INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

ESCOM

Trabajo Terminal

"Aplicación de cifrado contra de adversarios clasificadores, para el correo electrónico"

2015-A010

Presentan

Arcos Ayala Jonathan
Zepeda Ibarra Allan Ulises

Directores

Dra. Sandra Díaz Santiago
M. en C. Manuel Alejandro Soto Ramos

Noviembre 2015

Índice

Índice de Figuras

Índice de Tablas

Introducción

Actualmente, una gran cantidad de personas hacen uso del internet y de las nuevas tecnologías para comunicarse. Con ello, también se incrementa la cantidad de información que se transmite y/o almacena. En diversas ocasiones, esta información es susceptible a sufrir distintos tipos de ataques, tales como acceso no autorizado, modificación o destrucción de la misma, entre otros. Adicionalmente, cada día aparecen nuevos tipos de ataques a los sistemas de información. Por lo tanto, surge la necesidad de proteger dicha información.
Una de las tecnologías ampliamente usada para comunicarse es el correo electrónico [1]. Los mensajes que envían y reciben los usuarios de correo electrónico pueden ser de diferentes tipos: personales, transaccionales, de notificación o de publicidad. Por lo tanto, cada vez que se escribe y envía un correo electrónico, se está revelando información acerca de las preferencias y/o intereses del usuario. Estos datos, son el insumo más importante, para distintas entidades, entre las cuales están empresas que realizan publicidad en línea, proveedores de internet, instituciones de gobierno, entre otros [2]. El propósito de tener estos datos puede ser realizar publicidad efectiva, vender los datos a empresas de publicidad o averiguar si determinado usuario es una amenaza para el gobierno. Para obtener información acerca de los intereses y/o preferencias del usuario, se hace uso de programas de cómputo denominados *clasificadores*. Los clasificadores son herramientas informáticas que analizan una gran cantidad de información, haciendo uso de técnicas de aprendizaje máquina [3], y posteriormente clasifican un mensaje en determinada categoría o perfil. En este contexto, los clasificadores pueden constituir una amenaza para algunos usuarios del correo electrónico, por tal motivo de ahora en adelante a los programas que clasifican se les denominará *adversarios clasificadores*.
Ante tal escenario, surge la pregunta ¿cómo se puede proteger un usuario contra los adversarios clasificadores? Una posible respuesta es hacer uso de algoritmos de cifrado estándar. Sin embargo, hacer uso de tales algoritmos, implica que los participantes en la comunicación acuerden una clave de cifrado. Desafortunadamente, acordar una clave, no es un proceso sencillo para el usuario común. Otra desventaja de esta primera solución, es que los algoritmos de cifrado estándar ofrecen un alto nivel de seguridad, el cual resulta excesivo cuando se consideran los recursos y el objetivo de un adversario clasificador [4].

0.1. Justificación

La comunicación por medio del correo electrónico es atacada constantemente y por ellos se han creado diferentes herramientas para asegurar la transferencia de información entre los usuarios. Pero estas herramientas ofrecen un conjunto de servicios como confidencialidad, no repudio, autenticación, entre otros y es porque están pensadas para hacer frente a adversarios mejor capacitados en la adquisición de información de los usuarios de correo electrónico.

Estas herramientas al enfrentarse a adversarios más capacitados necesitan implementar esquemas y técnicas más sofisticadas para establecer una comunicación segura entre usuarios y el mayor reto que se les presenta es el intercambio de claves, porque si un adversario llega a obtener al menos una clave, el esquema de seguridad se considera roto y la comunicación es vulnerable al ataque del o los adversarios que tengan esa clave robada.
Si tomamos en cuenta que los adversarios clasificadores son programas de cómputo que solo leen el contenido del correo y buscan palabras específicas no necesitan tantos servicios criptográficos para detener sus ataques a los correos electrónicos. Sería suficiente con tener un esquema de cifrado que proporcione confidencialidad durante el envío de mensajes.
Pero este esquema no solo tiene que preocuparse por la confidencialidad en el envío de los mensajes, también se enfrenta al problema de intercambio de claves para poder descifrar el mensaje por el usuario que recibe el mensaje.
Por lo tanto en este trabajo terminal se propone utilizar CAPTCHAS para el envío de claves entre los usuarios. Los CAPTCHAS contienen una cadena de caracteres que al ser resueltos por un ser humano es posible calcular la clave con que fue cifrado el mensaje, y como el adversario clasificador es un programa de cómputo, le es muy complicado encontrar la clave para descifrar el mensaje y poderlo clasificar correctamente.

0.2. Objetivos

0.2.1. Objetivos Generales

Desarrollar una herramienta para un cliente de correo electrónico que permita cifrar el contenido de los mensajes para evitar su clasificación, haciendo uso de técnicas criptográficas simétricas y un servidor que verifique el envío y recepción de CAPTCHAS entre usuarios.

0.2.2. Objetivos Específicos

1. Desarrollar una herramienta en un cliente de correo electrónico para el envío y recepción de los correos cifrados y la generación, envío y recepción de CAPTCHAS.

2. Desarrollar un servidor de llaves que reciba, aloje y envíe los CAPTCHAS a los usuarios para descifrar los correos electrónicos.

3. Desarrollar un algoritmo de cifrado y descifrado basado en el envío y recepción de CAPTCHAS.

Capítulo 1

Preliminares

En este capítulo se hablará sobre el correo electrónico y algunos de los intermediarios que hacen posible que este servicio sea usado en toda la red de internet, también se describirá de las amenazas a las que se tiene que enfrentar este servicio para hacer llegar información de un usuario a otro de una manera segura y cuales han sido las respuestas de los proveedores de servicio de correo electrónico para proporcionar dicha seguridad a los usuarios.

1.1. Correo electrónico

El correo electrónico es el servicio de internet por el cual se pueden enviar mensajes entre dos usuarios que cuenten con este servicio. El correo electrónico o "e-mail" por sus siglas en inglés, basa su funcionamiento en las oficinas postales. Tomando esa analogía se puede afirmar que los usuarios tienen cartas (mensajes de correo electrónico) que desean enviar a otros usuarios; estas cartas son enviadas a las oficinas postales (servidores de correo electrónico) donde se almacenan con otras cartas que van dirigidas a otros usuarios en diferentes ciudades; estas oficinas envían las cartas a otras oficinas si es necesario; y por último estas se encarga de colocar el mensaje en el buzón del usuario correspondiente (buzón de correos electrónicos). Para poder entender la comunicación en correo electrónico se necesitan definir los siguientes conceptos:

- **Mensaje de correo electrónico** Los mensajes de correo electrónico al igual que las cartas tienen la dirección del receptor, la dirección del emisor y el cuerpo del mensaje, pero a diferencia de las cartas los correos electrónicos son enviados por internet y necesitan sus propios formatos. Para poder enviar un mensaje de correo electrónico es necesario tener los siguientes elementos como mínimo.
  Dirección del remitente: Esta dirección se compone por dos elementos importantes, el primero es el nombre de usuario seguido de un carácter "@", el segundo elemento es el dominio donde está alojado el servicio de correo electrónico.
  Direcciones de los receptores: En un correo electrónico debe haber al menos una dirección de receptor para ser enviado, esta tiene la misma estructura que la dirección del remitente.
  Contenido del mensaje: Es el texto que se desea transmitir entre el remitente y el receptor.
  El mensaje de correo electrónico cuenta con más elementos pero estos son opcionales y se pueden consultar en el RFC 2821 extensión MIME.

- **Dominio de internet**
  Es un nombre que identifica a los diferentes dispositivos interconectados en una red sin la necesidad de aprenderse un número de red. Estos dispositivos pueden proporcionar diferentes servicios como el servicio de correo electrónico.

- **Buzón de correo electrónico** Un buzón de correo electrónico es un espacio virtual proporcionado por el servicio de correo electrónico que sirve para almacenar los mensajes enviados y recibidos.

- **Usuario de correo electrónico** Se le conoce como usuario de correo electrónico a la persona que se registra en un dominio para obtener los servicios de mensajería proporcionados por un servidor de correo electrónico.

- **Cliente de correo electrónico** El cliente de correo electrónico es una interfaz por la cual el usuario de correo electrónico puede administrar sus mensajes enviados y recibidos.
  El cliente de correo electrónico puede ser de diferentes tipos como son:

  Cliente de correo electrónico para la pc: Estos clientes de correo electrónico son instalados en una computadora y es configurado para sincronizarse con un servidor de correo electrónico cada cierto tiempo o cuando el usuario se lo indique. Una de las principales características de éste cliente de correos es la capacidad de descargar los mensajes del servidor a la computadora para ser leídos sin la necesidad de tener una conexión de internet y en cuanto tiene conexión con el servidor otra vez descarga los nuevos mensajes y envía los que se tienen pendientes en la computadora.

  Cliente de correo electrónico web: Los clientes web necesitan un explorador de internet y una conexión permanente a la red para que funcione, estos clientes normalmente son proporcionados por el mismo servidor de correo electrónico y nunca descarga los correo en las computadora donde son utilizados.

  Cliente de correo electrónico para móviles: Un cliente de correo electrónico móvil se caracteriza por ser una aplicacion instalada en un dispositivo móvil. Esta aplicación descarga una copia temporal de los últimos mensajes recibidos.

- **Servidor de Correo electrónico** Un servidor de correo electrónico es un programa que se encarga de enviar y recibir los mensajes de correo electrónicos de sus usuarios registrados, este servidor puede recibir mensajes de usuarios de otros servidores de correos que sean dirigidos a sus usuarios registrados.
  Este servidor tiene que seguir algunos estándares que existen en internet para el envío(protocolo smtp) y recepción(protocolo pop3 o imap) de mensajes de correo electrónico.

- **Protocolo SMTP** El protocolo SMTP significa "protocolo para transferencia simple de correo" o "Simple Mail Transfer Protocol" por sus siglas en inglés, el cual se encarga de enviar los mensajes de correo electrónicos entre dispositivos que se encuentran interconectados en la red o en internet. Este protocolo solo se utiliza para mandar los mensajes entre servidores o entre el usuario emisor y su servidor de correo electrónico. Podemos revisar el protocolo completo en el RFC5321 [6].

- **Protocolo POP3** Este protocolo se encarga de descargar los mensajes del servidor a un cliente de correo electrónico que el usuario haya configurado previamente. Una de las características es que solo se sincroniza para la descarga de los mensajes de correo y no deja una copia de seguridad en el servidor de correo electrónico. Podemos consultar el funcionamiento detallado en el RFC1939 [7].
- **Protocolo IMAP** Este protocolo, al igual que el protocolo POP3, se encarga de la descarga de los mensajes del servidor a un cliente de correo electrónico con la diferencia de que la sincronización entre el servidor y el cliente es continua, manteniendo una copia de seguridad en el servidor. Con este protocolo es posible tener varios clientes de correo configurados con la misma cuenta y los cambios que se realicen en cualquiera de los clientes de correo se verán reflejados en el servidor y en los diferentes clientes sincronizados. Este protocolo puede ser consultado en el RFC 6851 [8].

Como se ha podido ver en el presente documento, el servicio de correo electrónico es un canal de comunicación que se ayuda de varios elementos para completar la comunicación entre dos usuarios de correo electrónico, no importando que estos estén registrados en 2 servidores de correo diferentes. Pero para poder entender por completo el comportamiento de este servicio se tiene que definir ciertas características de este servicio.
Este servicio establece una comunicación no orientada a conexión, lo que significa que el receptor no necesita estar conectado al servicio de correo electrónico para recibir el mensaje en su buzón de correo electrónico.
Los mensajes enviados por medio del correo electrónico transitan por el internet como archivos en texto plano, esto quiere decir que cualquiera que tenga una copia del mensaje puede abrir el correo electrónico y leer el mensaje siguiendo la estructura del protocolo SMTP.
Además de mandar mensajes tiene la facultad de enviar archivos multimedia en el mismo mensaje de correo electrónico, esta característica ha sido explotada bastante por empresas privadas y de gobierno para tener comunicados a sus empleados, departamentos, proveedores, socios, etc.
El correo permite enviar el mismo mensaje a más de un usuario sin la necesidad de hacer un mensaje para cada usuario, esto lo han utilizado muchas empresas de publicidad para hacer publicidad a gran escala y a muy bajo costo, a este servicio se le llama SPAM.
Este medio de comunicación es bastante rápido ya que se estima que un mensaje de correo electrónico tiene que llegar a su destino a más tardar en 5 minutos, no importando la ubicación geográfica del servidor de correo electrónico.
Las características mencionadas dan a notar que el correo electrónico tiene una baja seguridad al momento de enviar los mensajes, ya que la información que se manda es sumamente fácil de leer por cualquiera que pueda tener una copia del mensaje. Si se toma en cuenta que un mensaje de tiene que saltar por varios servidores antes de llegar a su destino, es fácil suponer que se puede interceptar una copia del mensaje en el envío de servidor a servidor.
Por estos motivos los servidores de correo electrónico han implementado diversos candados de seguridad para blindar la transferencia de mensajes entre servidores. Una de las maneras que han encontrado para proporcionar dicha seguridad ha sido a través de la implementación de técnicas criptográficas.

1.2. Criptografía

Es el estudio de técnicas matemáticas relacionadas con la seguridad para proveer, confidencialidad, integridad y autenticación. Estas técnicas están destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados. [11]
El objetivo fundamental de la criptografía es establecer una comunicación segura entre dos usuarios, usualmente esta comunicación es representada con Alicia y Bob, que son los interlocutores en un canal de comunicación y a una tercera persona que es el adversario, Oscar, que ataca la comunicación para obtener información.
Cuando Alicia manda un mensaje a Bob se le llama "texto en claro" y si Oscar intercepta el mensaje puede leer su contenido sin problemas. Ahora si Alicia cifra el texto en claro usando una clave, ella obtiene un texto cifrado y es enviado por el canal de comunicación a Bob, si Oscar intercepta este mensaje ya no podría leerlo porque no sabe como pasar del texto cifrado al texto el claro, pero Bob si descifrar el texto cifrado porque el si conoce la clave de cifrado. [12]
La idea anterior se puede describir formalmente usando una notación matemática de la siguiente manera.
Un sistema criptográfico es una quintupla $(\mathcal{M}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$ donde:

1. $\mathcal{M}$ es el espacio finito de posibles mensajes en claro.
2. $\mathcal{C}$ es el espacio finito de posibles mensajes cifrados.
3. $\mathcal{K}$ es el espacio finito deposibles claves a utilizar.
4. Para cada $K \in \mathcal{K}$, se tiene una regla $e_k \in \varepsilon$ y su correspondiente regla $d_k \in \mathcal{D}$. Donde $e_k : \mathcal{P} \to \mathcal{C}$ y $d_k : \mathcal{C} \to \mathcal{P}$ siendo funciones tales que $d_k(e_k(x)) = x$ siendo x un elemento del espacio de textos en claro $x \in \mathcal{P}$

Como ya se ha mencionado la criptografía provee diferentes servicios al momento del envío de mensajes entre usuarios, estos servicios son:

- **Confidencialidad**: La información sólo es accesible sólo para aquéllos que están autorizados.
- **Integridad**: La información sólo puede ser creada y modificada por quien esté autorizado a hacerlo.
- **Autenticación**: Se garantiza que el mensaje provino del aparente autor o fuente.
- **No repudio**: Este servicio evita que las entidades en una conexión nieguen compromisos establecidos.

1.2.1. Tipos de Ataques

A pesar de la implementación de estas técnicas criptográficas los ataques a las comunicaciones siguen existiendo y estos ataques se han clasificado dependiendo de cuanta información tenga disponible el adversario para poder romper el cifrado y obtener la información deseada. [13] Los tipos de ataques son:

- **Ataque con sólo texto cifrado(Ciphertext-only attack)**: En este tipo de ataque el adversario tiene acceso sólo al texto cifrado, y no tiene acceso al texto plano, pero es el ataque más débil debido a la falta de información.
- **Ataque de texto plano(Known plaintext attack)**: En este tipo de ataque el adversario se supone que tienen acceso a la lista en un número limitado de pares de texto plano y el texto cifrado correspondiente.
- **Ataque de texto cifrado elegido(Chosen ciphertext attack)**: Este ataque se puede elegir los textos cifrados arbitrariamente y tener acceso a texto planos después de ser procesados. Para que este ataque se lleve a cabo es necesario tener el extremo receptor de la comunicación y acceso al canal de la comunicación.
- **Ataque de texto plano elegido(Chosen plaintext attack)**: Este ataque es capaz de elegir una serie de textos planos para ser cifrados y tener acceso al texto cifrado resultante. Esto le permite explorar cualquier área del espacio de texto plano que desea y puede permitirle explotar el comportamiento no aleatorio que sólo aparecen con ciertos textos planos.

