon hizo precisa cuando habló sobre la capacidad de un canal de comunicación. También fue su teoría la que permitió usar los bits cuando se trata

de almacenar en el computador, imágenes, sonido y datos de diferentes formas. El Internet no habría sido posible sin la teoría de la información de Shannon. [14]

4.5.1 Entropía

La entropía sirve para dos fines relacionados entre sí: primero; es un indicador de la tendencia de los procesos naturales. Segundo; nos revela cuantitativamente la estructura estadística de movimientos internos de una forma muy parecida a como lo hace la teoría de la información con un conjunto de mensajes. Más aún, lo hace en forma análoga, ya que se toma la entropía como el algoritmo de la probabilidad termodinámica de un macro estado, tal como se mide la información de un mensaje por el algoritmo de la probabilidad de su aparición.

Evidentemente la entropía y la información son dos caras de la misma moneda, en el sentido que el orden interno u organización, implicando un mayor conocimiento o información de la composición interna del sistema, va siempre acompañado de su logaritmo o entropía.[15]

“La teoría de la información de Shannon tiene una característica muy importante: el concepto de entropía. La entropía se define como carencia de contenido de información en un mensaje. Estamos acostumbrados a pensar en información, como hechos, datos o evidencia en tal o cual asunto, pero en teoría de la información es incertidumbre, así que entre más bits de información se tenga más incertidumbre se tiene.

[14] PAREJA, Diego. Op. Cit., www.matematicasyfilosofiaenelaula.info/conferencias/Shannon.pdf

[15] SINGH, Jagjit. Teoría de la información del lenguaje y de la cibernética. 1982.

Dicho en pocos términos, información es lo que no se conoce. Si el receptor ya tiene la información, no se puede decir que una comunicación haya tenido lugar. Se recibe un mensaje del cual se desconoce su contenido. La teoría de la información habla precisamente acerca de los posibles mensajes que se pueden recibir o el número de mensajes entre los que se puede escoger, o en general, la teoría de la información habla de las propiedades estadísticas de un mensaje, comparado frente a la totalidad de los mensajes, sin tener en cuenta lo que el mensaje dice.

Un beneficio inmediato de la teoría de la información es que ofrece a los ingenieros las herramientas matemáticas necesarias para estimar la capacidad de los canales de comunicación, más exactamente, cuanta información, sin errores, puede ir de un punto A a otro B. La información que se quiere es la señal, la información que no se desea es ruido.

Si n es el número de posibles mensajes, el número x de bits necesarios para transmitir un mensaje entre los n posibles es logaritmo en base 2 de n.

Simbólicamente,

$$2^x = n$$

$$\lg n = x$$

Si se tienen n mensajes posibles, denominados $m_1, m_2, \dots, m_n$ y representados mediante la variable aleatoria M, cada uno con una probabilidad p_i asociada, entonces se define $H(M)$ como la incertidumbre asociada con la aparición del siguiente mensaje. Las propiedades que debe tener $H(M)$ son:

1. Debe ser continua en p_i
2. Si todos los p_i son iguales ($p_i = 1/n$) entonces H debe ser una función monotónica creciente de n.

3. Si un mensaje puede ser divido en dos mensajes diferentes, entonces el $H(M)$ original debe ser la suma ponderada de los valores individuales de $H(M)$.

Shannon demuestra en su artículo[16] que la única función que cumple con este requisito es:

$$H(M) = -\sum_{i=1}^{n} p_i \lg p_i$$

Con el fin de evitar la singularidad cuando se tienen mensajes con $p_i = 0$, se asume que $0 \lg 0 = 0$. Nótese que esto no representa ningún problema pues básicamente es lo mismo que decir que los mensajes con $p_i = 0$ no son tenidos en cuenta, lo cual es cierto pues no aportan nada a la *incertidumbre* del siguiente mensaje.

El valor H en la fórmula anterior, se conoce como entropía en la teoría de Shannon y representa la *incertidumbre* promedio asociada con los mensajes. Usualmente la entropía se mide en "bits por mensaje" o en una relación similar, si se está usando un conjunto de símbolos para transmitir el mensaje donde cada símbolo representa un mensaje, la entropía es el número de bits necesarios en promedio para representar un símbolo.

Por ejemplo, el conjunto extendido de caracteres ASCII tiene 256 caracteres. El logaritmo base 2 de 256 es 8, así que se requieren 8 bits por símbolo para poderlo representar. Para calcular la entropía se requieren las probabilidades p_i asociadas con cada símbolo. Si todos los símbolos son equiprobables, entonces $p_i = 1/n$ y por lo tanto $H = \lg 256 = 8$. Ahora, supongamos que sólo se utilizan 26 caracteres y que son equiprobables. En este caso $p_i = 1/26$ y $H = \lg 26 = 4{,}7004397$. Esto

[16] SHANNON, ibídem

quiere decir que se requieren únicamente $4{,}7$ bits para representar cada símbolo. Los demás bits son redundantes y hacen que la *incertidumbre* sea menor.

Shannon empleó pruebas matemáticas para mostrar que se puede usar la medida de entropía para medir la tasa promedio de transmisión de un canal sin ruido con capacidad C bits/sec. Si la fuente tiene entropía H y el canal una capacidad C bit/sec, entonces es posible codificar el mensaje de manera que en la transmisión utilice en promedio $H/C - \varepsilon$ símbolos por segundo, donde ε es arbitrariamente pequeño. Más aún, él probó que es imposible transmitir a una tasa superior a H/C.

En otras palabras: Se debe buscar una codificación que maximice la entropía H para hacer uso efectivo del canal. Para canales con errores, se define entropía condicional como

$$H(M)_y = -\sum_{i,j} p_{i,j} \log p_i(j)$$

donde

$$p_i(j) = \frac{p_{i,j}}{\sum_j p_{i,j}}$$

Para poder transmitir información se requiere que la capacidad del canal sea al menos $H(M)_y$ y se puede codificar de manera que se puedan corregir todos los errores salvo una pequeña fracción ε de los mismos.

"Shannon tomó las observaciones que la gente había hecho sobre el intercambio de información y las puso sobre bases sólidas, usando matemáticas y estadística."[17]

[17] PAREJA, Diego. Del bit a la revolución, un homenaje a Claude Shannon. www.matematicasyfilosofiaenelaula.info/conferencias/Shannon.pdf

4.6 SOFTWARE LIBRE

El término “software libre” es usado cuando los usuarios tienen libertades para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

El grupo GNU ha definido que para que un software sea considerado como libre, debe tener presentes las siguientes cuatro libertades:

- Libertad 0: Poder usar el programa sin importar el propósito que se tenga.
- Libertad 1: Poder acceder al código fuente del programa, para estudiar su funcionamiento y adaptarlo a necesidades particulares.
- Libertad 2: Poder distribuir copias.
- Libertad 3: Poder modificar el programa y hacer públicas dichas mejoras, beneficiando a toda la comunidad.

“Equivalente a software libre es el término Open Source Software (‘programas de fuente abierto’), promovido por Eric Raymond y la Open Source Initiative. Filosóficamente, el término es muy distinto, ya que hace énfasis en la disponibilidad de código fuente, no en la libertad”[18].

4.7 RECONOCIMIENTO DE PATRONES

Las búsquedas han sido y son un problema fundamental en la ingeniería de sistemas, ya que virtualmente están presentes en todas las aplicaciones. Las aplicaciones simples presentan problemas simples de búsqueda, sin embargo, las

[18] GONZÁLEZ, Jesús. Introducción al Software Libre. UOC – Formación de Posgrado.2003. p. 17 http://cv.uoc.edu/~fcaulas/20041/90.783/portada_Into.pdf

aplicaciones complejas han requerido de manera general, métodos más sofisticados de búsqueda.

Las búsquedas han sido aplicadas tradicionalmente sobre estructuras de datos claramente definida, que por lo general siempre son datos numéricos y / o alfanuméricos. Para estos casos se define la consulta de la búsqueda, y la cadena alfanumérica o de número que corresponda exactamente a la especificación de la consulta son recuperados. (Este tipo de búsquedas son muy comunes en los repositorios de Base de Datos)

Ahora bien, con la evolución de las Tecnologías de Información y comunicación, emergen por así llamarlo, nuevos repositorios de información no estructurados. Es así como ahora es común encontrar nuevos tipos de datos, como por ejemplo, texto, audio, imágenes, etc. y donde es difícil, de manera manual o computacional, estructurar las claves de búsqueda necesarias para realizar búsquedas sobre ellos. Aquí se requiere entonces, definir nuevos modelos y algoritmos de búsqueda, que puedan actuar eficientemente sobre estos nuevos repositorios no estructurados de información.

El concepto unificado de “búsqueda de similitud” o “búsqueda de proximidad”, trata de buscar elementos similares o cercanos a un elemento “base” dado. Por lo general, esto lo hace mediante el modelado de una función de distancia que cumple o satisface el triangulo de desigualdad, y utilizando lo que se denomina *espacio métrico*. El espacio métrico es un conjunto de objetos o elementos similares o cercanos entre sí, hecho que se formaliza matemáticamente con la noción de *métrica* o función de distancia.

En los espacios métricos, la única información disponible es la distancia entre los objetos. Por lo general, el cálculo de ésta es muy costoso, lo que hace necesario reducir el número de distancias evaluadas. Lo que se hace para los espacios métricos generales, es construir un “índice”, el cual es una estructura de datos diseñada para reducir el número de evaluaciones de distancia en una búsqueda.

Dado que el problema de las búsquedas ha aparecido en diferentes áreas, soluciones diversas también han aparecido en campos no relacionados como: estadística, geometría computacional, Inteligencia Artificial, Base de Datos, biología computacional, Data Mining y Reconocimiento de patrones. Por lo tanto no es de sorprender, que dado las múltiples soluciones, y debido a la falta de comunicación entre las comunidades científicas de éstas áreas, estas soluciones se reinventes una y otra vez.

4.7.1 Búsquedas de proximidad

Existe un conjunto de aplicaciones donde aparece el concepto de búsqueda de proximidad. Por lo general, para estas aplicaciones se utilizan algoritmos indexados para búsqueda de proximidad, los cuales consisten en la construcción de un conjunto de clases de equivalencias, donde se descartan algunas clases y se busca sobre exhaustivamente sobre el resto. Entre estas aplicaciones se encuentran las que realizan búsquedas de contenido (Base de Datos estructuradas y objetos multimedia), recuperación de texto, biología computacional y reconocimiento de patrones. En el contexto del presente trabajo, las búsquedas aproximadas se analizan pues el problema de identificar el tipo de un archivo es conceptualmente una búsqueda aproximada.

4.7.2 Reconocimiento de patrones y aproximación de funciones

Una definición simplificada de reconocimiento de patrones, es la construcción de un *function aproximator*. Para este caso, un problema tiene un número finito de muestras y cada muestra de datos es etiquetada como perteneciente a cierta clase. Cuando se provee una nueva muestra de datos, el sistema debe etiquetar esta nueva muestra con una de las etiquetas de datos conocidas. En otras palabras, se puede pensar que el clasificador es una función sobre el espacio de

objetos (datos) hacía el conjunto de etiquetas. En este sentido, todos los clasificadores son considerados los *function approximators*.

Si los objetos son vectores de números reales de m-dimensiones, entonces las redes neuronales y la lógica difusa (*fuzzy logic*) son las elecciones naturales. Otro tipo muy popular y universal de *function approximator* es el clasificador del k-vecino-mas-cercano, que consiste básicamente en encontrar los k objetos más cercanos a la muestra no etiquetada, y asignarle la etiqueta obteniendo el mayor entre los k objetos más cercanos.

