INSTITUTO POLITÉCNICO NACIONAL

Escuela Superior de Cómputo

ESCOM

Trabajo Terminal

"Aplicación de cifrado contra 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

Mayo 2016

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

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 [30]. 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 [10]. 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 [15], 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 [12].

1.1. Justificación

La comunicación por medio del correo electrónico es atacada constantemente y por ello 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 los 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.

1.2. Objetivos

1.2.1. 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.

1.2.2. 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.

1.3. 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 y 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. [31]

1.4. GPG (GnuPG o GNU Privacy Guard).

GnuPG es una herramienta de seguridad en comunicaciones electrónicas desarrollada por Werner Koch. GnuPG, o también conocida como GPG, implementa el sistema de seguridad OpenPGP. OpenPGP es la implementación libre de PGP y esta bajo la licencia GPL.

GPG, al igual que PGP, utiliza criptografía de clave pública para que los usuarios puedan comunicarse de modo seguro. También cuenta con sistema de cifrado híbrido para mejorar la velocidad en envió de información y la posibilidad de manejar claves de sesión. [18]

Capítulo 2

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 describirán las amenazas a las que se enfrenta 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.

2.1. 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 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 las envían a otras oficinas si es necesario; y por último estas se encargan 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. [17]

- **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:

  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 correos en la 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 electrónico 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 [16].

- **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 [20].

- **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 [14].

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, sin importar que 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 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.

2.2. Criptografía

El objetivo fundamental de la criptografía [26] es garantizar que dos entidades, usualmente representadas por Alicia y Bob, se comuniquen a través de un canal inseguro de tal manera que un oponente conocido como Óscar, no pueda entender lo que se dice. A dicho oponente se le conoce también como *adversario*. Cuando Alicia manda un mensaje a Bob también conocido como *texto en claro*, si Oscar intercepta el mensaje puede leer su contenido sin problemas, pero si Alicia cifra el texto en claro usando una clave, ella obtiene un texto cifrado el cual es enviado por el canal de comunicación a Bob. Si Oscar intercepta este mensaje ya no podrá leerlo porque no sabe como pasar del texto cifrado al texto en claro, pero Bob si será capaz de recuperar el texto en claro, por que él sí conoce la clave de cifrado.

Formalmente la criptografía [32] es el estudio de técnicas matemáticas relacionadas con la seguridad para proveer los siguientes servicios:

- **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 previamente.

Estas técnicas están destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados.

2.2.1. Tipos de Ataques

A continuación se describen los tipos de ataque a las comunicaciones, los cuales dependen de cuanta información tenga disponible el adversario para poder romper el cifrado [33].

- **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)**: En éste 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.

Definición 2.1 *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 de posibles claves a utilizar.*
4. *Para cada* $K \in \mathcal{K}$*, se tiene una regla* $e_k \in \mathcal{E}$ *y su correspondiente regla* $d_k \in \mathcal{D}$*. Donde* $e_k : \mathcal{M} \to \mathcal{C}$ *y* $d_k : \mathcal{C} \to \mathcal{M}$ *siendo funciones tales que* $d_k(e_k(x)) = x$ *siendo* x *un elemento del espacio de textos en claro* $x \in \mathcal{M}$

2.2.2. Cifrado Simétrico

Un sistema de cifrado simétrico [1] 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, esto se ilustra en la Figura 2.1.

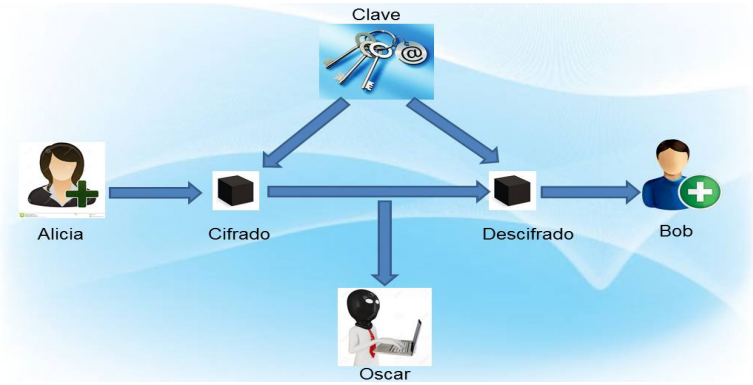


Figura 2.1: Diagrama cifrado simétrico

La sintaxis de un esquema de cifrado simétrico, esta dada por la siguiente definicón.

Definición 2.2 *Un esquema de cifrado simétrico está conformado por una tripleta de algoritmos* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, definidos como se describe a continuación:*

- *El algoritmo generador de claves* Gen *selecciona una llave* K *al azar del conjunto de llaves* $\mathcal{K}$*, esto se denotará como* $K \xleftarrow{\$} \mathcal{K}$*. Esta clave* K *será usada por los algoritmos* Enc *y* Dec*, esta clave la compartirán emisor y receptor.*
- *El algoritmo de cifrado* Enc*, toma como entrada un texto en claro* $M \in \mathcal{M}$ *y una clave* K *generada por* Gen *y regresa un texto cifrado* $C \in \mathcal{C}$*. Usualmente esto se denota como* $C \leftarrow \mathsf{Enc}_K(M)$*.*
- *El algoritmo de descifrado* Dec*, toma como entrada un texto cifrado* C *y una llave* K *y regresa* M*. Esta operación se denota por* $M \leftarrow \mathsf{Dec}_K(C)$*. Para que cualquier algoritmo de cifrado simétrico funcione correctamente, se debe garantizar que para todas las llaves posibles en* $\mathcal{K}$ *y todos los posibles mensajes* $\mathcal{M}$*,*

$$\mathsf{Dec}_K(\mathsf{Enc}_K(M)) = M.$$

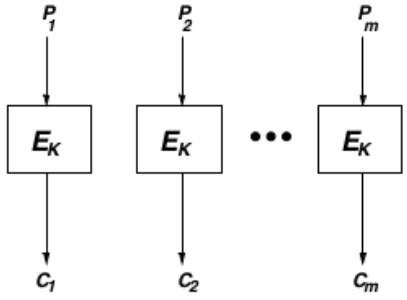
2.2.3. Cifrado por bloques

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 reemplazo de un bloque de n bits por otro bloque de n bits en un espacio de 2^k [11]. Los cifradores por bloques mas usados son AES (Advanced Encryption Standard, por sus siglas en inglés) y DES (Data Encryption Standard, por sus siglas en inglés).

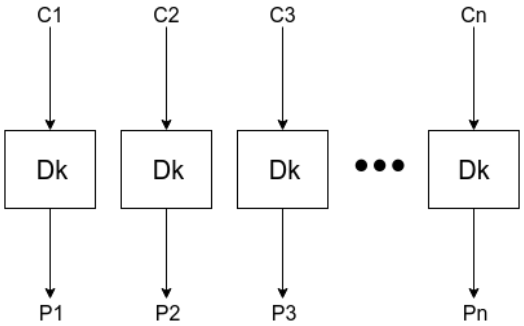
2.2.4. Modos de operación

Los modos de operación fueron desarrollados para el algoritmo DES, estos fueron estandarizados en Diciembre de 1980. Cuando la información es cifrada usando la misma clave, surge una serie de problemas de seguridad. En esencia los modos de operación son una técnica para mejorar el efecto criptográfico de los cifradores por bloques [25].

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

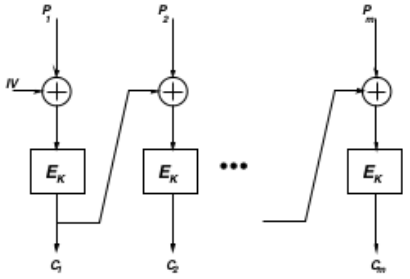


(a) Diagrama ECB Cifrado

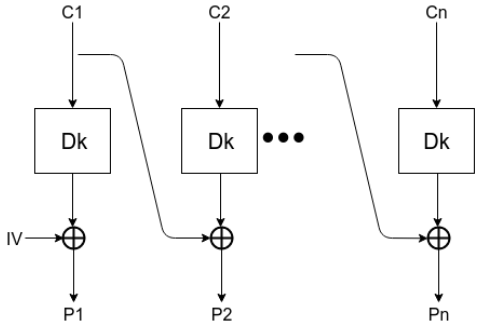


(b) Diagrama ECB Descifrado

CBC(Cipher-block chaining): Para 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.



(a) Diagrama CBC Cifrado

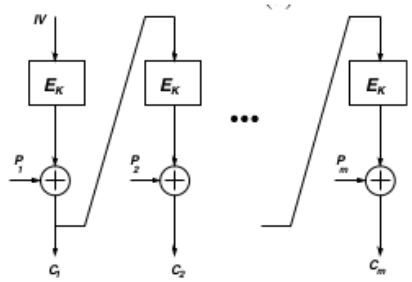


(b) Diagrama CBC Descifrado

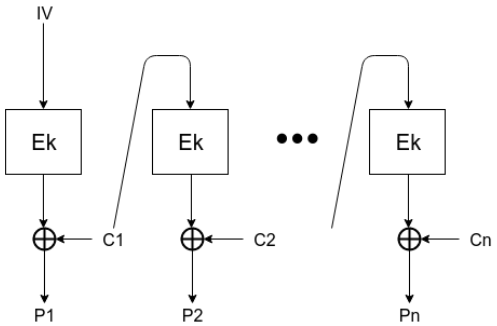
Algorithm $\text{CBC.Encrypt}_K^{\text{IV}}(P)$	**Algorithm** $\text{CBC.Decrypt}_K^{\text{IV}}(C)$
1. Partition P into $P_1, P_2, \ldots, P_m$	1. Partition C into $C_1, C_2, \ldots, C_m$
2. $C_1 \leftarrow E_K(P_1 \oplus \text{IV})$;	2. $P_1 \leftarrow E_K^{-1}(C_1) \oplus \text{IV}$
3. **for** $i \leftarrow 2$ 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** $P_1, P_2, \ldots, P_m$

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): En este modo de operación, los bloques de cifrado también están encadenados pero a la salida se produce de una manera muy diferente de la de CBC. Cada bloque de salida se le aplica XOR con el siguiente bloque de entrada.



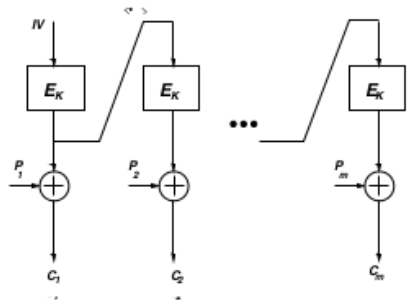
(a) Diagrama CFB Cifrado



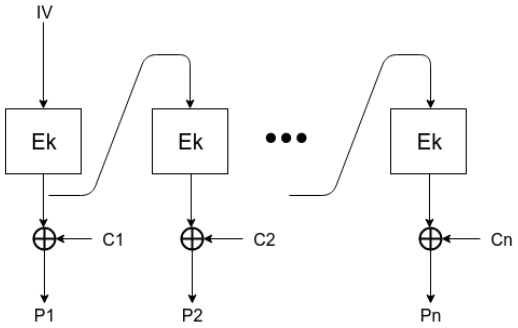
(b) Diagrama CFB Descifrado

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** $P_1, P_2, \ldots, P_m$

OFB(Output feedback): En este modo de operación el IV se cifra varias veces para obtener un flujo de bytes aleatorios, el resultado de esto se aplica XOR con el bloque de texto plano mientras que el flujo de bytes aleatorios se usa como parámetro del siguiente bloque. A diferencia de los otros modos en OFB ninguna parte del texto claro entra directamente a cifrarse.



(a) Diagrama OFB Cifrado



(b) Diagrama OFB Descifrado

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. $X \leftarrow \text{IV}$;	2. $X \leftarrow \text{IV}$
3. **for** $i \leftarrow 1$ to m	3. **for** $i \leftarrow 1$ to m
4. $\quad X \leftarrow E_K(X)$;	4. $\quad X \leftarrow E_K(X)$;
5. $\quad C_i \leftarrow X \oplus P_i$	5. $\quad P_i \leftarrow X \oplus C_i$
6. **end for**	6. **end for**
7. **return** $C_1, C_2, \ldots, C_m$	7. **return** $P_1, P_2, \ldots, P_m$

2.2.5. Funciones Hash

A continuación se describirán las características de las *funciones hash*, también conocidas como *funciones de resumen*. Las funciones hash basan su definición en funciones de un solo sentido (*one-way functions*, en inglés). Una función de un sólo sentido es aquella que para un valor x, es muy fácil calcular $f(x)$, pero es muy difícil hallar $f^{-1}(x)$. Es complicado en general, hallar funciones de éste tipo y probar que lo son.

Definición 2.3 *Una función hash, es una función de un sólo sentido cuya entrada m es un mensaje de longitud arbitraria y la salida es una cadena binaria de longitud fija. Al resumen o hash de un mensaje m, se le denotará como $h(m)$. Una función hash debe tener las siguientes propiedades:*

- *Para cualquier mensaje m, debe ser posible calcular $h(m)$ eficientemente.*
- *Dado $h(m)$, debe ser computacionalmente difícil, hallar un mensaje m', tal que $h(m) = h(m')$.*
- *Debe ser computacionalmente difícil, hallar dos mensajes m y m' tales que $h(m) = h(m')$.*

Entre las funciones hash que se usan para criptografía están: MD2, MD4, MD5, donde MD significa *Message Digest*, y el algoritmo estándar al momento de escribir éstas notas es el *Secure Hash Algorithm* por sus siglas en inglés SHA. La MD5 fue diseñada por Ron Rivest, toma como entrada un mensaje de longitud arbitraria y proporciona como salida una cadena binaria de 128 bits. El mensaje de entrada se procesa por bloques de 512 bits. La SHA fue diseñada por en NIST y se estableció como estándar en 1993. Recibe como entrada un mensaje con longitud menor a 2^{64} bits y como salida se obtiene una cadena binaria de 160 bits. Al igual que el MD5, se procesa en bloques de 512 bits [25].

2.3. Aritmética Modular

En criptografía a menudo se recurre a conceptos matemáticos del área de álgebra y teoría de números, para construir diversos esquemas criptográficos. En esta sección se describirán los conceptos matemáticos, necesarios para comprender los esquema de cifrado, que se utilizaron en el desarrollo de este trabajo terminal.

Definición 2.4 *Una relación de congruencia, denotada por el símbolo $\equiv$, se define como sigue: dados dos números enteros, $a, b \in \mathbb{Z}$ se dice que son congruentes módulo n, si y solo si n divide a $a - b$, dicho de otra manera si y sólo si $a - b = nq, q \in \mathbb{Z}$. Si a y b son congruentes módulo n se denota como $a \equiv b \bmod n$.*

Ejemplo 2.1 $27 \equiv 3 \bmod 6$ *ya que 6 divide a* $27 - 3 = 24$*;* $54 \equiv 3 \bmod 17$ *puesto que* $54 - 3 = 17(3)$*;* $-15 \equiv 6 \bmod 7$ *ya que* $-15 - 6 = 7(-3)$

La relación de congruencia es *reflexiva, transitiva y simétrica*,

- Reflexividad: $a \equiv a \bmod n$
- Transitividad: si $a \equiv b \bmod n$ y $b \equiv c \bmod n$ entonces $a \equiv c \bmod n$.

- Simetría: si $a \equiv b$ mód n entonces $b \equiv a$ mód n.

Una *clase de equivalencia* de un entero a, se define como el conjunto de enteros que son congruentes con a modulo n, donde n es un entero positivo.

Ejemplo 2.2 $3 \equiv 3$ mód 6 *ya que* $3 - 3 = 6(0)$*;* $35 \equiv 3$ mód 8 *y* $3 \equiv 11$ mód 8 *entonces* $35 \equiv 11$ mód 8*;* $159 \equiv 161$ mód 19 *por lo tanto* $161 \equiv 159$ mód 19

Definición 2.5 *El conjunto de enteros módulo n, denotado por $\mathbb{Z}_n$, se define como el conjunto de clases de equivalencia modulo n.*

$$\mathbb{Z}_n = \{0, 1, \ldots n-1\}$$

Dados dos enteros a y b, las operaciones de suma, resta y multiplicación en $\mathbb{Z}_n$ se llevan a cabo como sigue:

$$\begin{aligned} (a+b) \text{ mód } n &= [(a \text{ mód } n) + (b \text{ mód } n)] \text{ mód } n \\ (a-b) \text{ mód } n &= [(a \text{ mód } n) - (b \text{ mód } n)] \text{ mód } n \\ (a*b) \text{ mód } n &= [(a \text{ mód } n) * (b \text{ mód } n)] \text{ mód } n \end{aligned}$$

Definición 2.6 *Sea $a \in \mathbb{Z}_n$, el inverso multiplicativo de a módulo n es un entero $x \in \mathbb{Z}_n$, tal que $ax \equiv 1$ mód n. Si tal entero existe, entonces es único y se denota como a^{-1}, al cual se le denomina inverso multiplicativo módulo n.*

Para calcular el inverso multiplicativo, se utiliza el algoritmo extendido de Euclides. Para entender cómo funciona este algoritmo, primero se describirá el algoritmo de la división y el algoritmo de Euclides, este último es de utilidad para calcular el máximo común divisor.