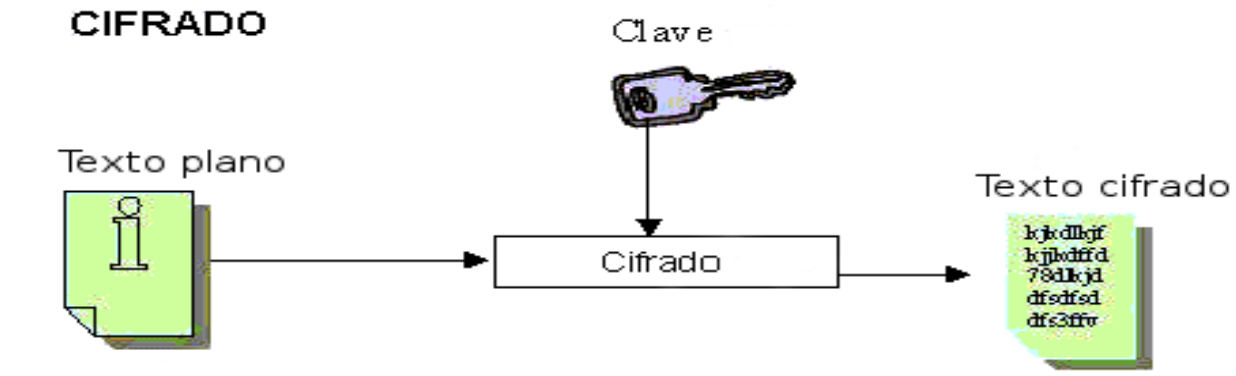

Figura 1.1: Diagrama cifrado

Cifrado Simétrico:
Un sistema de cifrado simétrico es un tipo de cifrado que usa una misma clave para cifrar y para descifrar. Las dos partes que se comunican mediante el cifrado simétrico deben estar de acuerdo en la clave a usar de antemano. Una vez de acuerdo, el remitente cifra un mensaje usando la clave, lo envía al destinatario, y éste lo descifra usando la misma clave.
Un buen sistema de cifrado pone gran parte de la seguridad en la clave y el resto en el algoritmo. En otras palabras, no debería ser de ninguna ayuda para un atacante conocer el algoritmo que se está usando. Sólo si el atacante obtuviera la clave, le serviría conocer el algoritmo.
Dado que toda la seguridad está en la clave, es importante que sea muy difícil adivinar el tipo de clave. Esto quiere decir que el abanico de claves posibles, o sea, el espacio de posibilidades de claves, debe ser amplio.

Hoy por hoy, las computadoras pueden adivinar claves con extrema rapidez, y ésta es la razón por la cual el tamaño de la clave es importante en los sistemas modernos. El algoritmo de cifrado DES usa una clave de 56 bits, lo que significa que hay 2^{56} claves posibles. 2^{56} son 72.057.594.037.927.936 claves. Esto representa un número muy alto de claves, pero una computadora de uso general puede comprobar todo el espacio posible de claves en cuestión de días. Una máquina especializada lo puede hacer en horas. Por otra parte, algoritmos de cifrado de diseño más reciente como DES, 3DES, AES, Blowfish e IDEA usan todos claves de 128 bits, lo que significa que existen 2^{128} claves posibles. Esto representa muchas, muchísimas más claves, y aun en el caso de que todas las máquinas del planeta estuvieran cooperando, todavía tardarían más tiempo que la misma edad del universo en encontrar la clave. [14]
A continuación veremos un diagrama de cómo es que viaja la información por medio de un esquema de cifrado simétrico.

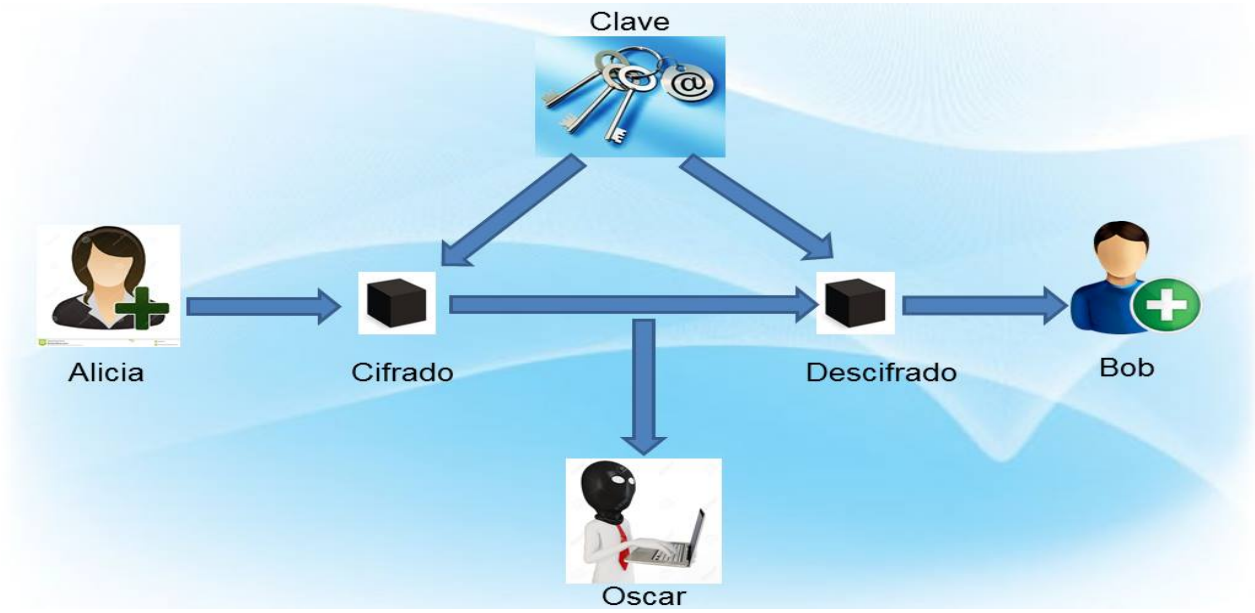

Figura 1.2: Diagrama cifrado simétrico

Cifrado por bloques:
Este tipo de cifrado toma bloques de información del texto plano y produce bloques de información cifrados de un tamaño fijo, normalmente es del mismo tamaño que el bloque de información del texto plano. Los bloques de cifrado tienen que cumplir la condición de ser lo suficientemente grandes como para evitar ataques de texto cifrado, otra condición que se debe cumplir es que la asignación de bloques de entrada a bloques de salida es uno a uno para hacer el proceso reversible.
Formalmente un cifrado en bloque se considera que es seguro si se comporta como una permutación pseudoaleatoria fuerte, es decir, un cifrado de bloques es seguro si un adversario no puede distinguir su salida de una permutación elegida al azar.
Para hacer la asignación de bloques los algoritmos de cifrado utilizan sustituciones y permutaciones en los bloques de texto plano hasta obtener un texto cifrado.
La sustitución es el remplazo de un bloque de k bits por otro bloque de k bits en un espacio de 2^k. [15]

Modos de operación:
ECB(Electronic codebook): Este modo de operación es probablemente el más simple de todos, el texto plano P está segmentado como $M = M_1||M_2||...||M_m$ donde cada M_i es un bloque de n bits. A continuación la funcion de cifrado E_k se aplica por separado a cada bloque M_i.
A continuación tenemos el diagrama de este modo de cifrado.

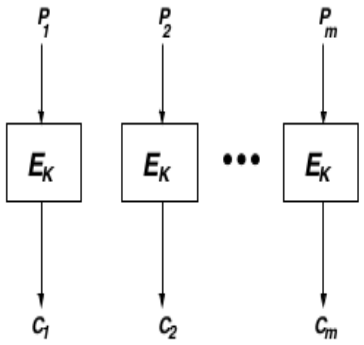

Figura 1.3: Diagrama ECB

CBC(Cipher-block chaining): Este modo de operación la salida de un bloque de cifrado se introduce en el siguiente bloque de cifrado junto con el siguiente bloque del mensaje.

Algorithm CBC.Encrypt$_K^{\text{IV}}(P)$	**Algorithm** CBC.Decrypt$_K^{\text{IV}}(C)$
1. Partition P into $P_1, P_2, \ldots, P_m$	1. Partition C into $C_1, C_2, \ldots, C_m$
2. $C_1 \leftarrow E_K(P_1 \oplus \text{IV})$;	2. $P_1 \leftarrow E_K^{-1}(C_1) \oplus \text{IV}$
3. **for** $i \leftarrow 2$ to m	3. **for** $i \leftarrow 2$ to m
4. $\quad C_i \leftarrow E_K(P_i \oplus C_{i-1})$	4. $\quad P_i \leftarrow E_K^{-1}(C_i) \oplus C_{i-1}$
5. **end for**	5. **end for**
6. **return** $C_1, C_2, \ldots, C_m$	6. **return** $P_1, P_2, \ldots, P_m$

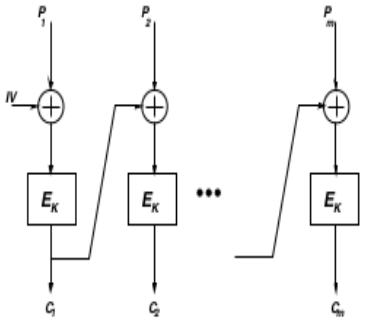

Figura 1.4: Diagrama CBC

CBC toma como bloques de mensajes de entrada M y un vector de inicialización (IV). Durante el cifrado, la salida del i-ésimo bloque depende de los i-1 bloques anteriores. Así, el cifrado CBC es intrínsecamente secuencial.

CFB(Cipher Feedback): Este modo de operación el cifrado por bloques también están encadenados pero a la salida se produce de una manera muy diferente de la de CBC. Para cada bloque, de salida se la hace XOR con el siguiente bloque de entrada.

Algorithm CFB.Encrypt$_K^{\text{IV}}(P)$	**Algorithm** CFB.Decrypt$_K^{\text{IV}}(C)$
1. Partition P into $P_1, P_2, \ldots, P_m$	1. Partition C into $C_1, C_2, \ldots, C_m$
2. $C_1 \leftarrow E_K(\text{IV}) \oplus P_1$;	2. $P_1 \leftarrow E_K(IV) \oplus C_1$
3. **for** $i \leftarrow 2$ to m	3. **for** $i \leftarrow 2$ to m
4. $\quad C_i \leftarrow E_K(C_{i-1}) \oplus P_i$	4. $\quad P_i \leftarrow E_K(C_{i-1}) \oplus C_i$
5. **end for**	5. **end for**
6. **return** $C_1, C_2, \ldots, C_m$	6. **return** $P_1, P_2, \ldots, P_m$

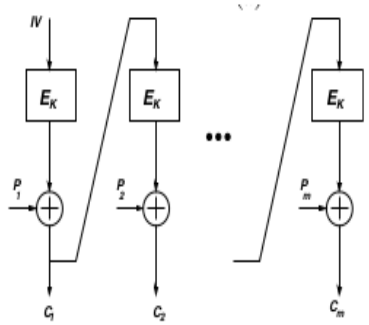

Figura 1.5: Diagrama CFB

OFB(Output feedback): Este modo de operación IV es encriptada varias veces para obtener una corriente de bytes aleatorios, el resultado de esto se hace XOR con el bloque de texto plano mientras que la corriente de bytes aleatorios se usa como parametro del siguiente bloque. A diferencia de los otros modos en la OFB ninguna parte del texto claro entra directamente a cifrarse se ejecuta de la mima manera que el modo OFB con la diferencia que le vector de inicialización es generado por un bloque llamado contador, el cual cambia una vez que es implementado en otro bloque de cifrado.

Algorithm CFB.Encrypt$_K^{\text{IV}}(P)$	**Algorithm** CFB.Decrypt$_K^{\text{IV}}(C)$
1. Partition P into $P_1, P_2, \ldots, P_m$	1. Partition C into $C_1, C_2, \ldots, C_m$
2. $X \leftarrow$ IV;	2. $X \leftarrow$ IV
3. **for** $i \leftarrow 1$ to m	3. **for** $i \leftarrow 1$ to m
4. $X \leftarrow E_K(X)$;	4. $X \leftarrow E_K(X)$;
5. $C_i \leftarrow X \oplus P_i$	5. $P_i \leftarrow X \oplus C_i$
6. **end for**	6. **end for**
7. **return** $C_1, C_2, \ldots, C_m$	7. **return** $P_1, P_2, \ldots, P_m$

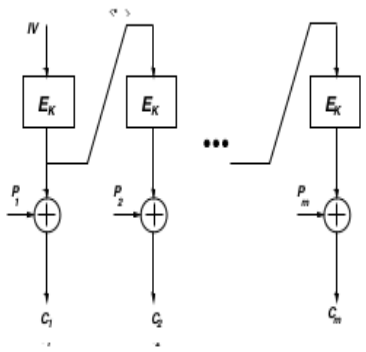

Figura 1.6: Diagrama OFB

1.3. Numeros Primos

Los números primos son aquellos números enteros que sólo son divisibles por si mismos y por la unidad. Los numeros que no son primos se llaman compuestos, excepto el numero 1 que no se considera ni primo ni compuesto. En el libro IX Euclides demostró que existe una infinidad de numeros primos.

1.4. aritmetica modular

En matemática, la aritmética modular es un sistema aritmético para clases de equivalencia de números enteros llamadas clases de congruencia. La aritmética modular fue introducida en

1801 por Carl Friedrich Gauss en su libro Disquisitiones Arithmeticae. La aritmética modular puede ser construida matemáticamente mediante la relación de congruencia entre enteros, que es compatible con las operaciones en el anillo de enteros: suma y multiplicación. La aritmética modular se basa en una relación de equivalencia, y las clases de equivalencia de un entero a se denota con [a]n (o simplemente [a] si sobreentendemos el módulo.) $a \mod n$. El conjunto de todas las clases de equivalencia se denota con $Zn = [0]n, [1]n, [2]n, ..., [n-1]n$. Un hecho importante sobre aritmética modular, cuando los módulos son números primos es el pequeño teorema de Fermat: si p es un número primo, entonces:

Si a es cualquier entero:

$a^p \equiv a(mod \quad p)$

Si a es un entero no divisible entre p:

$a^{p-1} \equiv 1(mod \quad p)$ [9]

1.5. Polinomio de Lagrange

El polinomio de Lagrange, llamado así en honor a Joseph-Louis de Lagrange, es una forma de presentar el polinomio que interpola un conjunto de puntos dado.

Dado un conjunto de $k+1$ puntos

$(x_0, y_0), ..., (x_k, y_k)$

donde todos los x_j se asumen distintos, el polinomio interpolador en la forma de Lagrange es la combinación lineal.

$$L(x) = \sum_{j=0}^{k} y_j l_j(x) \tag{1.1}$$

de bases polinómicas de Lagrange

$$l_j = \prod_{i=0, i \neq j}^{k} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} ... \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} ... \frac{x - x_k}{x_j - x_k} \tag{1.2}$$

[10]

Capítulo 2

Conceptos de base

2.1. CAPTCHA

Es un programa informático diseñado para diferenciar un ser humano de una computadora, CAPTCHA son las siglas de prueba de Turing completamente automática y pública para diferenciar computadoras de humanos (Completely Automated Public Turing test to tell Computers and Humans Apart). Un CAPTCHA es una prueba que es fácil de pasar por un usuario humano pero difícil de pasar por una máquina. Uno de los CAPTCHAs más comunes son imágenes distorsionadas de cadenas cortas de caracteres. Para un humano es generalmente muy fácil de recuperar la cadena original de la imagen distorsionada, pero es difícil para los algoritmos de reconocimiento de caracteres el recuperar la cadena original de la imagen distorsionada.
Un CAPTCHA en un algoritmo aleatorio G, que recibe como parámetro una cadena de caracteres STR y produce como resultado un CAPTCHA G(x)

2.2. PGP (Pretty Good Privacy).

Es un programa desarrollado por Phil Zimmermann y cuya finalidad es proteger la información distribuida a través de Internet mediante el uso de criptografía de clave pública, así como facilitar la autenticación de documentos gracias a firmas digitales.
PGP es un sistema híbrido que combina técnicas de criptografía simétrica y criptografía asimétrica, la velocidad de cifrado del método simétrico y la distribución de la claves del método asimétrico.
Cuando un usuario emplea PGP para cifrar un texto en claro, dicho texto es comprimido. La compresión de los datos ahorra espacio en disco, tiempos de transmisión; después de comprimir el texto, PGP crea una clave de sesión secreta que solo se empleará una vez. Esta clave es un número aleatorio generado a partir de los movimientos del ratón y las teclas que se pulsen; Esta clave de sesión se usa con un algoritmo para cifrar el texto claro; una vez que los datos se encuentran cifrados, la clave de sesión se cifra con la clave pública del receptor y se adjunta al texto cifrado enviándose al receptor.
El descifrado sigue el proceso inverso. El receptor usa su clave privada para recuperar la clave de sesión, simétrica, que PGP luego usa para descifrar los datos. [17]

2.3. Adversarios

Se aplica a la persona o grupo que es rival, competidor o contrario. En criptografía un adversario es un atacante con capacidades especiales que trata de romper la seguridad de un esquema criptográfico.
Clasificadores: Los adversario clasificadores son programas que se dedican a observar los mensajes que se intercambian entre los usuarios de correo electrónico, con el fin de clasificarlos e identificar a todos los usuarios que cumplan con cierto criterio. Esta clasificación se hace de manera masiva y la realiza haciendo una búsqueda de palabras clave dentro de los mensajes de los usuarios. Por ejemplo, el clasificador puede estar interesado en los mensajes que contienen la palabra clave "Bomba", así que todos los mensajes que contengan esta palabra serán etiquetados en una clasificación en específico, este proceso se lleva a cabo por medio de técnicas de "Reconocimiento de patrones" y "Aprendizaje Máquina" para encontrar y clasificar los mensajes que intercepta. [4, 5]
La clasificación de estos mensajes tiene diversos usos, ya que pueden ser clasificados con fines demográficos, con fines comerciales o con fines gubernamentales. Todo esto con el propósito de generar las estadísticas de comportamientos e intereses de los usuarios de correo electrónico.

En este trabajo terminal, se considera el tipo de ataque llamado "ataque de texto cifrado" o COA que hace referencia a su nombre en inglés " Ciphertext-only attack" [16]. Este ataque solo cuenta con los textos cifrados que va recopilando de un canal o base de datos, estos textos cifrados los utiliza para hacer un análisis criptográfico de cómo se comporta la técnica de cifrado y trata de hallar el texto en claro a partir de los textos cifrados que va recopilando. Este tipo de ataques es muy común en el internet aunque con muy baja efectividad cuando se implementa en comunicaciones altamente protegidas, y cuando se implementa en canales de comunicación desprotegidos la información obtenida llega a ser muy pobre. En los últimos años se han dado cuenta que si este tipo de adversarios atacan las comunicaciones sin cifrado se obtienen características valiosas sobre los usuarios que utilizan este tipo de canales de comunicación, este tipo de ataques son ejecutados por adversarios clasificadores.

2.4. Secreto Compartido de Shamir

El secreto compartido, es un método diseñado para compartir un objeto entre un grupo de participantes. Este método fue propuesto por Adi Shamir en 1997.