4.7.3 Espacios Métricos

A continuación se presenta las notaciones básicas para el problema de satisfacer las consultas de proximidad y para el modelo utilizado para agrupar y analizar los diferentes algoritmos.

El conjunto X denota el universo de *objetos validos*. Un subconjunto finito de éste, U, de tamaño $n = |U|$, es el conjunto de objetos donde buscamos. (U es conocido como el *diccionario*, *base de datos*, o simplemente conjunto de *objetos* o *elementos*) La función $d: X \times X \rightarrow R$, denota una medida de "distancia" entre objetos (i.e. entre más pequeña la distancia, más cercano o más similares son los objetos)

Las funciones de distancia tienen las siguientes propiedades:

(p1) Positiva: $\forall x, y \in X, d(x, y) \geq 0$

(p2) Simétrica: $\forall x, y \in X, d(x, y) = d(y, x)$

(p3) Reflexiva: $\forall x \in X, d(x, x) = 0$

(p4) Estrictamente positiva (en algunos casos): $\forall x, y \in X,\ x \neq y \Rightarrow d(x, y) > 0$

(p5) Desigualdad triangular: $\forall x, y, z \in X, d(x, y) \leq d(x, z) + d(z, y)$

La pareja (X, d), donde d cumple las propiedades (p1) a (p5), se denomina *espacio métrico*.

4.7.4 Consultas de proximidad

Existen básicamente tres tipos de consultas de interés en espacios métricos.

Consulta por rangos $(q, r)_d$. Recupera todos los elementos que se encuentran entre la distancia r a q. Esto es: $\{ u \in U \,/\, d\,(q, u\,) \leq r \}$. Este es el tipo más básico de consultas.

Consulta del vecino cercano $NN(q)$. Recupera los elementos cercanos a q en U. Esto es: $\{ u \in U \,/\, \forall v \in U\; d\,(q, u\,) \leq d\,(q, v\,)\}$

Consulta de los k *Vecinos cercanos* $NN_k(q)$ Recupera los k elementos cercanos a q en U. Esto es, recupera un conjunto $A \subseteq U$ tal que $|A| = k, \forall u \in A, v \in U - A, d(q, u) \leq d(q, v)$

El tiempo total para evaluar una consulta se puede definir como:

$T = evaluaciones\ de\ distancia * complejidad\ de\ d() + tiempo\ extra\ de\ CPU + tiempo\ I/O$, y lo que se pretende siempre es disminuir T, sin embargo para algunos casos evaluar d es tan costoso que los otros componentes del costo, pueden ser omitidos.

5. ESTADO DEL ARTE

5.1 IDENTIFICACIÓN DEL FORMATO DEL ARCHIVO

Los sistemas operativos no proporcionan ningún mecanismo formal para identificar el formato de un archivo. En algunos casos, como en MS-DOS, Windows y Unix, se utiliza parte del nombre para definir el formato. Ésta es la base del sistema MIME[19]. Tiene el problema de que cualquier agente, usuario o software, puede modificar fácilmente la descripción del archivo. En MacOS, en las primeras versiones, se almacenaban el formato y el creador como metadatos del archivo. Sin embargo, estos eran libremente modificables por un programador. La estrategia que han asumido la mayoría de los formatos estándares en Internet es utilizar un encabezado en el archivo, el cual se ajusta a ciertas convenciones. Ésta es la base del comando *file* de Unix.

5.1.1 Extensión del nombre del archivo

La extensión del archivo son los últimos caracteres del nombre del archivo, después del punto. Es un sufijo al final del nombre del archivo que indica de qué tipo es, por tanto puede ser considerada un tipo de metadato. Por ejemplo, en el archivo reporte.txt, la extensión es “txt”, e indica que el archivo es un documento de texto.

[19] Multipurpose Internet Mail Extensions.

La extensión, en algunos sistemas operativos como Windows, le dice al sistema operativo del computador qué programa debería ser usado para abrir el archivo. También es útil para que el usuario conozca que información puede tener un archivo sólo con mirar su nombre.

En sistemas operativos como Windows o Mac OS X, el usuario puede cambiar las extensiones del archivo, con lo que cambia también el programa que el computador usa para abrirlo. En algunos casos, esto evita que las aplicaciones puedan tener acceso a la información; por ejemplo, si un archivo .txt es renombrado a .doc, Microsoft Word debería poder abrirlo, pero si un .ppt es renombrado a .pdf, Acrobat Reader no reconocerá el archivo y no permitirá su visualización.

5.1.2 Número mágico

El número mágico, también conocido como firma del archivo[20], se encuentra ubicado en los primeros bytes del archivo, indicando su formato. Se llama número mágico porque su propósito y significado no es aparente si no se tiene un conocimiento adicional. Se puede usar un visor hexadecimal para acceder a este número, que generalmente, para los formatos de archivo más comunes, representa el nombre del tipo de archivo.

[20] Esta noción de firma no debe ser confundida con la firma digital utilizada en aplicaciones criptográficas.

Firma hexadecimal	Descripción ASCII[21] de la firma	Extensión del archivo
42 4D	BM	BMP Windows Bitmap Image
43 44 30 30 31	CD001	ISO ISO-9660 CD Disc Image
52 49 46 46 xx xx xx xx 57 41 56 45 66 6D 74 20	RIFF.... WAVEfmt	WAV Resource Interchange File Format
52 61 72 21 1A 07 00	Rar!...	RAR WinRAR compressed archive file
4C 00 00 00 01 14 02 00	L.......	LNK Windows shortcut file
4D 54 68 64	MThd	MID, MIDI Musical Instrument Digital Interface
4D 69 63 72 6F 73 6F 66 74 20 56 69 73 75 61 6C 20 53 74 75 64 69 6F 20 53 6F 6C 75 74 69 6F 6E 20 46 69 6C 65	Microsoft Visual Studio Solution File	SLN Visual Studio .NET Solution file
46 4C 56	FLV	SWF Flash video file

Tabla 1: Ejemplos de números mágicos para formatos de archivo comunes

No todos los formatos cuentan con un número mágico; los archivos de texto plano, como los HTML (hypertext markup language), los XHTML (extensible HTML), los XML (extensible markup language) y los archivos de código fuente no están identificados con una firma de archivo.

[21] American Standard Code for Information Interchange

5.1.3 Comando *File*

El comando *file* prueba cada argumento en un intento de clasificarlo. Hay tres conjuntos de pruebas, las cuales realiza en el siguiente orden: pruebas de sistema de archivos, pruebas de número mágico y pruebas de idioma. La primera prueba que tenga éxito hace que se imprima el tipo de archivo.

El tipo que se imprime usualmente contiene una de las siguientes palabras:

- text: El archivo contiene sólo caracteres imprimibles y unos pocos caracteres de control comunes y es probablemente seguro leerlo en una terminal ASCII.
- executable: El archivo contiene el resultado de compilar un programa en una forma entendible para algún kernel de Unix u otro.
- data: Para todo lo demás. El archivo es usualmente binario o no imprimible.

Los test del sistema de archivos se basan en el examen del resultado retornado por el comando *stat* (2) que es un llamado al sistema. El programa chequea si el archivo está vacío o si es un archivo de algún tipo especial. Cualquier tipo de archivo conocido apropiado para el sistema que se está corriendo son intuidos si son definidos en el encabezado de sistema del archivo <sys/stat.h>.

Las pruebas de número mágico son usadas para chequear si el archivo contiene datos de un formato particular. Estos archivos tienen un número mágico ubicado en un lugar específico cerca del inicio del archivo, el cual es parte integral de la definición del formato y es utilizado por las aplicaciones para identificar que el archivo esté bien formado. La información para identificar estos archivos es leída del archivo compilado *magic*. Éste es una base de datos que almacena información sobre todos los formatos conocidos por los desarrolladores del comando.

Si el archivo no coincide con ninguna de las entradas del archivo mágico este es examinado para ver si se trata de un archivo de texto. Si el archivo pasa

cualquiera de estas pruebas se reporta su conjunto de caracteres, adicionalmente el comando *file* identificará otras características de los archivos tipo texto.

Una vez el comando ha identificado el conjunto de caracteres usado en un archivo tipo texto, intentará de determinar en qué lenguaje está escrito. Las pruebas de lenguaje buscan cadenas de caracteres particulares que pueden aparecer en cualquier parte de unos cuantos bloques del principio del archivo.

Cualquier archivo que no haya podido identificado como escrito en alguno de los conjuntos de caracteres soportado es llamado data.

A continuación se listan las opciones recibidas como parámetro por el comando *file*, con el fin de obtener resultados más detallados.

-b, --brief

No anteponer los nombres de los archivos a la salida.

-c, --checking-printout

Causa una impresión de control de la forma identificada del número mágico. Esto es usualmente usado en conjunto con la opción –m para depurar un archivo mágico nuevo antes de su instalación.

-C, --compile

Escribe un archivo de salida magic.mgc que contiene una version pre-analizada del archivo.

-f, --files-from nombre del archivo

Lee los nombres de los archivos a ser examinados desde nombre del archivo (uno por línea) antes de la lista de argumentos. Al menos un nombre de archivo debe ser presentado.

-F, --separator separador

Usa la cadena de caracteres especificada como el separador entre el nombre del archivo y el resultado retornado.

-h, --no-dereference

Hace que los enlaces simbólicos no sean seguidos (en los sistemas que apoyan los enlaces simbólicos). Este es el valor por defecto si la variable de entorno POSIXLY_CORRECT no está definida.

-i, --mime

Hace que la salida del comando *file* sea tipo mime en vez del tradicional formato legible por el hombre.

-k, --keep-going

No pare en la primera concordancia, siga ejecutándose.

-L, --dereference

Hace que los enlaces simbólicos sean seguidos. Este es el default si la variable POSIXLY_CORRECT está definida.

-m, --magic-file lista

Especifica una lista alterna de archivos que contienen números mágico. Esto puede ser un solo archivo, o una columna separada de los archivos.

-n, --no-buffer

Fuerza *stdout* a ser vaciado después de comprobar cada archivo. Esto sólo es útil si se está comprobando una lista de archivos.

-p, --preserve-date

En los sistemas que apoyan *utime* (2) o *utimes* (2), intentando preservar el tiempo de acceso de archivos analizados, que el comado *file* nunca los modifique.

-r, --raw

No traduzca caracteres no imprimibles a \000. Normalmente el comando *file* traduce caracteres no imprimibles a su representación en octal.

-s, --special-files

Normalmente el comando *file* solamente intenta leer y determinar los archivos que el comando *stat*(2) reporta como ordinarios. Esto previene problemas, porque la lectura de archivos especiales puede conllevar a consecuencias peculiares. Especificar la opción –s hace que el comando *file* también lea los archivos que son catalogados como especiales. Esto es útil para determinar los tipos de sistemas de archivos de los datos en particiones de disco *raw*.

-v, --version

Imprime la versión del programa y sale.

-z, --uncompress

Trata de mirar dentro de archivos comprimidos

6. MÉTODO PROPUESTO

Hay momentos en los que se hace necesaria la identificación de un archivo a través de las propiedades de su estructura; los investigadores en computación forense tienen varias herramientas a su disposición que les permiten conocer el tipo de archivo con el que se encuentran trabajando, algunas de estas herramientas usan el número mágico para su identificación, sin embargo, hay archivos que no contienen el número mágico o que su secuencia de bytes representa un archivo parcial que no incluye una estructura clara para su reconocimiento. El siguiente método tiene como fin, presentar formalmente los pasos que deben seguirse a la hora de visualizar archivos incompletos o de tipo desconocido, haciendo uso de algunos métodos de reconocimiento de los mismos.

