Análisis del potencial de combinar esteganografía con criptografía para comunicación más segura

Daniel Montes De Oca B54621
Universidad de Costa Rica - ECCI
CI-1320 Redes de Computadoras
San José, Costa Rica
daniel.montesdeocab@gmail.com

Fernando Rojas B56219
Universidad de Costa Rica - ECCI
CI-1320 Redes de Computadoras
San José, Costa Rica
ferojasmel@hotmail.com

***Resumen*—Se propone un problema abierto y una implementación del mismo que integra esteganografía y criptografía. Se toman en cuenta métricas de esteganografía para intentar llegar a una solución efectiva al problema planteado.**

***Index Terms*—Esteganografía, criptografía, portador, canal encubierto, canal subliminal, canal legítimo**

I. Introducción

A través de la historia ha existido la necesidad de transmitir información oculta. Esto se ha logrado de diferentes maneras, una de ellas es el uso de un medio que no hace parecer que guarda la información que realmente contiene, a esta práctica se le llama esteganografía.

Se podría decir que la esteganografía busca ocultar secretos a simple vista. Su nombre proviene del griego *steganos* que significa oculto, y *graphos* que significa escritura [3]. En la antigüedad era común que la gente ocultara mensajes secretos mediante tatuajes, tinta invisible, o patrones en texto. Sin embargo, la llegada de la era de la computación le ha permitido convertirse en una de las técnicas de seguridad más relevantes de la actualidad. Esto ocurre porque se han desarrollado técnicas que logran esconder el mensaje de manera que hacen difícil saber que existe información encubierta.

Por otro lado, existe la criptografía, que permite enmascarar un mensaje. A veces, la criptografía o la esteganografía, no son suficiente para tener comunicación segura. Sin embargo, existe un gran potencial en combinar ambas técnicas para lograr la seguridad deseada [3].

En este artículo, se busca exponer el potencial de combinar ambas técnicas. De la misma manera, se propone una alternativa de implementación a una idea de comunicación que se beneficia de las utilidades. Se tiene el objetivo de resolver el problema de combinar ambas técnicas en una solución que tome en cuenta métricas de efectividad de las disciplinas. La meta es que se logre conocer si es útil combinar esteganografía y criptografía para reducir la detectabilidad en la transmisión de un mensaje.

Inicialmente, se provee una descripción teórica sobre las técnicas. Después, se expone el trabajo relacionado al tema. Posteriormente, se plantea el problema abierto, así como una posible implementación para resolverlo. Por último, se referencian los trabajos que nos guiaron en la producción de este documento.

II. Descripción teórica

II-A. Definiendo esteganografía

A la esteganografía se le puede definir como la ciencia o el procedimiento de incluir mensajes o información secretos dentro de otros datos, a los que se les llama portadores o *carriers* [1]. Su objetivo principal es la comunicación confidencial e incógnita entre un emisor y receptor (los cuales pueden cambiar de roles). El acto de transmisión del mensaje debe pasar inadvertido por quienes puedan accesar el medio por el que se transmite, basándose fuertemente en el concepto de seguridad por oscuridad: si nadie sabe que hay un mensaje escondido, nadie lo va a intentar obtener [1].

II-B. Ejemplos de portadores

La práctica de la esteganografía se ha realizado usando diferentes portadores o *carriers* [5]. Dentro de esos medios se pueden destacar:

- Texto: En este caso el mensaje se esconde entre patrones de letras o palabras. Se dice que este medio es vulnerable, ya que un atacante puede encontrar los patrones que se están usando para camuflar la información [5].
- Imágenes: Se busca disfrazar la información usando una imagen como medio. En tiempos modernos la aplicación de esta técnica se ha concentrado en esconder el mensaje entre los píxeles de la imagen [5].
- Audio: Se tiene como objetivo camuflar el mensaje en un archivo de audio. Se dice que es difícil esconder la información, debido a la alta sensibilidad de el sistema auditivo humano [5].

II-C. Aplicando esteganografía

La aplicación de la esteganografía se hace posible gracias a lo que se le llama *canales encubiertos* [1]. Los canales encubiertos son aquellos que no son destinados para que por ellos se envíe información, pero que de alguna forma se pueden utilizar para comunicarse. Al contrario, los *canales legítimos* son destinados y diseñados para el paso de información conforme a su protocolo indique [1]. En la esteganografía el emisor manipula el mensaje para poder transmitirlo mientras que el receptor hace la operación inversa para volver a construir el mensaje. El sustento de la esteganografía se basa en el uso de

un llamado *canal subliminal* que utiliza la existencia de un canal encubierto para disfrazar el mensaje y que cuando fuese visto por un tercero (alguien que no es ni emisor ni receptor) simule comunicación real a través de un canal legítimo [1].