Definición 2.7 (Algoritmo de la división) *Si a y b son enteros tales que $b \geq 1$ entonces al dividir a entre b se obtienen los enteros q y r tales que*

$$a = bq + r \quad 0 \leq r < b$$

El algoritmo de Euclides consiste en aplicar el algoritmo de la división repetidamente, el último residuo distinto de 0, será el máximo común divisor. Lo anterior expresado en notación matemática se ve de la siguiente manera.

Algoritmo de Euclides

$$\begin{aligned} a &= bq_0 + r_0 && 0 \leq r_0 < 0 \\ b &= r_0 q_1 + r_1 && 0 \leq r_1 < r_0 \\ r_0 &= r_1 q_2 + r_2 && 0 \leq r_2 < r_1 \\ &\vdots \\ r_{n-2} &= r_{n-1} q_n + r_n && 0 \leq r_n < r_{n-1} \end{aligned}$$

Ejemplo 2.3 *Dados $a = 42$ y $b = 30$ se tiene lo siguiente:*

$$\begin{aligned} 42 &= 30 \cdot 1 + 12 \\ 30 &= 12 \cdot 2 + 6 \\ 12 &= 6 \cdot 2 + 0 \end{aligned}$$

En este ejemplo el último residuo distinto de cero y por tanto el máximo común divisor de 42 y 30 es 6.

Para encontrar el inverso multiplicativo de manera eficiente, se utiliza una variante del algoritmo de Euclides, denominada: *algoritmo extendido de Euclides*. A continuación se muestra un ejemplo de cómo hacerlo.

Ejemplo 2.4 *Suponga que se desea obtener el inverso multiplicativo de 8 módulo 13. Inicialmente es posible aplicar el algoritmo de Euclides, tal y como se explicó en secciones anteriores.*

$$\begin{array}{rcrcr} 13 & = & 8 \cdot 1 & + & 5 \\ 8 & = & 5 \cdot 1 & + & 3 \\ 5 & = & 3 \cdot 1 & + & 2 \\ 3 & = & 2 \cdot 1 & + & 1 \\ 2 & = & 1 \cdot 2 & + & 0 \end{array}$$

Se despeja el primer residuo, en este caso 5:

$$5 = 13 + 8(-1)$$

observe que el lado derecho de la igualdad, no se simplifica. Se despeja el siguiente residuo, es decir 3:

$$3 = 8 + 5(-1)$$

en la expresión anterior, se sustituirá el 5:

$$3 = 8 + [13 + 8(-1)](-1)$$

simplificando se obtiene lo siguiente:

$$3 = 8 + 13(-1) + 8 = 8(2) + 13(-1)$$

El proceso anterior, se repite con el resto de los residuos.

$$\begin{array}{rcl} 2 & = & 5 + 3(-1) \\ & = & [13 + 8(-1)] + [8(2) + 13(-1)]\,(-1) \\ & = & 13 + 8(-1) + 8(-2) + 13 \\ & = & 13(2) + 8(-3) \end{array}$$

Finalmente, nos queda el último residuo, expresado en términos del 8 y del 13:

$$\begin{array}{rcl} 1 & = & 3 + 2(-1) \\ & = & [8(2) + 13(-1)] + [13(2) + 8(-3)](-1) \\ & = & 8(2) + 13(-1) + 13(-2) + 8(3) \\ & = & 8(5) + 13(-3) \end{array}$$

Recuerde que el último residuo es el máximo común divisor. Para poder hallar el inverso multiplicativo de 8 mód 13, se utilizará el siguiente resultado:

Teorema 2.1 *Si d es el máximo común divisor de a y b, entonces existen los enteros x_0 y y_0 tales que $d = ax_0 + by_0$.*

Ahora bien, dado $a \in \mathbb{Z}_n$, si se cumple que el máximo común divisor de a y el módulo n es 1, entonces aplicando el teorema anterior existen x_0 y y_0 tales que

$$1 = ax_0 + ny_0$$

si se reacomoda la expresión anterior:

$$1 - ax_0 = ny_0 \iff ax_0 \equiv 1 \text{ mód } n$$

es decir, el inverso multiplicativo de a mód n será justamente x_0, el entero que multiplica a a.

Continuando con el ejemplo, se obtuvo que $1 = 8(5) + 13(-3)$, reacomodando se tendrá lo siguiente:

$$1 - 8 \cdot 5 = 13(-3) \iff 8 \cdot 5 \equiv 1 \text{ mód } 13$$

es decir, 5 es el **inverso multiplicativo** de 8 módulo 13. Este hecho también se puede comprobar fácilmente si se lleva calcula (ab) mód n y se verifica que el resultado es igual a 1. En este caso:

$$5 \cdot 8 \text{ mód } 13 = 1$$

2.4. Primalidad

En varios algoritmos criptográficos, se requiere generar números primos. Y aunque parece una cuestión sencilla, en realidad no lo es.

Es conocida para la mayoría de las personas la definición de un número *primo*: *Un número primo es aquel que solamente es divisible entre sí mismo y la unidad.* Tal definición es fácilmente aplicable a números pequeños como 7, 13, 29. Sin embargo, si pensamos en un número con 10 dígitos, como 3283028807 ya no es tan fácil saber si tal número es primo o no y menos aún si se considera un número mayor como:

$$2326361098054328984297626699078322242 67$$

A lo largo de la historia se han descubierto varios métodos y resultados matemáticos que permiten averiguar si un número es primo o no. Un método muy sencillo, aunque poco práctico de usar en el caso de enteros muy grandes, es la criba de Eratóstenes. Este método nos sirve para hallar los números primos entre 2 y alguna cota superior. Consiste en listar a todos los enteros en el intervalo y posteriormente ir quitando de la lista los múltiplos de 2, de 3, de 4, de 5 etc. Por ejemplo, si buscamos a los números primos entre 1 y 50, tendríamos la siguiente lista:

$$2, 3, 4, 5, 6, 7, \ldots 48, 49, 50$$

se comienza eliminando a los múltiplos de 2, por lo que nos quedaría:

$$2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25,$$
$$27, 29, 31,33, 35, 37, 39, 41, 43, 45, 47, 49$$

luego se eliminarían los múltiplos de 3, quedando la lista como sigue:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47, 49$$

después los múltiplos de 5

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49$$

y así sucesivamente hasta que finalmente nos quedan los números primos en el intervalo de 2 a 50

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47$$

Sin embargo tal método no es aplicable a un número entero de 50 dígitos o más, pues tardaría muchísimo tiempo, aún usando una computadora. Para ello existen los algoritmos *probabilísticos*. Se les denomina así, puesto que dichos algoritmos indican con toda certeza si un algoritmo es compuesto, i.e, si no es primo, pero el algoritmo puede equivocarse indicando que un número es primo, aunque en realidad no lo sea. Sin embargo la probabilidad de que se equivoquen es muy pequeña. Tales algoritmos están basados en el teorema de Fermat.

Teorema 2.2 *(Teorema de Fermat) Sea p primo y $a \in \mathbb{Z}$ tal que $mcd(a, p) = 1$ entonces*

$$a^{p-1} \equiv 1 \text{ mód } p$$

Como se observa en el teorema si un número es primo efectivamente se cumplirá el resultado, pero puede ocurrir que un número sea compuesto y aún así el teorema puede cumplirse. Por ejemplo, $n = 341 = 11 \cdot 31$ y $a = 2$ satisface el teorema de Fermat puesto que $2^{340} \text{ mód } 341 = 1$.

Capítulo 3

Adversarios Clasificadores

Los adversarios clasificadores son programas de cómputo que se dedican a observar 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 a través de 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 [12, 13].
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 que un adversario clasificador solo es capaz de realizar ataques de texto cifrado (ciphertext only attack, en inglés). Como se mencionó en el capítulo anterior, en este ataque el adversario solo cuenta con los textos cifrados que va recopilando de un canal o base de datos. Posteriormente, el adversario utiliza estos textos cifrados para hacer un análisis criptográfico de cómo se comporta la técnica de cifrado y tratar 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.

3.0.1. Esquema Golle - Farahat

La única referencia que se tiene sobre un esquema criptográfico contra adversarios clasificadores es el que propusieron Golle y Farahat [13]. En este artículo se habla por primera vez de las características de este tipo de adversarios y se considera la posibilidad de utilizar un esquema de cifrado con un nivel de seguridad menor. Golle y Farahat proponen un protocolo

que hace uso de una función de cifrado, el cual 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 inglés. Para cifrar se utiliza una clave que se genera usando los datos de cabecera que acompañan al 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 tiene con una complejidad de cálculo moderadamente mas alta que las funciones hash estándar.
Este protocolo resulta inseguro para la criptografía moderna 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 se permite que el texto se vea como un texto en inglés pero indistinguible para los clasificadores y por lo tanto este clasifica incorrectamente el mensaje cifrado.

3.1. 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* por sus siglas en inglés). 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 comúnes son imágenes distorsionadas de cadenas cortas de caracteres. Para un humano es generalmente muy fácil recuperar la cadena original de la imagen distorsionada, pero es difícil para los algoritmos de reconocimiento de caracteres 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 $G(x)$

Figura 3.1: CAPTCHA

3.2. Esquema de Secreto Compartido de Shamir

El esquema de secreto compartido fue propuesto por Adi Shamir en 1997 [24]. 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 número primo $p \geq w + 1$ el cual define al conjunto $\mathbb{Z}_p$.

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

1. Se seleccionan w elementos distintos de cero del conjunto $\mathbb{Z}_p$ denotados como x_i donde $1 \leq i \leq w$.

2. Se seleccionan $u - 1$ elementos aleatorios de $\mathbb{Z}_p$ denotados como $a_1, ..., a_{u-1}$.

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

$$y(x) = K + \sum_{j=1}^{u-1} a_j x^j \text{ mód } \quad p \tag{3.1}$$

   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.

Ejemplo 3.1 *Si se considera el conjunto $\mathbb{Z}_{11}$ y se desea compartir el secreto $K = 8$, entre 5 participantes, de tal manera que solo cuando se reúnan cualesquiera 2 de ellos sea posible recuperar K, es decir $w = 5$ y $u = 2$.*

Se seleccionan los $u - 1$ elementos del conjunto $\mathbb{Z}_{11}$, puesto que $u = 2$ en este caso solo hay que escoger un elemento: $a_1 = 5$.

A continuación se seleccionan w elementos de $\mathbb{Z}_{11}$, por ejemplo $x_1 = 2, x_2 = 7, x_3 = 9, x_4 = 10, x_5 = 3$.

Posteriormente, se calcula el conjunto de elementos y_i por medio de la ecuación

$$y_i = k + \sum_{j=1}^{u-1} a_j x_i^j \text{ mód } \quad p$$

En el caso del ejemplo, la ecuación anterior queda como sigue:

$$y_i = K + a_1 x_i \text{ mód } \quad p$$

Y se obtienen los siguientes valores:

$$\begin{aligned} y_1 &= 8 + 5(2) \text{ mód } 11 = 7, \\ y_2 &= 8 + 5(7) \text{ mód } 11 = 10, \\ y_3 &= 8 + 5(9) \text{ mód } 11 = 9, \\ y_4 &= 8 + 5(10) \text{ mód } 11 = 3, \\ y_5 &= 8 + 5(3) \text{ mód } 11 = 1. \end{aligned}$$

Finalmente, se tienen los pares $S = \{(2,7),(7,10),(9,9),(10,3),(3,1)\}$

Para recuperar la llave K *es necesario seleccionar* $u = 2$ *pares del conjunto* S*, por ejemplo* $A_2(7,10)$*,* $A_4(10,3)$*. Con estos pares se puede crear el siguiente sistema de ecuaciones:*

$$\begin{aligned} a_0 + a_1 x_2 &= y_2 \\ a_0 + a_1 x_4 &= y_4 \end{aligned}$$

Es importante notar, que en tal sistema de ecuaciones, las incógnitas son a_0 *y* a_1*, las cuales son desconocidas para los* w *participantes. Para el esquema de secreto compartido de Shamir, es de particular interés* a_0*, ya que* $a_0 = K$*. Al sustituir los pares* A_2 *y* A_4 *en el sistema de ecuaciones anterior, se tiene:*

$$a_0 + 7a_1 = 10 \tag{3.2}$$

$$a_0 + 10a_1 = 3 \tag{3.3}$$

Para resolver este sistema se puede utilizar cualquiera de los métodos comúnes que se usan en álgebra, solo que respetando el conjunto $\mathbb{Z}_p$*, en este caso se resolverá por el método suma y resta.*
Multiplicamos la ecuación (3.2) por -1 *y obtenemos:*

$$-a_0 - 7a_1 = -10 \tag{3.4}$$

sumamos la ecuación (3.3) con (3.4) dándonos como resultado:

$$3a_1 = 4$$

de donde es posible despejara$_1$

$$a_1 = \frac{4}{3} \tag{3.5}$$

Puesto que 4 es el inverso multiplicativo de 3, la ecuación (3.5) queda de la siguiente forma

$$a_1 = (4)(4) = 16 \text{ mód } 11 = 5$$

Sustituimos a_1 *en la ecuación (3.3)*

$$a_0 + 10(5) = 3$$

Simplificamos y despejamos a_0

$$\begin{aligned} a_0 + (50 \text{ mód } 11) &= 3 \\ a_0 = -3 \text{ mód } 11 &= 8 \end{aligned}$$

Como $a_0 = 8$ *podemos ver que se recuperó a* K *exitosamente ya que* $a_0 = K$*.*

3.2.1. Polinomio de interpolación de Lagrange

Para ejemplificar el método de Lagrange se usará el mismo ejercicio que en el método de Shamir.

Ejemplo 3.2 *Si se considera el conjunto* $\mathbb{Z}_{11}$ *y se desea compartir el secreto* $K = 8$*, entre 5 participantes, de tal manera que solo cuando se reúnan cualesquiera 2 de ellos sea posible recuperar* K*, es decir* $w = 5$ *y* $u = 2$*.*

Se seleccionan los $u - 1$ *elementos del conjunto* $\mathbb{Z}_{11}$*, puesto que* $u = 2$ *en este caso solo hay que escoger un elemento:* $a_1 = 5$*.*

A continuación se seleccionan w *elementos de* $\mathbb{Z}_{11}$*, por ejemplo* $x_1 = 2, x_2 = 7, x_3 = 9, x_4 = 10, x_5 = 3$*.*

Posteriormente, se calcula el conjunto de elementos y_i *por medio de la ecuación*

$$y_i = k + \sum_{j=1}^{u-1} a_j x_i^j \text{ mód } \quad p$$

En el caso del ejemplo, la ecuación anterior queda como sigue:

$$y_i = K + a_1 x_i \text{ mód } \quad p$$

Y se obtienen los siguientes valores:

$$\begin{aligned} y_1 &= 8 + 5(2) \text{ mód } 11 = 7, \\ y_2 &= 8 + 5(7) \text{ mód } 11 = 10, \\ y_3 &= 8 + 5(9) \text{ mód } 11 = 9, \\ y_4 &= 8 + 5(10) \text{ mód } 11 = 3, \\ y_5 &= 8 + 5(3) \text{ mód } 11 = 1. \end{aligned}$$

Finalmente, se tienen los pares $S = \{(2, 7), (7, 10), (9, 9), (10, 3), (3, 1)\}$

Para recuperar la llave K *es necesario seleccionar* $u = 2$ *pares del conjunto* S*, por ejemplo* $A_2(7, 10)$*,* $A_4(10, 3)$*. Con estos pares se puede crear el siguiente sistema de ecuaciones:*

$$\begin{aligned} a_0 + a_1 x_2 &= y_2 \\ a_0 + a_1 x_4 &= y_4 \end{aligned}$$

Es importante notar, que en tal sistema de ecuaciones, las incógnitas son a_0 *y* a_1*, las cuales son desconocidas para los* w *participantes. Para el esquema de secreto compartido de Shamir, es de particular interés* a_0*, ya que* $a_0 = K$*. Al sustituir los pares* A_2 *y* A_4 *en el sistema de ecuaciones anterior, se tiene:*

$$a_0 + 7a_1 = 10 \tag{3.6}$$

$$a_0 + 10a_1 = 3 \tag{3.7}$$

Para resolver este sistema se puede utilizar cualquiera de los métodos comúnes que se usan en álgebra, solo que respetando el conjunto $\mathbb{Z}_p$*, en este caso se resolverá por el polinomio de interpolación de Lagrange.*

$$a_x = \sum_{j=1}^{u} y_i \prod \frac{x - x_j}{x_i - x_j} mod \quad p \tag{3.8}$$

Al sustituir los valor A_2 *y* A_4 *en la ecuación anterior se puede calcular* a_0*. Pero una simplificación es posible, si se toma en cuenta que no es necesario conocer todo el polinomio,*

bastará con sólo deducir el termino a_0. Por lo tanto en la ecuación (3.8) se sustituye $x = 0$ quedando de la siguiente forma.

$$l_i = \prod \frac{-x_j}{x_i - x_j} \tag{3.9}$$

$$a_0 = \sum_{j=1}^{u} y_i \, l_i \, mod \quad p \tag{3.10}$$

Sustituyendo el par A_2 en la ecuación (3.9), se tiene:

$$l_2 = \frac{-x_4}{x_2 - x_4} = \frac{-10}{7 - 10} = \frac{-10}{-3} \, mod \, 11 = \frac{1}{8} \tag{3.11}$$

Se realiza la misma sustitución en la ecuación (3.9) pero con el par A_4 dándonos como resultado:

$$l_4 = \frac{-x_2}{x_4 - x_2} = \frac{-7}{10 - 7} = \frac{-7}{3} \, mod \, 11 = \frac{4}{3} \tag{3.12}$$

Como la ecuación (3.10) es una sumatoria podemos desarrollarla, dándonos como resultado:

$$a_0 = (y_2 \, l_2 + y_4 \, l_4) \, mod \, 11 \tag{3.13}$$

realizamos la sustitución de y_2, l_2, y_4 y l_4.

$$a_0 = (10 \, (\frac{1}{8}) + 3 \, (\frac{4}{3})) \, mod \, 11 \tag{3.14}$$

Puesto que 7 es el inverso multiplicativo de 8 y 4 es el inverso multiplicativo de 3, la ecuación (3.14) queda de la siguiente forma

$$a_0 = ((10) \, (7) + (3) \, (4)) \, mod \, 11 = 8 \tag{3.15}$$

Como $a_0 = 8$ podemos ver que se recuperó a K exitosamente ya que $a_0 = K$.