Se parte del supuesto de que el archivo es legible, esto quiere decir que está en forma digital y puede ser leído byte a byte. Si el archivo hace parte de una investigación forense, es importante que se hayan seguido las consideraciones técnicas y jurídicas para su manejo[22], para que la evidencia digital se preserve válida y completa.

6.1 ETAPA 1: Identificación del archivo

La etapa de identificación del archivo tiene como objetivo sugerir el formato del archivo o fragmento de archivo que se está analizando. Para ello se tienen una

[22] TORRES, Daniel; CANO, Jeimy; RUEDA, Sandra., Op. Cit. http://www.acis.org.co/index.php?id=856

serie de estrategias, que pueden ser utilizadas dependiendo del tipo de problema que se enfrente, y que serán listadas a continuación.

6.1.1 Identificar la extensión del archivo

Identificar la extensión del archivo es la forma más fácil pero menos confiable de determinar el formato del archivo. En entornos gráficos, la extensión del archivo estará asociada a un ícono, cuando el formato es conocido (esto quiere decir que el sistema operativo a asociado un programa para abrir ese tipo de archivos). Desde el punto de vista técnico es poco confiable debido a que el usuario puede modificarla con facilidad, pero puede brindar una primera aproximación para hacer la identificación.

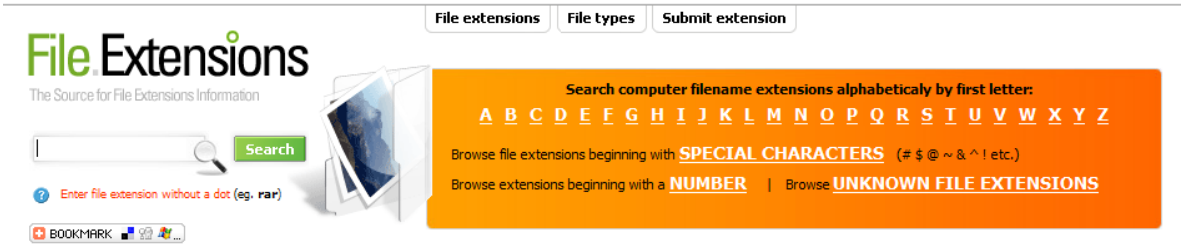

Ilustración 1: Página de búsqueda de tipos de archivo por extensión[23]

Existen páginas cuyo fin es investigar y recolectar una amplia lista de extensiones de archivo, muchas de ellas con explicaciones detalladas de la estructura del archivo y la forma como son usados. También, en algunos casos, se incluyen los programas asociados al tipo de extensión.

[23] File Extensions. 2008. http://www.file-extensions.org/

Si se tiene la extensión del archivo sobre el que se está trabajando, ésta podrá ser consultada en una de las bases de datos disponibles en la web para determinar el formato al que corresponde y el software con el cual puede ser visualizado.

6.1.2 Determinar si el archivo es texto

Para identificar si un archivo es texto, basta con acceder a él desde un editor hexadecimal. La visualización mostrada por el editor permitirá a un usuario identificar los patrones lingüísticos válidos.

También puede utilizarse el comando *strings* de Linux, el cual muestra las secuencias de más de cuatro caracteres imprimibles de largo, seguidos por un carácter no imprimible. Por defecto, *strings* sólo imprime las cadenas de caracteres de las secciones inicializadas y cargadas de archivos de objetos, para otros tipos de archivos imprime las cadenas de todo el archivo. *Strings* es útil para determinar el contenido de archivos que no son de texto.

Otra estrategia menos eficiente es comparar el archivo contra un diccionario, lo que permitiría también identificar el idioma del texto.

6.1.3 Ejecutar el comando *file*

Este comando de Linux es útil para ayudar a identificar el archivo. Para usarlo basta con abrir una consola en Linux y escribir el comando file seguido del nombre del archivo o fragmento de archivo a analizar.

El resultado podrá ser:

- *text*, para archivos de texto.
- *executable*, para archivos ejecutables en un *kernel* Linux.

- data, para archivos binarios.

Si el comando *file* identifica el archivo, imprime el formato al que pertenece, y en algunos casos, características presentes en los encabezados del formato del archivo.

6.1.4 Identificar el número mágico

Para acceder al número mágico del archivo es necesario contar con un editor hexadecimal[24]. El fragmento de archivo deberá estar completo en sus primeros bytes.

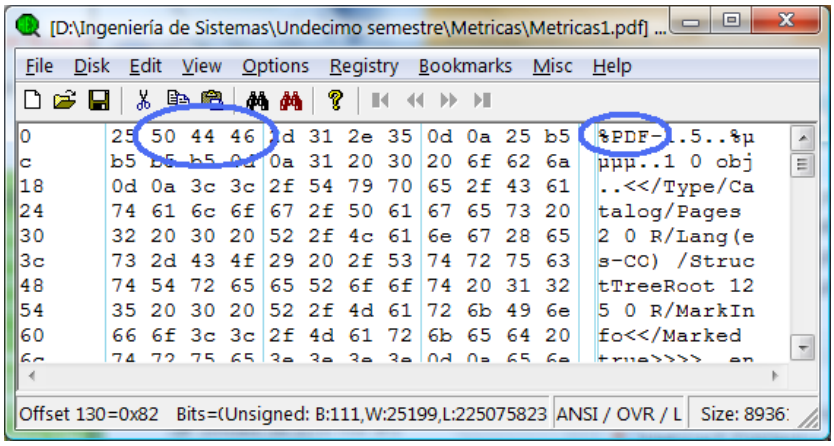

Ilustración 2: Identificación del número mágico en un archivo PDF

Una vez abierto el archivo, deberá buscarse en los primeros bytes los valores que se encuentran almacenados allí, y luego compararlos con una tabla de números mágicos.

[24] En este proyecto se trabajó con la versión 1.1.0 del editor hexadecimal frhed, desarrollado por Raihan Kibria, en el año 2000.

6.1.5 Utilizar herramientas y librerías públicas de identificación

- **NSRL**

El NSRL[25] es una biblioteca de referencia de software, mantenida por el NIST[26], que contiene firmas únicas que permiten identificar gran cantidad de archivos de aplicaciones comerciales o proveídos por manufactureros conocidos. Con estas firmas, los investigadores forenses pueden filtrar los archivos en un computador, descartando los que están en la base de datos, para tomar aquellos que puedan haber sido perturbados, duplicados o de contenido poco confiable. Las firmas del archivo son únicas para cada tipo de archivo, y pueden ayudar a identificar si:

- El archivo ha sido alterado.
- El archivo ha sido renombrado o se ha incurrido en algún intento para ocultarlo.
- El archivo es lo que parece ser.
- El archivo no está donde debería estar.
- El archivo se encuentra en el disco.

Los datos publicados por el NSRL son usados para identificar archivos rápidamente en sistemas informáticos, basándose únicamente en el contenido de los archivos.

En la mayoría de los casos, la base de datos del NSRL es usada para eliminar archivos conocidos, como los archivos del sistema operativo o de aplicaciones,

---

[25] National Software Reference Library. NIST, agency of the U.S. Commerce Department's Technology Administration. 2008. http://www.nsrl.nist.gov/

[26] National Institute of Standars and Technology

durante una investigación forense. Esto reduce el número de archivos que deben ser analizados manualmente, lo que ayuda a incrementar la eficiencia de la investigación.

- **ffident — Java metadata extraction / file format identification library**

Es una librería para Java que permite extraer información de los archivos e identificar su formato, ésta colecta información sobre los formatos más comunes y examina cada archivo contra una lista de conocidos. Usa la misma aproximación del comando file de los sistemas Unix[27].

- **FileAlyzer**

FileAlyzer es una herramienta que permite analizar los archivos con sus propiedades y su contenido en hexadecimal, capaz de interpretar contenido de archivos comunes a través de su estructura[28].

- **FileType**

Herramienta para reconocimiento de tipos de archivos que tiene su propio motor de detección de tipos y soporta amplia variedad de formatos[29]. Está basada en el comando file.

6.1.6 Medición de la Entropía en los archivos

En teoría de información, la entropía es la medida de la predictibilidad o aleatoriedad de los datos. Un archivo con una alta estructura predecible o un valor que se repite frecuentemente tendrán baja entropía. Dichos archivos serán

[27] SCHMIDT, Marco. Java metadata extraction / file format identification library. 2008. http://schmidt.devlib.org/ffident/index.html

[28] KOLLA, Patrick M. FileAlyzer. 2008. http://www.safer-networking.org/es/filealyzer/index.html

[29] Para más información de la herramienta diríjase a http://pldaniels.com/filetype/

considerados como de poca densidad información o contenido. Archivos donde el siguiente valor de byte es relativamente independiente del byte anterior se considera con entropía alta, lo cual puede significar un gran contenido de información.

Para determinar el tipo de información a visualizar se generó un programa en java basado en la propuesta presentada en el artículo "*Sliding Window Measurement for FileType Identification*", en donde se expone identificar el tipo de archivo a través de la medición de la Entropía, tomando cien ventanas de noventa bytes y aplicando la siguiente fórmula que da como resultado un valor de entropía para cada ventana.

$$H(M) = -\frac{1}{n}\sum_{c \in S_w} f(c)\lg f(c) + \lg n$$

donde $n = |S_w|$, S_w es el conjunto de símbolos diferentes en la ventana W, w es el tamaño de W *y* $f(c)$ es la frecuencia del carácter c en W.

Vale la pena aclarar que el estudio de la medición de la entropía que se llevó a cabo en este proyecto se hizo con el fin de proponer un método de identificación de formatos apoyado en el artículo, es decir, se buscó determinar si el cálculo de la entropía allí propuesto era una aproximación válida para el reconocimiento de formatos.

La propuesta se desarrolló de dos maneras, en la primera se usó la propuesta del artículo base (cien ventanas de noventa bytes), según los resultados de esta, se encontraron diferencias significativas con los hallados en el artículo, por lo tanto se decidió probar usando un tamaño de ventana más grande (quinientos doce bytes).

Se trabajó con la fórmula general de la entropía en lugar de la forma algebraica propuesta en el artículo, pues ésta nos generó problemas de redondeo en el

programa desarrollado en java, lo cual nos arrojaba resultados negativos para los valores de la entropía.

El programa genera un archivo csv[30] donde cada fila representa los 100 valores de entropía de un archivo. Se genera un archivo distinto para cada extensión identificada, utilizando la extensión para nombrarlo. Si no se identifica ninguna el archivo de salida es llamado entropía.csv.

Se hicieron pruebas del código con dos muestras diferentes, tomadas de los equipos de los autores, para trece formatos de archivos. En total se procesaron 61798 archivos, 32405 de la primera muestra y 29393 de la segunda. El número de archivos en la prueba de cada tipo de formato se muestra en la siguiente tabla.

Formato	**Número archivos Máquina 1**	**Número archivos Máquina 2**
.bmp	733	790
.dll	11380	13828
.doc	881	348
.exe	2878	3190
.html	7391	7038
.jpg	5359	3361
.mp3	634	2178
.pdf	1881	1790
.ppt	91	54
.rar	30	79
.txt	592	913

[30] Valores separados por comas (Comma Separated Values)

.xls	64	34
.zip	491	173

Tabla 2: Muestra de archivos para la medición de la entropía

Para cada uno de estos archivos se calculó la entropía en cien ventanas de quinientos doce bytes, estos valores quedaron consignados en trece archivos, uno por formato, de los cuales se realizaron gráficos para análisis. Para cada tipo de archivo las medidas fueron promediadas y se calcularon desviaciones estándar para cada punto correspondiente.