Cuando se aplica la esteganografía es necesario considerar cuatro aspectos fundamentales para medir su efectividad [2]:

- El primero es la **capacidad de incrustación** del medio, es decir, cuanta información podemos incluir o incrustar en el portador sin afectar su calidad original.
- El segundo es que tan **indetectable** es la información que se va a transmitir, si su existencia es aparente, la esteganografía falla.
- Como tercer aspecto está la **robustez** del algoritmo de incrustación, que depende de si la información sigue retenida en el medio de forma recuperable inclusive después de haber sido comprimido y descomprimido.
- **La resistencia al "manoseo"**, es decir, si alguien detecta la presencia de un mensaje, este debe ser difícil de eliminar o editar para el atacante.

II-D. Diferenciando esteganografía y criptografía

Al igual que la esteganografía, la criptografía es un proceso por el cual se esconde información de personas para los que no está dirigida. A pesar de tener el mismo propósito, es claro que no son lo mismo ya que la esteganografía camufla un mensaje dentro de otro medio, sin verdaderamente esconder el mensaje para quien lo encuentre. Por otro lado, la criptografía convierte su mensaje en un cripto-mensaje, con una llave secreta, y solo puede ser descifrado teniendo en posesión esa llave secreta [2].

De estas diferencias parten otras en distintos aspectos que nos permiten analizar los dos procesos y ver como varían según la situación. Por ejemplo, la seguridad de los datos en la criptografía depende meramente de que tan segura sea la llave, en la esteganografía depende de algo más circunstancial como qué tanto pueda notar alguna persona no autorizada el mensaje incrustado. También es importante tener en cuenta que en cuanto los ataques la criptografía es más propensa a sufrirlos ya que la aparición de información encriptada puede provocar curiosidad o le puede indicar a alguien que esa es la información valiosa que busca. En la criptografía esto no pasa ya que el paso de información secreta ni siquiera es detectado [2]. A continuación se detallan las fuentes que nos permitieron llegar a un problema abierto que relacionara ambos temas.

III. Trabajo Relacionado

Para la elaboración de este trabajo, se estudiaron distintos documentos relacionados con el tema de esteganografía y otros enfocados en la combinación de esta técnica con la criptografía. Existe mucha información acerca del tema e investigando en publicaciones y tesis se puede encontrar mucha información valiosa. A continuación se explica el proceso de aproximación a un problema abierto, en base a las lecturas analizadas.

Para empezar, la lectura de Deymonnaz [1], aporta información sobre investigaciones que mencionan el cifrado del mensaje con esteganografía como una posibilidad de mejora a la comunicación.

Además, Johri [2] detalla distintas aplicaciones esteganográficas como en audio, vídeo y redes. Brinda información sobre como medir o analizar la efectividad de el proceso de esteganografía. Como problema a futuro propone analizar y profundizar en la esteganografía híbrida, lo cual confirmó el potencial en el campo de la esteganofrafía.

Por otra parte, Baby [3] explica apectos como el origen del término esteganografía, e información sobre criptografía, así como una confirmación de que estas prácticas se pueden combinar a futuro. Es el el artículo que más problemas propone sobre el tema del documento. Algunas de las posibles aplicaciones que expone citando a otros documentos son:

- Ocultar mensajes en imágenes usando redes neuronales de forma encriptada, de forma que sea casi indetectable y si es detectada, que no se pueda descifrar si no se posea la llave.
- Ocultar imágenes secretas dentro de otras imágenes en redes inalámbricas, la imagen secreta es dividida en cuadrículas que son desordenadas usando criptografía para luego ser desencriptada.
- Ocultar mensajes relacionados a ambientes del Internet de las Cosas, en donde se encripta y se incrusta información valiosa en otros archivos (pone de ejemplo imágenes) para enviarse de cliente a servidor hogar y del servidor hogar al servidor en la nube. Los servidores se encargan de comparar los datos recién llegados con datos anteriormente recibidos para comprobar que se mantuvo su integridad.
- Ocultar mensajes cifrándolos con el algoritmo RSA para posteriormente utilizar el mensaje encriptado en audio utilizando la técnica denominada LSB.
- Ocultar mensajes encriptados mediante el cifrado de Vernam en imágenes usando esteganografía.
- Ocultar un mensaje mapeando cada caracter a un numero, que genera una imagen, que luego es ocultada en otra imagen mediante esteganografía.
- Cifrar imágenes mediante el algoritmo de Blowfish e incrustarlas esteganográficamente en otras imágenes portadoras.