El objetivo de este método es dividir un secreto K en w partes,que son dadas a w participantes. Para recuperar el secreto es necesario tener al menos u elementos de las w partes siendo $u \leq w$. Y no es posible recuperar el secreto si se tienen menos que u partes.

Para construir el esquema del secreto compartido primero es necesario seleccionar un $p \geq w+1$ el cual define el anillo Z_p.

El procedimiento para dividir un secreto K en w partes es el siguiente:

1. Se seleccionan w elementos distintos de cero del anillo Z_p denotados como x_i donde

$1 \leq i \leq w$.

2. Se seleccionan $u-1$ elementos aleatorios de Z_p denotados como $a_1, ..., a_{u-1}$.

3. Se construye el polinomio y_x de la siguiente forma. Sea

$$y(x) = K + \sum_{j=1}^{u-1} a_j x^j mod \quad p \tag{2.1}$$

   Por medio de este polinomio se calculan los elementos y_i.

4. La salida es el conjunto $S = \{(x_1, y_1), ..., (x_w, y_w)\}$.

Para recuperar el secreto solo tenemos que resolver un sistema de ecuaciones que es definido por el polinomio característico $a(x) = a_0 + a_1 x + ... + a_{u-1} x^{u-1}$.

Posteriormente se seleccionan u pares de elementos (x_w, y_w) con los que obtendremos nuestro sistema de ecuaciones a resolver. El elemento que nos interesa obtener del sistema de ecuaciones es a_0 ya que este es el valor de nuestro secreto K.

2.5. Codificacion de caracteres a enteros

Se tiene un conjunto de caracteres AL compuesto por $AL = \{A, B, ..., Z\} \cup \{a, b, ..., z\} \cup \{0, 1, ..., 9\} \cup \{+, /\}$ con una cardinalidad $|AL| = 64$.

Para obtener una representación binaria de 64 elementos son necesarios 6 bits por lo que para todos los elementos $\sigma \epsilon AL$ existe una representación binaria. Una vez establecido esto el procedimiento para realizar la conversión es el siguiente:

1. Tomamos una cadena de caracteres y la separamos caracter por caracter y los intercambiamos por su correspondiente número entero en AL $\alpha_0 || \alpha_1 || ... || \alpha_m$

2. Posteriormente cada uno de los enteros lo convertimos en un binario de 6 bits y se concatenan uno detrás del otro $\Psi \longleftarrow bin_6(\alpha_0) || bin_6(\alpha_1) || ... || bin_6(\alpha_m$

3. La cadena binaria Ψ la convertimos a entero $v \longleftarrow toInt(\Psi)$

El entero v que obtenemos es el valor entero.

2.6. Decodificacion de enteros a caracteres

Tambien es necesario convertir un entero a una cadena de caracteres y para esto se realiza el proceso inverso:

1. El entero v es convertido en un número binario $z = toBin_6(v)$

2. Separamo z en cadenas de 6 bits y cada una de ellas la interpretamos como un entero $toInt(z_0) || toInt(z_1) || ... || toInt(z_w)$

3. Cada uno de estos valores son convertidos a su correspondiente caracter en AL y concatenados para generar la cadena de caracteres final.

2.7. Antecedentes

La única referencia que se tiene sobre un esquema criptográfico contra adversarios clasificadores es el que encontramos en el artículo "Defending Email Communication Against Profiling" de Philippe Golle y Ayman Farahat, ambos miembros del "Palo Alto Research Center".
En su artículo se aborda el ataque de adversarios clasificadores sobre los mensajes de correo electrónico, en el cual se proponen un protocolo para la comunicación por correo electrónico. Este protocolo hace uso de una función de cifrado que sustituye cada una de las palabras del mensaje por otra de la misma extensión y frecuencia gramatical, esta función esta pensada para textos en idioma ingles. Esta función tiene como parámetro una clave K. Para calcular esta clave se usan los datos de cabecera que acompañan el mensaje los cuales pueden ser dirección del remitente, la dirección del destinatario, la hora a la que se envía el correo electrónico y potencialmente otros campos. Estos datos se introducen en una función hash lenta y el resultado de esta función es la clave K. Estas funciones hash pueden ser conocidas pero con una complejidad de calculo moderadamente mas alta.
Este protocolo resulta inseguro para la criptografía clásica pero es efectivo contra el ataque de clasificadores. Por otro lado este protocolo resuelve dos problemas, le permite a los usuarios calcular la clave de cifrado y descifrado fácilmente ya que los datos del mensaje con que se calcula son públicos, resolviendo así el intercambio de claves. Al usar un cifrado de tipo semántico permite que el texto se vea como una texto en inglés pero indistinguible para los clasificadores y por lo tanto este clasifica incorrectamente el mensaje cifrado.
Tomando en cuenta que esta es la única referencia sobre el tema que se está abordando, se puede decir que se esta trabajando en el mismo sentido, ya que esta propuesta de solución está basada en el artículo "On Securing Communication from Profilers" [4, 5] en el que se propone una solución similar al ataque de clasificadores, pero con la diferencia de que el intercambio de llaves opera por medio del envío y resolución de CAPTCHAS y no por los datos públicos del mensaje.

Capítulo 3

Propuesta de solución

Tomando en cuenta la información ya vertida en este documento, a continuación se explicará detalladamente la propuesta de solución.
En la figura 3.1 se tiene el diagrama general del sistema, se puede apreciar la comunicación entre las diferentes entidades que se usaran, que datos se mandan y reciben y por que canales transitan estos datos. A continuación se describe de manera general como es el proceso de envío y recepción de correos electrónicos ideado para este esquema.

1. Envío
   - El remitente escribe el correo electrónico y le da enviar.
   - El correo electrónico pasa por el complemento del cliente de correo.
   - El cliente genera a partir del correo una clave que usaremos para cifrar el mensaje.
   - Se cifra y se empaqueta el mensaje con el protocolo SMTP.
   - Se coloca una bandera en el mensaje.
   - La clave se convierte en CAPTCHA y es enviada al servidor de CAPTCHAS.
   - Se envía el mensaje de correo electrónico al destinatario.
2. Recepción
   - El receptor abre un correo electrónico cifrado con el presente esquema.
   - El cliente lo descarga del servidor por medio del protocolo POP3 o IMAP.
   - Se hace una petición al servidor de CAPTCHAS para recuperar los CAPTCHAS del correo.

- El usuario resuelve el CAPYCHA y se recalcula la clave de descifrado.
- Se descifra el mensaje y se le muestra al usuario.

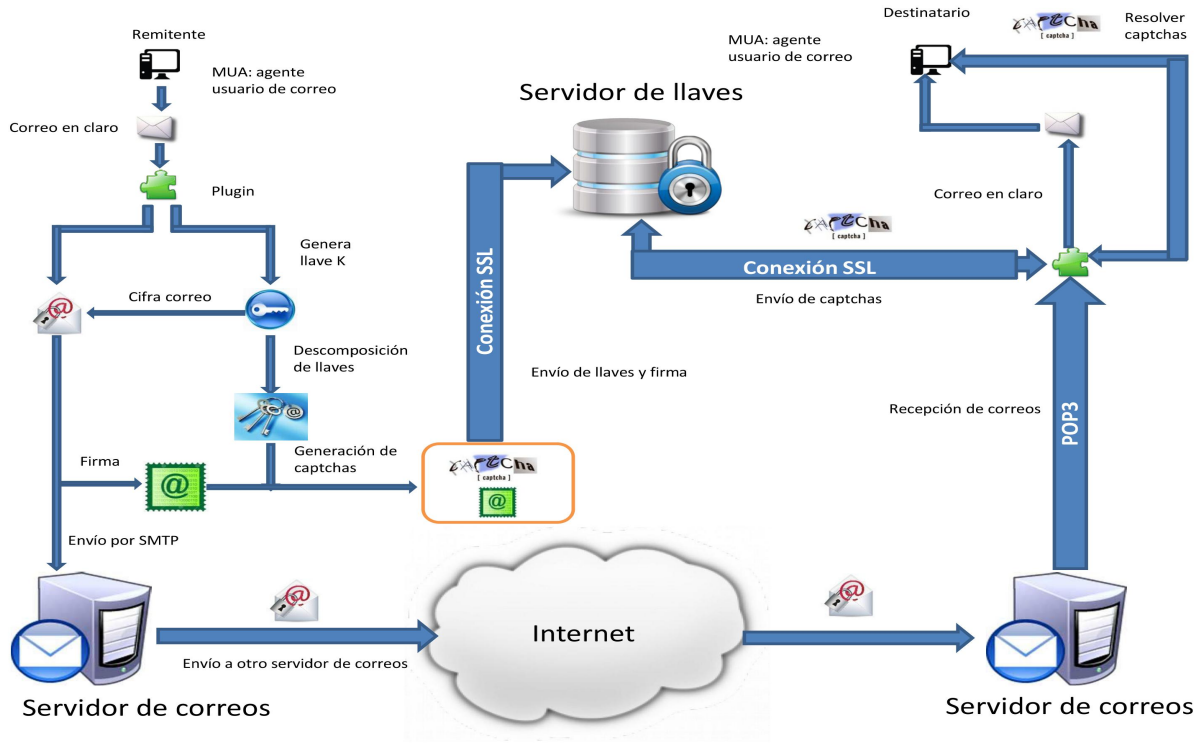

Figura 3.1: Diagrama General del sistema

3.1. Tecnologías

Como ya se ha visto en el esquema anterior se necesita hacer uso de las herramientas adecuadas para poder desarrollar este esquema de cifrado. Las herramientas que se analizaron se describen en las siguientes secciones.

3.1.1. Cliente de correo electrónico

Un cliente de correo electrónico es necesario para el desarrollo de este proyecto ya que en él se instalará un complemento que cifre el mensaje, envíe los CAPTCHAS y descifre los mensajes de correo electrónico. Para ello buscamos un cliente de correo electrónico que cuente con el soporte de los protocolos POP3, SMTP y IMAP; sus licencias son de código libre; soporte la instalación de APIs externas; y tenga soporte en los sistemas operativos ***Windows***, ***IOS*** y ***Linux***. Por lo tanto se investigaron los siguientes clientes de correo electrónico que se encuentra en el mercado:

Cliente de correo electrónico	Sistema Operativo	Protocolos soportados	Código Libre	Agregar funcionalidad	Extra	Gratuita o de paga
eM client	Windows 7, 8 & 10 ; IOS	POP3, SMTP, IMAP, EWS, AirSyn	NO	NO	100 % compatible con gmail y sus APIs	Ambos
Postbox	Windows, IOS	POP3, SMTP, IMAP	NO	SI por medio de APIs	Sincronización con Dropbox, OneDrive, Facebook y Twitter	Ambos
Zimbra	Windows, IOS & Linux	POP3, SMTP, IMAP	SI	SI por medio de APIs	Una plataforma de nivel empresarial y capas se soportar sincronización con múltiples servicios	Ambos
Opera Mail	Windows, IOS & Linux	POP3, SMTP, IMAP	SI	NO	La plataforma para desarrollar en Opera se actualiza cada semana	Gratuito
Thunderbird	Windows, IOS & Linux	POP3, SMTP, IMAP	SI	SI por medio de APIs	Cliente de correo versátil y fácilmente escalable y una comunicad de desarrollo bastante amplia	Gratuito
Nylas N1	Windows, IOS & Linux	POP3, SMTP, IMAP	SI	Si directamente compilando		Gratuito

- El cliente de correo electrónico ***eM client*** tiene una sincronización a 100 % con las cuentas de ***Gmail*** y sus APIs, cuenta con una versión gratuita y una versión de paga; puede hacer migración de mensajes de correo electrónico y contactos de diversos clientes de correo electrónico y tiene una compatibilidad con muchos servidores de correo electrónico. [21]
  Su desventaja es que su código es cerrado y permite agregar funcionalidades.

- El cliente de correo electrónico ***Postbox*** esta soportada en los sistemas operativos ***Windows 7*** o posteriores y ***IOS***, esta aplicación es generada por el servidor de correo electrónico ***Postbox*** por lo tanto cuenta con una sincronización al 100 % con este servidor, también soporta otros servidores de correo como ***Gmail*** y ***Outlook***; este

cliente puede sincronizarse con ***Dropbox***, ***Onedrive*** y redes sociales como ***Facebook***, ***Twitter***, entre otras. Es posible agregar más funcionalidades con la instalación de APIs.
Una desventaja de esta aplicación es que su código es cerrado, pero gracias a que esta basado en código de ***Mozilla*** puedes desarrollar APIs para agregarle tus propias funciones. [22]

- El cliente de correo electrónico ***zimbra*** es la aplicación más completa que se analizó, tiene compatibilidad con el servidor ***zimbra*** pero es capaz de soportar otros servidores de correo electrónico, se encuentra en los 3 sistemas para PC, ***Windows***, ***IOS*** & ***Linux***, da la facilidad de agregarle funcionalidades por medio de la instalación de APIs y gracias a que su código es abierto se pueden programar funciones propias. Este cliente cuenta con la versión gratuita y la versión de paga. Una gran ventaja que tiene es que oferta certificaciones en el desarrollo APIs para este cliente de correo electrónico. [23]
  La única desventaja que se encontro en este cliente de correo es que la plataforma es demasiado grande y el tiempo que se necesita invertir al estudio del código es demasiado y el tiempo de desarrollo de este proyecto es muy corto.

- ***Opera mail*** es un cliente de correo electrónico que salió al mercado recientemente y se puede instalar en los sistemas operativos ***Windows***, ***IOS*** & ***Linux***, es capaz de comunicarse con diversos servidores de correo electrónico y su código es de libre acceso. Su principal desventaja es que las funcionalidades que se quieran agregar no pueden ser adquiridas por medio de la instalación de complementos o APIs. [24]

- Por último tenemos a ***Thunderbird*** que es un cliente de correo electrónico desarrollado por ***Mozilla***, este cliente es de código abierto y la instalación de APIs para agregar funcionalidad es demasiado sencilla; es un cliente de correo que puede ser instalado en los sistemas operativos ***Windows***, ***IOS*** y ***Linux***. [25]

Por lo tanto el cliente de correo electrónico que se usará es ***Thunderbird***, al ser un cliente de correo electrónico casi tan completo como ***zimbra*** pero con la facilidad de desarrollar APIs más rápido.

3.1.2. Lenguajes de programación.

Uno de los objetivos que se tienen en este proyecto es generar un complemento para un cliente de correo electrónico por lo tanto al escoger a ***Thunderbird*** como cliente tenemos que programar con el lenguaje que fue desarrollado para tener la mayor compatibilidad.
Para el desarrollo de este proyecto se utilizará [25]:

- Lenguaje Python
- HTML 5
- CSS3

A pesar de ser una aplicación de escritorio este cliente de correo electrónico está construido con lenguajes web.

3.1.3. Tipos de CAPTCHAS

En el esquema de cifrado es necesario esconder la palabra que genera clave en un CAPTCHA para ser enviado a otro usuario y descifre el mensaje, pero existen varios tipos de CAPTCHAS que se pueden utilizar [26].
Los CAPTCHAS se pueden clasificar de la siguiente forma:

- CAPTCHAS de texto: Este tipo de CAPTCHAS genera una pregunta sencilla donde la respuesta a la pregunta es la respuesta al CAPTCHA.
- CAPTCHAS de imágenes: Este tipo de CAPTCHAS nos muestran en una imagen una cadena de caracteres distorsionados, esta cadena de caracteres es la repuesta del CAPCHA transformada en una imagen.
- CAPTCHAS de audio: Este tipo de CAPTCHAS se caracterizan por ser un audio con ruidos de fondo, donde nos dirán la respuesta oculta.
- CAPTCHAS de video: Este tipo de CAPTCHAS nos muestran un video de unos cuantos segundos donde una palabra aparece alrededor de la pantalla, esta palabra es la respuesta al CAPTCHA.
- CAPTCHAS de acertijos: Este tipo de CPATCHAS es versatil, ya que se trata de pequeños acertijos que resolver donde la respuesta no es una palabra si no una acción o serie de acciones. Los CAPTCHAS de acertijos pueden ser armar un rompecabezas pequeño, seleccionar la imagen que no corresponde, unir figurar geométricas, etc.

Para poder decidir qué tipo de CAPTCHAS se utilizará se consideró los siguientes puntos:

- Facilidad de creación.
- Peso en bytes del CAPTCHA.
- Forma del despliegue.
- Tipo de respuesta.

Por lo tanto se necesita un tipo de CAPTCHA con poco peso, cuya respuesta sea una cadena de caracteres y su forma de despliegue sea fácil de implementar.
Considerando las necesidades anteriores las opciones son CAPTCHAS de imágenes y CAPTCHAS de texto, pero en este proyecto se utilizarán los CAPTCHAS de imágenes porque en ellos se almacenaran las palabras claves de cifrado de los diferentes mensajes de correo electrónico y estas son cadenas de caracteres que no se les puede generar una pregunta coherente.

3.1.4. Bases de datos para almacenar los CAPTCHAS.

Para escoger un gestor de base de datos que controle la información de los usuarios registrados en la aplicación propuesta, la información de los mensajes que envían entre usuarios y los CAPTCHAS asociados a los mensajes para ser descifrados se consideraron 3 características principales en un gestor base de datos relacional:

- Rapidez en transferencias de información.
- Número de usuarios conectados que soporta.
- Facilidad de comunicación entre los lenguajes Python, HTML con los gestores de base de datos.