Se cuenta con un gráfico de uno de los archivos procesados del formato indicado (Entropía vs Número de ventanas), un gráfico para las medidas promediadas de cada muestra y un gráfico para las medidas promediadas de los valores consolidados de ambas muestras. Los trece archivos generados y sus respectivas gráficas se encuentran en los anexos de este proyecto.

Al comparar los resultados de estas pruebas con los del artículo anteriormente nombrado se encontró que estos en la mayoría de los casos eran significativamente diferentes lo que llevó a descartar la medida como confiable para la identificación de formatos en archivos.

Sin embargo, los valores y gráficas generados para formatos comprimidos, como son: mp3, zip y jpg evidencian una entropía alta lo cual permite que sean más fácilmente diferenciables usando el programa generado en el proyecto. Esto es perfectamente coherente con el hecho de que los procesos de compresión están basados en la teoría de la información y utilizan una codificación que maximice la entropía. En la prueba no se utilizaron archivos cifrados porque desde el punto de vista de la teoría de la información tienen las mismas características que los

archivos comprimidos. De hecho el cálculo de la entropía de los archivos es una técnica común en las aplicaciones forenses para identificar archivos cifrados.

Lo anterior puede verse a continuación:

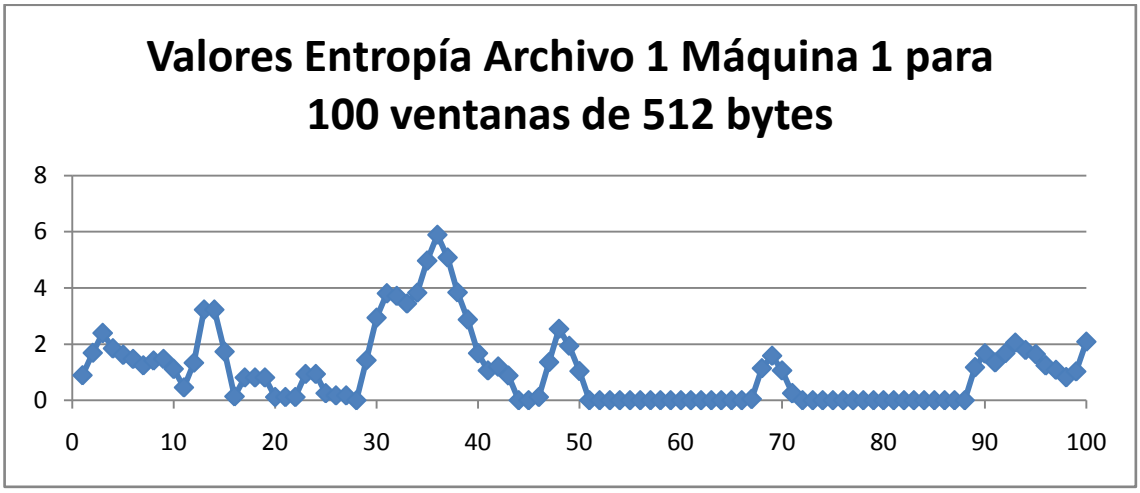

Ilustración 3: Valores de entropía para un archivo .doc

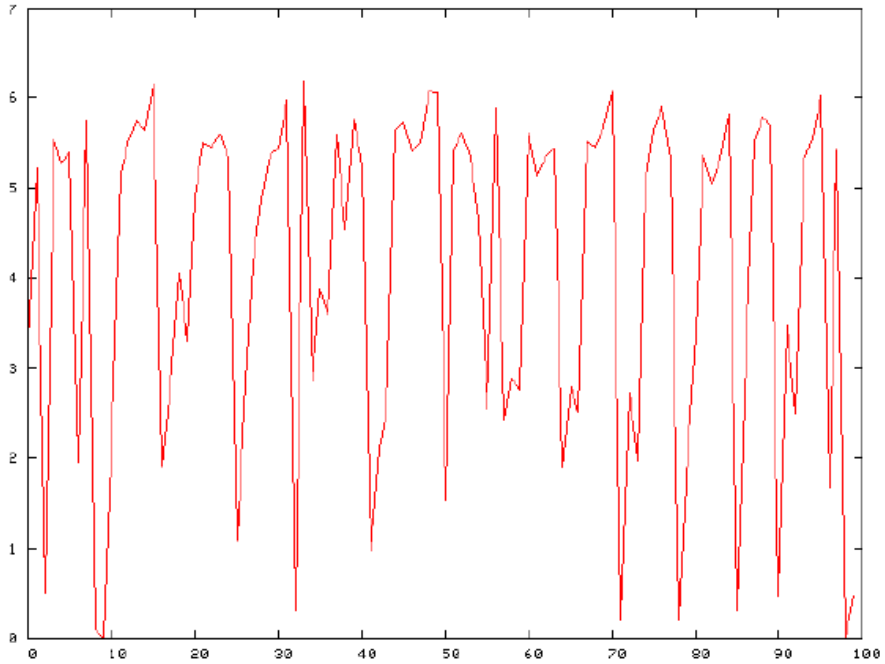

Ilustración 4: Valores de entropía para un archivo .doc. Tomada de artículo: "Sliding Window Measurement for File Type Identification"

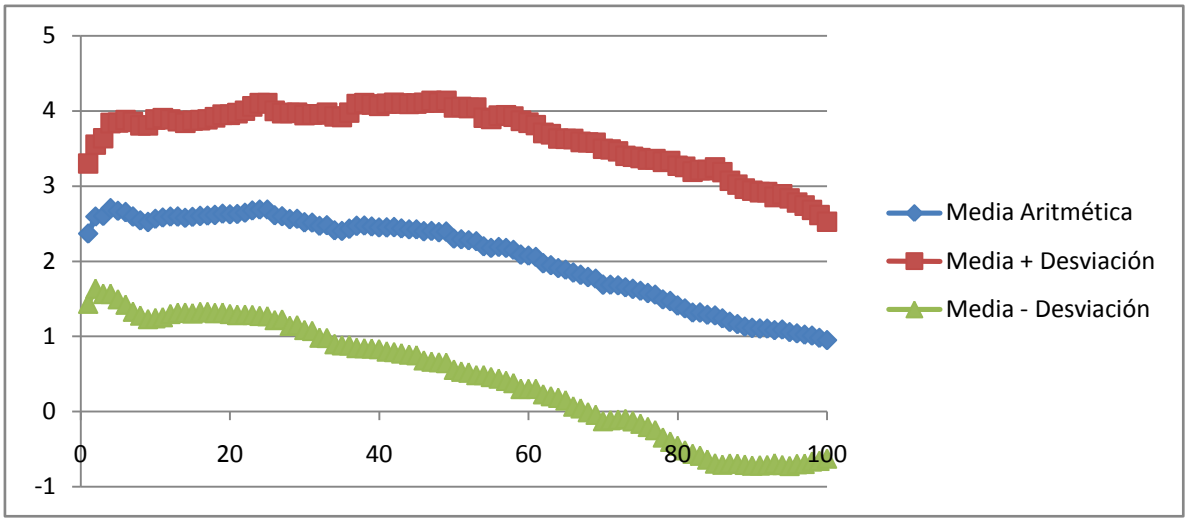

Ilustración 5: Gráfica de valores consolidados promediados para 1229 archivos .doc

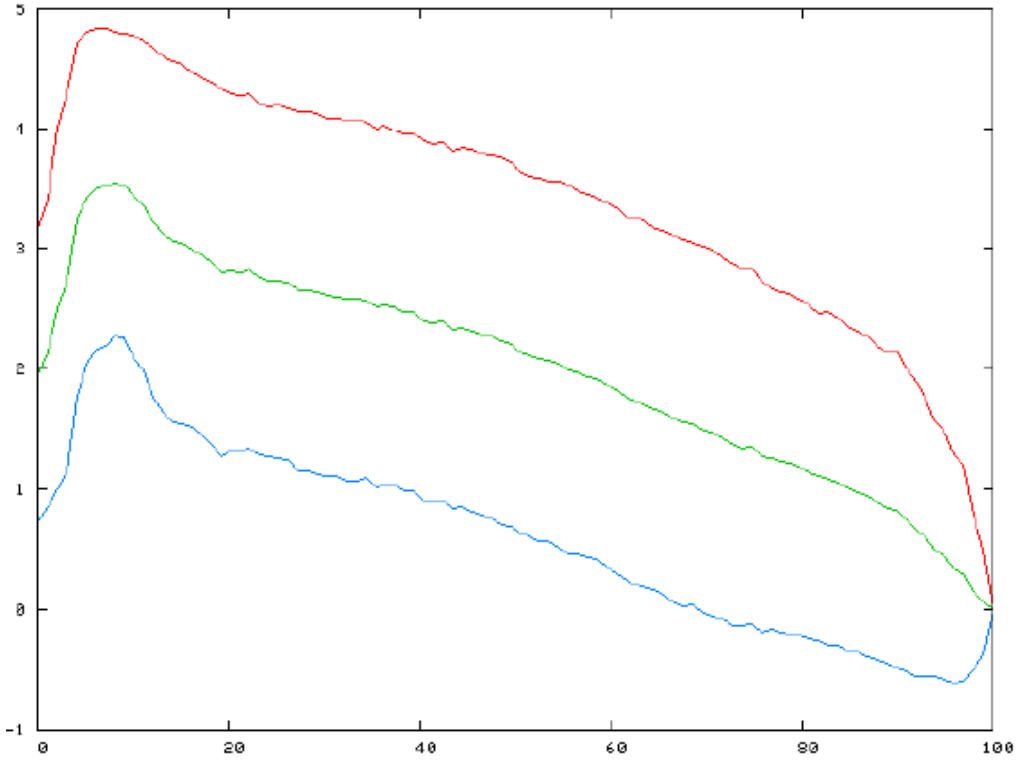

Ilustración 6: Gráfica de valores promediados para formato .doc. Tomada de artículo: "Sliding Window Measurement for File Type Identification"

Al tomar las gráficas generadas por el programa y las del artículo, para un archivo .doc, como se muestra en las ilustraciones 3, 4, 5 y 6 se puede observar que los valores de entropía varían significativamente, de tal forma que la gráfica no permite sugerir el formato al que pertenecen los archivos. Lo mismo sucede con los formatos xls, ppt y exe.