3.3. Esquema Díaz - Chakraborty

Otro esquema para combatir a los adversarios clasificadores fue propuesto en 2012 por Díaz y Chakraborty [12]. El esquema Díaz-Chakraborty utiliza CAPTCHAs y un algoritmo de clave secreta, para proteger el correo electrónico. Para obtener la clave se genera una cadena al azar, a la se le aplica una función hash, con esta clave se cifra el mensaje. Tanto el mensaje cifrado como el CAPTCHA se envían al receptor. El receptor debe resolver el CAPTCHA, para obtener la cadena, aplicarle la función hash y así obtener la clave de cifrado. Puesto que un adversario clasificador es un programa de cómputo, no podrá resolver un CAPTCHA y por tanto no podrá obtener la clave de cifrado. Este esquema se muestra en la Figura 3.2.

Contemplando que es muy común que el usuario no consiga resolver el CAPTCHA Díaz y Chakraborty propusieron una variante del esquema anterior, el cual se describe a continuación. Se genera una cadena de caracteres aleatoriamente llamada STR la cual se codifica

Protocol $\mathbb{P}(x)$

1. $k \xleftarrow{\$} \mathsf{STR}$;
2. $k' \leftarrow G(k)$;
3. $K \leftarrow H(k)$;
4. $c \leftarrow E_K(x)$;
5. **return** (c, k')

Figura 3.2: Protocol Díaz-Chakraborty.

a un valor entero. El valor entero es dividido en 5 pares (x, k') por medio del algoritmo de Secreto Compartido, cada uno de los elementos k' de los pares generados es decodificado a su correspondiente valor en cadena de caracteres para posteriormente ser convertidos en CAPTCHAS. Para finalizar la cadena STR se introduce en una función Hash para generar la llave K. Con esta llave se cifra el mensaje de correo y se envía junto con los pares de $(x, CAPTCHA)$. Este esquema se puede observar en la Figura 3.3.

Protocol $\mathbb{P}'(x)$

1. $k \xleftarrow{\$} \mathsf{STR}$;
2. $k' \leftarrow \mathsf{ENCD}(k, 0)$;
3. $\{(x_1, k'_1), \ldots, (x_w, k'_w)\} \leftarrow \mathsf{SHARE}^p_{u,w}(k')$;
4. **for** $i = 1$ **to** w;
5. $\quad (k_i, \lambda_i) \leftarrow \mathsf{ENCD}^{-1}(k'_i)$;
6. $\quad c_i \leftarrow G(k_i)$;
7. **end for**
8. $K \leftarrow H(k)$;
9. $C \leftarrow E_K(x)$;
10. **return** $[C, \{(x_1, c_1, \lambda_1), \ldots, (x_w, c_w, \lambda_w)\}]$

Figura 3.3: Variante del protocolo Díaz-Chakraborty

Este nuevo esquema se creó pensando en que el usuario pueda tener más oportunidades de recuperar el mensaje cifrado y esto sucede gracias a el algoritmo de Secreto Compartido, ya que no este podemos tener la misma llave repartida en n CAPTCHAS. A continuación se describe las funciones ENCD y ENCD^{-1}, cuyo propósito es convertir una cadena de caracteres a enteros y viceversa.

3.3.1. Codificación 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 cadena 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 $v \longleftarrow toInt(\Psi)$

Ejemplo:
Tenemos la cadena $STR =$ 'ABC' de la cual cambiaremos cada caracter por su correspondiente valor entero en AL quedando de la siguiente manera $\alpha = \{0, 1, 2\}$

Ahora cada uno de los elementos de α lo convertiremos a su correspondiente representacion binaria, $bin_6(0) = 000000, bin_6(1) = 000001, bin_6(2) = 000010$ y concatenamos cada una quedando $\Psi = 000000000001000010$.

La cadena binaria Ψ se convertirá en un entero $v = toInt(\Psi)$ que da como resultado $v = 66$. El entero v que obtenemos es el valor entero.

3.3.2. Decodificación de enteros a caracteres

También 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. Separamos 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 se convierte a su correspondiente caracter en AL se concatenan para generar la cadena de caracteres final.

Ejemplo:
El entero $v = 66$ se representa como una cadena de 18 bits $z = 000000000001000010$, la cual se divide en sub cadenas 6 bits quedando $z_0 = 000000, z_1 = 000001, z_2 = 000010$, para cada uno de estos números binarios se procede a convertirlo en un entero $toInt(z_0) = 0, toInt(z_1) = 1, toInt(z_2) = 2$, por último estos son intercambiados por sus correspondientes caracteres en AL y concatenados resultando en $s =$ 'ABC'

Capítulo 4

Tecnologías usadas

Tomando en cuenta la información ya vertida en este documento, a continuación se explicará detalladamente la propuesta de solución.
En la figura 4.1 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 CAPTCHA y se recalcula la clave de descifrado.
- Se descifra el mensaje y se le muestra al usuario.

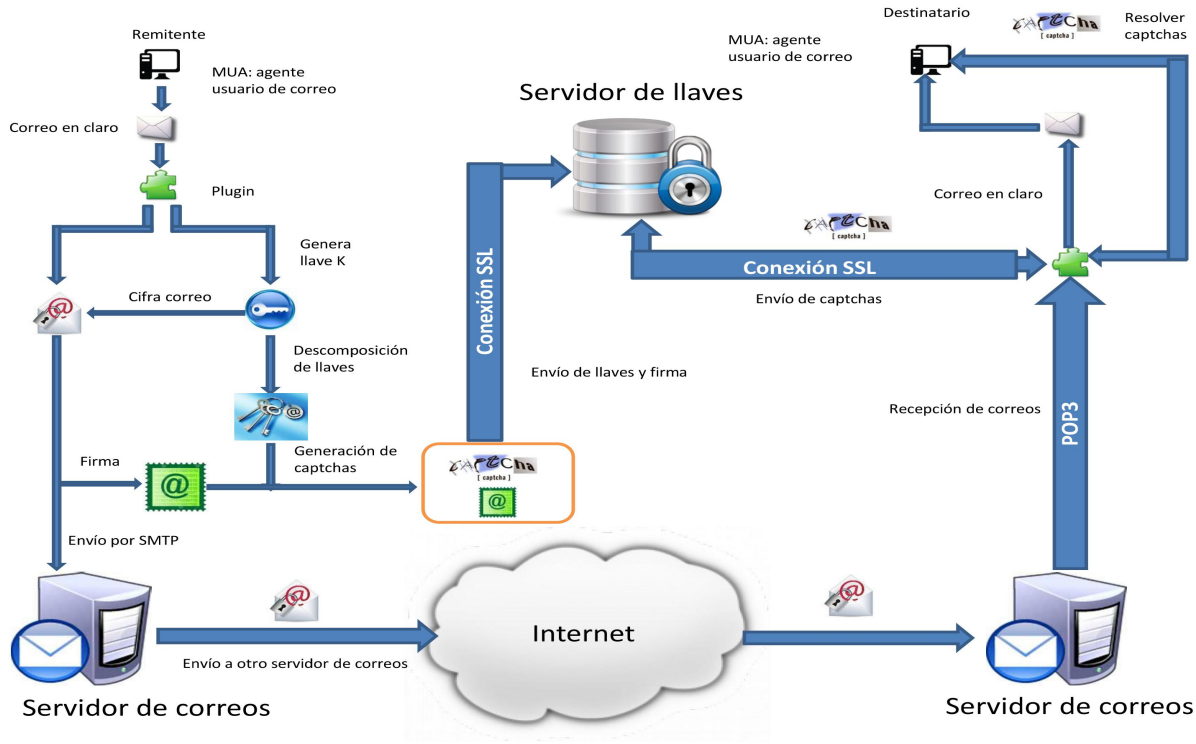


Figura 4.1: Diagrama General del sistema

4.1. 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.

4.1.1. 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. [2]
  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. [4]

- 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. [6]
  La única desventaja que se encontró 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. [3]

- 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***. [5]

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.

4.1.2. 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 [5]:

- Lenguaje Python
- HTML 5
- CSS3

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

4.1.3. Tipos de CAPTCHAS

En el esquema de cifrado es necesario esconder la palabra que genera la clave en un CAPTCHA para ser enviado a otro usuario y descifre el mensaje, pero existen varios tipos de CAPTCHAS que se pueden utilizar [22].
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 seran almacenadas 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.

4.1.4. 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. [19]
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 5

Análisis y Diseño

5.1. Diagramas de caso de uso

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

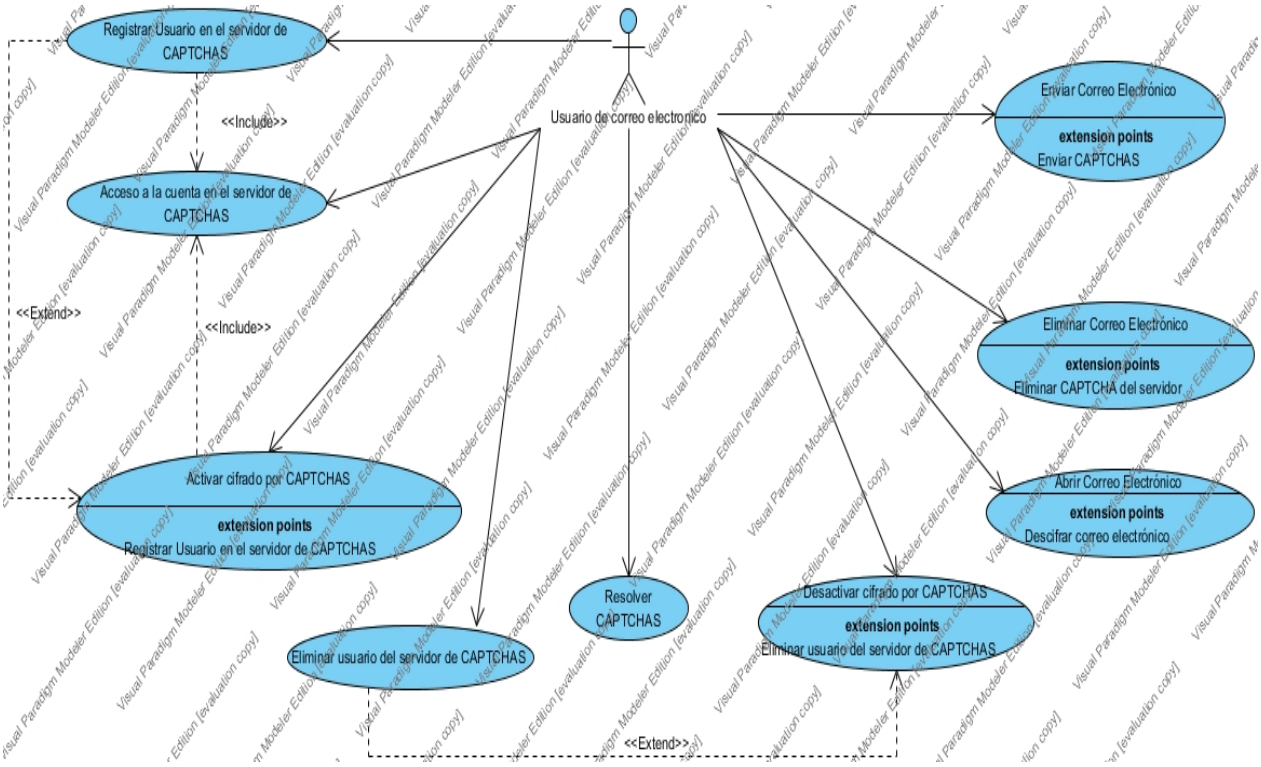


Figura 5.1: Diagrama General de caso de uso

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

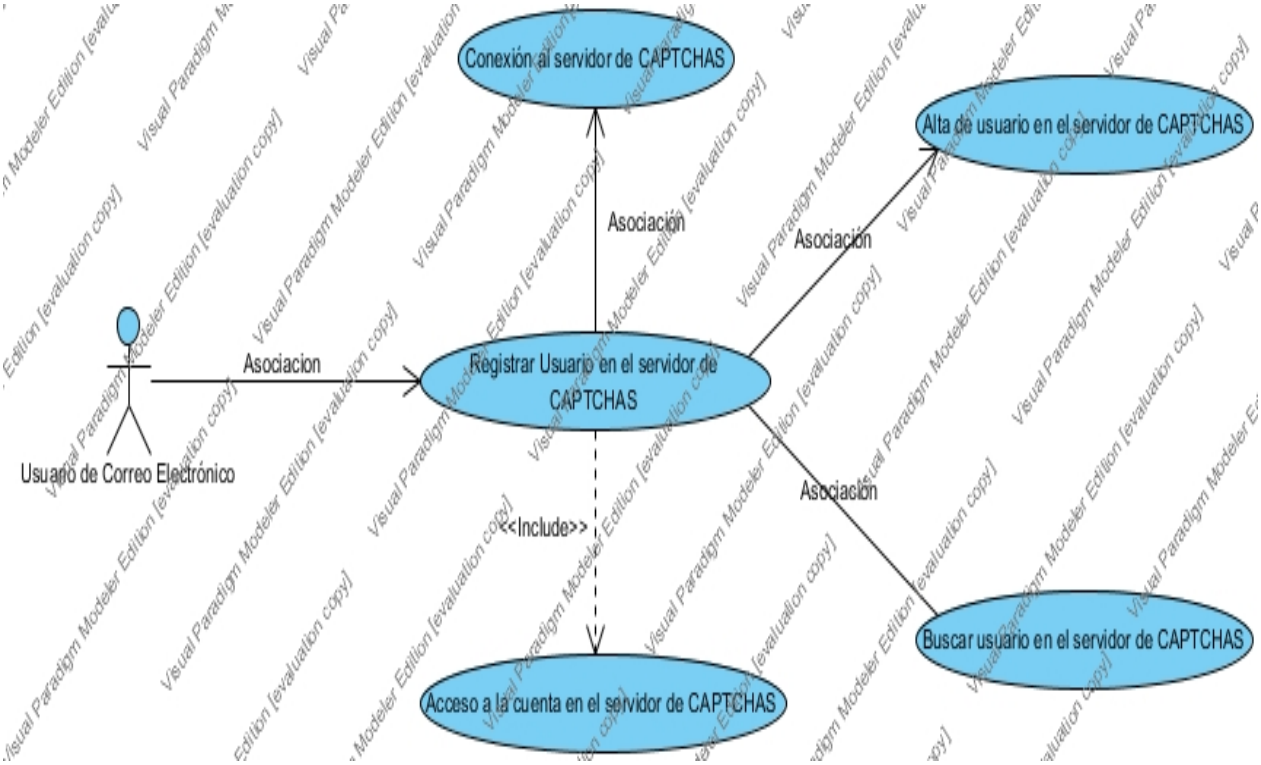


Figura 5.2: Diagrama de casos de uso CU2 Registrar usuario en el servidor de CAPTCHAS

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 5.1: Descripción CU2.