Los 3 gestores que se analizaron fueron SQLite, MySQL y PostGreSQL.
SQLite es un gestor de base de datos sumamente ligero que soporta hasta 2 terabytes de información, esta base de datos es compatible con python y lenguajes de programación web. Este gestor por su misma ligereza es posible implementarla en dispositivos móviles, pero no es recomendable cuando múltiples usuarios están haciendo peticiones al mismo tiempo ya que su rendimiento baja [27].
MySQL es un gestor de base de datos que se caracteriza por su transferencia al hacer consultas de datos almacenados; es uno de los gestores libres más utilizados en aplicaciones web y cuenta con diferentes APIs para hacer la comunicación con los lenguajes Python, PHP, C++, etc. Y soporta peticiones de múltiples usuarios gracias a la implementación de hilos.
PostGreSQL es un gestor de base de datos que puede hacer operaciones complejas como subconsultas, transacciones y rollbacks's. Soporta las peticiones de múltiples usuarios pero en cuanto a la velocidad de transferencia de información, comparado con MySQL, es muy lenta pero lo compensa manteniendo una velocidad casi invariable a pesar de que la base se mantenga con un número grande de registros. [28]
Se eligió el gestor de base de datos MySQL porque el proyecto necesita mayor rapidez en la transferencia de información más que generar filtros muy complejos para la búsqueda de información.

Capítulo 4

Análisis y Diseño

4.1. Diagramas de caso de uso

Para el desarrollo de esta propuesta se muestran los siguientes diagramas del sistema.

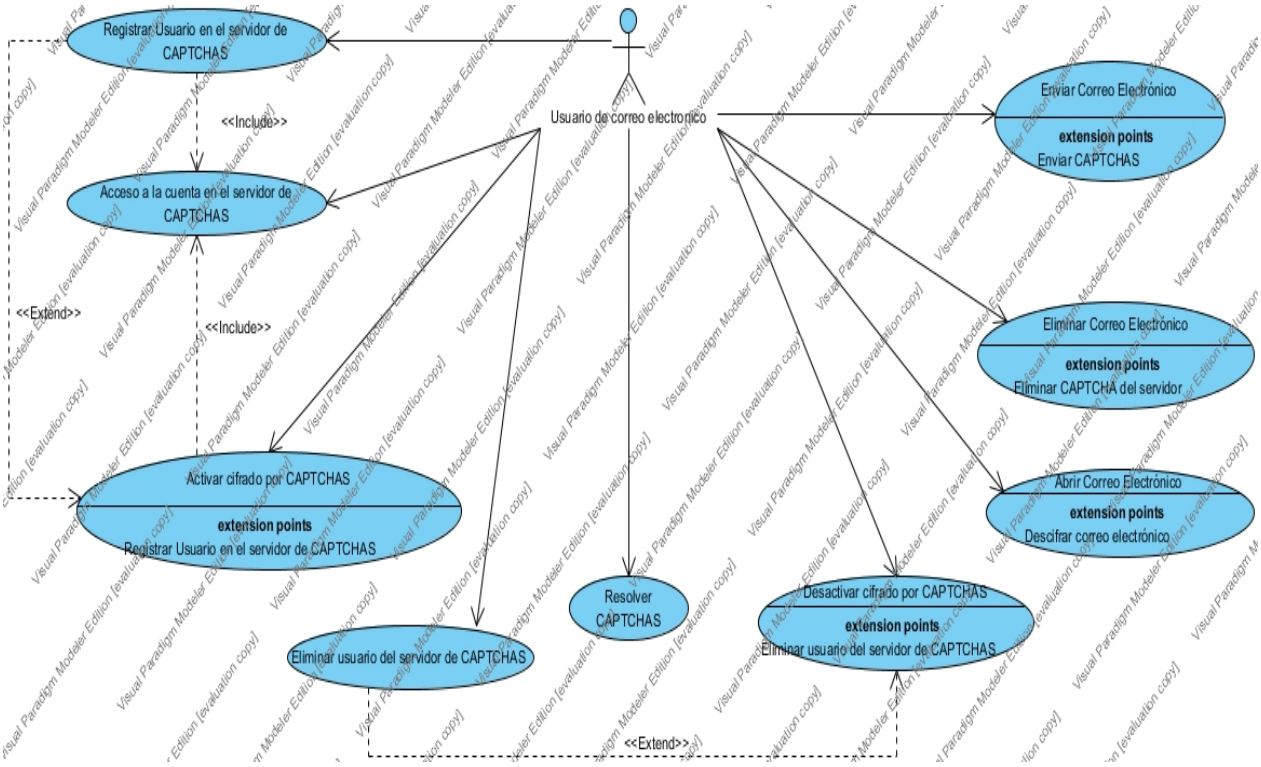

Figura 4.1: Diagrama General de caso de uso

4.1.1. Diagrama de casos de uso CU2 Registrar usuario en el servidor de CAPTCHAS

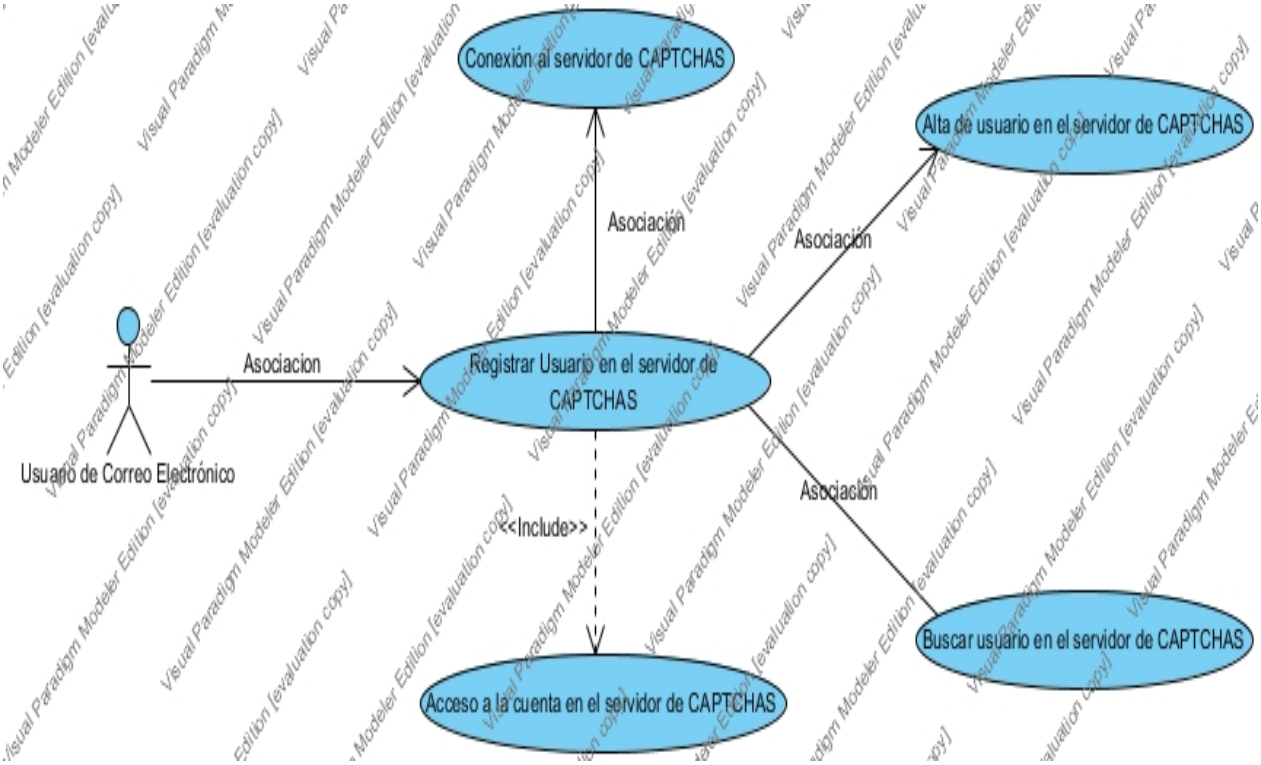

Figura 4.2: Diagrama de casos de uso CU2 Registrar usuario en el servidor de CAPTCHAS

Caso de Uso	CU2 Registrar Usuario en el servidor de CAPTCHAS		
Actor	Actor1. Usuario de Correo Electrónico		
Descripción	Describe los pasos necesarios para registrar un nuevo usuario en el servidor de CAPTCHAS.		
Pre-condiciones	Tener una cuenta de correo electrónico.		
Post-condiciones	Activación del módulo de cifrado por CAPTCHAS.		
Puntos de inclusión	Acceso a la cuenta en el servidor de CAPTCHAS.		
Puntos de extensión			
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	El usuario selecciona la opción registrarse en el servidor de CAPTCHAS
	2	Sistema	El cliente de correo contesta un formulario con la información necesaria para dar de alta en el servidor de CAPTCHAS.
	3	Actor	Completa el formulario y oprime el botón de registrar.

	4	Sistema	El sistema valida los datos proporcionados por el usuario.
	5	Sistema	Se conecta con el servidor y valida si el usuario ya está registrado. <FA01 - Usuario ya registrado><FA02 - Falla en la conexión con el servidor>
	6	Sistema	Manda la información del usuario y lo da de alta.
	7	Sistema	Despliega el siguiente mensaje "El usuario se ha dado de alta correctamente"
			Fin del flujo principal.
	FA01 - Usuario ya registrado.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Despliega el siguiente mensaje "El usuario ya está registrado favor de proporcionar otra cuenta de correo electrónico"
	2		El flujo continúa en el paso 3 del flujo principal.
			Fin del flujo alternativo
	FA02 - Falla en la conexión con el servidor.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Despliega el siguiente mensaje "La conexión con la red es nula o limitada, favor de realizar esta operación más tarde"
	2		El flujo continúa en el paso 1 del flujo principal.
			Fin del flujo alternativo

Tabla 4.1: Descripción CU2.

4.1.2. Diagrama de casos de uso CU3 Acceso a la cuenta en el servidor de CAPTCHAS

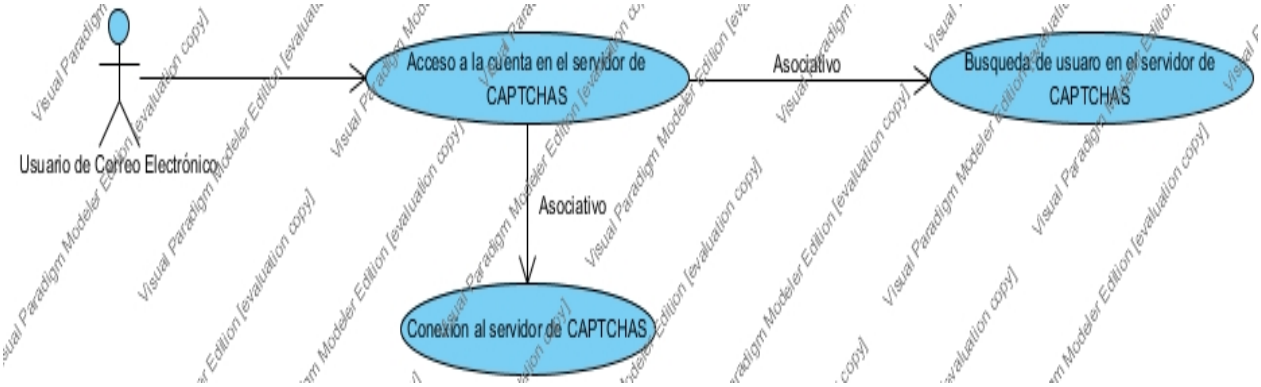

Figura 4.3: Diagrama de casos de uso CU3 Acceso a la cuenta en el servidor de CAPTCHAS

4.1.3. Diagrama de casos de uso CU4 Abrir Correo Electrónico.

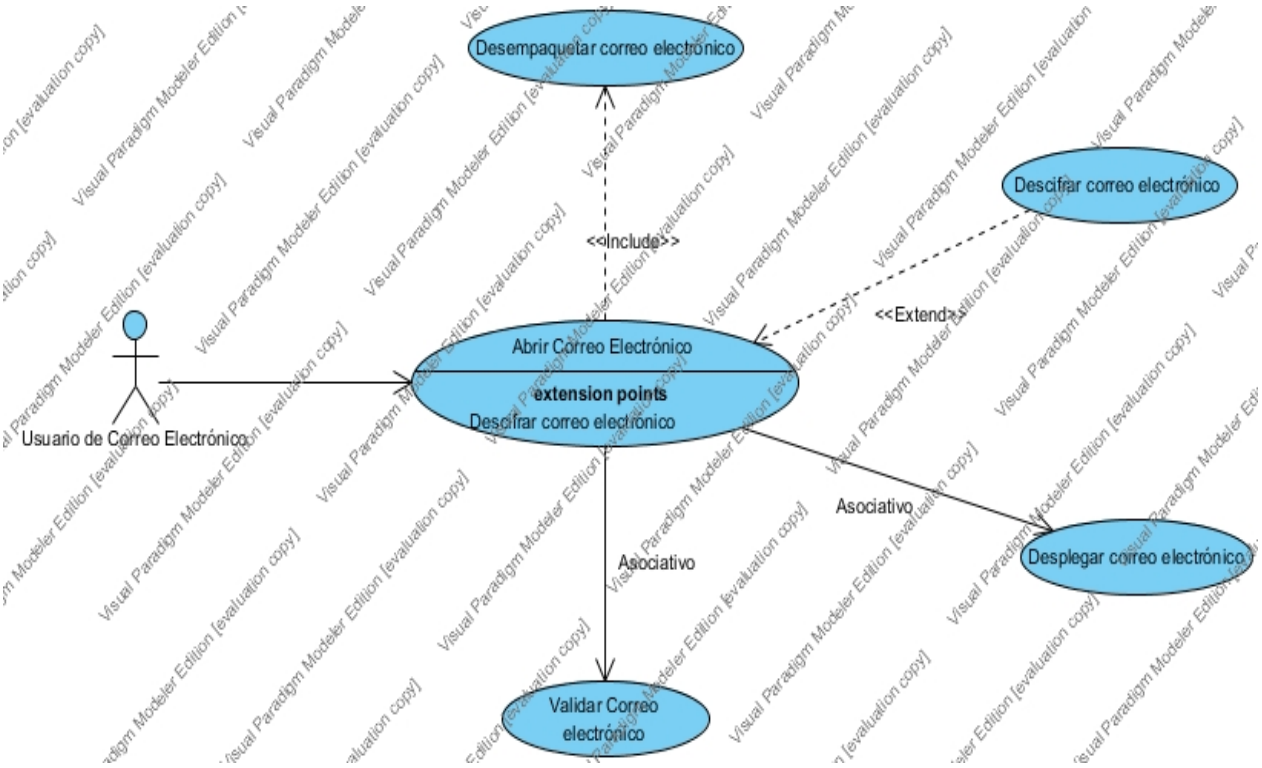

Figura 4.4: Diagrama de casos de uso CU4 Abrir Correo Electrónico.

Caso de Uso	CU4 Abrir correo electrónico		
Actor	Actor 1. Usuario de correo electrónico		
Descripción	Describe los pasos necesarios para abrir un mensaje de correo electrónico.		
Pre-condiciones	1. Iniciar sesión con su servidor de correo electrónico. 2. Descargar el correo electrónico que se desea abrir.		
Post-condiciones	Despliegue del mensaje de correo electrónico descifrado.		
Puntos de inclusión	Desempaquetar correo electrónico		
Puntos de extensión	Descifrar correo electrónico		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	El caso de uso comienza cuando el usuario selecciona el correo que desea abrir.
	2	Sistema	El sistema manda a llamar a la función de desempaquetar correo electrónico.
	3	Sistema	Valida si el mensaje viene timbrado. <FA01 - El mensaje no viene timbrado>
	4	Sistema	Invoca al caso de uso <CU Descifrar correo electrónico>
	5	Sistema	Recibe el mensaje de correo electrónico descifrado
	6	Sistema	Despliega el contenido completo del mensaje al usuario
			Fin del flujo principal.
	FA01 - El mensaje no viene timbrado.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	El flujo continúa en el paso 6 del flujo principal.
			Fin del flujo alternativo

Tabla 4.2: Descripción CU4.

4.1.4. Diagrama casos de uso CU5 Activar cifrado por CAPTCHAS.

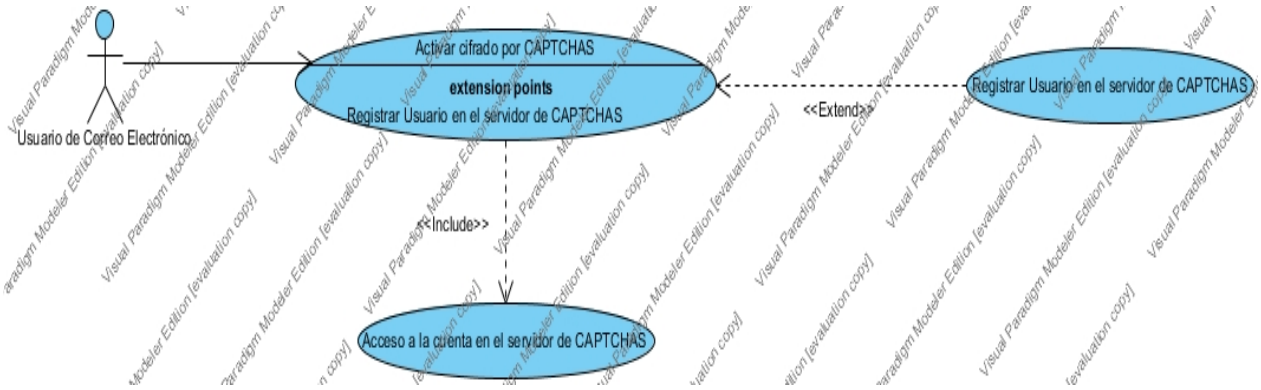

Figura 4.5: Diagrama casos de uso CU5 Activar cifrado por CAPTCHAS.