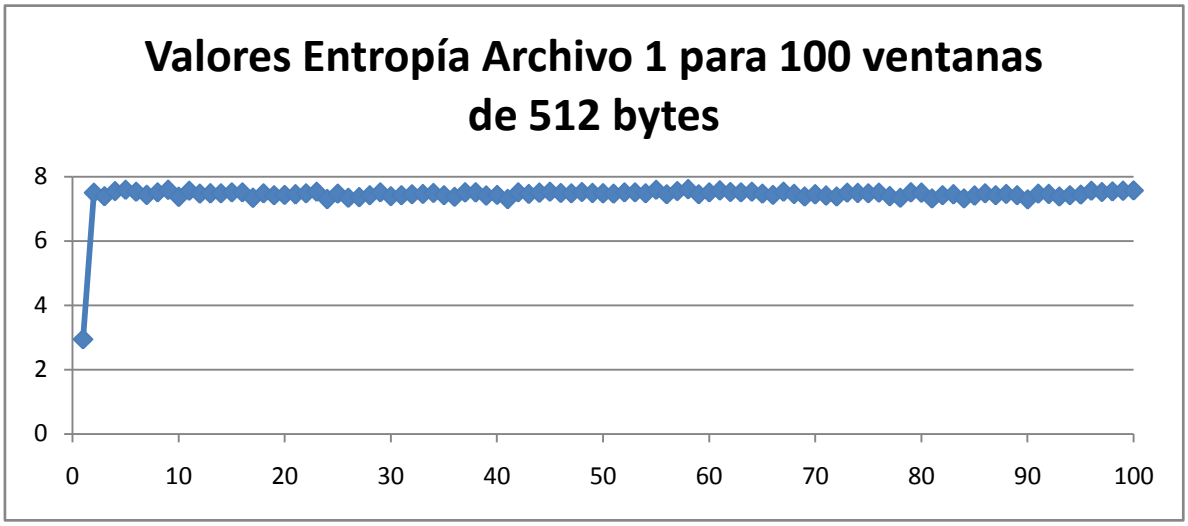

Ilustración 7: Valores de entropía para un archivo .mp3

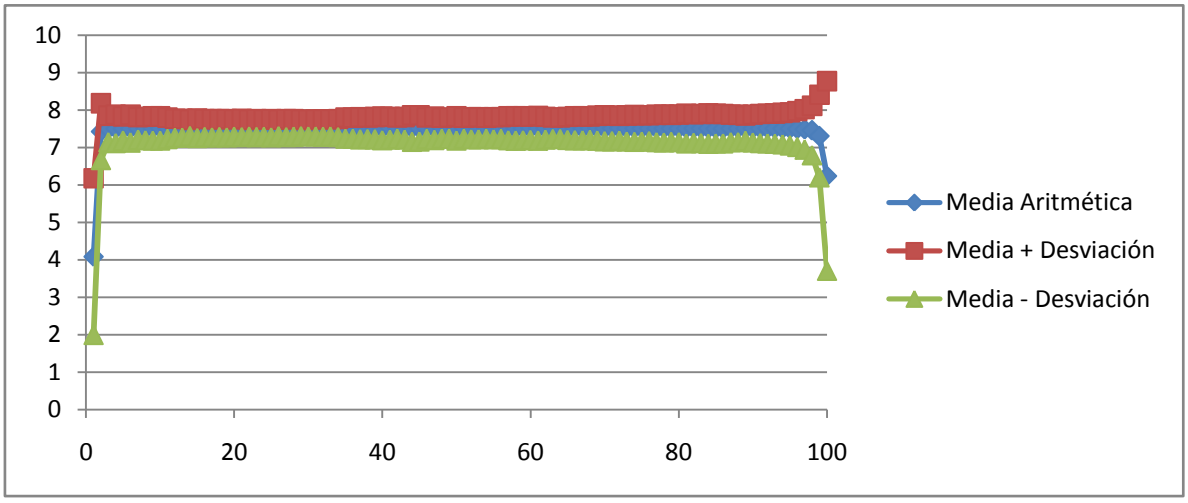

Ilustración 8: Gráfica de valores consolidados promediados para 2812 archivos .mp3

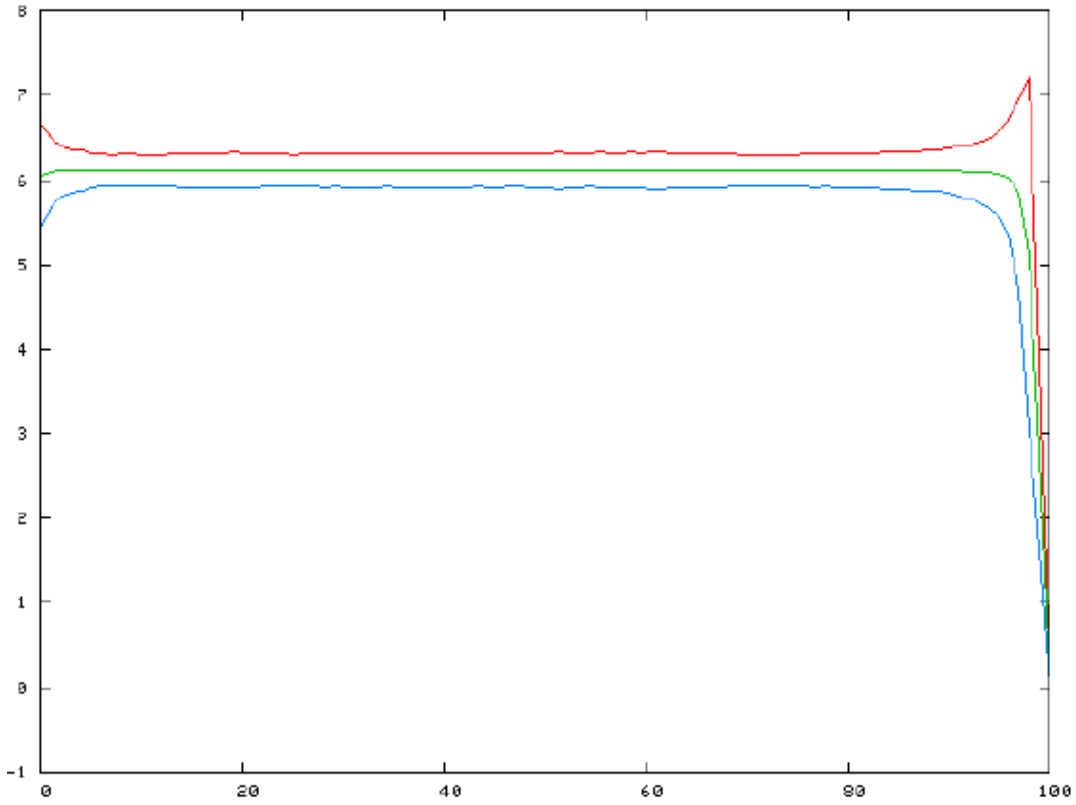

Ilustración 9: Gráfica de valores promediados para formato .mp3. Tomada de artículo: "Sliding Window Measurement for File Type Identification"

Como se dijo anteriormente, la medición de la entropía en un tipo de archivo comprimido como lo es el mp3, da valores muy grandes y las gráficas permiten sugerir que el archivo a analizar puede o no ser de este tipo, tal y como se ve en las ilustraciones 7, 8 y 9.

La entropía no da el tipo de confiabilidad que se desea, sin embargo es necesario realizar un trabajo futuro para determinar la forma como el tamaño del símbolo y el tamaño de la ventana afectan la medición. Se nota en algunos tipos de archivos, como se puede apreciar en las gráficas incluidas en los anexos, que hay regiones de menor entropía al comienzo y al final de archivo. Esto da pie para pensar que se puede tratar de identificar de manera automática las zonas que contienen

metadatos e información descriptiva, pero dicha exploración va más allá del alcance de este proyecto.

6.2 ETAPA 2: Formateado o reconstrucción del archivo

Para la visualización del archivo se tienen dos posibles escenarios:

- El formato del archivo pudo ser identificado o sugerido en la primera etapa del método.
- El archivo o fragmento de archivo aún continúa con un formato desconocido. Si éste último es el caso, tendría que aplicarse la segunda etapa del método, a ensayo y error, con los formatos de archivo más probables.

Si el formato del archivo pudo ser sugerido y el archivo está completo, éste puede ser accedido con una herramienta específica para su visualización.

Si el archivo está incompleto, éste tiene que ser reconstruido. Para ello, el primer paso es conocer su estructura interna, la cual se encuentra detallada en un documento de especificación de formato creado por el desarrollador del formato, en el que se describe exactamente cómo son codificados lo datos, y cuya accesibilidad depende de si el formato es o no propietario. En caso de ser un formato propietario, la reconstrucción se dificultaría, pues sería necesario hacer ingeniería inversa para aprender la manera en que trabaja el formato, o adquirir la especificación, pagando el valor determinado por sus desarrolladores.

Si se cuenta con dicha especificación, se puede proceder a construir la herramienta para formatear, la cual tomará el archivo o fragmento de archivo y lo estructurará en uno nuevo de acuerdo a los parámetros ingresados por el usuario.

Los formatos de archivo pueden ser vistos como un segmento de datos unido a una serie de tipos de datos primitivos en los que se especifican los metadatos sobre dicho segmento.

A continuación se expondrán los dos formateadores desarrollados. Se utiliza *bitmap* porque es el formato básico de las imágenes. En principio, cualquier formato puede ser convertido a *bitmap* mediante el uso de un decodificador adecuado; y lo mismo aplica para WAV, que es el formato básico de audio.

En la herramienta formateadora deberá definirse como serán determinados los valores de los encabezados, pues algunos de ellos deben ser ingresados por el usuario y otros pueden ser calculados a partir de dichas entradas y del propio fragmento de archivo. También es necesario especificar el byte inicial y el byte final del fragmento que se formateará, pues en algunos casos, el archivo a analizar podrá tener datos dañados o que no se desean acceder. Una vez completada la información, el formateador escribirá un nuevo archivo válido para esa especificación de formato.

6.2.1 Formateador BMP

Por ejemplo, en un archivo BMP, el formato se encuentra estructurado en datos de uno, dos o cuatro bytes, de la siguiente manera:

Encabezado del archivo BMP: Es útil para la identificación del archivo. El número mágico va en formato *big-endian*, mientras que los otros valores enteros se almacenan en formato *little-endian*.

Posición	Tamaño	Contenido

0	2	El número mágico usado para identificar el archive BMP: 0x42 0x4D (son los caracteres ASCII equivalentes a la **B** y la **M**)
2	4	Tamaño del archivo completo en bytes.
6	4	Valor reservado para uso posterior, depende de la aplicación que creó la imagen.
10	4	Offset (dirección inicial), byte desde el cual se encuentran los datos del bitmap.

Tabla 3: Encabezado del archivo BMP

Información del Bitmap: En este bloque se tiene información detallada sobre la imagen, y es útil para mostrar la imagen en pantalla.

Posición	Tamaño	Contenido
14	4	Longitud del Encabezado. Se tomará el encabezado Windows V3, cuya longitud es 40 bytes.
18	4	Longitud horizontal del bitmap, en pixeles (entero con signo).
22	4	Longitud vertical del bitmap, en pixeles (entero con signo).
26	2	Número de planos en el bitmap. Debe ser 1.
28	2	Número de bits por pixel, es la profundidad de color de la imagen. Los posibles valores son: 1 – monocromático 4 – 16 colores 8 – 256 colores 16 - 16bit (color de alta densidad) 24 - 24bit (color verdadero) 32 - 32bit (color verdadero)
30	4	Método de compresión utilizado. Los valores posibles son: 0 - none (BI_RGB)

		1 - RLE 8-bit / pixel (BI_RLE4) 2 - RLE 4-bit / pixel (BI_RLE8) 3 - Bitfields (BI_BITFIELDS) 4 – JPEG (BI_JPEG) 5 – PNG (BI_PNG)
34	4	Tamaño de la imagen (bitmap data).
38	4	Resolución horizontal de la imagen, expresada en pixeles por metro (entero con signo).
42	4	Resolución vertical de la imagen, expresada en pixeles por metro (entero con signo).
46	4	Número de colores usados en el bitmap, (0 - 2^n , n = número de bits por pixel)
50	4	Número de colores importantes usados, 0 cuando cada color es importante. Es generalmente ignorado.