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

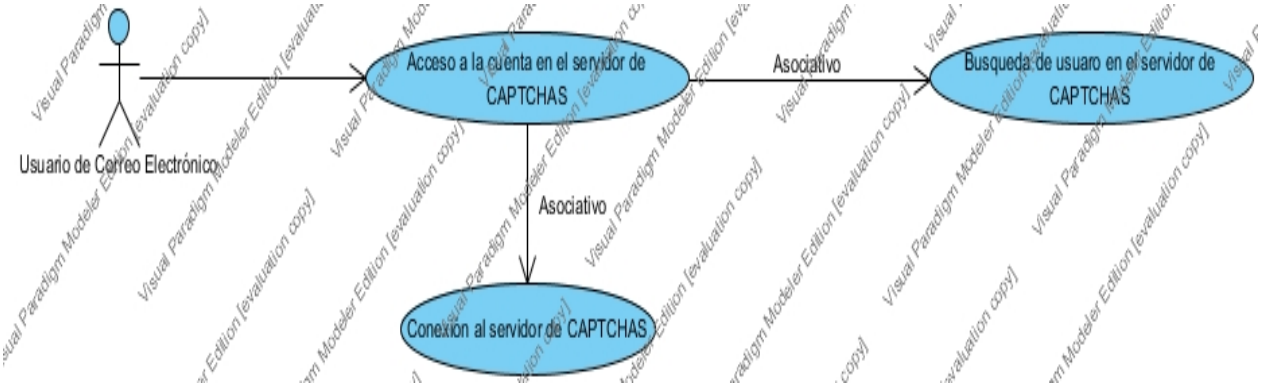


Figura 5.3: Diagrama de casos de uso CU3 Acceso a la cuenta en el servidor de CAPTCHAS

Caso de Uso	CU3 Acceso a la cuenta en el servidor de CAPTCHAS		
Actor	Actor 1. Usuario de correo electrónico		
Descripción	Describe los pasos necesarios para dar acceso al los archivos guardados en el servidor de CAPTCHAS.		
Pre-condiciones	Tener una cuenta de correo electrónico.		
Post-condiciones	Activación del modulo de cifrado por CAPTCHAS.		
Puntos de inclusión	Desempaquetar correo electrónico		
Puntos de extensión	Descifrar correo electrónico		
Flujo principal		Actor/Sistema	Acción a realizar
	1	Actor	El usuario ingresa su correo electrónico, nombre de usuario y contraseña.
	2	Sistema	El cliente de correo abre una conexión con el servidor de CAPTCHAS.<FA01. Servidor no responde.>
	3	Sistema	El cliente de correo envía los datos ingresados por el actor al servidor de CAPTCHAS.<FA02. Datos erróneos.>
	4	Sistema	Envía la respuesta satisfactoria al usuario.
			Fin del flujo principal.
	FA01 - Servidor no responde..		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Despliega el siguiente mensaje "No se pudo establecer comunicación con el servidor de CAPTCHAS"

			El flujo continua en el paso 1 del flujo principal.
			Fin del flujo alternativo
	FA02 - Datos erróneos..		
Flujo alternativo		Actor/Sistema	Acción a realizar
	1	Sistema	Despliega el siguiente mensaje "Los datos ingresados son inválidos"
			El flujo continua en el paso 1 del flujo principal.
			Fin del flujo alternativo

Tabla 5.2: Descripción CU3.

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

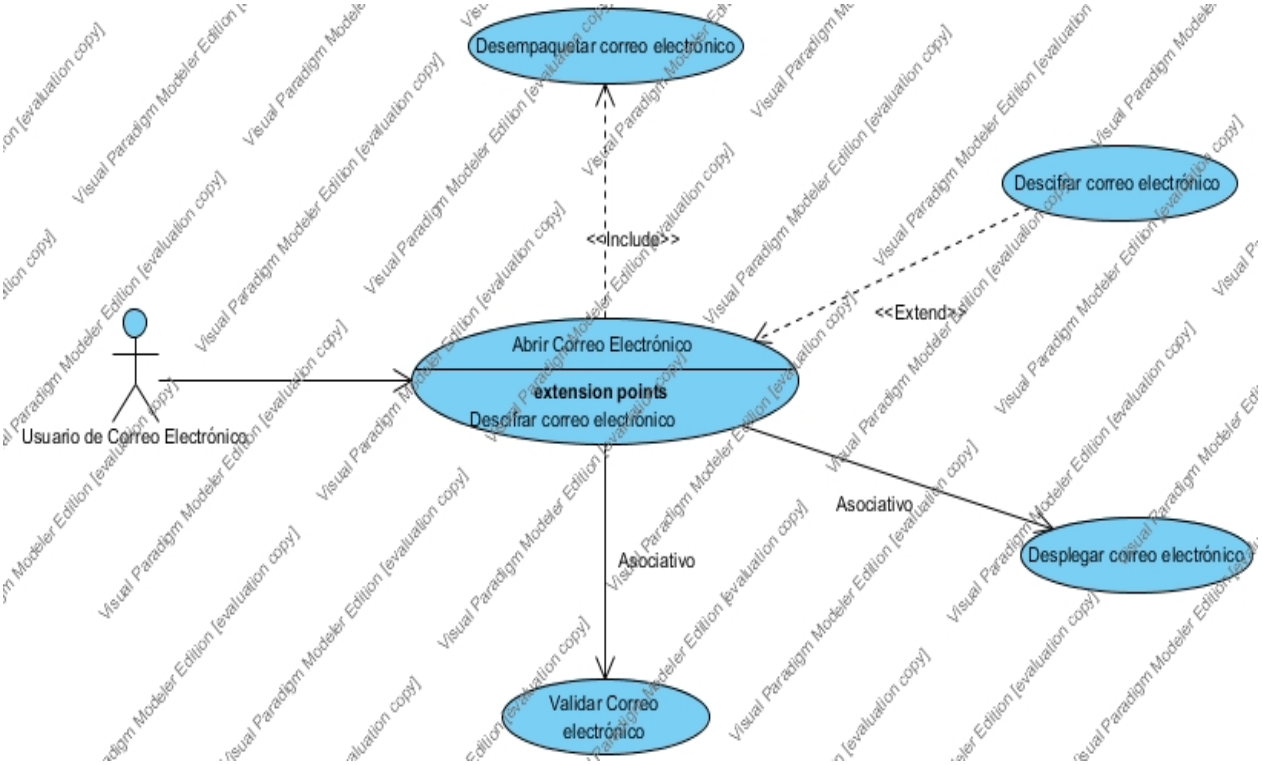


Figura 5.4: Diagrama de casos de uso CU4 Abrir Correo Electrónico.

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 5.3: Descripción CU4.

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

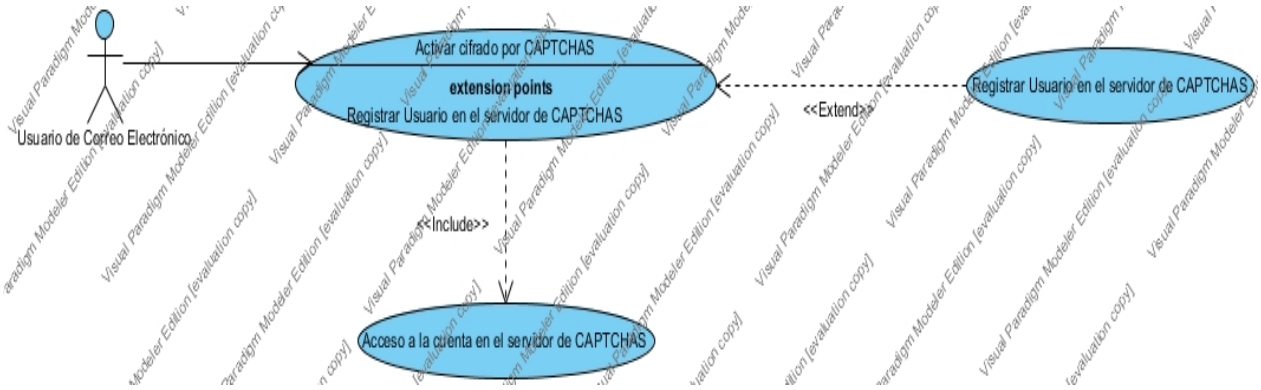


Figura 5.5: Diagrama casos de uso CU5 Activar cifrado por CAPTCHAS.

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 5.4: Descripción CU5.

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

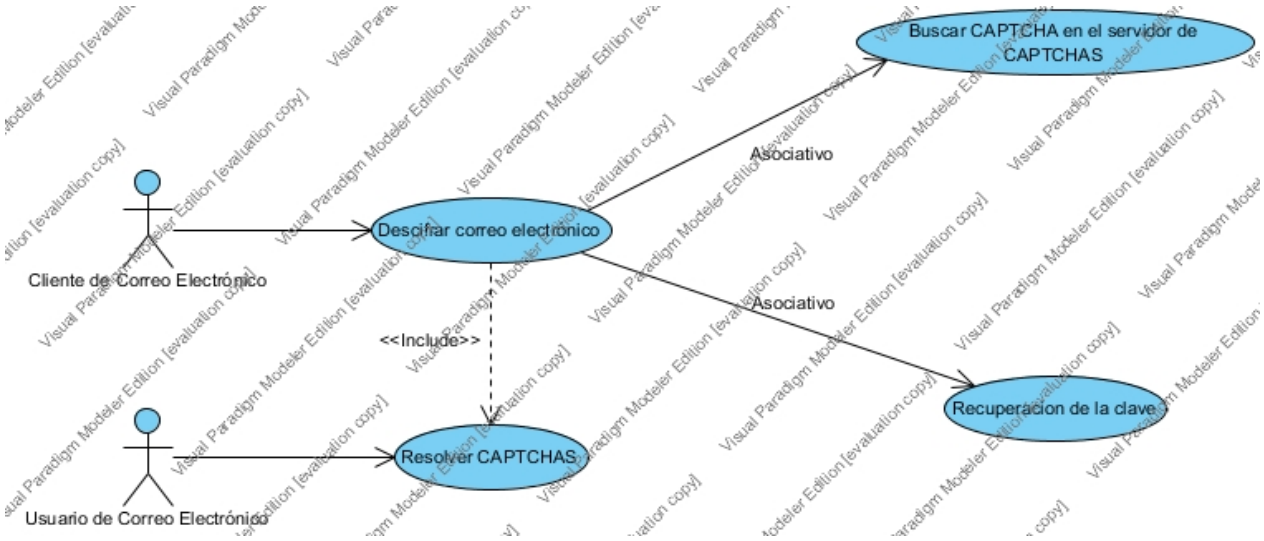


Figura 5.6: Diagrama de casos de uso CU6 Descifrar correo electrónico.

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 5.5: Descripción CU6.

5.1.6. Diagrama de casos de uso CU7 Enviar CAPTCHAS

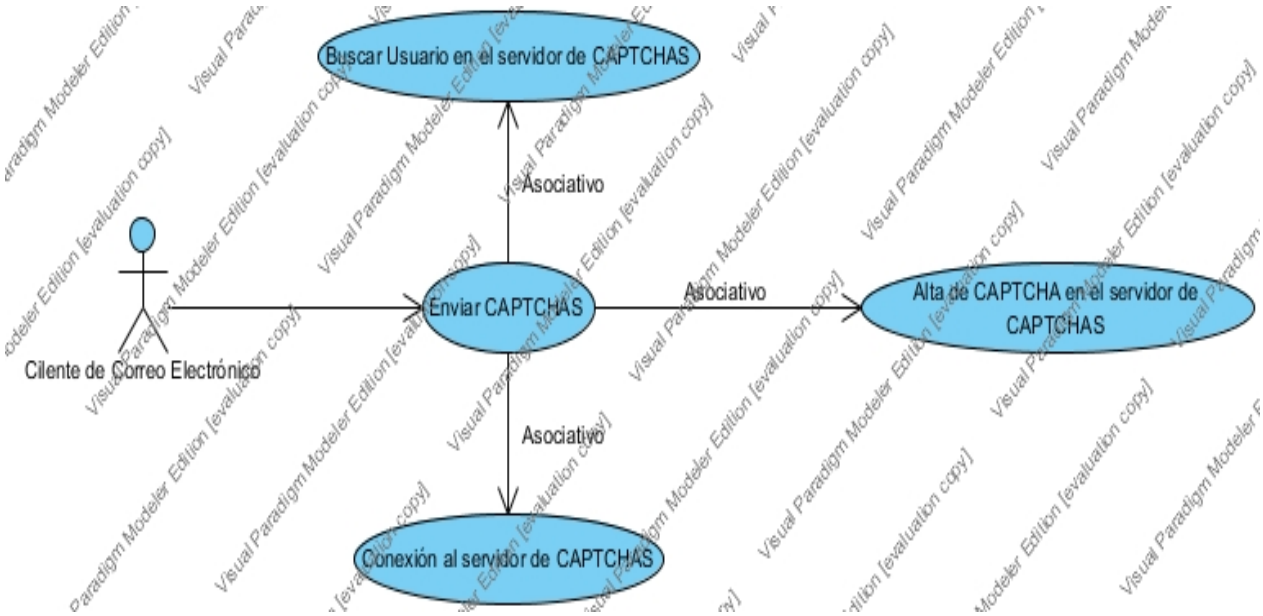


Figura 5.7: Diagrama de casos de uso CU7 Enviar CAPTCHAS

Caso de Uso	CU7 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 5.6: Descripción CU7.

5.1.7. Diagrama de casos de uso CU8 Enviar correo electrónico.

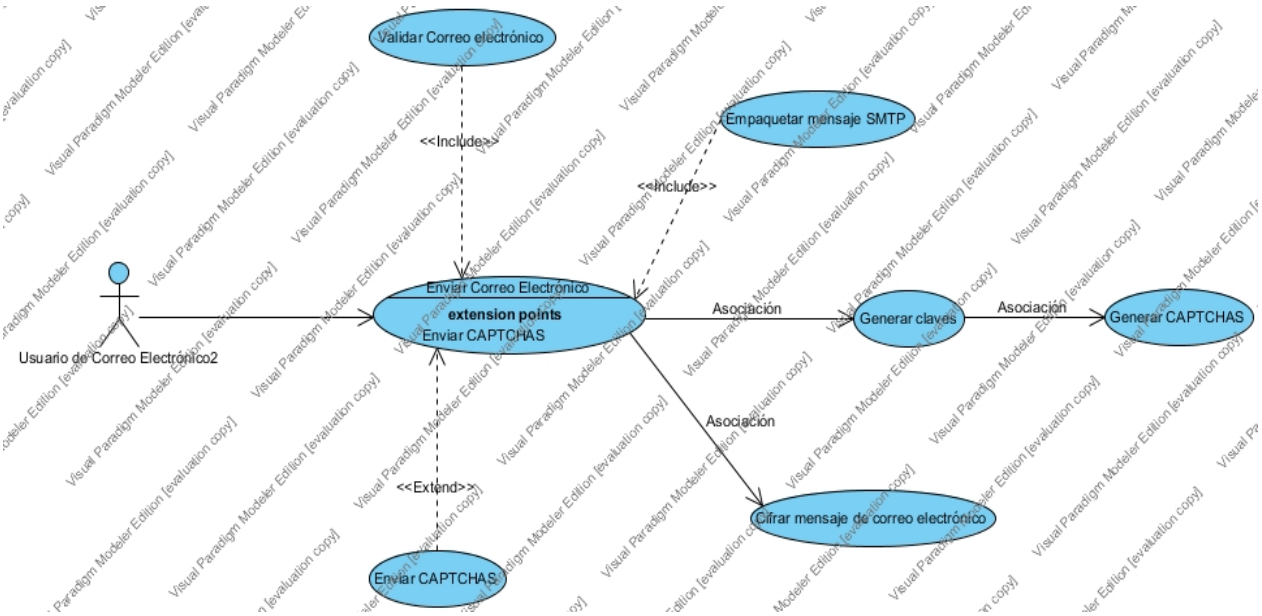


Figura 5.8: Diagrama de casos de uso CU8 Enviar correo electrónico.

Caso de Uso	CU8 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 5.7: Descripción CU8.

5.2. 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.