Varias otras alternativas son mencionadas.

Un ejemplo es Challita [4], quién expone la posibilidad de combinar criptografía con esteganografía, así como un problema no resuelto de enviar un vector encriptado que contiene la información de donde encontrar el mensaje en una imagen, combinando criptografía y esteganografía.

Similarmente, Song provee información introductoria a esteganografía, así como una explicación de su uso en diferentes medios. También menciona un posible problema abierto sobre la creación de un protocolo de comunicación avanzado que implemente esteganografía y criptografía. Esto ayudó a comprobar la idea de la combinación de estas disciplinas.

Por úlitmo, **LSB based image steganography using**

dynamic key cryptography [6] También propone soluciones generales al problema de combinar esteganografía con criptografía, y propone utilizar criptografía utilizando semillas y esteganografía normal para transportar mensajes a través de imágenes.

IV. Problema abierto

La importancia de la confidencialidad de los datos, así como los esfuerzos por violentarla, han motivado la búsqueda de nuevas métodos para asegurar la privacidad de la información. Recientemente, se ha propuesto como respuesta a este problema la combinación de esteganografía y criptografía [3]. Se perfila como una solución muy completa, ya que une el camuflado del mensaje que logra la esteganografía, con el enmascarado del mensaje que logra la criptografía.

La esteganografía y la criptografía son campos muy explorados. En ambos se tienen muchas alternativas para trabajar. En criptografía se tienen varios algoritmos, al punto que se tienen que categorizar por tipo de llave (simétrica o asimétrica) [3]. Así como en esteganografía existe una variedad de medios, tales como imágenes y texto [5].

Al unir esteganografía y criptografía, se han usado diferentes combinaciones de algoritmos y medios [3]. Y en algunos de ellos se han logrado producir soluciones bastante completas, como en un caso de esteganografía de imágenes con criptografía de llave dinámica [6]. Sin embargo, todavía se presenta una gran variedad de problemas sin resolver, con propuestas bastante concretas [3].

Por otro lado, existen ideas que están en un estado inicial, que también pueden tener mucha proyección a futuro. Tal es el caso de un método en el cual no se modifica el portador o *carrier*, sino que se envía por aparte cuales partes de este portador se deben interpretar como mensaje. Challita y Farhat [4] proponen un ejemplo más concreto de este tipo de comunicación, y deja claro que no existe algo parecido en la literatura o implementado actualmente.

Más específicamente este documento expone la posibilidad de que primeramente se envíen varios *carriers*, de forma tal que le sea muy difícil a un intruso saber cual de todos contiene el verdadero mensaje; por esto es importante que el receptor y emisor antes se pongan de acuerdo sobre cual es el verdadero portador y cuales son meramente para despistar a los intrusos [4]. Como el mensaje en realidad no está contenido dentro del *carrier* como tal, resulta imposible que sepa cual de todos los portadores es el correcto sin información externa que le ayude a averiguarlo. Para poder pasar el mensaje lo que se hace en vez de modificar el portador es encontrar diferentes partes del portador que sean iguales al mensaje y guardarlas en un vector, este vector luego es enviado al receptor y este sabrá cuales partes del portador corresponden a información relacionada al mensaje [4].

Esta técnica podría tener una capa adicional de seguridad si el emisor y el receptor logran compartir los portadores por otra vía. Es decir que de alguna manera, la fuente y el destinatario compartan el *carrier* desde un momento previo al envío del vector. Ya que si algún atacante lograra interceptar el vector, no podría tener una manera fácil de conocer el portador.

Para implementar lo anterior primero es necesario afrontar dos importantes desafíos, el primero es como encontrar partes del portador (que sería una imagen) que correspondan a partes del mensaje. El segundo reto es cómo enviar este vector que contiene la información del mensaje al receptor de forma segura.