Tabla 4: Información del Bitmap

Datos del bitmap: Este bloque de bytes describe la imagen pixel por pixel, desde la esquina inferior izquierda, fila por fila, hasta la esquina superior derecha de la imagen.

Para reconstruir los encabezados anteriores, el formateador recibe como parámetros:

- El nombre del archivo o fragmento de archivo al que se le quiere dar formato de imagen.
- El nombre del archivo de salida en el que se genera el archivo formateado como BMP.
- El primer valor para la relación de aspecto.
- El segundo valor para la relación de aspecto.

- El valor para la profundidad de color, que según la especificación del formato puede ser: 1, 4, 8, 16, 24 o 32

- El valor para el método de compresión. Según la especificación del formato, puede ser: 0, 1, 2, 3, 4 ó 5.

- Las posiciones inicial final, en bytes, del fragmento de archivo. Esto será útil en caso de que se deseen ignorar algunos bytes del archivo, o que este deba ser visualizado por partes.

Estos parámetros servirán para calcular los demás metadatos del encabezado. Los valores de la relación de aspecto son útiles para calcular el ancho y el alto de la imagen. Para ello, es necesario hallar un factor, así:

$$factor = \sqrt{\frac{\frac{tamaño\ del\ fragmento}{profundidad\ de\ color}}{8}}{valRelacion1 * valRelacion2}$$

$$ancho = valRelacion1 * factor$$

$$alto = valRelacion2 * factor$$

Para el valor del tamaño de la imagen, que es diferente al tamaño del archivo, se toma el tamaño del fragmento de archivo a formatear, pues es a éste al que se le agregarán los encabezados y a partir del cual se originará el archivo con formato BMP.

La resolución horizontal y vertical no es utilizada por la mayoría de visualizadores, por tanto tomará por defecto el valor cero. De igual manera, se tomará este valor para los metadatos correspondientes al número de colores usados en el bitmap y al número de colores importantes, pues son generalmente ignorados.

Finalmente, los valores de los encabezados serán escritos en el archivo de salida, construyendo finalmente un archivo BMP correcto.

6.2.2 Formateador WAV

El archivo WAV se encuentra estructurado de la siguiente manera:

Encabezado RIFF: Contiene el número mágico del archivo. Es de 12 bytes de longitud.

Posición	Tamaño	Contenido
0	4	"RIFF" (Caracteres ASCII)
4	4	La longitud del archivo desde este número. (Binary, little endian)
8	4	"WAVE" (Caracteres ASCII)

Tabla 5: Encabezado RIFF del archivo WAV

Encabezado Format: Es el encabezado que contiene los metadatos del archivo, en donde se especifica el formato del audio. Es de 24 bytes de longitud.

Posición	Tamaño	Contenido
12	4	"fmt_" (Caracteres ASCII)
16	4	Longitud del encabezado format (siempre vale 24)
20	2	Siempre es 1
22	2	Número de canales. Mono=1, Estéreo=2
24	4	Tasa de muestreo, en Hertzios
28	4	Bytes por segundo.
32	2	Bytes por muestra. 1=8 bit Mono 2=8 bit Stereo

		2=16 bit Mono 4=16 bit Stereo
34	2	Bits por muestra

Tabla 6: Encabezado format del archivo WAV

Encabezado de datos: Contiene los datos de audio. Su tamaño varía dependiendo del archivo.

Posición	Tamaño	Contenido
36	4	"data" (Caracteres ASCII)
40	4	La longitud de los datos que siguen.
44	Depende del archivo	Datos (las muestras de audio).

Tabla 7: Encabezado de datos del archivo WAV

Para reconstruir los encabezados anteriores, el formateador recibe como parámetros:

- El nombre del archivo o fragmento de archivo al que se le quiere dar formato de audio.
- El nombre del archivo de salida en el que se genera el archivo formateado como WAV.
- El número de canales, puede valer 1 ó 2.

- El valor para la tasa de muestreo (en Hz).
- El número de bits por muestra, puede ser 8, 16, 24 o 32.
- Las posiciones inicial final, en bytes, del fragmento de archivo. Esto será útil en caso de que se deseen ignorar algunos bytes del archivo, o que este deba ser visualizado por partes.

A partir de estos parámetros pueden calcularse todos los metadatos necesarios. La tasa de bytes se calcula a partir de los valores de la tasa de muestreo, el número de canales y los bits por muestra, así:

$$tasa\ de\ bytes = \frac{tasa\ de\ muestreo * número\ de\ canales * bits\ por\ muestra}{8}$$

Los bytes por muestra se calculan así:

$$bytes\ por\ muestra = \frac{número\ de\ canales * \ bits\ por\ muestra}{8}$$

Con todos los metadatos definidos, finalmente podrán reconstruirse los encabezados de un archivo con formato WAV.

6.3 ETAPA 3: Visualización

Una vez el archivo ha sido reconstruido, puede ser visualizado haciendo uso de los programas disponibles para tal fin, y será el usuario quien finalmente determine si la información mostrada es válida y tiene sentido. En caso de que el resultado no sea coherente, el usuario deberá modificar los parámetros recibidos por el formateador hasta obtener una visualización válida, o en caso contrario, determinar que el formato sugerido no fue correcto o que el archivo no pudo ser reconstruido.

7. EJEMPLO DE APLICACIÓN DEL MÉTODO

Partimos de un archivo o fragmento de archivo digital del cual se desconoce su formato, pero que puede ser visualizado por medio de un editor hexadecimal.

CASO 1

7.1 ETAPA 1: Identificación del formato del archivo

7.1.1 Identificar la extensión del archivo

El archivo no tiene extensión, no se puede sugerir ningún tipo de formato a partir de este ítem.

7.1.2 Determinar si el archivo es texto

Al acceder el archivo con un editor hexadecimal, se determinó que no era texto, pues no se visualizaron cadenas de caracteres que formaran palabras o estructuras lingüísticas válidas.

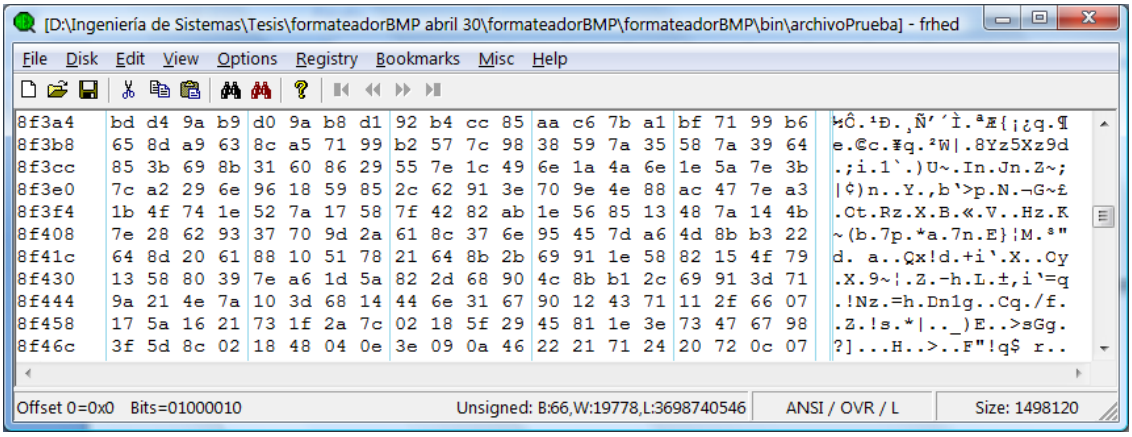

Ilustración 10: Visualización del archivo en un editor hexadecimal

En el marco derecho del editor hexadecimal utilizado, se muestran en ASCII los caracteres imprimibles del archivo, los no imprimibles se muestran como un punto.

Para verificar que el contenido de la imagen no era texto también se utilizó el comando strings, con el cual se obtuvo como salida el mismo resultado logrado en el editor hexadecimal, que permitió concluir que el archivo no es de tipo texto.

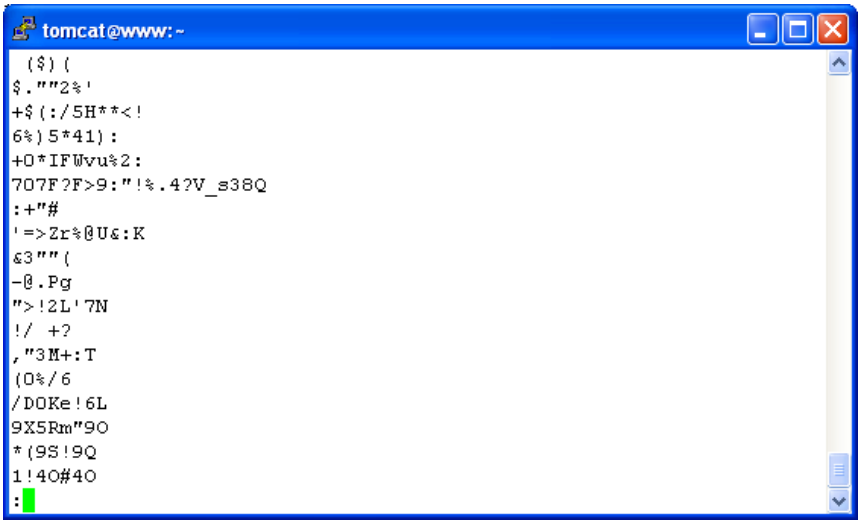

Ilustración 11: Uso del comando strings en archivoPrueba

7.1.3 Ejecución del comando file

Al ejecutar el comando file con este fragmento de archivo, se obtuvo el siguiente resultado:

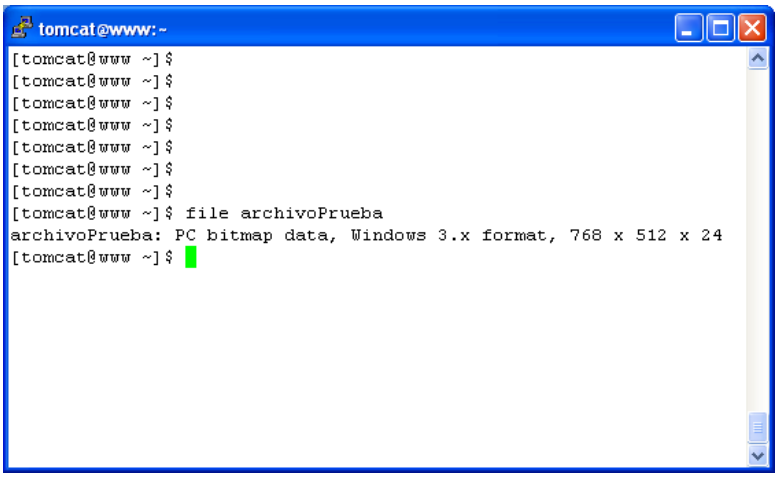

Ilustración 12: Uso del comando file sobre el archivo de prueba

El comando file puede lograr, en la mayoría de los casos, sugerir el tipo de archivo con el que se está trabajando. Si el tipo es desconocido, el comando file lo identificaría como “data”. En el caso de prueba, el comando file pudo identificar al archivo como mapa de bits, mostrando también el alto, el ancho y la profundidad de color de la imagen, información de la que no se tiene ciento por ciento de certeza debido a la incompletitud del archivo.

7.1.4 Identificar el número mágico

Para ver el número mágico del archivo deben visualizarse los primeros bytes del mismo en un editor hexadecimal.