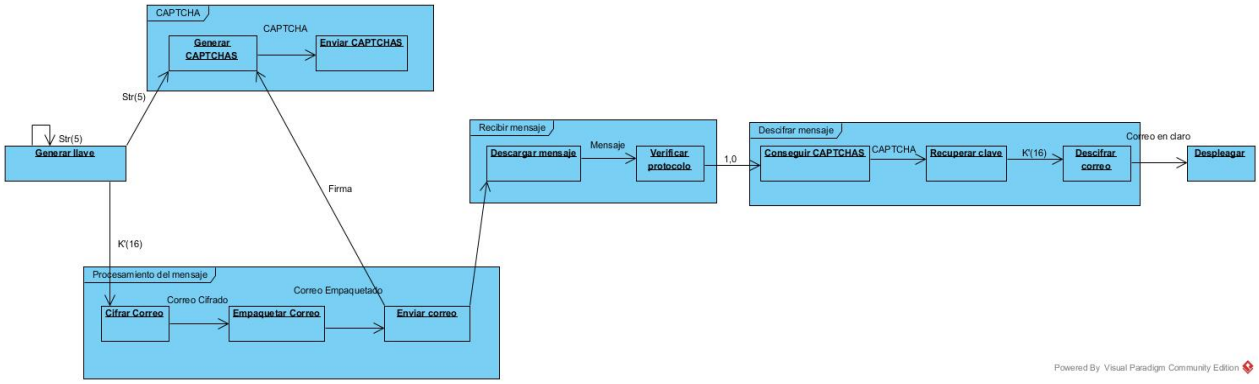


Figura 5.9: Diagrama a bloque 0 general del sistema

	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 5.8: Diagrama a bloques 0 general

5.2.1. Diagrama a bloques 1 Generar clave

Figura 5.10: 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 de enteros módulo 67 -.,+*[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 5.9: Diagrama a bloques 1 general clave

5.2.2. Diagrama a bloques 2 Cifrado

	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 5.10: Diagrama a bloques 2 Cifrar Correo

5.2.3. Diagrama a bloques 3 Empaquetar Correo

Figura 5.11: 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 5.11: Diagrama a bloques 3 Empaquetar Correo

5.2.4. Diagrama a bloques 4 Enviar correo

Figura 5.12: 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 5.12: Diagrama a bloques 4 Enviar correo

5.2.5. Diagrama a bloques 5 Generar CAPTCHA

Figura 5.13: 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 5.13: Diagrama a bloques 5 Generar CAPTCHA

5.2.6. Diagrama a bloques 6 Enviar CAPTCHAS

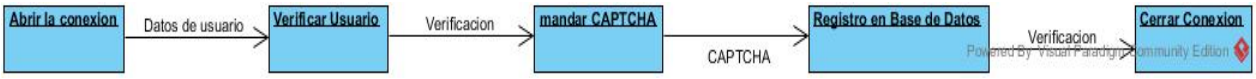


Figura 5.14: Diagrama a bloques 6 Enviar CAPTCHAS (Usuario existente)

	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 5.14: Diagrama a bloques 6 Enviar CAPTCHAS (Usuario existente)

5.2.7. Diagrama a bloques 7 Enviar CAPTCHAS

Figura 5.15: 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 5.15: Diagrama a bloques 7 Enviar CAPTCHAS (Usuario inexistente)

5.2.8. Diagrama a bloques 8 Descargar mensaje

Figura 5.16: 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 5.16: Diagrama a bloques 8 Descargar mensaje

5.2.9. Diagrama a bloques 9 Verificar protocolo

Figura 5.17: 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 5.17: Diagrama a bloques 9 Verificar protocolo (con protocolo válido)

5.2.10. Diagrama a bloques 10 Verificar protocolo

Figura 5.18: Diagrama a bloques 10 Verificar protocolo (con protocolo inválido)

	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 5.18: Diagrama a bloques 10 Verificar protocolo (con protocolo inválido)

5.2.11. Diagrama a bloques 11 Conseguir CAPTCHAS

Figura 5.19: 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 5.19: Diagrama a bloques 11 Conseguir CAPTCHAS (Usuario existente)

5.2.12. Diagrama a bloques 12 Recuperar clave

Figura 5.20: 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 5.20: Diagrama a bloques 12 Recuperar clave

5.2.13. Diagrama a bloques 13 Descifrar correo

	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 5.21: Diagrama a bloques 13 Descifrar correo

Capítulo 6

Desarrollo de prototipos

6.1. 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 éstas es estándar de alguna biblioteca de C o C++, así que se tendrá 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.

6.2. 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. El codigo fuente de las funciones se encuentra en el Anexo 1.

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 logró 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.

6.3. Prototipo 3

Objetivo Generar una imagen 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 ingresa 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 las bibliotecas de imagenes encontradas no hacen rotaciones, inclinaciones, cambio de tamaños y colores variables para generar CAPTCHAS, por lo tanto las bilbiotecas encontradas tenían que adaptarse a la funcionalidad del prototipo.
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 decidió cambiar de lenguaje de programación.

6.4. 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 se levantaron en una computadora con un sistema operativo Xubuntu 15.04; primero se instaló el servidor HTTP [8], posteriormente se pasó a la instalación del servidor DNS y configuración de un dominio [23]; se seguió 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 [9].
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             ; minimo (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             ; minimo (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”
- Por último se ejecuta el siguiente comando “nslookup www.correocifrado.edu” y nos dará un resumen de los DNS configurados.

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 último 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 seleccionó 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 configuró en el servidor DNS “correocifrado.net”
- Regresará al menú principal y se teclea la letra Q para salir de la configuración.
- Preguntará si queremos guardar los cambios y se teclea la letra Y.
- Por último 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 podrá 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 creará 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.

6.5. Prototipo 5

Objetivo: Familiarizarse con el uso de la tecnología y estructura del cliente de correo electrónico Thunderbird.
Se planeó 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 documentación en la página de desarrollo de Mozilla resulta evidente que 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.

6.6. 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 pública directamente en su página oficial, además de contar con breves tutoral de como usarse. Se implementó sobre la interfaz 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 métodos que el mismo N1 proporciona.

- Comunicación con el servidor de CAPTCHAS: En complemento creado por nosostros se generó 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 implementadas 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 después de que ya se ejecutó 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.