Caso de Uso	CU5 Activar cifrado por CAPTCHAS		
Actor	Actor 1. Usuario de correo electrónico		
Descripción	Describe los pasos necesarios para activar el módulo de cifrado CAPTCHAS en el cliente de correo electrónico.		
Pre-condiciones	1. Instalar el módulo de cifrado por CAPTCHAS		
Post-condiciones	Activación del cifrado y descifrado por CAPTCHAS.		
Puntos de inclusión			
Puntos de extensión	Registrar usuario del servidor de CAPTCHAS		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	El caso de uso inicia cuando el actor seleccionar la opción “Activar cifrado por CAPTCHAS”
	2	Sistema	El sistema verifica si la dirección de correo del usuario está registrada en el servidor de CAPTCHAS<FA01 -Usuario no registrado en el servidor>
	3	Sistema	Despliega una ventana con el mensaje “Activación del módulo de cifrado por CAPTCHAS”
			Fin del flujo principal.
	FA01 -Usuario no registrado en el servidor.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	El sistema despliega una ventana con las opciones de “Registrarse” y “Cancelar”. <FA02 Cancelar activación>

	2	Actor	Oprime el botón de “Registrarse”
	3	Sistema	El sistema invoca al caso de uso <CU Registrar usuario en el servidor de CAPTCHAS>
	4	Sistema	El sistema obtiene una respuesta satisfactoria del registro
	5		El flujo continúa en el paso 2 del flujo principal.
			Fin del flujo alternativo
	FA02 - Cancelar activación.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Actor	El Actor selecciona “Cancelar”
	2	Sistema	Cierra la ventana de selección
	3		El flujo continúa en el paso 1 del flujo principal.
			Fin del flujo alternativo

Tabla 4.3: Descripción CU5.

4.1.5. Diagrama de casos de uso CU6 Descifrar correo electrónico.

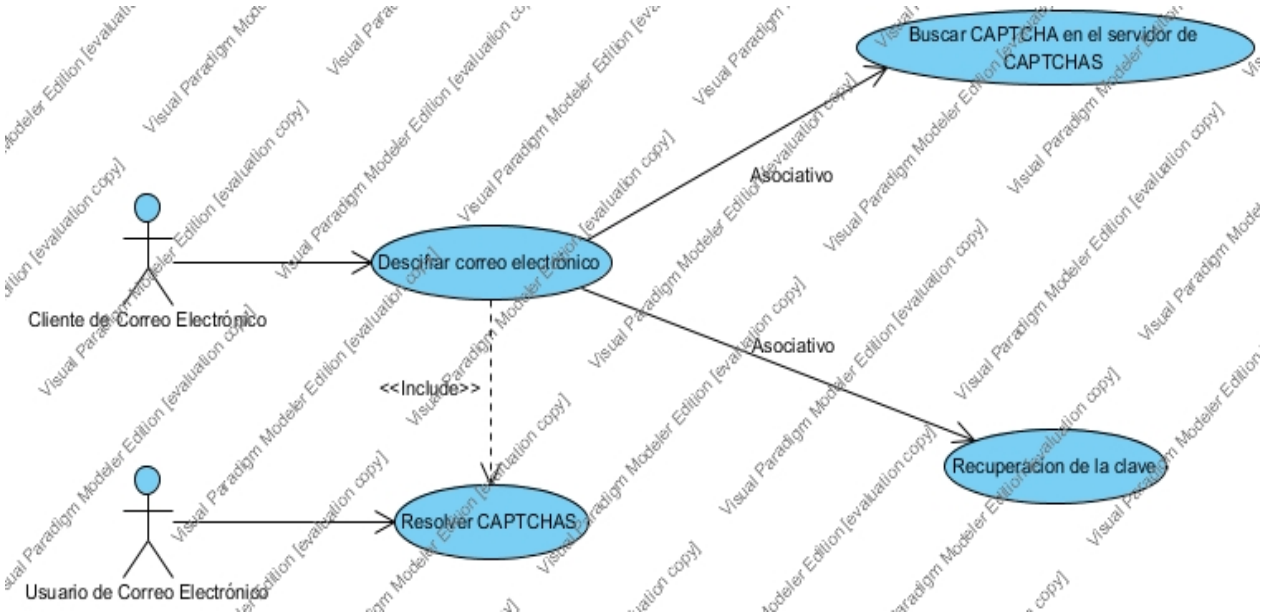

Figura 4.6: Diagrama de casos de uso CU6 Descifrar correo electrónico.

Caso de Uso	CU6 Descifrar correo electrónico.		
Actor	Actor 1. Usuario de correo electrónico		
Descripción	Describe los pasos necesarios para desactivar el módulo de cifrado CAPTCHAS en el cliente de correo electrónico		
Pre-condiciones	1. Activar cifrado por CAPTCHAS. 2. Registrar usuario en el servidor de CAPTCHAS		
Post-condiciones	Desactivación del cifrado y descifrado por CAPTCHAS.		
Puntos de inclusión			
Puntos de extensión	Eliminar usuario del servidor de CAPTCHAS		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	El caso de uso inicia cuando el actor seleccionar la opción "Desactivar cifrado por CAPTCHAS"
	2	Sistema	El sistema despliega la venta con las opciones de “Desactivar cifrado” y “Eliminar usuario” <FA01 - Eliminar usuario>
	3	Actor	Selecciona la Desactivación del cifrado por CAPTCHAS
	4	Sistema	El sistema desactiva el módulo de cifrado por CAPTCHA

			Fin del flujo principal.
	FA01 - Eliminar usuario.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Actor	El Actor selecciona "Eliminar usuario"
	2	Sistema	El sistema despliega una ventana con las opciones de "Aceptar" y "Cancelar" para confirmar la eliminación del usuario. <FA02 - Cancelar acción eliminar usuario>
	3	Actor	Oprime el botón de "Aceptar"
	4	Sistema	Establece la conexión con el servidor de CAPTCHAS <FA03 - Fallo en la conexión con el servidor>
	5	Sistema	Busca y elimina al usuario de la base de datos desplegando la confirmación del servidor.
	6	Actor	Oprime el botón de "Aceptar"
	7	Sistema	Desactiva el módulo de cifrado por CAPTCHA
			Fin del flujo alternativo
	FA02 - Cancelar acción eliminar usuario.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Actor	El Actor selecciona "Cancelar"
	2	Sistema	Cierra la ventana de confirmación
			Fin del flujo alternativo
	FA03 - Fallo en la conexión con el servidor.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Despliega una ventana de alerta con el mensaje "No se ha podido establecer la conexión con el servidor, es probable que no se tenga conexión a internet. Favor de intentarlo más tarde"
	2	Actor	Cierra la ventana de alerta
	3		El flujo continúa en el paso 1 del flujo principal

			Fin del flujo alternativo

Tabla 4.4: Descripción CU6.

4.1.6. Diagrama de caos de uso CU7 Eliminar CAPTCHA del servidor.

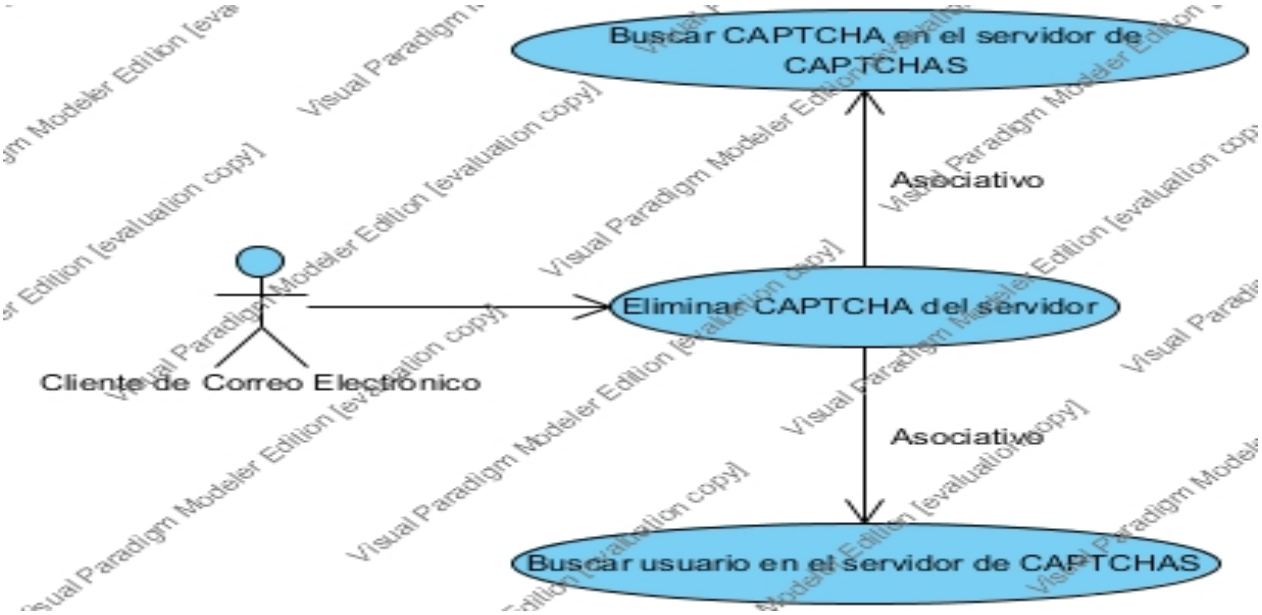

Figura 4.7: Diagrama de caos de uso CU7 Eliminar CAPTCHA del servidor.

Caso de Uso	CU7 Eliminar CAPTCHA del servidor.		
Actor	Actor 1. Cliente de correo electrónico.		
Descripción	Describe los pasos necesarios para eliminar los CAPTCHAS del servidor de CAPTCHAS.		
Pre-condiciones	1. Solicitar eliminar un mensaje de correo electrónico		
Post-condiciones			
Puntos de inclusión			
Puntos de extensión			
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	Solicita eliminar CAPTCHA del servidor de CAPTCHAS
	2	Sistema	Busca al usuario y el CAPTCHA a eliminar en el servidor de CAPTCHAS
	3	Sistema	Elimina el CAPTCHA solicitado
	4	Sistema	Regresa la confirmación de que se eliminó el CAPTCHA.
			Fin del flujo principal.

Tabla 4.5: Descripción CU7.

4.1.7. Diagrama de casos de uso CU8 Eliminar correo electrónico.

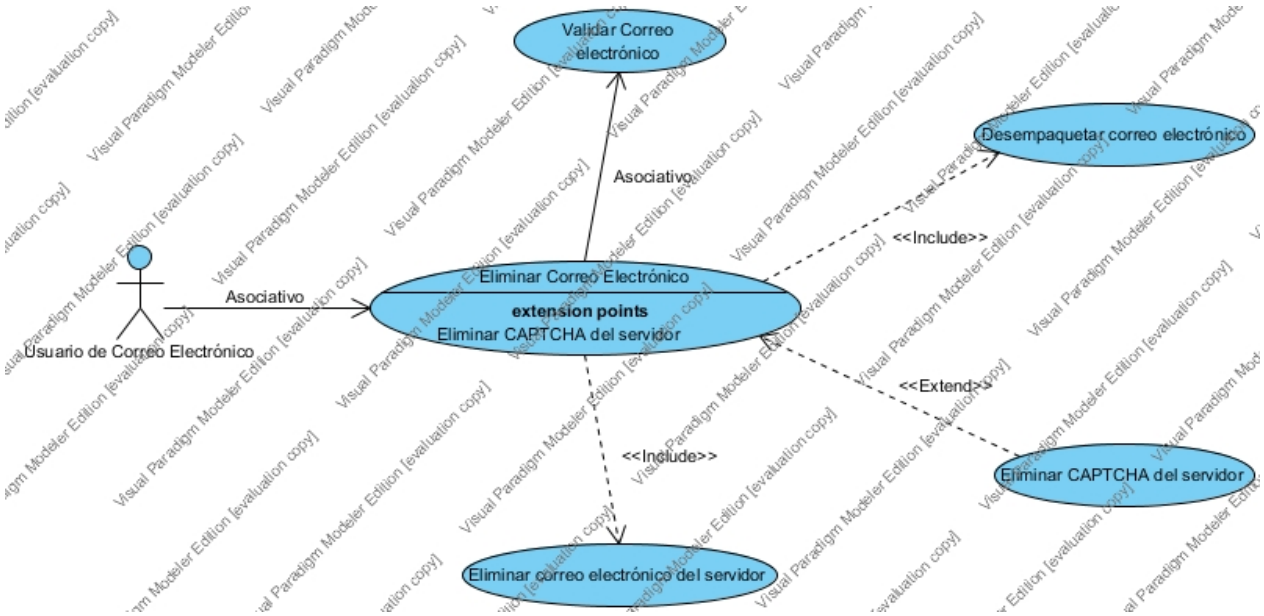

Figura 4.8: Diagrama de casos de uso CU8 Eliminar correo electrónico.

Caso de Uso	CU8 Eliminar correo electrónico.		
Actor	Actor 1. Usuario de correo electrónico.		
Descripción	Describe los pasos necesarios para eliminar un mensaje de correo electrónico.		
Pre-condiciones	1. Seleccionar un mensaje de correo electrónico		
Post-condiciones	Mensaje y CAPTCHA eliminados.		
Puntos de inclusión	1. Desempaquetar correo electrónico. 2. Eliminar correo electrónico del servidor		
Puntos de extensión	Eliminar CAPTCHA del servidor de CAPTCHAS		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	Selecciona un mensaje de correo electrónico a eliminar.
	2	Sistema	Desempaqueta el mensaje de correo electrónico
	3	Sistema	Valida si el mensaje esta timbrado. <FA01 - Mensaje no timbrado>
	4	Sistema	Invoca al caso de uso <CU Eliminar CAPTCHA del servidor>
	5	Sistema	Elimina el mensaje de correo electrónico y despliega el mensaje "El correo se ha eliminado correctamente"
			Fin del flujo principal.
	FA01 - Mensaje no timbrado.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	El sistema continúa a partir del paso 5 del flujo principal.
			Fin del flujo alternativo

Tabla 4.6: Descripción CU8.

4.1.8. Diagrama de casos de uso CU9 Enviar CAPTCHAS

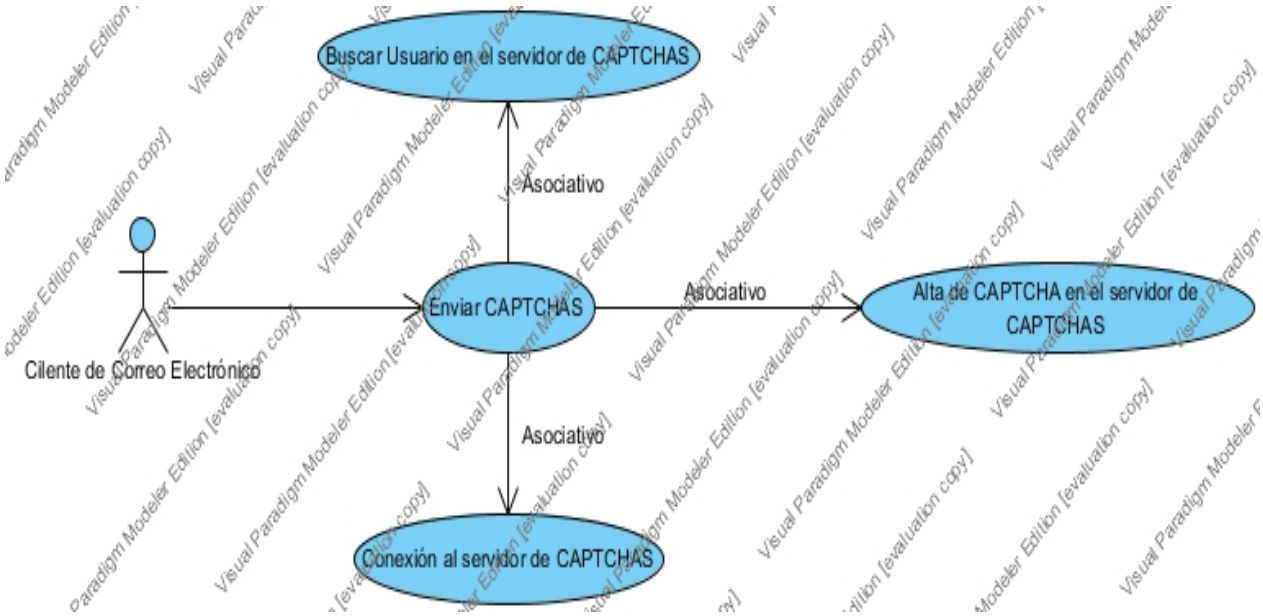

Figura 4.9: Diagrama de casos de uso CU9 Enviar CAPTCHAS

Caso de Uso	CU9 Enviar CAPTCHAS		
Actor	Actor 1. Cliente de correo electrónico.		
Descripción	Describe los pasos necesarios para enviar el CAPTCHA el servidor de CAPTCHAS		
Pre-condiciones	1. Solicitar el envió de un nuevo mensaje de correo electrónico.		
Post-condiciones	Envío del CAPTCHA al servidor de CAPTCHAS		
Puntos de inclusión			
Puntos de extensión			
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	Solicita el envío de CAPTCHA al servidor
	2	Sistema	Abre la conexión y busca al usuario en el servidor de CAPTCHAS
	3	Sistema	Da de alta el CAPTCHA en el servidor asociándolo con el usuario.
	4	Sistema	Regresa la confirmación de que se dio de alta el CAPTCHA
			Fin del flujo principal.

Tabla 4.7: Descripción CU9.