Para atacar el primer problema y poder mapear partes de la imagen a partes del mensaje el texto propone una solución de fácil implementación y relativamente eficiente [4]. Primeramente la imagen y el mensaje los dos se convierten a cadenas de bits. Luego para poder codificar el mensaje en la imagen se intenta encontrar la subcadena más grande que exista entre los dos utilizando una estructura de datos llamada árbol de sufijos generalizado, el cual tiene un orden de duración lineal. Entonces el algoritmo utilizaría una estrategia de divide y vencerás de la siguiente manera [4]: empieza con todos los bits del mensaje e intenta encontrar una cadena igual en la cadena de bits de la imagen, si la encuentra, guarda los índices de los bits de la cadena de la imagen (el primero y el ultimo) que contienen el mensaje en un vector, si no la encuentra, el algoritmo recursivamente parte el mensaje a la mitad y vuelve a intentar encontrar una cadena correspondiente en la imagen, así una y otra vez hasta que todo el mensaje esté mapeado en el vector [4].

Un ejemplo simple de lo anterior podría ser algo de esta forma: la cadena de la imagen es 10101111 la cadena del mensaje es 1011, por lo que en el vector se guardaría el par (3,6) pues corresponde a los índices de la cadena de la imagen donde está contenido el mensaje (si se inicia la cuenta del índice en uno).

Para solucionar el segundo desafío de como transmitir o pasar el vector al receptor es donde es necesario combinar la esteganografía con la criptografía. El texto propone de alguna manera encriptar el vector y su información con criptografía clásica y enviarlo al receptor, el cual sería en teoría el único que lo podría ver pues es el único que contiene la llave. En caso de que de alguna manera algún intruso pudiera desencriptar el mensaje del vector, se le haría muy difícil saber en primer lugar como interpretarlo, y en segundo saber a cual de los distintos portadores que se mandaron corresponde el mapeo del mensaje.

V. Pregunta de investigación

El problema abierto expuesto genera la siguiente pregunta: **¿Es útil combinar esteganografía y criptografía para aumentar la capacidad de incrustación y reducir la detectabilidad?** En teoría la gran ventaja de combinar estas disciplinas es que una práctica compensa las debilidades de la otra [4]. Uniéndolas se puede enviar un mensaje no entendible, por un medio que no parece llevar el mensaje, lo cual brinda una capa adicional de protección que provee un nivel mejorado de seguridad. La seguridad informática es un área que con el paso del tiempo se vuelve más crítica, es por esto que se busca desarrollar e investigar sobre esta posible solución.

Es importante recalcar que esta es una propuesta poco explorada. Y que en etapas iniciales se puede empezar a desarrollar usando algoritmos de cifrado sencillos, o formatos de medios que faciliten la extracción de la información. Se espera a través del desarrollo que se implementará en función del tema investigado marcar el principio de un camino para implementaciones más complejas que utilicen la misma lógica y conceptos.

VI. Diseño del producto

Se pretende desarrollar un producto que ayude a entender mejor la importancia y posible alcance de la solución en desarrollo, y que ayude a responder la pregunta de investigación.

VI-A. Descripción del producto

El producto se basa en una implementación a pequeña escala en forma de programa que comunica dos entes los cuales utilizarán el método propuesto para comunicar mensajes secretos de forma segura. El lenguaje de programación Java provee las características necesarias para el desarrollo que se plantea y por lo tanto será el utilizado para crear la aplicación.

Para la implementación existen cuatro principales desafíos:

1. Convertir una imagen a cadena de bits y viceversa.
2. Convertir un texto a cadena de bits y viceversa.
3. Programar el algoritmo que mapee el mensaje a una imagen.
4. Cifrar la información contenida en el vector.
5. Transmitir y recibir mensajes entre los dos entes que se comunican.

Se quiere simular una conexión que permita comunicación entre los dos entes teniendo a cada uno como un programa, y utilizando *sockets* para que se simule una conexión entre los dos. Estos se transmitirán solo tres tipos de mensajes, los URL de las imágenes portadoras y no portadoras, el vector cifrado, y mensajes de confirmación de recepción.

El programa transmisor creará el vector que mapea el mensaje a alguna de las imágenes, lo cifrará, y lo enviará a través de la conexión junto con varios URL que correspondan a distintas imágenes. Para crear el vector tiene que convertir el mensaje y alguna de las imágenes a una cadena de bits, para que el primero pueda ser mapeado dentro de la imagen.