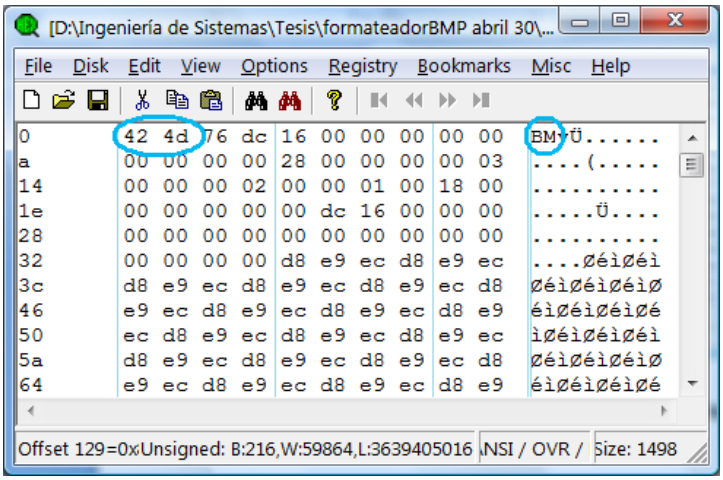

Ilustración 13: Visualización de los primeros bytes del archivo

En el ejemplo de aplicación del método se encontraron los números 42H y 4dH equivalentes a los caracteres BM, en ASCII. Al buscar en una tabla de números mágicos se obtuvo el siguiente resultado:

Firma hexadecimal: `42 4D`

Descripción ASCII de la firma: `BM`

Extensión del archivo: BMP - Windows Bitmap Image

Esto permite sugerir que el archivo defectuoso puede ser un mapa de bits.

Si el número mágico hubiese estado defectuoso, el comando file no habría podido hacer la identificación de la imagen, tal y como se muestra a continuación, en donde se modificó el segundo byte del archivo.

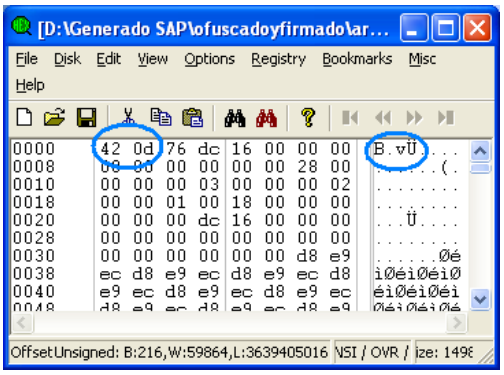

Ilustración 14: Modificación del número mágico en el archivo de prueba

Luego, al ejecutar el comando file nuevamente, la salida fue “data”, pues el formato no pudo ser identificado.

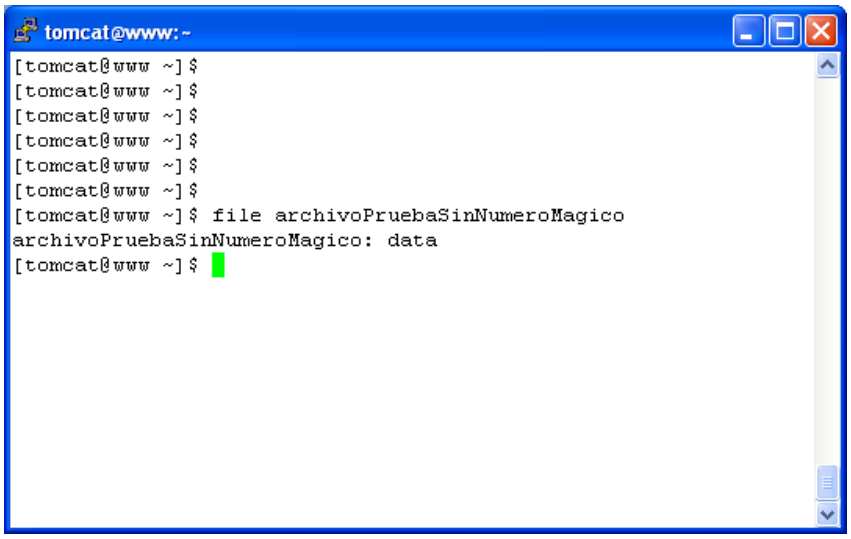

Ilustración 15: Salida del comando file para el archivo con el número mágico alterado.

7.1.5 Medición de la entropía en el archivo de prueba

Para medir la entropía para cien ventanas de quinientos doce bytes, en el archivo de prueba, se ejecuta el siguiente comando que llama al programa de entropía realizado para el proyecto.

java ProcesadorArchivos archivoPrueba

donde ProcesadorArchivos es el nombre del programa a ejecutar y archivo prueba es el nombre del archivo a procesar. Se debe tener en cuenta que si no se está en la ruta donde se encuentran los archivos ProcesadorArchivos.class y archivoPrueba se deben cambiar los anteriores por su ruta completa.

El resultado del programa es un archivo de extensión csv que contiene cien valores de entropía, los valores se muestran en la ilustración 16.

Archivo de Prueba	7.052595752	5.97799446	5.696918825
2.060697278	6.962991903	7.116235643	5.857804475
6.83793229	7.270516184	5.762758604	5.191566644
6.911124051	6.8405592	7.036774737	6.153905388
6.708729114	6.094654898	7.133367924	6.690175455
6.318991877	6.247884578	6.736653691	6.624218269
6.091019512	6.626412951	7.315756007	6.061600941
5.955798715	7.008303975	6.485456442	5.480741736
5.739348711	6.716503287	6.038362393	6.080740931
5.895103222	7.168023851	6.155444713	5.982862157
6.720640248	7.356643375	6.079389744	5.412111975
6.97481727	6.663408958	6.923543665	
6.945797048	6.175592847	7.07649675	
6.373825677	5.795258242	6.890894651	
7.046283879	6.439658931	7.053827446	
6.931507868	7.123154535	6.244870468	
6.17422281	7.199286157	5.358529589	
5.887956142	6.957201284	5.728595606	
6.634930353	7.239475053	5.987479318	
7.032026817	7.221663694	5.709605923	
6.861940495	6.42673406	5.640057764	
7.037592969	5.805333276	6.406925891	
6.943274304	5.946412411	6.719778851	
7.157669804	6.769896011	6.523590148	
7.098143443	7.057782635	5.003727727	
6.641132599	7.20677051	5.360817534	
6.408164504	7.003755793	6.223152152	
6.953109738	6.646067295	6.34438982	
6.655878863	6.957510438	5.208336762	
6.838966796	5.928861686	4.995826478	

Ilustración 16: Valores entropia para archivoPrueba

Con los valores del archivo generado por el programa, se hace un gráfico de los valores de entropía contra el número de ventanas usadas, en este caso cien. El gráfico se muestra en la ilustración 17.

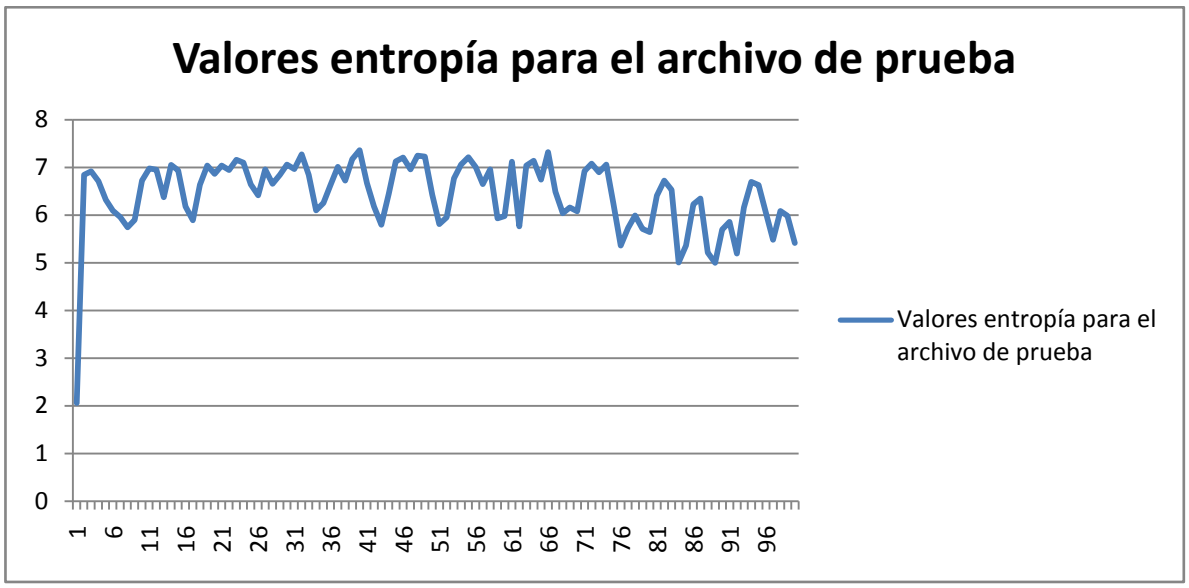

Ilustración 17: Gráfica valores entropía para el archivo de prueba

Se recuerda que al presentar el programa y el análisis de los datos arrojados, se llegó a la conclusión de que la medida de la entropía aunque se usa para reconocer el tipo de archivo, no es una medida acertada y confiable para hacerlo, por tal motivo la aplicación de esta no tiene más trascendencia que la que se acaba de mostrar.

7.2 ETAPA 2: Formateado del archivo (reconstrucción)

En este punto, se tiene que el fragmento de archivo puede ser una imagen BMP. Para poder llevar a cabo la visualización, el archivo debe ser reconstruido, para lo cual se hará uso del formateador BMP.

El usuario deberá ingresar los parámetros especificados para el formateador, al momento de ejecutar el programa. Para resultados más eficientes, se sugiere crear un archivo .bat en el que se hagan pruebas con varias combinaciones de parámetros.

Para nuestro problema, se tiene un archivo .bat, con las siguientes líneas, dependiendo del tipo de parámetros:

java byteToBMP/FormateadorBMP archivoPrueba bmp0 16 9 24 0 500 4000000

java byteToBMP/FormateadorBMP archivoPrueba bmp1 4 3 24 0 0 1000000

java byteToBMP/FormateadorBMP archivoPrueba bmp2 1 1 24 0

java byteToBMP/FormateadorBMP archivoPrueba bmp3 4 3 16 0

java byteToBMP/FormateadorBMP archivoPrueba bmp4 4 3 32 0

java byteToBMP/FormateadorBMP archivoPrueba bmp5 4 3 24 0

7.3 ETAPA 3: Visualización

En la etapa de visualización, el usuario debe abrir los archivos generados con un visualizador de imágenes común, para luego determinar si se obtuvo un resultado válido. De acuerdo al resultado visualizado, el usuario puede continuar modificando los parámetros hasta obtener una imagen coherente.