4.1.9. Diagrama de casos de uso CU10 Enviar correo electrónico.

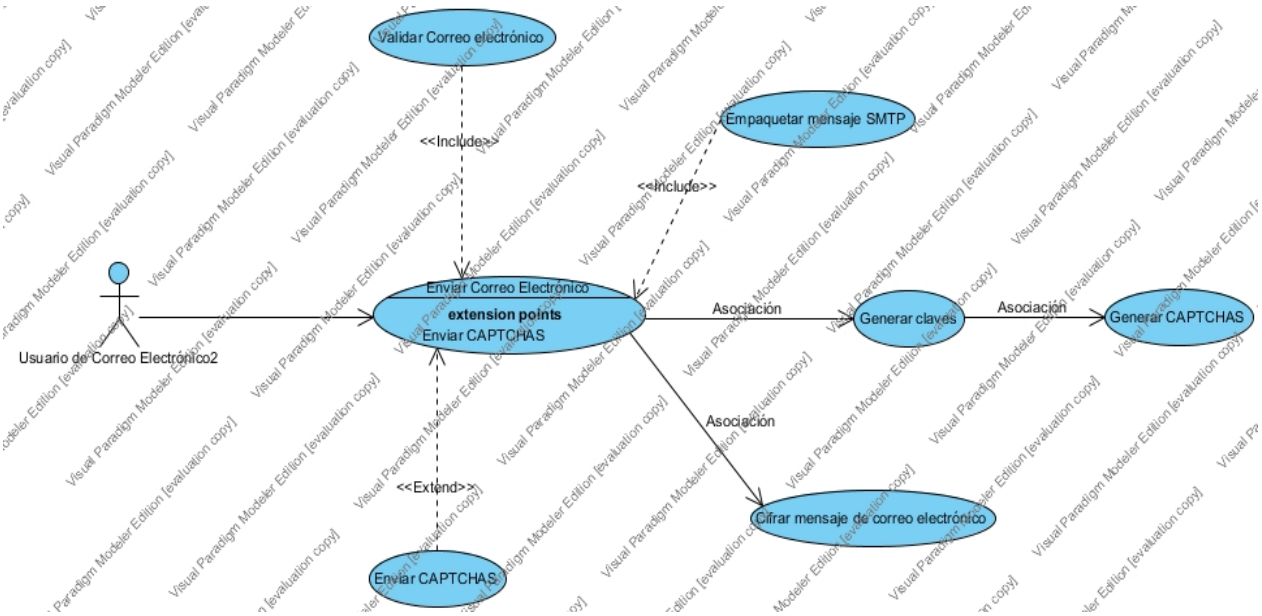

Figura 4.10: Diagrama de casos de uso CU10 Enviar correo electrónico.

Caso de Uso	CU10 Enviar correo electrónico.		
Actor	Actor 1. Usuario de correo electrónico.		
Descripción	Describe los pasos necesarios para enviar un mensaje de correo electrónico cifrado a otro usuario de correo electrónico.		
Pre-condiciones	1. El usuario tiene que redactar un mensaje de correo electrónico que contenga la dirección del destinatario.		
Post-condiciones	Envió de un mensaje cifrado al servidor de correo electrónico y el registro del CAPTCHA en el servidor de CAPTCHAS.		
Puntos de inclusión	1. Validar correo electrónico. 2. Empaquetar mensaje de correo electrónico SMTP.		
Puntos de extensión	Enviar CAPTCHA		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	Oprime el botón "Enviar"
	2	Sistema	Valida que el mensaje de correo electrónico contenga los datos mínimos.<FA01 - Campos no completados>
	3	Sistema	Genera una llave de cifrado
	4	Sistema	Con una palabra aleatoria se genera el CAPTCHA y cifra el mensaje de correo electrónico.
	5	Sistema	Toma el mensaje cifrado y es empaquetado para enviarse al servidor de correo electrónico
	6	Sistema	Toma el CAPTCHA y se envía al caso de uso <CU Enviar CAPTCHA>
	7	Sistema	Despliega el mensaje de "envío satisfactorio"
			Fin del flujo principal.
	FA01 - Campos no completados.		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Notifica al usuario cuales campos han sido mal proporcionados, para poder enviar el mensaje correctamente
	2	Actor	Modifica los campos solicitados
	3		El flujo continúa en el paso 1 del flujo principal
			Fin del flujo alternativo

Tabla 4.8: Descripción CU10.

4.2. Diagramas a bloques

A continuación se presentan los diagramas a bloques, en donde se muestra cuál es la secuencia de procesos a realizar. Esto servirá para comprender cómo se comunican los diferentes módulos de manera interna, y cómo hacen los procesos.

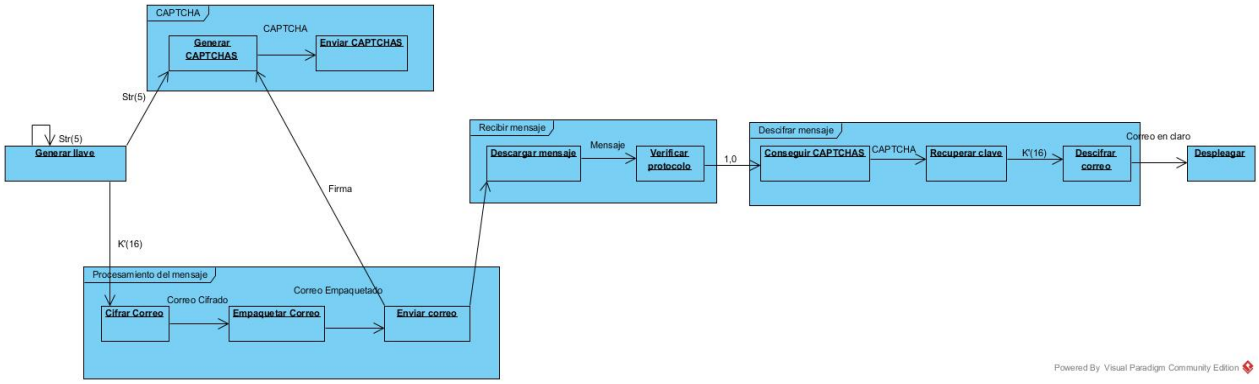

Figura 4.11: Diagrama a bloque 0 general del sistema

	Generar clave	**Generar CAPTCHA**	**Procesamiento del mensaje**	**Recibir mensaje**	**Descifrar mensaje**	**Desplegar**
Entradas	*Señal de activación	*Cadena de 5 caracteres: Str(5)	*Clave de 16 bytes: K'(16). *Mensaje de correo.	*Correo Cifrado	Verificación (1,0)	*Correo en claro
Salidas	*Cadena de 5 caracteres: Str(5) *Clave de 16 bytes: K'(16)	*Señal de envió	*Correo Cifrado	*Verificación (1,0)	*Correo en claro	
Descripción	Se activa el proceso generar clave, este crea una palabra de 5 caracteres (Str(5)), procesa la palabra Str(5) por medio de una función hash obteniendo una palabra de 256 caracteres (K(256)) y recorta esta clave a una palabra de 16 caracteres (K'(16)).	Toma la entrada Str(5) y la convierte en una imagen CAPTCHA, Posteriormente inicia una conexión con el servidor de CAPTCHAS para mandarlo a este.	Cifra el mensaje de correo con la clave K'(16), posteriormente lo firma y genera un timbre para saber que fue creado con este esquema y lo empaqueta para su envió.	El cliente hace una petición al servidor y descarga el mensaje de correo electrónico, lo des empaqueta verifica la firma y el timbrado para saber de quién viene y si está cifrado bajo este esquema.	Se hace una petición al servidor de CAPTCHAS, se descargan los CAPTCHAS asociados al correo, ya con el CAPTCHA este se resuelve y se recupera la cadena Str(5), esta se pasa por una función hash y se recupera K(256), esta se corta a K'(16), con esto se descifra el mensaje.	Se muestra el correo descifrado en la interfase del cliente de correo electrónico.

Tabla 4.9: Diagrama a bloques 0 general

Figura 4.12: Diagrama a bloques 1 Generar clave

	Generar Str(5)	**Aplicar Función Hash**	**Recortar Hash K'**
Entradas	*Llamada a Función	*Cadena de 5 caracteres: Str(5)	*Digesto K(128)
Salidas	*Cadena de 5 caracteres: Str(5)	*Digesto K(128)	*K'(16)
Descripción	Toma una función random módulo 67, para formar una palabra con 5 caracteres aleatorios tomados del siguiente conjunto.Anillo67- .,+*[a-z][A-Z]	Se pasa la cadena Str(5) por una función hash SHA-1 para obtener un digesto único de esta palabra.	Se copian a otro string lo primeros 16 caracteres del digesto K(128) para formar la clave K'(16)

Tabla 4.10: Diagrama a bloques 1 general clave

	Cifrar
Entradas	*Clave K'(16) *Mensaje de correo
Salidas	*Correo cifrado
Descripción	Se cifra el mensaje con un algoritmo de llave simétrica (AES o DES) usando una llave de 16bytes o 128bits.

Tabla 4.11: Diagrama a bloques 2 Cifrar Correo

Figura 4.13: Diagrama a bloques 3 Empaquetar Correo

	Empaquetamiento SMTP	**Timbrar Correo**
Entradas	*Mensaje Cifrado	*Correo Empaqueta
Salidas	*Correo Empaquetado	*Correo Timbrado
Descripción	Se toma el correo y se integra en el formato del correo marcado en el RFC822	Se timbra el mensaje colocando una marca después del final del mensaje. Para señalar que el correo enviado está cifrado bajo este protocolo.

Tabla 4.12: Diagrama a bloques 3 Empaquetar Correo

Figura 4.14: Diagrama a bloques 4 Enviar correo

	Abrir conexión SMTP	**Envió de Correo por SMTP**
Entradas	*Petición	*Correo empaquetado
Salidas	*Canal de comunicación	*Confirmación de envió
Descripción	Se genera una petición para conexión SMTP	Se manda el correo electrónico al servidor por medio del protocolo SMTP

Tabla 4.13: Diagrama a bloques 4 Enviar correo

Figura 4.15: Diagrama a bloques 5 Generar CAPTCHA

	Generar respuesta del CAPTCHA	**Firmar CAPTCHA**	**Transformar palabra**
Entradas	*Cadena de caracteres: Str(5)	*Firma	*Señal de confirmación
Salidas	*Señal de confirmación	*Archivo Firmado	*Imagen CAPTCHA
Descripción	Genera un archivo con la respuesta del CAPTCHA	Se firma el CAPTCHA por medio de un Hashing del mensaje.	Convierte el Str(5) en una imagen distorsionada que llamaremos CAPTCHA

Tabla 4.14: Diagrama a bloques 5 Generar CAPTCHA

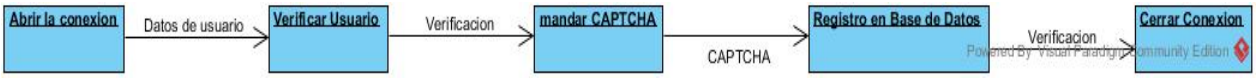

Figura 4.16: Diagrama a bloques 6 Enviar CAPTCHAS (Usuario existente)

	Abrir conexión	**Verificar Usuario**	**Mandar CAPTCHA**	**Registrar en base de datos**	**Cerrar Conexión**
Entradas	*CAPTCHAS	*Datos de Usuario	*Verificación de usuario	*Datos de usuario *CAPTCHA	Verificación
Salidas	*Datos de usuario	*Verificación de usuario	*CAPTCHA	Verificación	
Descripción	Se genera una petición para poder entablar una conexión con el servidor de CAPTCHAS	Se verifica la existencia del usuario en el servidor, si existe se le da acceso	Ya verificado el usuario se manda el CAPTCHA al servidor	Se registran los datos del CAPTCHA en la base de datos y se envía una verificación	Se cierra la conexión y se guardan los datos

Tabla 4.15: Diagrama a bloques 6 Enviar CAPTCHAS (Usuario existente)

Figura 4.17: Diagrama a bloques 7 Enviar CAPTCHAS (Usuario inexistente)

	Abrir conexión	**Registrar usuario en Base de Datos**	**Verificar Usuario**	**Mandar CAPTCHA**	**Registrar en base de datos**	**Cerrar Conexión**
Entradas	*CAPTCHAS	*Datos de Usuario	*Verificación de registro	*Verificación de usuario	*Datos de usuario *CAPTCHA	Verificación
Salidas	*Datos de usuario	*Verificación de registro	*Verificación de usuario	*CAPTCHA	Verificación	
Descripción	Se genera una petición para poder entablar una conexión con el servidor de CAPTCHAS	Se da de alta al usuario en la base de datos	se le da acceso al usuario	Ya verificado el usuario se manda el CAPTCHA al servidor	Se registran los datos del CAPTCHA en la base de datos y se envía una verificación	Se cierra la conexión y se guardan los datos

Tabla 4.16: Diagrama a bloques 7 Enviar CAPTCHAS (Usuario inexistente)

Figura 4.18: Diagrama a bloques 8 Descargar mensaje

	Abrir conexión al servidor de correo	**Descargar mensaje por IMAP o POP3**
Entradas	*Señal de activación	*Confirmación
Salidas	*Confirmación	*Correo electrónico
Descripción	El receptor se conecta al servidor de correo electrónico e inicia la sesión	Descarga del servidor de correo electrónico todos los mensajes que aún no se hayan descargado.

Tabla 4.17: Diagrama a bloques 8 Descargar mensaje

Figura 4.19: Diagrama a bloques 9 Verificar protocolo (con protocolo válido)

	Abrir mensaje	**Verificar mensaje**
Entradas	*Correo electrónico	*mensaje
Salidas	*Mensaje	*verificación
Descripción	Se toma el mensaje descargado del servidor y se des empaqueta para dejar solo el texto del mensaje	Se verifica que el mensaje tenga la bandera correspondiente a que está cifrado con este esquema

Tabla 4.18: Diagrama a bloques 9 Verificar protocolo (con protocolo válido)

	Abrir mensaje	Verificar mensaje
Entradas	*Correo electrónico	*mensaje
Salidas	*Mensaje	*verificación
Descripción	Se toma el mensaje descargado del servidor y se des empaqueta para dejar solo el texto del mensaje	Si la verificación es negativa se manda directamente al bloque de Despliegue

Tabla 4.19: Diagrama a bloques 10 Verificar protocolo (con protocolo inválido)

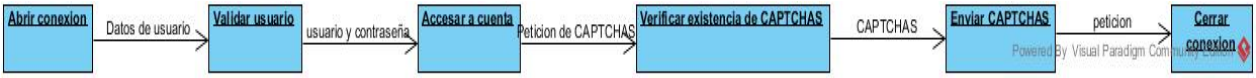

Figura 4.20: Diagrama a bloques 11 Conseguir CAPTCHAS (Usuario existente)

	Abrir Conexión	Validar Usuario	Accesar a cuenta	Verificar existencia de CAPTCHA	Enviar CAPTCHA
Entradas	*Confirmación	*Datos usuario	*Contraseña	*Petición de CAPTCHAS	*Confirmación
Salidas	*verificación	*Contraseña	*confirmación	*confirmación	*CAPTCHA
Descripción	Se abre una conexión con el servidor de CAPTCHAS	Se verifica que el usuario este dado de alta en el servidor mandándole una petición a la base de datos, si el usuario existe se accesa	Si esta dado de alta en el servidor se manda la contraseña para que pueda tener acceso a los CAPTCHAS de su cuenta	Se verifica que los CAPTCHAS que están ligados al mensaje que realizo la petición existan	Si existen estos CAPTCHAS son enviados de regreso al mensaje

Tabla 4.20: Diagrama a bloques 11 Conseguir CAPTCHAS (Usuario existente)

Figura 4.21: Diagrama a bloques 12 Recuperar clave

	Resolver CAPTCHA	**Aplicar Función Hash**	**Recortar llave**
Entradas	*CAPTCHA	*Cadena de 5 caracteres: Str(5)	*Digesto K(128)
Salidas	*Str(5)	*Digesto K(128)	*K'(16)
Descripción	Se despliega el CAPTCHA para que el usuario pueda resolverlo	Se pasa la cadena Str(5) por una función hash SHA-1 para obtener un digesto único de esta palabra	Se copian a otro string lo primeros 16 caracteres del digesto K(128) para formar la clave K'(16)

Tabla 4.21: Diagrama a bloques 12 Recuperar clave

	Descifrar
Entradas	*Clave K'(16) *Mensaje de correo
Salidas	*Correo descifrado
Descripción	Se descifra el mensaje con un algoritmo de llave simétrica (AES o DES) usando una llave de 16bytes o 128bits.

Tabla 4.22: Diagrama a bloques 13 Descifrar correo

Capítulo 5

Desarrollo de prototipos

5.1. Prototipo 1

Objetivo del prototipo: Conocer el uso, funcionamiento e implementación de herramientas de cifrado, hashing y generación de CAPTCHAS, con el fin de conocer la integración de estos módulos en diferentes lenguajes de programación.
Se implementó un módulo de cifrado de mensajes de texto en lenguaje C++. Tratando de simular el proceso de cifrado del esquema que se esta usando.
La primera parte del proceso es abrir el mensaje para lo cual se estan usando los métodos estándar definidos en las bibliotecas nativas de C++, posteriormente se generará una palabra aleatoria de 5 caracteres usando una función Rand() %100 y transformando el valor de salida a un char.
Al resultado se pasa por una función de hashing, esta función no es nativa de ninguna biblioteca estándar de C++ ni de C, por lo que se tuvo que conseguir una en internet y probar que efectivamente funcionara como se necesita.
Posteriormente este hash se usará como clave para cifrar el mensaje que ya se ha abierto, para esto se necesita una función AES o DES, ninguna de estas es estándar de alguna biblioteca de C o C++, así que se tendra que buscar y verificar su funcionamiento.

Conclusión: Podemos ver que en C++ el proceso es simple pero se necesita buscar muy bien las bibliotecas externas que se usarán, ya que no siempre están funcionando correctamente, en algunos casos estas ni siquiera compilan.
Este caso fue particularmente evidente al buscar una biblioteca de C o C++ que pudiera realizar el cifrado con AES o DES, se encontro con bibliotecas que cifraban mal ya que al meter la misma llave no descifraban e incluso bibliotecas que no se lograron compilar.