El receptor entonces confirmará que los datos llegaron de forma satisfactoria y descifrará el vector. Procederá a convertir la imagen portadora en una cadena de bits, y utilizará el vector para saber qué partes de la imagen corresponden al mensaje, luego este mensaje, que por el momento esta en bits, tiene que ser convertido a texto para poder ser leído.

Las imágenes se manejan dentro del programa como direcciones URL para que su transmisión sea más fácil y rápida, además Java tiene una forma fácil de implementar el procesamiento de imágenes a través de estas direcciones. Otra ventaja es que se pueden utilizar más imágenes para que la información sea aún más segura.

Se piensa dividir el desarrollo del trabajo en dos entregables, los cuales incluirán lo siguiente:

1. La conversión del mensaje a bits y de bits al mensaje, además del mapeado que nos va a llevar al vector.
2. La encripción del vector y la transmisión del mismo y los medios.

Seguidamente se discuten las herramientas que se planean utilizar para el desarrollo del trabajo.

VI-B. Conceptos teóricos

Arreglo de bytes final: Se identificó la posibilidad de que una imagen no contuviera todos los bytes posibles (de 0 a 255). Este arreglo contiene la imagen en bytes, además de los bytes que no están contenidos en la imagen (los cuales se ponen al final del arreglo). Este arreglo tiene $3 * x * y + m$, donde x corresponde al ancho, y al alto, y m a la cantidad de bytes faltantes (el 3 es por RGB). Los bytes faltantes se agregan al final del archivo, en orden ascendente.

VI-C. Detalles de implementación

Muchos lenguajes de programación proveen herramientas que ayudan a resolver los desafíos propuestos. Para la conversión de texto e imágenes a bits y viceversa existen muchos métodos, pero uno de los más eficientes es el uso de *toBinaryString* para el texto y de *ByteArrayOutputStream* para imágenes (se utilizan nombres de Java, sin embargo, otros lenguajes proveen utilidades similares). Con esta ayuda la programación del algoritmo de mapeo al vector y de conversión de vuelta a texto o a imagen se vuelve mucho más simple.

Para el cifrado de la información del vector existe la biblioteca *crypto.Cipher* que incluye toda la implementación de un cifrado seguro y fácil de utilizar para el producto. Es óptimo para el proyecto porque no se llega a mucho detalle a la hora de utilizarlo, ya que ese no es el enfoque de la investigación.

Para la transmisión y recepción de los mensajes entre los dos entes se puede utilizar la funcionalidad de sockets, la cual puede usarse para crear una conexión entre dos programas. Esta conexión se puede realizar de forma local, lo cual es bueno porque el aspecto de la conexión como tal no es la que se está investigando a fondo en el proyecto.

Utilizando las herramientas, se realizó la implementación con base en lo expuesto por Challita y Farhat [4]. La propuesta estaba en un estado muy inicial, lo cual nos llevó a tener que tomar decisiones sobre el desarrolló de la solución. Seguidamente se listan algunos aspectos relevantes sobre la misma:

- Era muy fuerte para el sistema hacer un análisis a nivel de bits, por lo tanto, se tomó la decisión de hacerlo a nivel de bytes. Esto llevo a la necesidad de rellenar los bytes faltantes (si existen), llevando al concepto de arreglo de bytes final, explicado en la sección “Conceptos teóricos”. Este arreglo se crea en ambos lados (emisor y receptor), ya que se puede replicar, dado a que si recibe una misma imagen, el algoritmo de creación siempre generará el mismo resultado.

- Se manejó el vector como un objeto, el cual tenía que ser serializable, para que el servidor pudiera ensamblarlo correctamente.
- La implementación se realizó para el paso de mensajes en forma de texto. Sin embargo, se le podrían realizar pequeñas modificaciones para que funcione con archivos. Esta posibilidad ofrecería la ventaja de poder compartir una mayor variedad de contenido de forma segura.

VII. Experimentos

Con el objetivo de obtener posibles respuestas a la pregunta de investigación y lograr saber si la implementación es verdaderamente efectiva se realizan los siguientes experimentos que estudian las fortalezas y debilidades del producto.