6.7. 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 número aleatorio del 0 al 63 y este se manda a otra función que lo mapea a su caracter correspondiente en el conjunto de enteros $AL = \{A-Z\} \cup \{a-z\} \cup \{0-9\} \cup \{+,/\}$

```
  var map;map = [];

map=["A","B","C","D","E","F","G","H","I","J","K","L","M","N"
,"O","P","Q","R","S","T","U","V","W","X","Y","Z","a","b","c"
,"d","e","f","g","h","i","j","k","l","m","n","o","p","q","r"
,"s","t","u","v","w","x","y","z","0","1","2","3","4","5","6"
,"7","8","9","+","/"];
//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 usará 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 métodos 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
Oransa2011RashaOsama Oransa2012Osama Oransa2011RashaOsama
Oransa2012Osama Oransa2011Rasha";
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.

6.8. Prototipo 8

Objetivo: Crear una biblioteca en lenguaje Python que contenga el esquema de cifrado por CAPTCHA. El cliente de correo electrónico ***Thunderbird*** 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 métodos 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"número de 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. El código fuente se encuentra en el Anexo 2.

- 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 conjunto de enteros $AL = \{A-Z\} \cup \{a-z\} \cup \{0-9\} \cup \{+, /\}$

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

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

- Codificar: La función Codificar toma como parámetro, una cadena de caracteres y la transforma en un número entero, primero caracter a caracter se busca su correspondiente número entero en el conjunto de enteros AL posteriormente este se pasa a una representación en 6 bits y por último se convierte a un número entero.

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

- Generar Partes: Se reciben como parámetro el conjunto de enteros módulo Zp , el número de partes en que se dividirá el secreto w, el número 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.

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

- Cifrar: Este método recibe como parámetro el cuerpo del mensaje, el asunto y de manera opcional recibe la opción de cifrado, el número de caracteres de los CAPTCHAS, el número de partes en que se divide el secreto, y el número 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 conjunto de enteros módulo 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 número la semilla por medio de Codificar, con este número se generan los pares del secreto compartido, estos son mapeados a cadena de caracteres y convertidos en CAPTCHAS, se crea la llave y por último 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 número x y su correspondiente imagen CAPTCHA.

- Descifrar: este método hace le proceso inverso que el método cifrar, este recibe como parámetro 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 parámetro CAPTCHA una lista que contiene los pares $x, CAPTCHA$ de este se obtiene el tamaño del CAPTCHA y se calcula el conjunto de enteros módulo Zp, con esto se toman los CAPTCHAS y se mapean a su representación numérica por medio de Codificar, los pares de números se ingresan al método que calcula el secreto compartido, lo arrojado por este método 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.

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

6.9. Prototipo 9

Objetivo: Dar de alta un servidor en el lenguaje PHP donde se alojen, busquen y se realice el intercambio de CAPTCHAS entre los usuarios que utilicen los protocolos P y P' del esquema Díaz – Chakraborty.

En éste prototipo se tienen implementados por separado 3 archivos PHP que se encargan, cada uno, de dan de alta a los usuarios que desean utilizan el esquema "On Securing Communication from Profilers"; subir al servidor los CAPTCHAS que contiene la clave de descifrado de los mensajes redactados por los usuarios previamente registrados; y realizan la descarga de los mismos al momento de que el usuario destino desee recuperar el mensaje, esto con el fin de poder hacer el intercambio y resguardo de las claves para el descifrado de los mensajes de correo electrónico entre usuarios registrados. Tambien se cuenta con una base de datos que nos guarda la relación entre los mensajes enviados y los CAPTCHAS que contienen la clave para ser descifrados. El código fuente de encuentra en el anexo 3

- Alta de usuarios: El servidor cuenta con un archivo PHP donde recibe las peticiones de los nuevos usuarios que quieran hacer uso de los protocolos P y P' bajo el esquema propuesto en este documento, este archivo PHP recibe un formulario HTML llenado previamente con el correo electrónico del usuario, nombre de usuario que lo identificara en el servidor y una contraseña. El nombre de usuarios y la contraseña son utilizados para autenticar a los usuarios y se lleve un control de los CAPTCHAS que se tienen listos para ser intercambiados.
  La respuesta entregada por este archivo PHP es una respuesta HTML, la cual contiene la respuesta con códigos para saber si se realizo correctamente la operación.

- Cargar CAPTCHAS: Este servidor cuenta con un archivo PHP que recibe las peticiones de los usuarios ya registrados que quieren subir los CAPTCHAS generados por los protocolos P y P'. La carga de los CAPTCHAS se hace mediante un formulario HTML llenado previamente y archivo PHP se encarga de darnos un código para saber si la carga de los CAPTCHAS fue realizada satisfactoriamente.

- Descargar CAPTCHAS: Por último, el servidor cuenta con un archivo PHP que recibe por medio de un formulario HTML el correo destino, el correo origen y la firma que identifica al mensaje de correo electrónico que se desea descifrar. Este archivo nos devuelve una URL para descargar los CAPTCHAS deseados ó un mensaje de error en caso de pedir los CAPTCHAS de un mensaje inexistente en la base de datos.

Conclusión: La funcionalidad de este servidor es simple ya que se limitan a la distribución y resguardo de los CAPTCHAS generados por los usuarios registrados que utilizan los protocolos P y P' del prototipo anterior. Su implementación corresponde al esquema propuesto en "On Securing Communication from Profilers" y se integra en el prototipo siguiente para completar dicho esquema.

6.10. Prototipo 10

Objetivo: Crear un cliente de correo electrónico de escritorio que utilice el esquema descrito en este documento.

El cliente de correo electrónico se programó utilizando la biblioteca grafica GTK3+, el entorno gráfico de GNOME 3 y el lenguaje de programación Python. Para poder inicial el desarrollo de este prototipo es necesario instalar previamente la biblioteca gráfica GTK3+ y el entorno gráfico GNOME 3, ver anexo 4.

Se inicio el desarrollo de este prototipo generando una interfaz gráfica que ayude a establecer las configuraciones básicas para conectarse con los servidores de correo electrónico por los protocolos POP3 y SMTP. Esta interfaz también establece la configuración necesaria para conectarse con el servidor de CAPTCHAS y para escoger el protocolo de cifrado de mensajes que se utilizará.

- Configuración básica POP3: El cliente de correo electrónico establece una conexión POP3 con un servidor de correo electrónico para poder descargar los mensajes de un usuario, para poder hacerlo se necesita nombre de host, puerto de comunicación, nombre de usuario, contraseña del usuario y si se utilizara una conexión segura. Todos estos datos son proporcionados por el servidor de correo electrónico con el que se desea comunicar.

- Configuración básica SMTP: El envío de mensajes de correo electrónico se realiza estableciendo una comunicación con nuestro servidor de correo electrónico, para ello se necesita nombre de host, puerto de comunicación, nombre de usuario y contraseña del usuario. Al igual que en la configuración POP3 estos datos son proporcionados por el servidor de correo electrónico con el que se desea comunicar.

- Configuración con el servidor de CAPTCHAS: El envío de la imágenes CAPTCHAS generadas después del cifrado de los mensajes necesitan ser resguardadas en el servidor de CAPTCHAS, prototipo 9, para ello es necesario enviar al servidor su dirección de correo electrónico, un nombre de usuario y una contraseña. El servidor valida si el usuario ya esta registrado, en caso contrario el servidor realiza el registro del usuario con los datos proporcionados.

Estas 3 configuraciones se establecen llenando los campos de la interfaz, ver figura n, la cual llamaremos ventana de configuración. Esta ventana genera un archivo JSON donde se guardan estos datos para poder ser utilizado mas adelante para el envío y recepción de mensajes de correo electrónico, así como la subida y descarga de las imágenes CAPTCHA y la selección entre los protocolos P y P' para cifrar los mensajes de correo electrónico.

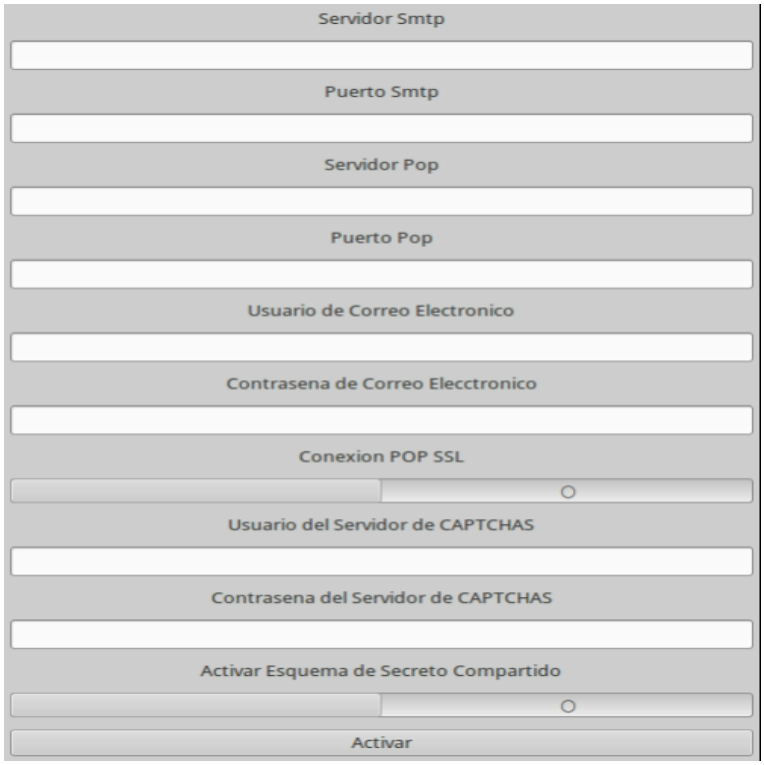


Figura 6.1: Ventana de Configuración

Posteriormente se genero una interfaz gráfica principal para visualizar los mensajes de correos electrónicos y una segunda interfaz para la redacción de los mismos.

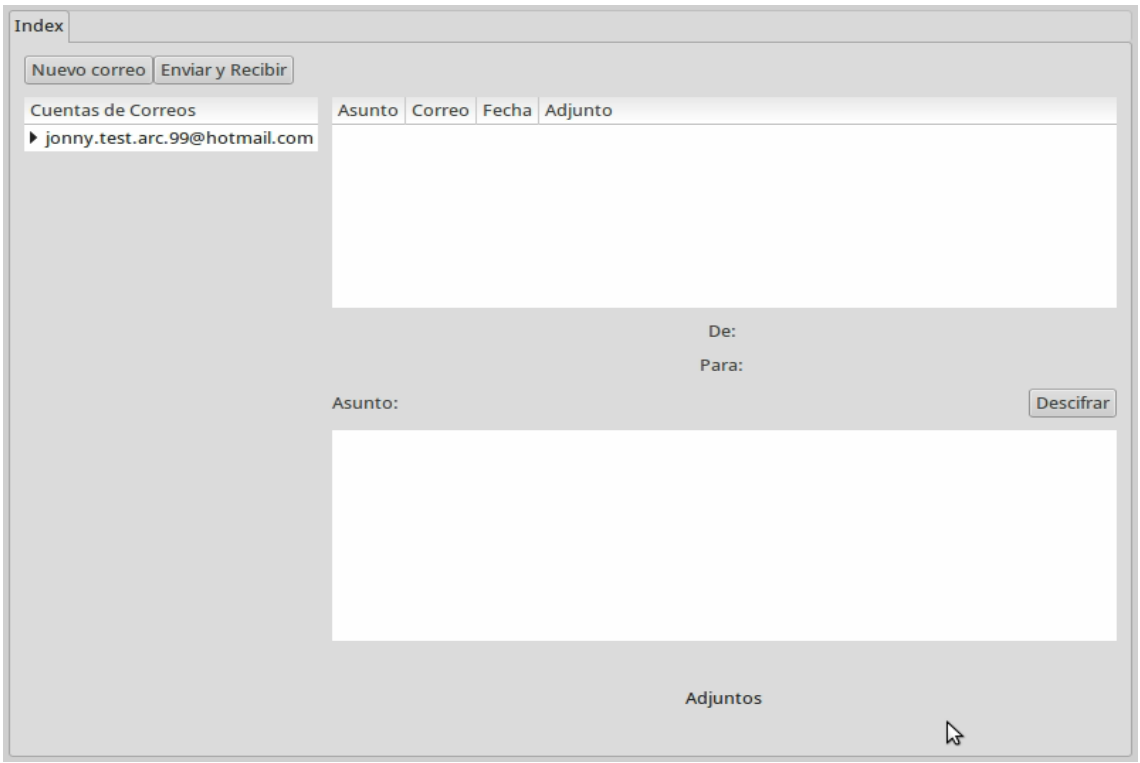


Figura 6.2: Ventana Principal

La primera interfaz, también llamada ventana principal, esta dividida en 3 parte una barra lateral, un listado y un visualizador de mensajes de correo electrónico. En la barra lateral encontramos las carpetas donde se almacenan los correo electrónicos, en el listado encontramos los mensajes de correos electrónicos que se han almacenado en la carpeta seleccionada de la barra lateral y por ultimo tenemos el visualizador de mensajes, el cual despliega la dirección de correo del usuario que mando ese mensaje, los destinatarios a donde fue dirigido el mensaje y por ultimo el cuerpo del mensaje, ver Figura 6.2.

La segunda interfaz, también llamada ventana de envío de mensajes, tiene un diseño simple para redactar los mensajes de correo electrónico, esta interfaz cuenta con 3 espacios, el primero es para escribir la dirección de correo donde se enviará el mensaje; el segundo espacio es para escribir el asunto que se adjunta al mensaje; y por último espacio es para la redacción del mismo, ver Figura 6.3.

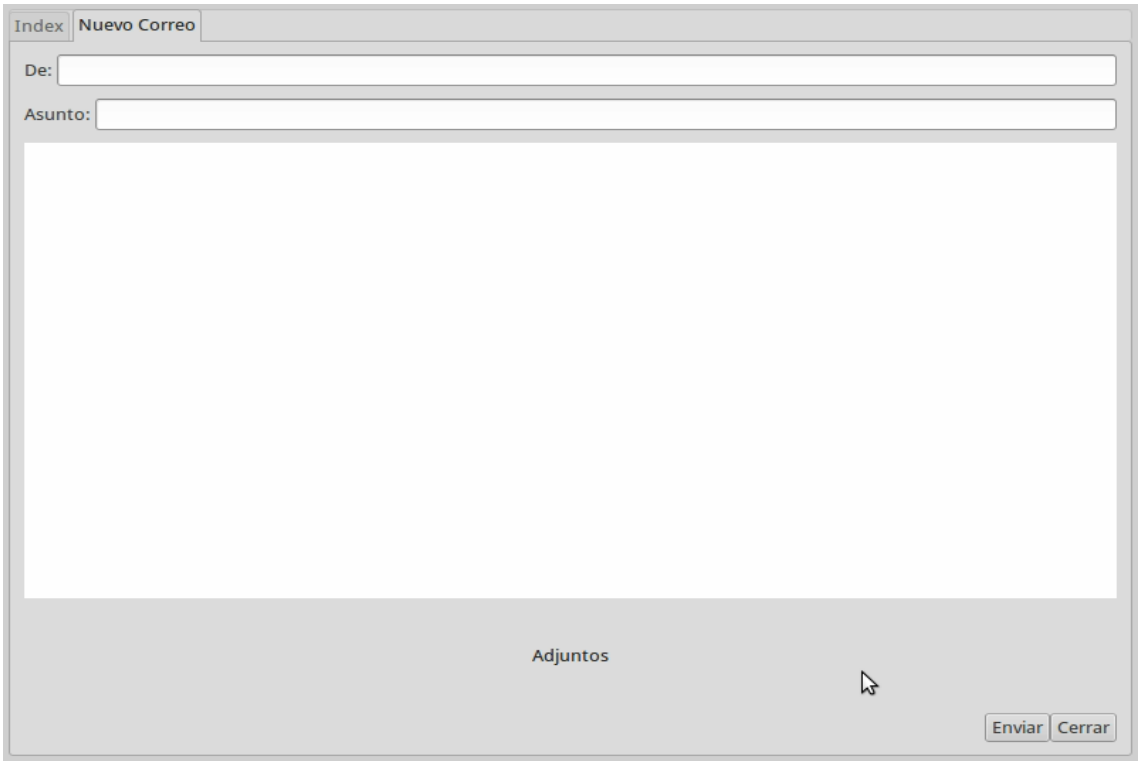


Figura 6.3: Ventana de Nuevo Correo

Una vez que se tienen las interfaces listas se procede a darles funcionalidad, para ello se llevaron a cabo las siguientes actividades.