Para el ejemplo de aplicación del método, se obtuvieron las siguientes imágenes:

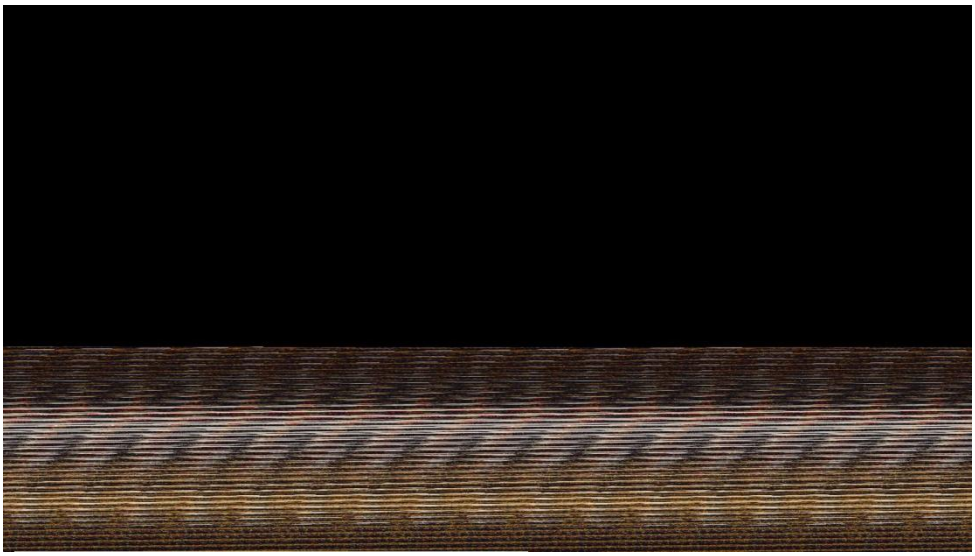

Ilustración 18: Imagen bmp0.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- Nombre Archivo de entrada:archivoPrueba
- Nombre Archivo de Salida: bmp0
- Relación de aspecto:Valor 1:16, valor 2:9.
- Profundidad de color: 24
- Método de compresión:0
- Posición inicial y final del fragmento del archivo: 500 - 4000000

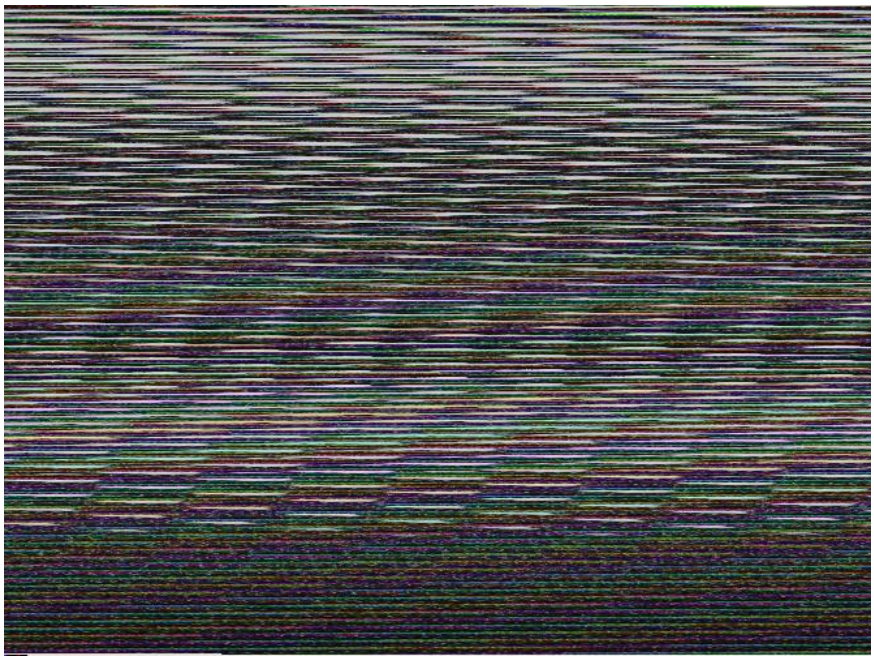

Ilustración 19: Imagen bmp1.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- *Nombre Archivo de entrada:archivoPrueba*
- *Nombre Archivo de Salida: bmp1*
- *Relación de aspecto:Valor 1:4, valor 2:3*
- *Profundidad de color: 24*
- *Método de compresión:0*
- *Posición inicial y final del fragmento del archivo: 0 - 1000000*

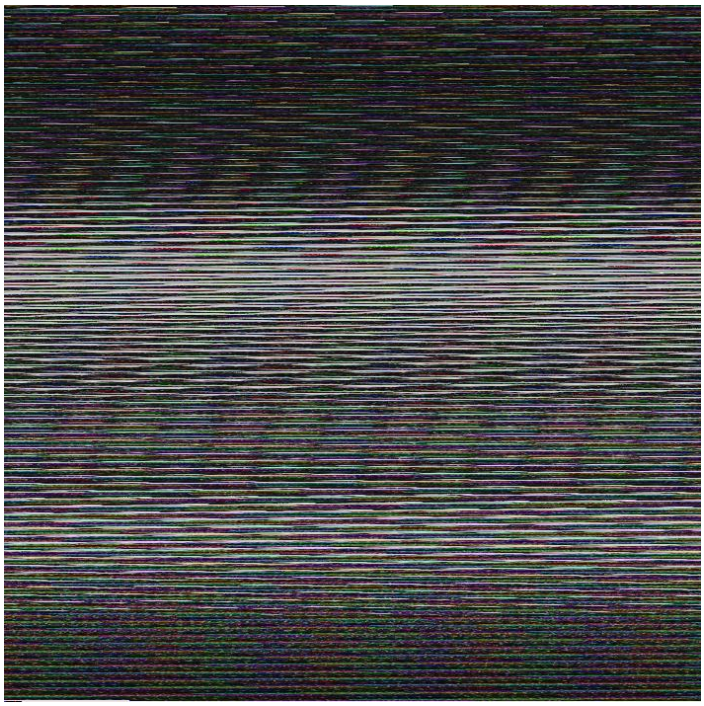

Ilustración 20: Imagen bmp2.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- *Nombre Archivo de entrada:archivoPrueba*
- *Nombre Archivo de Salida: bmp2*
- *Relación de aspecto:Valor 1:4, valor 2:1*
- *Profundidad de color: 24*
- *Método de compresión:0*
- *Posición inicial y final del fragmento del archivo: Se omitieron, por tanto, se tomará todo el archivo.*

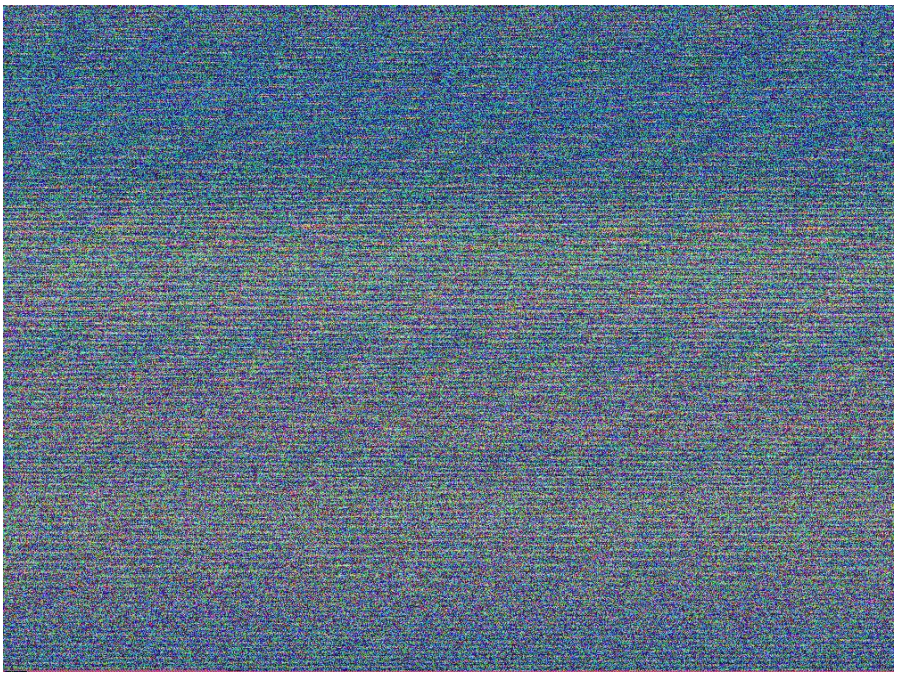

Ilustración 21: Imagen bmp3.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- *Nombre Archivo de entrada:archivoPrueba*
- *Nombre Archivo de Salida: bmp3*
- *Relación de aspecto:Valor 1:4, valor 1:3*
- *Profundidad de color: 16*
- *Método de compresión:0*
- *Posición inicial y final del fragmento del archivo: Se omitieron, por tanto, se tomará todo el archivo.*

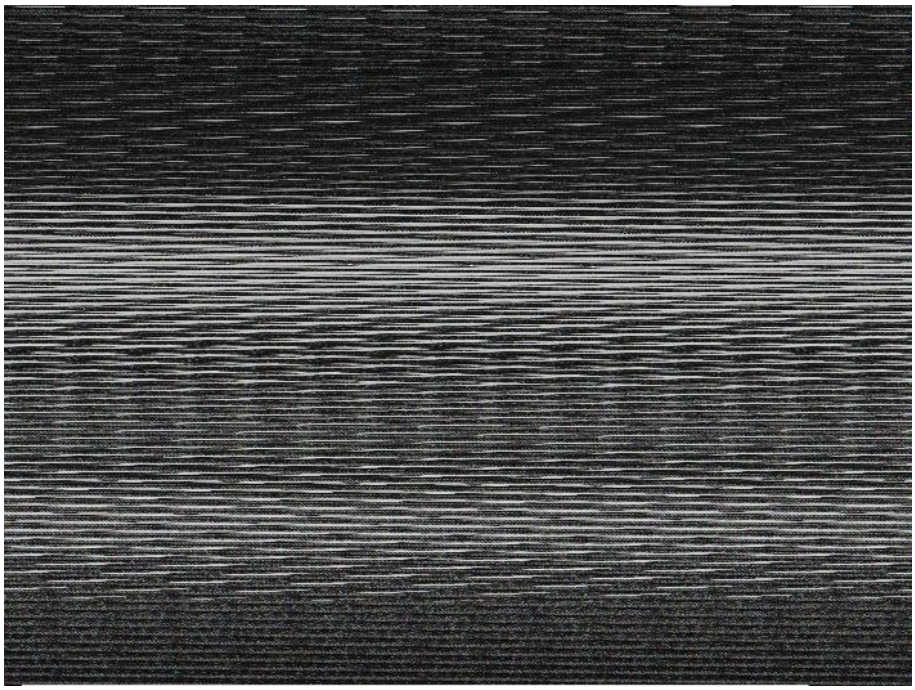

Ilustración 22: Imagen bmp4.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- *Nombre Archivo de entrada:archivoPrueba*
- *Nombre Archivo de Salida: bmp4*
- *Relación de aspecto:Valor 1:4, valor 1:3*
- *Profundidad de color: 32*
- *Método de compresión:0*
- *Posición inicial y final del fragmento del archivo: Se omitieron, por tanto, se tomará todo el archivo.*

Ilustración 23: Imagen bmp5.bmp

Esta imagen corresponde a ejecutar el formateador con los siguientes parámetros:

- *Nombre Archivo de entrada:archivoPrueba*
- *Nombre Archivo de Salida: bmp5*
- *Relación de aspecto:Valor 1:4, valor 1:3*
- *Profundidad de color: 24*
- *Método de compresión:0*
- *Posición inicial y final del fragmento del archivo: Se omitieron, por tanto, se tomará todo el archivo.*

Finalmente, de podrá determinar que el fragmento de archivo si era un mapa de bits, y que la reconstrucción a partir de la siguiente ejecución del programa: "*java byteToBMP/FormateadorBMP archivoPrueba bmp5 4 3 24 0*", y por medio de la cual se generó la imagen bmp5.bmp fue correcta, concluyendo el método de visualización exitosamente.