VII-A. Primer experimento: Medición de bytes añadidos

Cuando se pasa la imagen a un arreglo de bytes y esta tiene bytes faltantes (es decir que no tiene todos los posibles valores para estos) es necesario rellenar al final del arreglo los que no están. Esto es necesario ya que podrían haber mensajes que no pueden ser incrustados en la imagen si falta alguno.

El máximo número de bytes que podrían faltar para una imagen son solo 255: para una imagen monocromática. Generalmente para una imagen cualquiera no es necesario agregar bytes, la mayoría de las veces ya todos están cuando esta se pasa al arreglo, sin embargo algunas imágenes que aparentan tener gran cantidad de información como muchos tipos de colores y que son de gran tamaño algunas veces necesitan que se les completen los bytes faltantes.

A continuación se presenta una tabla con los resultados del experimento en donde se quiere observar cuantos bytes se deben agregar en distintos casos de imagen. Primeramente imágenes de pocos colores y luego imágenes aleatorias que simulan un caso más parecido al de una aplicación real del método.

Imagen	Bytes agregados
Todo blanco	255
Todo negro	255
Todo verde	253
Todo rojo	253
Dos colores	247
Cinco colores	238
Cinco colores (2)	250
Nueve colores	145
Once colores	102
Setenta colores	0
Ochenta colores	3
Todos los colores	0
Promedio imágenes de pocos colores	166.75
Imagen aleatoria 11	26
Imagen aleatoria 13	4
Imagen aleatoria 20	9
Otras diecisiete imágenes aleatorias	0
Promedio imagenes aleatorias	1.95

VII-A1. Análisis primer experimento: Agregar los bytes faltantes al arreglo en realidad no es muy costoso computacionalmente, pero lo que sí lo es es recorrer byte por byte cada imagen para ver cuales están y cuales no. Lo anterior se hace realmente malo cuando son imágenes muy grandes las que se quieran utilizar, pues la cantidad de tiempo que se gasta recorriendo la imagen es considerable (aproximadamente 4 segundos para una imagen 3072 x 2048). Esto significa que sería muy lento y poco factible este método si se quisiera usar en un ambiente en que se mandan muchos mensajes en una corta cantidad de tiempo con imágenes de alta resolución.

La primera observación importante que se puede derivar del experimento es sobre las imágenes simples de pocos colores y la cantidad de bytes que hay que insertar en estas. Para una imagen de un solo color es lógico que tenga poca cantidad de distintos valores de bytes, pues no contiene gran variedad de información. Este es de los peores casos porque recorremos toda la imagen para luego darnos cuenta que tenemos que añadir prácticamente todos los valores de bytes que hay a excepción de unos cuantos.

Por otro lado, para las imágenes de pocos colores se puede observar que conforme aumenta la cantidad de colores e información en la imagen, se reduce rápidamente la cantidad de bytes que deben ser agregados. Cuando pasamos de cinco a nueve colores la cantidad de bytes que debemos agregar se reduce a casi la mitad, y ya para cuando hay setenta o más colores en la imagen generalmente hay suficiente información para que no se tenga que agregar casi ningún byte. Es muy extraño que si se quiere incrustar algún mensaje en una imagen se utilice una así de simple, pero es un buen punto de referencia comparativo; para estos casos de imágenes muy simples en promedio se deben agregar 167 bytes al arreglo, por lo que buscar toda la imagen para encontrarlos es más o menos consecuente.

En el caso de las imágenes aleatorias, para una prueba de 20 imágenes que simulaban posibles usos reales del método, es decir imágenes con mucha cantidad de información y de buen tamaño, solo tres requirieron que se les agregaran bytes faltantes a su arreglo. Es decir, la mayoría de las imágenes ya tenía los 256 posibles valores para un byte en su arreglo. En promedio solo es necesario agregar 2 bytes al arreglo de este tipo de imágenes, lo cual es bastante bajo. Al solo tener que agregar una pequeña cantidad de bytes se puede considerar un desperdicio de tiempo y poder computacional tener que escanear toda la imagen para luego casi no modificarla, por lo que este es un punto débil de la implementación.

Una posible solución al gasto computacional que se hace cuando innecesariamente se explora toda una imagen para luego no modificarla, es asegurarse de ante mano una lista de imágenes las cuales ya se han recorrido sus arreglos y se sabe que no tienen bytes faltantes. Esta lista es la única que se utilizaría para incrustar los mensajes y de esta manera no habría que hacer el anterior chequeo. Implementar esto tendría implicaciones negativas en el aspecto de seguridad, pues al delimitar las imágenes que se pueden usar se le facilitaría a un intruso buscar la imagen en que se está mapeando el mensaje.