5.2. Prototipo 2

Se implementó un módulo de cifrado, descifrado y generación de CAPTCHAS en Python, simulando el proceso antes del envío del correo y el que se hace después de la recepción de los correos electrónicos.
Para este se usó el formato estándar del correo electrónico especificado en el RFC 822, también se usaron bibliotecas ya estandarizadas de python para la implementación de las funciones de

hashing, funciones de cifrado y descifrado (AES o DES), funciones aleatorias y la generación de los CAPTCHAS.

```
 #! /usr/bin/env python
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
from captcha.image import ImageCaptcha
import os
import random
hash = SHA256.new()
semilla=""
r=0
image = ImageCaptcha(fonts=['./fon/A.ttf', './fon/B.ttf'])
for i in range(5):
        r=random.randrange(100)
        semilla=semilla+chr(r)
        print str(i)+" "+str(r)+" "+chr(r)+" "+semilla

print semilla+"\n"

data = image.generate(semilla)
image.write(semilla, '/tmp/out.png')
image.write(semilla, 'out.png')

os.remove("/tmp/out.png")

hash.update(semilla)
otra=hash.digest()
llave = ""
print otra


for i  in range(16):

        llave=llave+otra[i]
        print str(i)+" "+otra[i]+" "+llave

print "\n"
print llave
archy=open('llave.txt','w')
archy.write(semilla)
archy.close()
arc=open('cifrado.txt','w')
archi=open('1443750804.V805Idc01e2M920300.jonnytest:2,S','r')
obj = AES.new(llave, AES.MODE_ECB)
lineas=''
```

```
c=0
while lineas!="":
        c=c+1
        lineas=archi.read(16)

        if (((len(lineas))<16)and((len(lineas))>0)):
                c=16-(len(lineas))
                aux=lineas
                for i in range(c):
                        aux=aux+" "
        else:
                aux=lineas

        ciphertext = obj.encrypt(aux)
        arc.write(ciphertext)
        print str(c) +"   " + lineas + "   "+str(len(lineas))+"   "
        +str(len(aux))+"   "+ciphertext


archi.close()
arc.close()
```

```
 #!/usr/bin/python
import Tkinter
import Image, ImageTk
imagenAnchuraMaxima=300
imagenAlturaMaxima=200
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
import random
hash = SHA256.new()
# -*- coding: utf-8 -*-

def funcion():

        a=e.get()
        print(a)
        hash.update(a)
        otra=hash.digest()
        llave = ""
        print otra
        for i  in range(16):

                llave=llave+otra[i]
                print str(i)+" "+otra[i]+" "+llave

        print "\n"
```

```
        print llave
        archi=open('cifrado.txt','r')
        arc=open('descifrado.txt','w')
        obj = AES.new(llave, AES.MODE_ECB)
        lineas=' '
        c=0
        while lineas!="":
                c=c+1
                lineas=archi.read(16)

                if (((len(lineas))<16)and((len(lineas))>0)):
                        c=16-(len(lineas))
                        aux=lineas
                        for i in range(c):
                                aux=aux+" "
                else:
                        aux=lineas

                ciphertext = obj.decrypt(aux)
                arc.write(ciphertext)
                print str(c) +"   " + lineas + "   "+str(len(lineas))+"
"+str(len(aux))+"   "+ciphertext


        archi.close()
        arc.close()
        root.quit()



# abrimos una imagen
img = Image.open('out.png')

img.thumbnail((imagenAnchuraMaxima,imagenAlturaMaxima), Image.ANTIALIAS)
root = Tkinter.Tk()
# titulo de la ventana
root.title("Mostrar imagen")
# Convertimos la imagen a un objeto PhotoImage de Tkinter
tkimage = ImageTk.PhotoImage(img)
# Ponemos la imagen en un Lable dentro de la ventana
label=Tkinter.Label(root, image=tkimage, width=imagenAnchuraMaxima, height=
valor = ""
e = Tkinter.Entry(root)
e.pack()

buttonStart2=Tkinter.Button(root, text="Cerrar",command=funcion).pack()
# Mostramos la ventana
```

```
root.mainloop()
```

Conclusión: Se logró generar todo el proceso de envío y parte del proceso de recepción de mensajes. En cuanto al envío se logró leer el mensaje, crear una palabra a partir de funciones random, crear la clave con dicha cadena de caracteres y cifrar el correo exitosamente, además de esto se logró leer el archivo de mensaje de correo electrónico y cifrar únicamente el mensaje que viene en este.
Por su parte el módulo de generación de CAPTCHAS mostró muchos problemas para generarlos, ya que no se logro hacer que el intérprete pudiera encontrar correctamente las funciones de la biblioteca de generación CAPTCHAS por lo que al no poder generar un CAPTCHA la recuperación no se puede realizar como se planteó, para verificar únicamente que las funciones trabajan correctamente se implementó el descifrado del mensaje en el mismo método.

5.3. Prototipo 3

Objetivo Generar una imagen un CAPTCHA a partir de una cadena de caracteres ingresada desde una interfaz gráfica. Este prototipo se construyó en 2 partes; la primera parte fue la interfaz gráfica y sus herramientas, y la segunda en las herramientas para generar la imagen a partir de una cadena de caracteres.
Para la interfaz gráfica se utilizaron las siguientes herramientas para desarrollar este prototipo:
Biblioteca ***Qt*** y ***Qt creator***: Utilizamos esta biblioteca para generar la interfaz gráfica con la que ingresara la cadena de caracteres y el IDE ***Qt Creator*** para facilitar la gestión de las clases.
La interfaz gráfica consta de un apartado para ingresar la cadena de caracteres y un botón para convertir la cadena a una imagen de CAPTCHAS.
Para generar CAPTCHAS se utilizaron las siguientes herramientas:
Lenguaje PHP: se utilizó para genera las imágenes CAPTCHAS con la cadena de caracteres proporcionada anteriormente.
En un principio se buscó una biblioteca que generara las imágenes CAPTCHAS en el lenguaje C++ pero su implementación no estaba optimizada y necesitaba ser adaptada casi en su totalidad que tener el funcionamiento deseado, por esta razón se busco otra biblioteca que se adaptara más a la funcionalidad del prototipo, por lo tanto se optó por utilizar el lenguaje PHP ya que tiene librerías optimizadas para generar imágenes CAPTCHAS.
Conclusión. La generación de imágenes CAPTCHAS es rápida y fácil de implementar, pero durante la investigación llegamos a la conclusión que el cliente de correo "Thunderbird" está desarrollado en el lenguaje de programación Python y al no tener una biblioteca nativa en el lenguaje C++ para convertir una cadena de caracteres en CAPTCHAS y se decidio cambiar de lenguaje de programación.

5.4. Prototipo 4

Objetivo del prototipo. Instalar y configurar un servidor de correo electrónico para el envío de mensajes de correo electrónico entre diferentes usuarios.
Instalación y configuración de un servidor de correo electrónico y un servidor DNS.

Para el desarrollo de este prototipo fue necesario instalar el servidor de correo electrónico con el protocolo pop y imap, un cliente de correo electrónico web, un servidor DNS y el servidor HTTP Apache. Estos 3 servicios fueron levantados en una computadora con un sistema operativo Xubuntu 15.04; primero se instaló el servidor HTTP [18], posteriormente se pasará a instalar el servidor DNS y configurar un dominio [19]; se seguirá con la instalación del servidor de correo electrónico y los protocolos pop y imap; y por último se instaló y configuró el cliente de correo web [20].
Para la instalación de servidor HTTP fue necesario seguir los siguientes pasos:

- Se abre una terminal en Ubuntu y se escribe el comando: "sudo apt-get install apache2"
- Se abre como administrador el archivo /etc/apache2/sites-enabled/00-default.conf y se escribe la siguiente configuración:

```
<VirtualHost *:80>
    ServerAdmin nombredelsitio@example.com
    ServerName nombredelsitio
    ServerAlias www.nombredelsitio.com
    DocumentRoot /var/www/nombredelsitio.com/public_html/
    ErrorLog /var/www/nombredelsitio.com/logs/error.log
    CustomLog /var/www/nombredelsitio.com/logs/access.log combined
</VirtualHost>
```

- Se levanta el servicio http con el siguiente comando: "sudo service apahce2 start"
- Para verificar la instalación Se abre un explorador y escribirlos en la barra de búsqueda la siguiente dirección: http://localhost/ y nos aparecerá la siguiente pantalla.

Una vez instalado el servidor HTTP se prosigue a instalar el servidor DNS, para levantar este servicio es necesario seguir los siguientes pasos:

- Se selecciona un nombre de dominio, para fines prácticos nuestro dominio privado será "correocifrado.edu".
- Se abre una terminal en Ubuntu y se escribe el siguiente comando: "sudo apt-get install bind9"
- Realizar una copia de respaldo del archivo de configuración original con el comando "cp /etc/bind/named.conf.local
  /etc/bind/named.conf.local.original"
- Se edita el archivo de configuración con: "nano /etc/bind/named.conf.local"
- Se agrega al final del archivo lo siguiente:

```
zone "correocifrado.edu" {
type master;
file "correocifrado.edu.zone";
};

zone "10.168.192.in-addr.arpa" {
type master;
file "db.192.168.10";
};
```

- Se procede a crear los (nuevos) archivos de zona, esos archivos contienen los registros del DNS y en Ubuntu se encuentran en el directorio /var/cache/bind/ "nano /var/cache/bind/db.isti.edu.ni.zone"

- En el archivo Se agrega el siguiente texto:

```
$ORIGIN correocifrado.edu.
$TTL 86400              ; 1 dia
@       IN      SOA ns.correocifrado.edu. info.correocifrado.edu. (
        2014112401      ; serie
        6H              ; refresco (6 horas)
        1H              ; reintentos (1 hora)
        2W              ; expira (2 semanas)
        3H              ; m nimo (3 horas)
)

@       IN      NS      ns
@       IN      MX 10   mail
ns      IN      A       192.168.10.10
mail    IN      A       192.168.10.10
www     IN      A       192.168.10.10
```

- De igual manera el archivo de zona de búsqueda inversa:
nano /var/cache/bind/db.192.168.10

- Se agrega la siguiente configuración:

```
$ORIGIN 10.168.192.in-addr.arpa.
$TTL 86400      ; 1 dia
@       IN  SOA ns.correocifrado.edu. info.correocifrado.edu. (
        2014112401      ; serie
        6H              ; refresco (6 horas)
        1H              ; reintentos (1 hora)
        2W              ; expira (2 semanas)
        3H              ; m nimo (3 horas)
)
```

```
@       IN      NS      correocifrado.edu.
10      IN      PTR      correocifrado.edu.
10      IN      PTR      mail.correocifrado.edu.
10      IN      PTR      www.correocifrado.edu.
```

- Se procede a re-iniciar el servicio con el comando “service bind9 restart”
- Cambiar el primero de los servidores DNS por la IP del nuestro: “nameserver 192.168.10.10”
- Lo único que quede es realizar las pruebas en el cliente “nslookup www.correocifrado.edu”

Se prosigue con la instalación del servidor de correo electrónico y los servicios del protocolo pop y imap con la aplicación courier-pop y courier-imap:

- Se abre una terminal y se escribe el siguiente comando: “sudo apt-get install postfix”
- Durante la instalación aparecerá una pantalla de configuración, se da enter para aceptar la configuración.
- En tipo genérico de configuración de correo se selecciona "Sitio de Internet".
- A continuación se indica el nombre de sistema de correo, normalmente la dirección del dominio registrado, en este caso çifradocorreo.net".
- Con esto se verá que postfix termina de instalarse y se procede a editar el archivo “/etc/postfix/main.cf”.
- Se añade al final del fichero main.cf las líneas:

```
  inet_protocols = ipv4
home_mailbox = emails/
```

- Una vez guardado el archivo que se edita se procede a reiniciar el servidor con el comando “sudo /etc/init.d/posrfix restart”

Una vez instalado el servicio de correo electrónico se procede a instalar el courier-pop y el courier-imap.

- Se abre una terminal el Ubuntu y se escribe el siguiente comando “sudo apt-get install courier-pop”.
- Nos mostrará una ventana de configuración de courier-base, se selecciona “NO”.
- Se procede a instalar courier-imap con el siguiente comando “sudo apt-get install courier-imap”.
- Se espera a que finalice la instalación.

Por ultimo es necesario instalar una aplicación webmail para enviar correos entre usuarios del correo electrónico.

- Se abre una terminal en Ubuntu y se escribe el siguiente comando “sudo apt-get install squirrelmail”.

- Tras la instalación de SquirrelMail se configura ejecutando el siguiente comando "sudo squirrelmail-configure"
- Se selecciona la letra D y se da enter.
- En este nuevo menú se teclea la opción courier y se da enter.
- Nos dará un informe de la configuración que se selecciono y se da enter para continuar.
- Se regresa al primer menú, ahora se teclea el número 2 y se da enter.
- Se selecciona en este nuevo menú la opción 1 y se da enter nuevamente.
- Pedirá nuestro nombre de dominio, en este caso es el dominio que se configuro en el servidor DNS "correocifrado.net"
- Regresara al menú principal y se teclea la letra Q para salir de la configuración.
- Preguntara si queremos guardar los cambios y se teclea la letra Y.
- Por ultimo se ejecuta el siguiente comando para levantar SquirrelMail en Apache "sudo ln -s /usr/share/squirrelmail /var/www/webmail"
- Se reinicia el servicio apache con el comando "sudo service restart apache2".
- Se podra entrar a la aplicación escribiendo el explorador "www.correocifrado.edu/webmail"

Para poder enviar correos se necesitan usuarios que desean enviar mensajes entre usuarios, primero se creara un usuario

- Se abre una terminal de Ubuntu y se escribe el siguiente comando "sudo adduser nombreusuario".
- Se introduce la nueva contraseña de UNIX: introduciremos la contraseña para el usuario, es importante que sea segura (números, letras, mayúsculas y minúsculas) pues con el usuario y la contraseña podremos acceder vía web al servidor de correo electrónico desde cualquier parte del mundo.
- Vuelva a escribir la nueva contraseña de UNIX: se repite la contraseña.
- Full Name: Se introduce el nombre completo, por ejemplo "Alicia Robles Maldonado".
- Room Number: Número de oficina.
- Work Phone: teléfono del trabajo.
- Home Phone: teléfono particular.
- Other: otros datos del usuario.
- Se responde "S" a la pregunta "¿Es correcta la información?". Y se tendrá el usuario creado en el sistema operativo, que también servirá como usuario (buzón) para el servidor de mail.

- Ahora se generará el buzón con el siguiente comando “sudo maildirmake /home/nombreusuario/emails”

- Se cambian los permisos de las carpeta emails con el comando “sudo chdown nombreusuario:nombreusuario /home/nombreusuario/emails -R

Para crear otro usuario es necesario repetir los pasos anteriores.

5.5. Prototipo 5

Objetivo: Familiarizarse con el uso de la tecnología y estructura del cliente de correo electrónico thunderbird.
Se planeo la creación de un completo para el cliente de correo electrónico thunderbird, para dar paso al desarrollo del complemento es necesaria la documentación de dicho cliente de correo, la cual debe de ser debidamente requisitada a su desarrollador que en este caso es Mozilla. El cliente thunderbir al ser un cliente de software libre debe de contar con una documentación pública. Al buscar ser documentación en la página de desarrollo de Mozilla resulta evidente que está no está, pero puede ser pedida a Mozilla por medio de la misma página. La documentación fue pedida el 03-03-16 y a la fecha de escritura de este reporte 20-04-16 no se ha obtenido una respuesta por parte de Mozilla.
Conclusión: Por el tiempo de respuesta y la falta de documentación implementar un complemento para thunderbird en el tiempo proporcionado para este trabajo terminal no es viable.

5.6. Prototipo 6

Objetivo: Familiarizarse con el uso de las tecnologías y estructuras de Nylas N1, así como verificar su viabilidad como solución factible para el presente trabajo.
La documentación de este cliente de correo electrónico es fácil de conseguir ya que es publica directamente en su pagina oficial, además de contar con breves tutórales de como usarse. Se implemento sobre la interfase incluir imágenes y mostrar información nueva sobre el panel auxiliar, también la obtención directa del cuerpo del mensaje para poder procesarlo, agregar una clase dentro del complemento que permita la comunicación con el servidor de CAPTCHAS.

- Inclusión de texto e imagen en la interfase: El mismo N1 da la posibilidad de generar tu propio complemento de correo ya que el mismo te proporciona un formato estándar y una opción para modificar los textos de sus diferentes áreas, se modifico justo la opción de texto en la barra lateral derecha pero para agregar una imagen y texto al mismo tiempo.

- Obtención del cuerpo del mensaje: Despues de analizar la estructura del complemento se creo uno propio en el cual se mandaron llamar el cuerpo del mensaje y el asunto por medio de metodos que el mismo N1 proporciona.

- Comunicación con el servidor de CAPTCHAS: En complemento creado por nosostros se genero una clase que hace una llamada al servidor de CAPTCHAS que espera una

respuesta para poder empaquetar el nuevo cuerpo del mensaje y poder mandarlo por correo. Todas estas clases están implementasdas en CoffeScript este lenguaje no es secuencial si no concurrente, como la función de empaquetamiento del cuerpo espera la respuesta del servidor y esta no llega si no hasta despues de que ya se ejecuto esta función genera un error.

Conclusión: No es posible hacer una sincronización con los servidores externos que necesitamos, por lo que la implementación no es viable en Nylas N1.

5.7. Prototipo 7

Objetivo: Evaluar la viabilidad y compatibilidad del algoritmo de cifrado así como su integración con Nylas N1.
Se implementaron los algoritmos de cifrado, descifrado, generación de llave y generación de CAPTCHA en el lenguaje JavaScript. considerando que Nylas N1 esta implementado en CoffeScript que es una versión de escritorio de JavaScript.

- Generación de llave: Se genera una cadena de caracteres aleatorio de tamaño 5. Primero se genera un numero aleatorio del 0 al 63 y este se manda a otra función que lo mapea a su caracter correspondiente en el anillos $AL = \{A-Z\} \cup \{a-z\} \cup \{0-9\} \cup \{+,/\}$

```
  var map;map = [];
map=["A","B","C","D","E","F","G","H","I","J","K","L","M","N","O","P","Q
//alert(map.length);


function CtoI(let){
        var a;
        for (var i = 0; i < map.length; i++) {
                if (map[i]==let) {
                        a=i;
                        //alert(a);
                }
        }
        return a;
}

function ItoC(num){
        return map[num];
}
```

de esta cadena generaremos un digesto por medio de una función hash sha256 y recortada a una cadena de caracteres tamaño 16 la cual se usara como llave para la función de cifrado.