- Cifrado de mensajes de correo electrónico por el protocolo P y P': Esta actividad se inicia al redactar un correo electrónico en la ventana de envío de mensaje y pulsar el botón enviar. Lo primero que hace es toma la fecha actual de la computadora y la concatena con las dirección de correo destino y origen, a esta cadena generada se le obtiene un digesto MD5, el cual sera utilizado como firma para el mensaje de correo. Posteriormente se toma el mensaje redactado por el usuario y es enviado a la biblioteca de cifrado, prototipo 8, especificando el protocolo a usar. Esta biblioteca nos regresa el mensaje cifrado junto con la ruta de la imágenes CAPTCHAS que descifran el

mensaje. Después se toma este mensaje cifrado y es concatenado con la firma generada anteriormente y con una cabecera que nos indicara si el mensaje esta cifrado o no al momento de visualizarlo. A continuación toma las direcciones de correo, origen y destino, el asunto redactado y el mensaje cifrado para generar un mensaje de correo electrónico y guardalo en la carpeta de salida, éste mensaje será enviado posteriormente por el protocolo SMTP al servidor de correos. Por último esta actividad activa el envío de imágenes CAPTCHAS al servidor de CAPTCHAS.

- Envío de imágenes CAPTCHAS al servidor de CAPTCHAS: Esta actividad se inicia al momento de pulsar el botón "Enviar y Recibir" de la ventana principal o al termino del cifrado de mensajes de correo electrónico. Esta actividad inicia tomando un listado de los mensajes de correo que se tienen pendientes de envío en la carpeta de salida y buscando los CAPTCHAS correspondientes a cada mensaje. Cada uno de estos CAPTCHAS son enviados al servidor junto con las direcciones de correo origen y destino, la firma del mensaje de correo y los datos de configuración del archivo JSON por medio de una petición HTTP. Por último, por cada CAPTCHA enviado exitosamente se envía su correspondiente mensaje al servidor de correo por el protocolo SMTP.

- Envío de mensajes por el protocolo SMTP: Esta actividad se inicia al momento de pulsar el botón "Enviar y Recibir" de la ventana principal o al termino de un envío exitoso de un CAPTCHA. Para hacer el envío de un mensaje de correo electrónico se necesitan los datos de configuración que se tienen el el archivo JSON junto con el mensaje que se desea enviar. En caso de error el mensaje se almacena en la carpeta de salida.

- Descargar mensajes por POP3: Esta actividad se inicia al momento de pulsar el botón "Enviar y Recibir" de la ventana principal. Para iniciar la descarga de los mensajes de correo electrónico se toman los datos básicos del archivo JSON para establecer comunicación con el servidor. Una vez establecida la conexión el servidor de correo electrónico nos dará uno a uno los mensajes y el cliente de correo electrónico guardará cada mensaje en un archivo txt en la carpeta de entrada.

- Descarga de imágenes CAPTCHAS del servidor de CAPTCHAS: Esta actividad se inicia al momento de pulsar el botón "Descifrar" de la ventana principal. Para saber si el mensaje esta cifrado se busca en el cuerpo del mensaje la cabecera de cifrado de donde obtenemos la firma del mensaje. Con la firma del mensaje se buscan las imágenes de descifrado en la carpeta CAPTCHA, esta carpeta se crea con la instalación del prototipo, en caso de no encontrar las imágenes en la carpeta el cliente de correo hacer una petición HTTP al servidor de CAPTCHAS adjuntando la firma del mensaje, las direcciones de origen y la dirección destino. El servidor contesta enviando la dirección URL de la imágenes de donde el cliente descarga las imágenes y las guarda en la carpeta CAPTCHA. Después de guardarlas, el cliente despliega la o las imágenes CAPTCHA en una ventana para que el usuario lo resuelva, esta ventana la llamaremos ventana de Descifrado. El despliegue de una o mas imágenes dependerá del protocolo que se haya utilizado para cifrar.

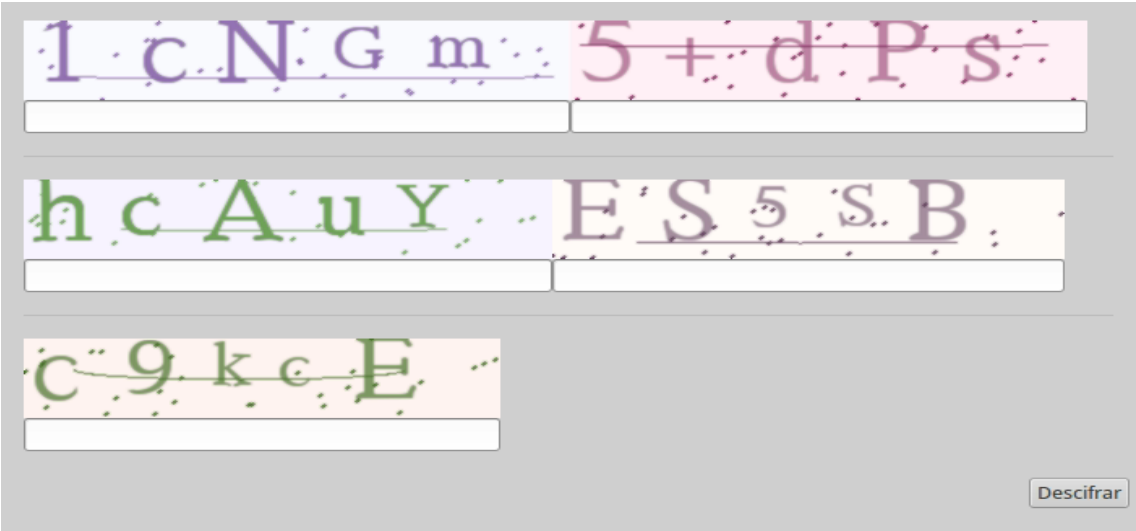


Figura 6.4: Ventana Multi-CAPTCHAS

- Descifrado de mensajes de correo electrónico por el protocolo P y P': Esta actividad se inicia al momento de pulsar el botón "Descifrar" de la ventana de Descifrado. Una vez que el usuario resuelve los CAPTCHAS se toman las respuestas junto con el cuerpo del mensaje cifrado sin la cabecera de cifrado y se envían a la biblioteca de cifrado, prototipo 8, la cual nos regresa el texto descifrado. En caso de que los CAPTCHAS sean ingresados incorrectamente el texto regresado por la biblioteca sera ilegible y el cliente de correos lo detectara.

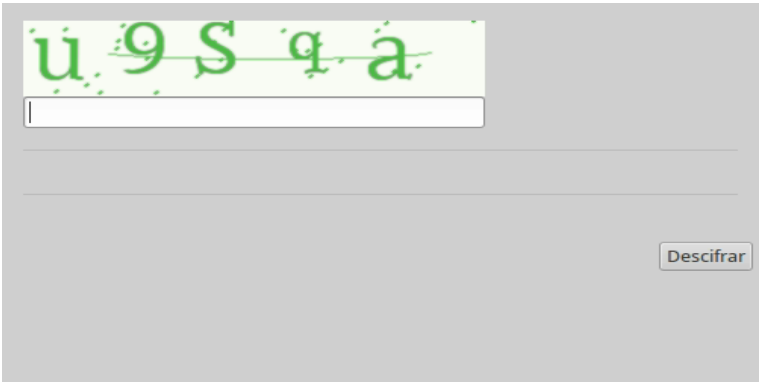


Figura 6.5: Ventana CAPTCHAS

Conclusión: El cliente de correo electrónico que se describió en este prototipo es completamente funcional y en conjunto con los prototipos 8 y 9 cumplen con el funcionamiento del esquema propuesto en este documento para implementar los protocolos P y P' del esquema Díaz -Chakraborty. Para ver el codigo completo del prototipo 10 ver el anexo 5.

Capítulo 7

Pruebas

En este capitulo se muestran los resultados de las pruebas realizadas sobre el prototipo 10. Las pruebas realizadas fueron de tiempo de respuesta y de trafico de información en la red.

7.1. Prueba de rendimiento, Cifrado y Descifrado de un solo CAPTCHA

En la figura 7.1 se muestra la relación de metodos que permiten el cifrado del mensaje y la generación de un solo CAPTCHA. El tiempo total que tarda en hacer todo el proceso es de 0.17s donde las funciones que tardan mas en realizar sus tareas son Ek_din.crearCAPTCHA que se encarga de crear el CAPTCHA y la función Ek_din.primoSig que calcula el numero primo mas cercano dependiendo del tamaño del CAPTCHA a realizar.

En la figura 7.2 se muestra la relación de metodos que permiten el descifrado del mensaje, este proceso es muy sencillo para un solo CAPTCHA ya que lo que introduce en usuario es lo que se canvierte en la clave de descifrado. El tiempo que tarda este proceso es de 0.004s.

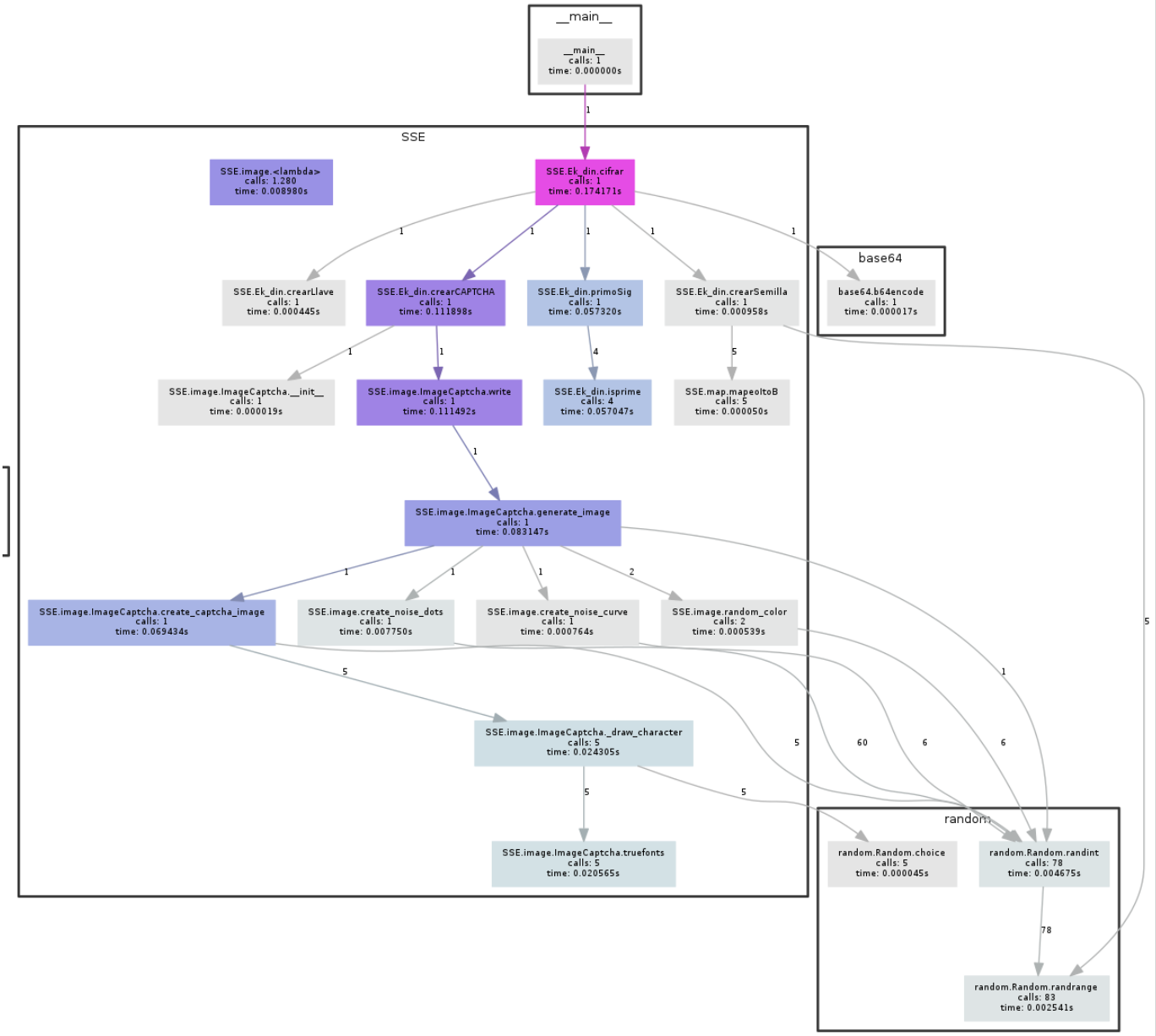


Figura 7.1: Rendimiento del esquema para un solo CAPTCHA

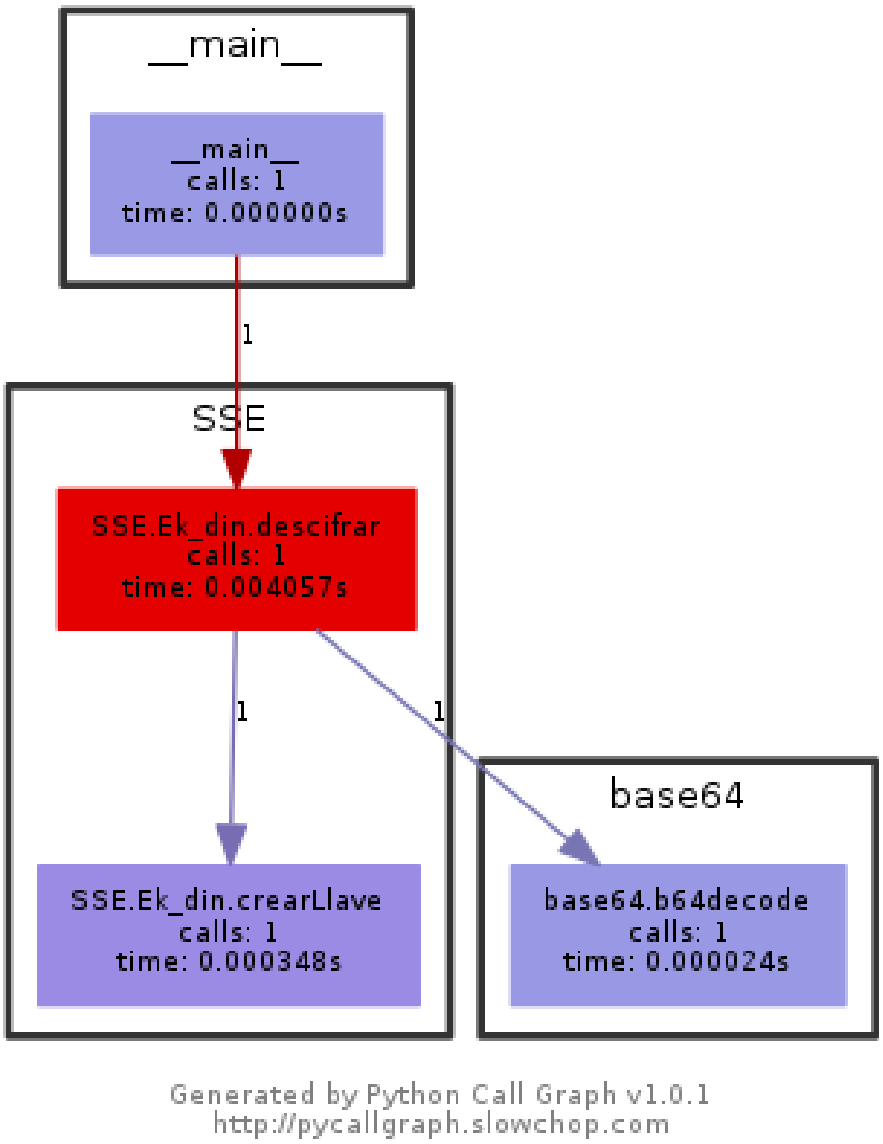


Figura 7.2: Rendimiento del esquema para un solo CAPTCHA

7.2. Prueba de rendimiento, Cifrado y Descifrado de multiples CAPTCHA's

En la figura 7.3 se encuentra la relación de funciones que generan múltiples CAPTCHAS y cifran el mensaje, en este caso la llamada a funciones es mayor que para un solo CAPTCHA, se puede ver que al igual que en el esquema de un solo CAPTCHA las funciones que mas tardan son las de Ek_din.crearCAPTCHA y la función Ek_din.primoSig dandonos un tiempo total para todo el proceso de 0.47s.

En la figura 7.4 se encuentra la relación de funciones que resuelve el algoritmo de secreto compartido y descifra el mensaje, este es mas elaborado que el descifrado para un solo

CAPTCHA en este la función que mas tarda es Ek_din.primoSig por el tamaño de numero que esta manejando, el tiempo total de termino de la tarea es 0.062s.

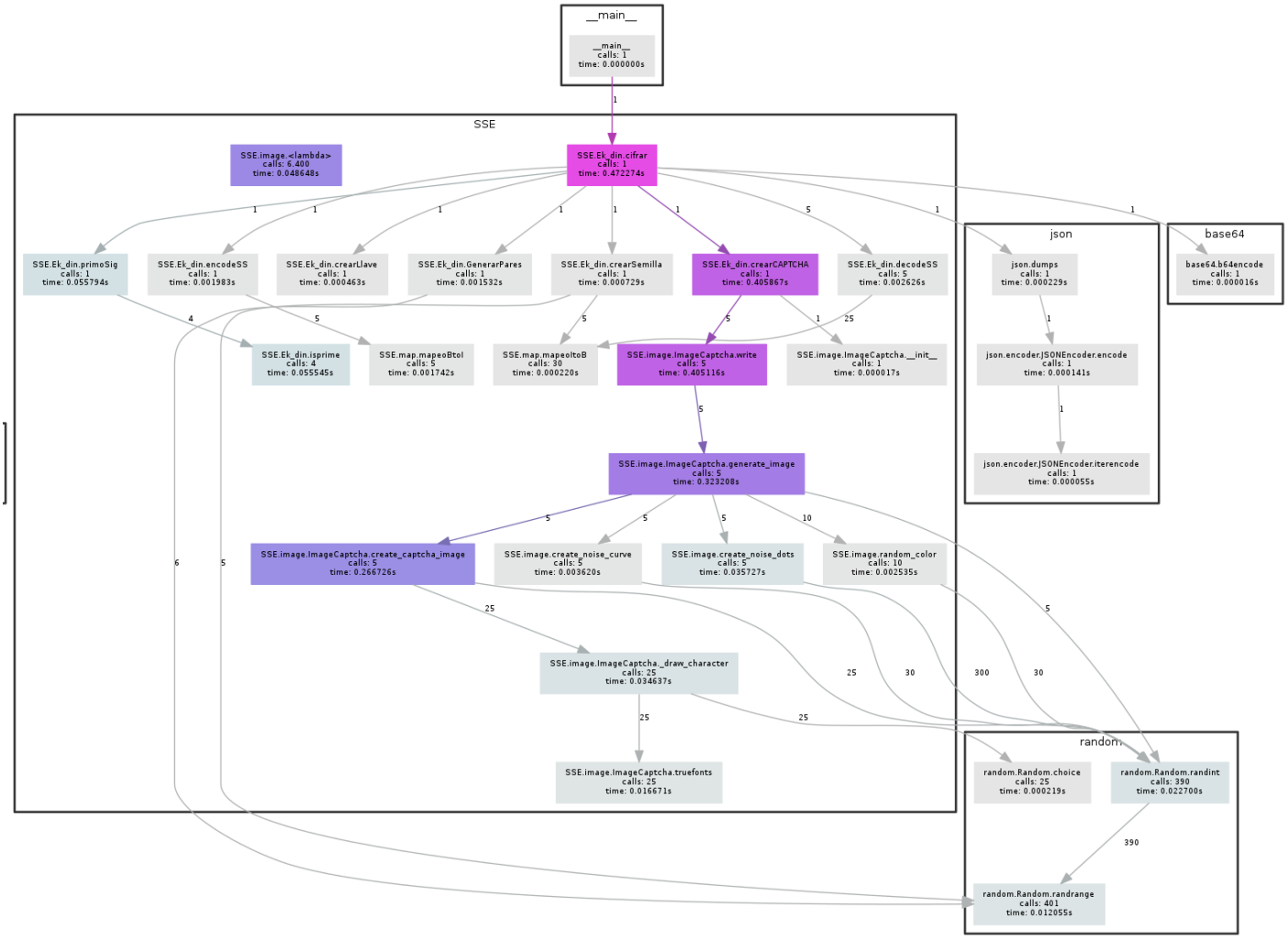


Figura 7.3: Rendimiento del esquema multiCAPTCHA

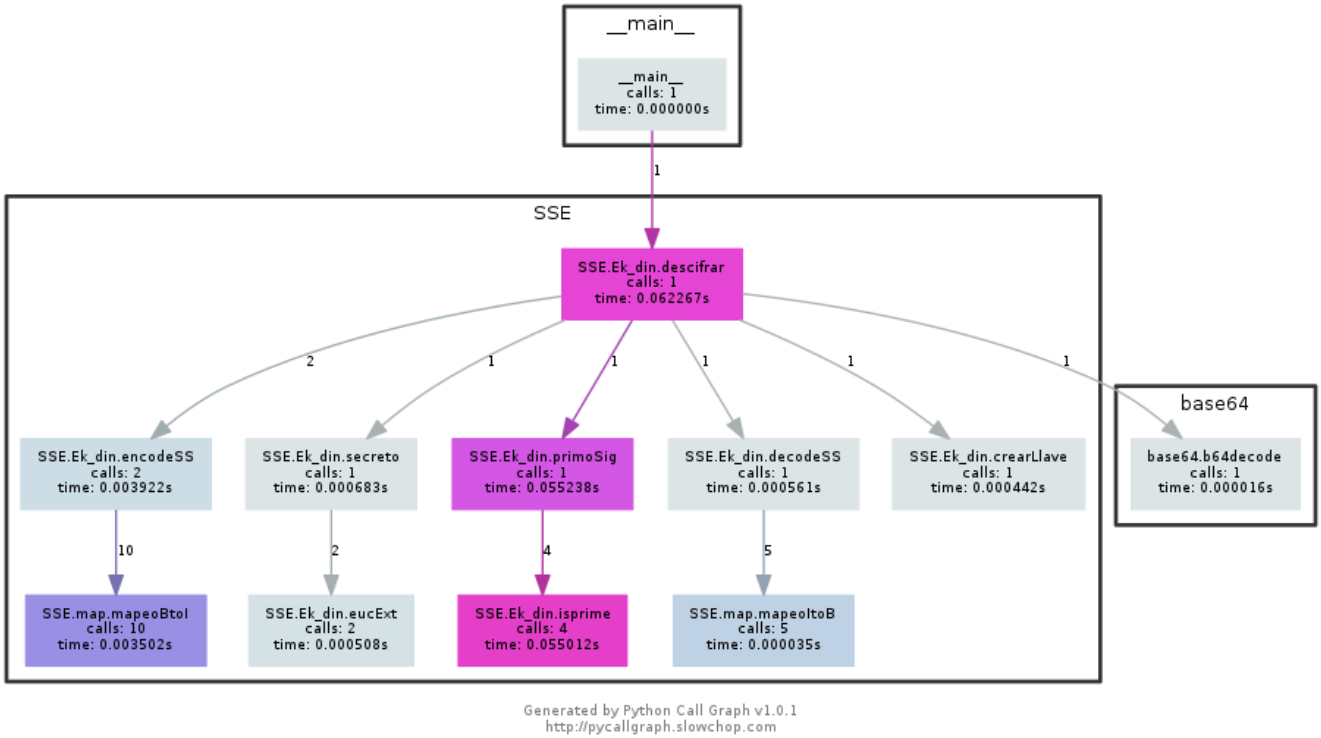


Figura 7.4: Rendimiento del esquema multiCAPTCHA

Capítulo 8

Conclusiones y Trabajo a Futuro

8.1. Conclusiones

En el desarrollo de este trabajo terminal se encontraron varios problemas al desarrollar complementos para clientes de correo electrónico comerciales los cuales describiremos a continuación.

El primer problema encontrado fue la gran cantidad de tiempo que se invierte en la investigación, desarrollo, revisión y correcciones de los complementos que se implementan para los clientes de correo estándar, ya que si se desea publicar un complemento con la empresa que desarrollo el cliente, éstos son sometido a una evaluación para verificar que no altere el funcionamiento de otros módulos de su cliente.

Otro problema a tomar en cuenta es la fase de desarrollo en la que se encuentra el cliente de correo que se desea ocupar, ya que si se encuentra en una etapa muy temprana de desarrollo se encontrara poca documentación; las funciones disponibles serán limitadas; y muy probablemente cambien la compatibilidad entre módulos de una versión a otra.

La solución que se implemento fue desarrollar un cliente de correo electrónico que tuviera las funciones básicas de envío y recepción de mensajes de correo electrónico por los protocolos POP3 y SMTP, junto con la implementación de los protocolos P y P' del esquema Díaz – Chakraborty para el cifrado y descifrado de los mensajes de correo electrónico por medio de CAPTCHAS. Se observó que el intercambio de clave y la implementación de los protocolos P y P' del esquema Díaz - Chakraborty se llevó con éxito. También se concluye que estos esquemas pueden implementarse en los modelos actuales de comunicación de correo electrónico de una manera trasparente al usuario al momento del envío y recepción de los correos electrónicos. Cabe destacar que es la primera implementación funcional que se tiene del esquema de secreto compartido de Adi Shamir para el correo electrónico e inhibiendo los ataques de los agentes clasificadores.

Por último se encontró que las comunicaciones que se establecen actualmente entre los servidores de correo electrónico y los usuarios son canales seguros. Lo cual fue confirmado por las pruebas realizadas a la aplicación, por lo tanto se concluye que el ataque de los adversarios clasificadores se hace en los servidores de correo electrónico donde son almacenados los mensajes en claro y se tiene acceso a un gran número de mensajes para su clasificación.

8.2. Trabajo a futuro.

Las líneas de trabajo que sugieren los autores de este trabajo terminal son las siguientes.

- La implementación del esquema Díaz – Chakraborty y el esquema de intercambio de claves en un cliente de correo electrónico de escritorio comercial o en otros clientes de correo como los clientes web o móviles.
- En el esquema de intercambio de claves en este trabajo terminal se utilizó un servidor que aloja y distribuye los CAPTCHAS entre los usuarios, por lo tanto se sugiere trabajar en la gestión de los CAPTCHAS para mejorar el intercambio de claves entre los usuarios.
- El cifrado del contenido de los mensaje de correo electrónico no es detectado por los protocolos de SMTP y POP3, pero en algunos países el uso de algoritmos criptográficos esta prohibido. Para evitar una sanción por parte de estos países y que a su vez los adversarios clasificadores no puedan clasificar los mensajes se sugiere implementar un cifrado no estandar como los son AES y DES.
- Por último se sugiere una implementación de una biblioteca de creación de CAPTCHA's en el lenguaje PYTHON para mejorar las imágenes generadas.

Apéndice A

Código fuente del prototipo 2

A continuación se muestra el código fuente desarrollado en el prototipo 2.

- Archivo de cifrado (cifrado.py).