VII-B. Segundo experimento: Medición del tamaño del vector

Se tomó la decisión de hacer un segundo experimento, que relacionara tamaños en bytes, con los conceptos de capacidad de incrustación e indetectabilidad. Por la naturaleza de la solución, se le hacen pocas modificaciones a la imagen. Además estas modificaciones a la imagen no son transmitidas (ya que emisor y receptor crean su propio arreglo de bytes final).

Viendo la solución, se podría decir que la capacidad de incrustación es casi infinita. Esto ocurre si se toma en cuenta el aspecto de la definición que dice que es cuanta información podemos esconder sin perder calidad, ya que por lo general las modificaciones a la imagen son mínimas o nulas (algo que se comprobó en el primer experimento). Usando esto se podría mapear un mensaje de casi cualquier longitud a una imagen, y lo único que ocurriría es que el vector crecería de acuerdo al largo del mensaje. Se usó esto como motivación para ver cuál era el mayor largo de mensaje que se podía enviar usando el método estudiado, sin que el vector excediera el tamaño del archivo original.

La fortaleza del método que se relaciona a este experimento es el hecho de no tener que hacer un envío de la imagen. Se notó la posibilidad de hacer una comparación entre el tamaño de la imagen que se hubiera enviado, al tamaño del vector encriptado. Para esto, se tomó una imagen aleatoria con bastantes colores (tamaño $256*256$), la cual pesaba aproximadamente 27 kilobytes. Después, se generó texto aleatorio (del tipo Lorem Ipsum), de diferentes longitudes, y se midió el tamaño (en bytes) del vector en forma de archivo, para compararlo al tamaño de la imagen.

Caracteres mensaje	Vector encriptado (bytes)
500	4160
1000	8288
1500	15328
2000	16160
2500	21360
3000	30400

En promedio, el tamaño del vector encriptado, superaba 8,93 veces el tamaño del mensaje. Se puede observar que el último resultado excedía el tamaño de la imagen. Esto implica que en un escenario donde una imagen se puede modificar lo suficiente, y no se pueden enviar más de 27 kilobytes, la única manera de transmitir el mensaje usando esteganografía, es con esteganografía convencional. Esto se cubrirá mejor en el análisis de este experimento, que se expone seguidamente.

VII-B1. Análisis segundo experimento: Al hacer este estudio, se notaron ciertos aspectos relacionados al tamaño del vector.

Lo que se envía (vector encriptado pasado a archivo), es de un tamaño muy grande, para lo que contiene. En otras palabras, el vector se pudo haber implementado de manera que al convertirlo a un archivo solo contuviera enteros en estado *raw*, es decir, sin ninguna información adicional al valor del entero. Sin embargo, en el estado actual de la implementación, se utilizan listas enlazadas de un objeto que se creó con el fin de facilitar la creación de índices para el mapeo. Eso puede tener un impacto relevante en el tamaño final del vector.

Existe una posibilidad de beneficio que es difícil de controlar en el caso de que un mensaje "se parezca" a la imagen. Esto ocurre cuando un cantidad considerable de caracteres de un mensaje coinciden con los bytes de una imagen. Esto implica un mapeo muy favorecedor, que lleva a un vector pequeño para el tamaño del mensaje. Esto se puede observar en la tabla contenida en la descripción de este experimento. En ese ejemplo, se puede ver que en algunos casos, el mensaje coincidió bastante con la imagen, como en el ejemplo del mensaje de 2000 caracteres, donde el vector fue poco más de 8 veces más grande que el mensaje. Por otra parte, algunas casos fueron menos eficientes, como el de 3000 caracteres, donde el tamaño del vector era más de 10 veces más grande que el mensaje, algo que también ocurrió en el caso de 1500 caracteres.

Realizar estos experimentos, llevó a notar deficiencias de la solución. Por otra parte, esas pruebas permitieron imaginar posibles mejoras. Esas ideas y limitaciones se detallan en la siguiente sección.