- Cifrado: se cifra el texto con una función AES 128bits usando la llave generada anteriormente

- generación de CAPTCHA: JavaScript no tiene metodos propios de edición de imagen ni bibliotecas de creación de CAPTCHAS, pero se pueden pasar las variables declaradas en JavaScript a PHP y que este lenguaje termine el proceso, solo que es necesaria la creación de un formulario HTML para que esto suceda, en el caso nuestro esto no es factible ya que Nylas N1 no hace uso de estas herramientas.

```
 var semilla="";

for (var i = 0; i < 5; i++) {
        var r = Math.floor((Math.random() * 63) + 1);
        semilla=semilla + ItoC(r);
}

alert(semilla);

var shaObj = new jsSHA("SHA-1", "TEXT");
shaObj.update(semilla);
var hash = shaObj.getHash("HEX");

alert(hash);
var llave;
llave = hash[0];
//llave = llave + hash[1];
for (var i = 1; i < 16; i++) {
        llave=llave + String(hash[i]);
}

alert(String(llave));

usedKey=llave;
myStr="Osama Oransa2012Osama Oransa2011RashaOsama Oransa2012Osa  Oransa2011I
alert(myStr);

var key=init(usedKey);
alert(key);
encrypted=encryptLongString(myStr,key);
alert('after encrypt='+encrypted);
decrypted=decryptLongString(encrypted,key);
alert('after decrypt='+decrypted);
finish();
```

Conclusión: El uso de JavaScript para implementar el esquema propuesto en este trabajo no es totalmente posible ya que el mismo lenguaje no nos permite la creación de CAPTCHAS ni la llamada a sistema por lo que limita la capacidad de desarrollo.

5.8. Prototipo 8

Objetivo: Crear una biblioteca en lenguaje python que contenga el esquema de cifrado por CAPTCHA. El cliente de correo electrónico thunderbir tiene una parte implementada en python, pensado para esto se implemento el presente esquema en el lenguaje python. Para continuar con el desarrollo del presente trabajo se decidió crear una biblioteca en el lenguaje python.
En esta biblioteca se implementaron en metodos por separado: la creación de la cadena original llamada semilla, la creación de la llave, la creación del CAPTCHA, cifrado y descifrado. En esta biblioteca se esta considerando un esquema en el cual se pueda generar un solo CAPTCHA por mensaje cifrado o n CAPTCHA's por cada mensaje cifrado, para el esquema multi CAPTCHA se hace uso de el algoritmo de secreto compartido por lo que para este se implementaron los metodos: Codificación, decodificación, repartir el secreto y recuperar el secreto

- Crear Semilla: para este método originalmente esta configurado para hacer una palabra de 5 caracteres, pero puede introducir manualmente el numero de caracteres deseados, esto es generando 5 números aleatorios en un rango de 0 a 63 que posteriormente son asignados a su correspondiente caracter en el anillo $AL = \{A - Z\} \cup \{a - z\} \cup \{0 - 9\} \cup \{+, /\}$

```
def crearSemilla(tam):
        r=0
        semilla=""
        for i in range(tam):
                r=random.randrange(64)
                semilla=semilla+str(mapeoItoB(r))
        return semilla
```

- Crear Llave: En este metodo se manda la semilla y a esta se le genera un digesto con una funcion SHA256 que posteriormente es recortada a 16 bits.

```
def crearLlave(semilla1):
        aux=""
        llave=""
        hash = SHA256.new()
        hash.update(semilla1)
        aux=hash.digest()
        llave = ""
        for i  in range(16):
                llave=llave+aux[i]
        return llave
```

- Crear CAPTCHA: este método recibe la opción, la semilla, y el asunto. La opción define cual es el funcionamiento de este método, si la opción es 0 el método toma la semilla y crea un CAPTCHA a partir de ella usando la biblioteca CAPTCHA de python. Por el contrario si la opción es 1 lo que recibe la función en el parametro semilla, es un arreglo de caracteres y uno a uno lo convierte en CAPTCHA.

```
def crearCAPTCHA(op,semilla2,asunto):
        imagen=""
        aux=""
        ax=[]
        xa=""
        s=[]
        c=0
        asunto=asunto.replace(" ","_")
        os.mkdir('./'+asunto+"_CAPTCHA",0755)
        image = ImageCaptcha(fonts=['./fon/A1.ttf', './fon/A1.ttf'])
        if (op==0):
                aux='./'+asunto+'_CAPTCHA/CAPTCHA00.png'
                image.write(semilla2, aux)
                return './'+asunto+"_CAPTCHA"
        else:
                for x in semilla2:
                        #print(x)
                        aux='./'+asunto+'_CAPTCHA/CAPTCHA'+str(c)+'.png
                        xa='CAPTCHA'+str(c)+'.png'
                        ax.append(xa)
                        s.append(ax)
                        ax=[]
                        image.write(x, aux)
                        c=c+1
                return (s,'./'+asunto+"_CAPTCHA")
```

En el caso particular de la generación de CAPTCHAS, se modifico la biblioteca que los crea, ya que esta función arrojaba CAPTCHAS ilegibles.

```
def _draw_character(c):
        font = random.choice(self.truefonts)
        w, h = draw.textsize(c, font=font)

        #dx = random.randint(4, 6)
        #dy = random.randint(4, 8)
        im = Image.new('RGBA', (w+30 , h+25 ))
        Draw(im).text((0, 0), c, font=font, fill=color)

        # rotate
        #im = im.crop(im.getbbox())
        #im = im.rotate(random.uniform(-30, 30), Image.BILINEAR, ex

        # warp
        #dx = w * random.uniform(0.1, 0.3)
        #dy = h * random.uniform(0.2, 0.3)
        #x1 = int(random.uniform(-dx, dx))
        #y1 = int(random.uniform(-dy, dy))
        #x2 = int(random.uniform(-dx, dx))
```

```
            #y2 = int(random.uniform(-dy, dy))
            #w2 = w + abs(x1) + abs(x2)
            #h2 = h + abs(y1) + abs(y2)
            #data = (
            #    x1, y1,
            #    -x1, h2 - y2,
            #    w2 + x2, h2 + y2,
            #    w2 - x2, -y1,
            #)
            #im = im.resize((w2, h2))
            #im = im.transform((w, h), Image.QUAD, data)
            return im

        images = []
        for c in chars:
            images.append(_draw_character(c))

        text_width = sum([im.size[0] for im in images])

        width = max(text_width, self._width)
        image = image.resize((width, self._height))

        average = int(text_width / len(chars))
        rand = int(0.25 * average)
        offset = int(average * 0.1)

        for im in images:
            w, h = im.size
            mask = im.convert('L').point(lambda i: i * 1.97)
            image.paste(im, (offset, int((self._height - h) / 2)), mask
            offset = offset + w + random.randint(-rand, 0)

        return image
```

- Encode: La función encode toma como parametro, una cadena de caracteres y la transforma en un numero entero, primero caracter a caracter se busca su correspondiente numero entero en el anillo AL posteriormente este se pasa a una representación en 6 bits y por ultimo se convierte a un numero entero.

```
def encodeSS(strin):
        bina=""
        aux=""
        for i in range(len(strin)):
                aux=bin(mapeoBtoI(strin[i])).replace("0b","")
                if (len(aux)==6):
                        bina=bina+aux
                else:
                        while ((len(aux))<6):
```

```
                aux="0"+aux
            bina=bina+aux
    return int(str(bina),2)
```

- Decode: La función decode recibe como parametro un numero entero y el numero de partes en las que tiene que partir el numero, esto lo hace convirtiendo el numero entero a su representación binaria, posteriormente se separa en números binarios de 6 bits y cada uno es convertido en un numero entero e intercambiado por su correspondiente caracter del anillo AL.

```
def decodeSS(strr,w0):
        c=0
        s=""
        capt=""
        letras=[]
        z=bin(strr).replace("0b","")
        while (len(z)<(6*w0)):
                z="0"+z
        for i in z:
                if (c==5):
                        c=0
                        s=s+i
                        letras.append(s)
                        s=""
                else:
                        c=c+1
                        s=s+i
        for j in letras:
                capt=capt+mapeoItoB(int(str(j),2))
        return capt
```

- Generar Partes: Se reciben como parametro el anillo Zp , el numero de partes en que se dividirá el secreto w, el numero de partes necesarias para recuperar el secreto t y el secreto k. se generan aleatoriamente w que son las x e igualmente de manera aleatoria se generan t elementos que son los elementos a, con esto se genera la sumatoria correspondiente para generar los elementos y. La función retorna pares de números que están conformados por x, y.

```
def GenerarPares(p=7,w=5,t=2,k=0):
    pares =[]
    a = [k]
    for aux in range(0,w):
        pares.append([randrange(p),0])
    for aux in range(1,t):
        a.append(randrange(p))
    for aux in pares:
        suma = k
        for aux2 in range(1,t):
```

```
            suma = (suma+(a[aux2]*(aux[0]**aux2)))%p
        aux[1] =suma
    return pares
```

- Recuperar secreto: por medio del algoritmo de la grange se resuelve el sistema de ecuaciones y así recuperando el secreto

```
def secreto(pares,p):
    suma = 0
    for aux in pares:
        ind = pares.index(aux)
        lis = pares[:ind] + pares[(ind+1):]
        num=1
        den=1
        for aux2 in lis:
            num = (num*(aux2[0])*-1)%p
            den = (den*((aux[0]-aux2[0])%p))%p
        den = eucExt(den,p)
        suma += (den*aux[1]*num)%p
    return suma%p
```

- cifrar: Este método recibe como parametro el cuerpo del mensaje, el asunto y de manera opcional recibe la opción de cifrado, el numero de caracteres de los CAPTCHAS, el numero de partes en que se divide el secreto, y el numero de pares necesarios para recuperar el secreto. Este método hace las funciones de un main, ya que en este método se invocan todos los demás para poder realizar el funcionamiento del esquema completo. Este método tiene dos funciones, la primera es la opción 0 en la que se crea la semilla, se calcula el anillo Zp, con la semilla se crea el CAPTCHA con opción 0, con la semilla se crea la llave y con esta se procede a cifrar el cuerpo del mensaje siempre cuidando que los bloques sean del tamaño de la llave, en este caso el meto retorna el cuerpo cifrado y la ruta en la que están los CAPTCHAS. Este método en opción 1 genera la semilla, despues calcula Zp, mapea a numero la semilla por medio de encode, con este numero se generan los pares del secreto compartido, estos son re mapeados a cadena de caracteres y convertidos en CAPTCHAS, se crea la llave y por ultimo se cifra el cuerpo del mensaje con esta, en este caso el método retorna el cuerpo del mensaje cifrado y una lista de pares donde esta el numero x y su correspondiente imagen CAPTCHA.

```
def cifrar(body,asunto1,op1=1,ta=5,w1=5,t1=2):
        ruta=""
        salida=""
        if t1>w1:
                salida=""
                ruta=None
                print("w1 < t1")
                return (salida,ruta)
        semilla3=crearSemilla(ta)
        num=0
        cap=[]
```

```
zp=2**(6*ta)
#print(semilla3)
if (op1==0):
        ruta=crearCAPTCHA(0,semilla3,asunto1)
else:
        ruta=[]
        num=encodeSS(semilla3)
        pares=GenerarPares(zp,w1,t1,num)
        #print(pares)
        for x in pares:
                cap.append(decodeSS(x[1],w1))
        ruta=crearCAPTCHA(op1,cap,asunto1)
        num=0
        #print(cap)
        for x in pares:
                ruta[0][num].insert(0,x[0])
                num=num+1
k=crearLlave(semilla3)
obj = AES.new(k, AES.MODE_ECB)
salida=""
ax=0
c=0
strr=""
#print len(body)
while (ax < len(body)):
        while (c<16):
                if (ax>=len(body)):
                        strr=strr+" "
                else:
                        strr=strr+body[ax]
                c=c+1
                ax=ax+1
                #print str(c) +" " + str(ax)
        c=0
        #print strr
        ciphertext = obj.encrypt(strr)
        salida=salida+ciphertext
        strr=""
return (salida,ruta)
```

- descifrar: este método hace le proceso inverso que el método cifrar, este recibe como parametro el cuerpo cifrado, una cadena con el CAPTCHA o una lista de pares, $x, CAPTCHA$. Con la opción 0 recibe una cadena de caracteres en la opción captcha con este se genera la llave y se descifra el cuerpo obteniendo el mensaje original. En caso de tener la opción 1 se recibe en el parametro captcha una lista que contiene los pares $x, CAPTCHA$ de este se obtiene el tamaño del CAPTCHA y se calcula el anillo Zp, con esto se toman los CAPTCHAS y se mapean a su representación numérica por

medio de encode, los pares de números se ingresan al método que calcula el secreto compartido, lo arrojado por este metodo se mapea a su representación en cadena de caracteres y se genera la llave con esto se descifra el cuerpo del mensaje y se obtiene el mensaje original. Este metodo retorna el cuerpo del mensaje original.

```
def descifrar(body1,capt1,op2):
        aux=[]
        ax=0
        pares=[]
        zp=0

        if (op2==0):
                k=crearLlave(capt1)
        else:
                w=len(capt1[0][1])
                zp=2**(6*(len(capt1[0][1])))
                for x in capt1:
                        aux=x
                        aux[1]=encodeSS(x[1])
                        pares.append(aux)
                #print(pares)
                ax=secreto(pares,zp)
                #print(ax)
                semilla4=decodeSS(ax,w)
                #print(semilla4)
                k=crearLlave(semilla4)
                #print(k)
        obj = AES.new(k, AES.MODE_ECB)
        salida=""
        ax=0
        c=0
        strr=""
        #print len(body1)
        while (ax < len(body1)):
                while (c<16):
                        if (ax>=len(body1)):
                                strr=strr+" "
                        else:
                                strr=strr+body1[ax]
                        c=c+1
                        ax=ax+1
                        #print str(c) +" " + str(ax)
                c=0
                #print strr
                ciphertext = obj.decrypt(strr)
                salida=salida+ciphertext
                strr=""
```

```
return salida
```

Conclusión: La funcionalidad de la biblioteca ya implementada, corresponde al esquema propuesto en "On Securing Communication from Profilers" [4, 5], al tener todas las funciones separadas lo hace tener una funcionalidad modular. Esta implementacion se usara para integrarla al siguiente prototipo.

5.9. Prototipo 9

Referencias

[1] Email, Internet: http://en.wikipedia.org/wiki/Email, Mayo, 2015

[2] Interactive Advertising Bureau, Marcelo Brodsky, "Reflexiones jurídicas sobre el e-marketing en Chile", Internet: http://www.iab.cl/reflexiones-juridicas-sobre-ele-marketing-en-chile.

[3] D. Jurafsky, Text Classification, Stanford University Natural Language Processing.

[4] S. Díaz Santiago y D. Chakraborty., "On Securing Communication from Profilers." Proceedings of International Conference on Security and Cryptography, Secrypt 2012, pp.154-162, Rome, Italy, 2012.

[5] Philippe Golle and Ayman Farahat. "Defending Email Communication Against Profiling Attacks" Proceedings of the 2004 ACM workshop on Privacy in the electronic society, ACM New York, NY, USA ©2004pp 39-40

[6] J. Klensin, "Simple Mail Transfer Protocol", Patent 5321, October 2008.

[7] J. Myers, "Post Office Protocol - Version 3", Patent 1939, May 1996.

[8] A. Gulbrandsen, "Internet Message Access Protocol (IMAP) - MOVE Extension", Patent 6851, January 2013.

[9] Algebra Modular, Internet: https://es.wikipedia.org/wiki/Aritm%C3%A9tica_modular, Abril, 2016

[10] Polinomio de Lagrange, Internet: https://es.wikipedia.org/wiki/Interpolaci%C3%B3n_polin%C3%B3mica_de_Lagrange

[11] Criptografia, Internet: http://www.matem.unam.mx/rajsbaum/cursos/web/presentacion_seguridad_1.pdf, Noviembre2015

[12] Douglas R. Stinson, "Cryptography Theory and Practice Third Edition", Ontario, Canada: University of Waterloo

[13] Tipos de ataques, Internet: http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/, Noviembre2015

[14] Cifrado simetrico, Internet: https://www.gnupg.org/gph/es/manual/c190.html#AEN201, Noviembre2015.

[15] Debrup Chakraborty and Francisco Rodríguez-Henríquez, "Block Cipher Modes of Operation from a Hardware Implementation Perspective" Computer Science Department, Centro de Investigación y de Estudios Avanzados del IPN, México D.F.

[16] Ciphertext-only attack Internet: https://en.wikipedia.org/wiki/Ciphertext-only_attack,Noviembre2015.

[17] PGP Internet: https://es.wikipedia.org/wiki/Pretty_Good_Privacy, Noviembre2015

[18] HTTP Internet: http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=506, Noviembre2015

[19] DNS Internet:http://www.servermom.org/install-apache-php-mariadb-ubuntu-15-04/2208/, Noviembre2015

[20] web Internet: https://www.howtoforge.com/tutorial/ubuntu-perfect-server-with-apache-php-myqsl-pureftpd-bind-postfix-doveot-and-ispco Noviembre2015

[21] emclient Internet: http://www.emclient.com/, Noviembre2015

[22] postbox Internet: https://www.postbox-inc.com/, Noviembre2015

[23] zimbra Internet: https://www.zimbra.com/, Noviembre2015

[24] opera Internet: http://www.opera.com/es-419/computer/mail, Noviembre2015

[25] thunderbird Internet https://www.mozilla.org/es-ES/thunderbird/, Noviembre2015

[26] CAPTCHAS Internet http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.444.8759&rep=rep1&type=pdf, Noviembre2015

[27] Bases de Datos Internet https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-syste Noviembre2015

[28] sql Internet http://danielpecos.com/documents/postgresql-vs-mysql/, Noviembre2015