VII-C. Alcances y limitaciones

La implementación logra cumplir los objetivos propuestos, sobre todo, con fines de estudio. Por ejemplo, el mapeo que hace del mensaje dentro de la imagen es eficiente y eficaz. También se logra cifrar el vector y transmitir el mensaje correctamente en la implementación usada, para todo tipo de tamaños de mensaje. Se logró responder la pregunta de investigación satisfactoriamente, se observó una mejora en la capacidad de incrustación e indetectabilidad al combinar las dos disciplinas estudiadas. En el primer experimento queda claro que los cambios que eventualmente habría que hacerle a la imagen son solo de a los sumo 255 bytes, lo cual es muy bajo, por lo que la capacidad de incrustación es alta y por consiguiente también la indetectabilidad. En el segundo experimento salen a relucir propiedades que ayudan a la indetectabilidad como que solo se pase por el canal el vector cifrado, aunque con algunas debilidades pues este crece muy rápido. Sin embargo, todavía tiene limitaciones y posibles puntos a mejorar. Entre las limitaciones se pueden mencionar:

- Inicialmente se había pensado en implementar la solución con un sistema que enviaba varios URLs para confundir a los posibles receptores, sin embargo esto no se implementó. Hubiera sido un método útil para poder de paso compartir las imágenes en las que se iba a esconder el mensaje.
- El estado actual de la solución solo permite contraseñas de 16 caracteres, debido a las herramientas utilizadas para la parte de criptografía. A pesar de que es un número que provee mucha seguridad (gran cantidad de permutaciones), no es tan cómodo a la hora de usarlo. Esta limitación añade también la incapacidad de usar la incertidumbre de no saber el tamaño de la contraseña

para hacer el producto más seguro. Se restringe el posible dominio de la contraseña y con esto se echa a perder una importante parte de la seguridad criptográfica.

- En el primer experimento se expone la falencia del producto de tener que buscar en toda la imagen siempre los bytes faltantes, y lo ineficiente que esto se puede volver. La lentitud que provoca este problema puede ser un gran inconveniente si el producto se quiere usar para una carga de trabajo fuerte.

Por otro lado, algunos puntos de mejora son:

- Si se aplicaba la idea de los varios URL en la implementación, existía la posibilidad de crear una utilidad en el programa que detectara cual de los URL proveía la imagen con el mapeo más eficiente.
- Como el mapeo es en bytes, la implementación para transmitir otros tipos de mensaje que no sean texto puede realizarse con este mismo método, inclusive se podrían usar otros portadores también, como audio, conservando las mismas ventajas de capacidad de incrustación que se tiene con imágenes.
- Como se mencionó en el experimento 2, la utilización de estructuras relativamente complejas en el vector, afectan el tamaño del mismo. Esto implicó un código ordenado, pero con el costo de vectores más grandes de lo que era necesario.
- Idear una manera de completar los bytes faltantes en el arreglo de bytes de cada imagen sin tener que buscar en toda la imagen haría el producto mucho más eficiente, y quitaría una de las mayores trabas presentes en este momento. Una posible idea de solución es que el emisor y el receptor se den a entender en el vector que cierta parte del mensaje no se encuentra en la imagen. Por ejemplo, ir a otro lugar a conseguir el mensaje, es decir que se maneje por aparte una lista de todos los bytes siempre y que los busque ahí en vez de en la imagen si cierta bandera está encendida.

REFERENCIAS

[1] P. Deymonnaz, "Análisis de vulnerabilidades esteganográficas en protocolos de comunicación IP y HTTP", tesis, Universidad de Buenos Aires, 2012.

[2] P. Johri, A. Mishra, S. Das, A. Kumar, "Survey on steganography methods (text, image, audio, video, protocol and network steganography)", 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, 2016, pp. 2906-2909.

[3] Baby, A., and Krishnan, H. (2017). Combined Strength of Steganography and Cryptography-A Literature Survey. International Journal of Advanced Research in Computer Science, 8(3).

[4] Challita, K., and Farhat, H. (2011). Combining steganography and cryptography: new directions. International Journal of New Computer Architectures and their Applications (IJNCAA), 1(1), 199-208.

[5] Song, S., Zhang, J., Liao, X., Du, J., and Wen, Q. (2011). A novel secure communication protocol combining steganography and cryptography. Procedia Engineering, 15, 2767-2772.

[6] N. Patel and S. Meena, "LSB based image steganography using dynamic key cryptography", 2016 International Conference on Emerging Trends in Communication Technologies (ETCT), Dehradun, 2016, pp. 1-5.