```
 #! /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()
```

- Ventana de despliegue de CAPTCHAS (ventana.py)

```
 #!/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()
```

```
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=imagenAlturaMaxima).pack()

valor = ""
e = Tkinter.Entry(root)
e.pack()

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

root.mainloop()
```

Apéndice B

Código fuente del prototipo 8

A continuación se muestra el código fuente desarrollado en el prototipo 8.

- Biblioteca de cifrado (Ek.py).

```
from Crypto.Hash import SHA256
import os
import random
import base64
import json
from random import randrange
from map import mapeoBtoI
from map import mapeoItoB
from image import ImageCaptcha
from Crypto.Cipher import AES

def isprime(n):

 n = abs(int(n))
 # 0 y 1 no son primos
 if n < 2:
  return False
 # 2 es el unico primo par
 if n == 2:
  return True
 # El resto de pares no son primos
 if not n & 1:
  return False
 # El rango comienza en 3 y solo necesita subir
 # hasta la raiz cuadrada de n
 # para todos los impares
 for x in range(3, int(n**0.5)+1, 2):
  if n %x == 0:
   return False
 return True
```

```
def primoSig(num):
 buscar=True
 while buscar:
  if isprime(num):
   buscar=False
  else:
   num+=1
 return num
def crearSemilla(tam):
 r=0
 semilla=""
 for i in range(tam):
  r=random.randrange(64)
  semilla=semilla+str(mapeoItoB(r))
 return semilla

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

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

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)

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

def eucExt(a,b):
 r = [a,b]
 s = [1,0]
 i = 1
 q = [[]]
```

```
 while (r[i] != 0):
  q = q + [r[i-1] // r[i]]
  r = r + [r[i-1] %r[i]]
  s = s + [s[i-1] - q[i]*s[i]]
  i = i+1
 return s[i-1]%b

def GenerarPares(p=7,w=5,t=2,k=0):
 pares =[]
 a = [k]
 for aux in range(0,w):
  print(aux)
  pares.append([randrange(p),0])
 print("X->")
 print(pares)
 for aux in range(1,t):
  print(aux)
  a.append(randrange(p))
 print("A->")
 print(a)
# for aux in range(0,w):
#  suma = k+(a[1]*pares[aux][0])
#  pares[aux][1] = suma%p
 for aux in pares:
  print("suma")
  suma = k
  print(suma)
  for aux2 in range(1,t):
   print("sin ecuacion")
   print(suma)
   suma = (suma+(a[aux2]*(aux[0]**aux2)))%p
   print("con ecuacion")
   print(suma)
  aux[1] =suma
 return pares

def secreto(pares,p):
 suma = 0
# print("pares")
# print(pares)
 for aux in pares:
#  print("par")
#  print(aux)
  ind = pares.index(aux)
#  print("index")
#  print(ind)
```

```
   lis = pares[:ind] + pares[(ind+1):]
#   print("otros pares")
#   print(lis)
   num=1
   den=1
   for aux2 in lis:
#     print("numerodor")
     num = (num*(aux2[0])*-1)%p
#     print(num)
#     print("denominador")
     den = (den*((aux[0]-aux2[0])%p))%p
#     print(den)
#   print("Euclides")
   den = eucExt(den,p)
#   print(den)
   suma += (den*aux[1]*num)%p
#   print("suma")
#   print(suma)
  return suma%p

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=primoSig(2**(6*ta))
  disc={}
  #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
  for i in ruta[0]:
   disc[i[1]]=i[0]
  print(ruta)
  lista=open(ruta[1]+"/lista.json","w")
  lista.write(json.dumps(disc))
  lista.close
 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=""
 salida = base64.b64encode(salida)
 return (salida,ruta)

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

 if (op2==0):
  k=crearLlave(capt1)
 else:
  w=len(capt1[0][1])
  zp=primoSig(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)
 body1 = base64.b64decode(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
```

- Generador de imágenes CAPTCHAS (imagen.py).

```
# coding: utf-8

import os
import random
from PIL import Image
from PIL import ImageFilter
from PIL.ImageDraw import Draw
from PIL.ImageFont import truetype
try:
 from cStringIO import StringIO as BytesIO
except ImportError:
 from io import BytesIO
```

```
try:
 from wheezy.captcha import image as wheezy_captcha
except ImportError:
 wheezy_captcha = None
DATA_DIR = os.path.join(os.path.abspath(os.path.dirname(__file__))
                        , 'data')
DEFAULT_FONTS = [os.path.join(DATA_DIR, 'DroidSansMono.ttf')]

if wheezy_captcha:
 __all__ = ['ImageCaptcha', 'WheezyCaptcha']
else:
 __all__ = ['ImageCaptcha']

class _Captcha(object):
 def generate(self, chars, format='png'):
  im = self.generate_image(chars)
  out = BytesIO()
  im.save(out, format=format)
  out.seek(0)
  return out

 def write(self, chars, output, format='png'):
  im = self.generate_image(chars)
  return im.save(output, format=format)

class WheezyCaptcha(_Captcha):
 def __init__(self, width=200, height=75, fonts=None):
  self._width = width
  self._height = height
  self._fonts = fonts or DEFAULT_FONTS

 def generate_image(self, chars):
  text_drawings = [wheezy_captcha.warp(),wheezy_captcha.rotate(),
                   wheezy_captcha.offset(),]
  fn = wheezy_captcha.captcha(
   drawings=[
    wheezy_captcha.background(),
    wheezy_captcha.text(fonts=self._fonts, drawings=text_drawings),
    wheezy_captcha.curve(),
    wheezy_captcha.noise(),
    wheezy_captcha.smooth(),],
   width=self._width,
   height=self._height,
  )

  return fn(chars)
```

```
class ImageCaptcha(_Captcha):

 def __init__(self,width=160,height=60,fonts=None,font_sizes=None):

  self._width = width
  self._height = height
  self._fonts = fonts or DEFAULT_FONTS
  self._font_sizes = font_sizes or (46, 58, 68)
  self._truefonts = []

 @property
 def truefonts(self):

  if self._truefonts:
   return self._truefonts
  self._truefonts = tuple([
   truetype(n, s)
   for n in self._fonts
   for s in self._font_sizes
  ])
  return self._truefonts

 @staticmethod
 def create_noise_curve(image, color):

  w, h = image.size
  x1 = random.randint(0, int(w / 5))
  x2 = random.randint(w - int(w / 5), w)
  y1 = random.randint(h / 5, h - int(h / 5))
  y2 = random.randint(y1, h - int(h / 5))
  points = [(x1, y1), (x2, y2)]
  end = random.randint(160, 200)
  start = random.randint(0, 20)
  Draw(image).arc(points, start, end, fill=color)
  return image

 @staticmethod
 def create_noise_dots(image, color, width=3, number=30):
  draw = Draw(image)
  w, h = image.size
  while number:
   x1 = random.randint(0, w)
   y1 = random.randint(0, h)
   draw.line(((x1, y1), (x1 - 1, y1 - 1)), fill=color, width=width)
   number -= 1
  return image
```

```
def create_captcha_image(self, chars, color, background):

 image = Image.new('RGB', (self._width, self._height), background)
 draw = Draw(image)

 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+30 ))
  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,expand=1)
  # 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

 def generate_image(self, chars):
  """Generate the image of the given characters.

  :param chars: text to be generated.
  """
  background = random_color(238, 255)
  color = random_color(0, 200, random.randint(220, 255))
  im = self.create_captcha_image(chars, color, background)
  self.create_noise_dots(im, color)
  self.create_noise_curve(im, color)
  im = im.filter(ImageFilter.SMOOTH)
  return im


def random_color(start, end, opacity=None):
 red = random.randint(start, end)
 green = random.randint(start, end)
 blue = random.randint(start, end)
 if opacity is None:
  return (red, green, blue)
 return (red, green, blue, opacity)
```

- Empaquetado de imágenes CAPTCHA (empaquetar.py).

```
from Ek_din import cifrar
from subprocess import call
import types
import os
from os import path

def listFiles(folder):
 return [d for d in os.listdir(folder)
         if path.isfile(path.join(folder, d))]

def empaquetar(body,asunto,op):
 s=cifrar(body,asunto,op)
 asunto=asunto.replace(" ","_")
 disc={}
```

```
#print(s[1])
ass=asunto+".zip"
if type(s[1])==types.StringType:
 zi=call("zip -r "+ass+" "+s[1], shell=True)
 mv=call("mv ./"+ass+" ./CAPTCHAS", shell=True)
 rmm=call("rm -rf "+s[1], shell=True)
 return (s[0],"./CAPTCHAS/"+ass)
else:
 zi=call("zip -r "+ass+" "+s[1][1], shell=True)
 mv=call("mv ./"+ass+" ./CAPTCHAS", shell=True)
 rmm=call("rm -rf "+s[1][1], shell=True)
 return (s[0],"./CAPTCHAS/"+ass)
```

Apéndice C

Código fuente del prototipo 9

A continuación se muestra el código fuente desarrollado en el prototipo 9.

- Estructura de la base de datos (script.sql).

```
CREATE TABLE IF NOT EXISTS `Mensaje` (
  `firma_digital` varchar(255) CHARACTER SET utf8 NOT NULL,
  `correo_destino` varchar(50) CHARACTER SET utf8 NOT NULL,
  `ruta_archivo` varchar(255) CHARACTER SET utf8 NOT NULL,
  `correo_electronico` varchar(50) CHARACTER SET utf8 NOT NULL,
  PRIMARY KEY (`correo_destino`,`firma_digital`)
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

CREATE TABLE IF NOT EXISTS `Usuario` (
  `correo_electronico` varchar(50) CHARACTER SET utf8 NOT NULL,
  `nombre` varchar(150) CHARACTER SET utf8 NOT NULL,
  `contrasena` varchar(20) COLLATE utf8_unicode_ci NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;
```

- Alta de usuario en el servidor de CAPTCHAS (AltaUsuario.php).

```
<html>
<head>
 <title>Alta de usuario</title>
</head>

<body>
<?php
$usuario = trim($_POST["nombre"]);
$contra = trim($_POST["contrasena"]);
$correo = trim($_POST["correo_electronico"]);
if (empty($usuario)){
 echo '<p name="respuesta">0</p>';
}elseif (empty($contra)) {
 echo '<p name="respuesta">1</p>';
```

```
}elseif (empty($correo)) {
 echo '<p name="respuesta">2</p>';
}else{
 $enlace = new mysqli('mysql.hostinger.mx', 'u715698692_corre',
                      'correocifrado','u715698692_corre');
 if($enlace->connect_errno){
  echo '<p name="respuesta">3</p>';
  die("Error en conexion");
 }

 if (!file_exists("./Usuarios/".$correo)) {
  mkdir("./Usuarios/".$correo);
 }

 $query = "SELECT nombre FROM Usuario
           WHERE correo_electronico like '$correo'";
 $result = $enlace->query($query);
 $aux = $result->num_rows;
 if($aux == 1){
  echo '<p name="respuesta">5</p>';
  $enlace->close();
 }else{
  $result->free();
  if($enlace->query("INSERT INTO `Usuario`
  (`correo_electronico`, `nombre`, `contrasena`)
  VALUES
  ('".$correo."','".$usuario."','".$contra."')") === TRUE){
   echo '<p name="respuesta">5</p>';
   $enlace->close();
  }else{
   echo '<p name="respuesta">6</p>';
   $enlace->close();
  }
 }
}
?>
 </body>
</html>
```

- Carga de imágenes CAPTCHAS en el servidor (AltaMensage.php).

```
<html>
<head>
 <title>Alta de Mensaje</title>
</head>
```

```
 <body>
<?php
$usuario = trim($_POST["nombre"]);
$contra = trim($_POST["contrasena"]);
$correo = trim($_POST["correo_electronico"]);
$firma = trim($_POST["firma"]);
$correo_des = trim($_POST["correo_destino"]);

if (empty($usuario)){
 echo '<p name="respuesta">0</p>';
 die();
}elseif (empty($contra)) {
 echo '<p name="respuesta">1</p>';
 die();
}elseif (empty($correo)) {
 echo '<p name="respuesta">2</p>';
 die();
}elseif (empty($firma)) {
 echo '<p name="respuesta">3</p>';
 die();
}elseif (empty($correo_des)) {
 echo '<p name="respuesta">4</p>';
 die();
}elseif (!is_uploaded_file($_FILES["archivo"]["tmp_name"])){
 echo '<p name="respuesta">6</p>';
 die();
}else{
 $enlace = new mysqli('mysql.hostinger.mx', 'u715698692_corre',
           'correocifrado','u715698692_corre');
 if($enlace->connect_errno){
  echo '<p name="respuesta">7</p>';
  die("Error en conexion");
 }
 $query = "SELECT nombre, contrasena
           FROM Usuario
           WHERE Correo_Electronico like '$correo' ";
 $result = $enlace->query($query);
 $aux = $result->num_rows;

 if($aux >0){
  $row = $result->fetch_array(MYSQLI_ASSOC);
 }else{
  echo '<p name="respuesta">11</p>';
  $enlace->close();
  die("Error de autenticacion");
 }
```

```
if(!(strcmp($row["nombre"],$usuario) == 0)){
 echo '<p name="respuesta">8</p>';
 $enlace->close();
 die("Error de autenticacion 1");
}
if (!(strcmp($row["contrasena"],$contra) == 0)) {
 echo '<p name="respuesta">8</p>';
 $enlace->close();
 die("Error de autenticacion 2");
}
$result->free();
if(strcmp($_FILES['archivo']['type'], "application/zip")==0){
 $file= sha1($correo_des.$firma.$correo).".zip";
 $ruta=join(DIRECTORY_SEPARATOR,array("./Usuarios",
       $correo,$file));
 $query = "SELECT ruta_archivo
           FROM Mensaje
           WHERE Correo_Electronico like '$correo'
           and firma_digital like '$firma'
           and correo_destino like '$correo_des'";
 $result = $enlace->query($query);
 $aux = $result->num_rows;

 if($aux >0){
  echo '<p name="respuesta">12</p>';
  $enlace->close();
  die("Error de autenticacion");
 }else{
  $result->free();
  if (!file_exists($ruta)) {
   move_uploaded_file($_FILES['archivo']['tmp_name'], $ruta);
   echo "<pre>";
   print_r($ruta);
   if($enlace->query("INSERT INTO `Mensaje`
   (`firma_digital`, `correo_destino`, `ruta_archivo`,
   `correo_electronico`)
   VALUES ('".$firma."','".$correo_des."',
   '".$ruta."','".$correo."')") === TRUE){
    echo '<p name="respuesta">5</p>';
    $enlace->close();
   }else{
    echo '<p name="respuesta">9</p>';
    $enlace->close();
   }
  }else{
   echo '<p name="respuesta">13</p>';
```

```
    $enlace->close();
    die("Error de autenticacion");
   }
  }
 }else{
  echo '<p name="respuesta">10</p>';
  $enlace->close();
 }

}
?>
 </body>
</html>
```

- Descarga de imágenes CPATCHAS del servidor (BusquedaArchivo.php).

```
 <html>
 <head>
  <title>Busqueda de Archivos</title>
 </head>

 <body>
<?php

$correo = trim($_POST["correo_electronico"]);
$firma = trim($_POST["firma"]);
$correo_des = trim($_POST["correo_destino"]);

if (empty($correo)) {
 echo '<p name="respuesta">0</p>';
 die();
}elseif (empty($firma)) {
 echo '<p name="respuesta">1</p>';
 die();
}elseif (empty($correo_des)) {
 echo '<p name="respuesta">2</p>';
 die();
}else{

 $enlace = new mysqli('mysql.hostinger.mx', 'u715698692_corre',
           'correocifrado','u715698692_corre');
 if($enlace->connect_errno){
  echo '<p name="respuesta">7</p>';
  die("Error en conexion");

 }
```

```
 $query = "SELECT ruta_archivo
           FROM Mensaje
           WHERE Correo_Electronico like '$correo'
           and firma_digital like '$firma'
           and correo_destino like '$correo_des'";
 $result = $enlace->query($query);
 $aux = $result->num_rows;
 if($aux == 1){
  $row = $result->fetch_array(MYSQLI_ASSOC);
  $ruta = "http://correocifrado.esy.es".$row["ruta_archivo"];
  echo '<p name="respuesta">'.$ruta.'</p>';
  $enlace->close();
 }else{
  echo '<p name="respuesta">4</p>';
  $enlace->close();
  die("Error de autenticacion");
 }

}
?>
 </body>
</html>
```

Apéndice D

Intalación de biblioteca GTK+ 3 y entorno gráfico GNOME 3

A continuación se muestra los pasos a seguir para la intalación de las bibliotecas GTK3+ y el entorno grafico GNOME 3.

D.1. Instalación del entorno gráfico GNOME 3.

La instalación del entorno gráfico GNOME 3 se realizo en un sistema operativo XUBUNTU 15.1 y XUBUNTU 14.1. A continuación se explica los pasos a seguir para la instalación de este entorno gráfico.

- Se abre una terminal del sistema operativo.
- Se ingresan los siguientes comandos a la terminal para instalar los repositorios de descarga.

  ```
  sudo add-apt-repository ppa:gnome3-team/gnome3
  sudo add-apt-repository ppa:gnome3-team/gnome3-staging
  ```

- Posteriormente se ingresa el siguiente comando a la terminal para actualizar los repositorios de descarga.

  ```
  sudo apt-get update
  ```

- Una vez terminada la actualización se ingresa este último comando para terminar con la instalación del entorno gráfico GNOME 3.

  ```
  sudo apt-get dist-upgrade
  ```

  Una vez que la instalación termine se reinicia el equipo para activar el entorno gráfico GNOME 3.

D.2. Instalación de la biblioteca gráfica GTK+ 3.

La instalación del biblioteca gráfica GTK+ 3 se realizo en un sistema operativo XUBUNTU 15.1 y XUBUNTU 14.1 siguiendo los tutoriales proporcionados por la pagina de Python GTK+ 3 Tutorial y GNOME developer. Uno de los requisitos previos para la instalación de GTK+ 3 es la instalación de JHBuild la cual se instalo siguiendo el tutorial de GNOME developer encontrado en la siguiente pagina web:
`https://developer.gnome.org/jhbuild/unstable/getting-started.html.es`

Después de la instalación de JHBuild se prosiguió con la instalación de la biblioteca gráfica GTK+ 3. A continuación se presentan los pasos a seguir para la instalación de de la biblioteca.

- Se abre una terminal del sistema operativo.
- Se ingresan los siguientes comandos.

```
$ jhbuild build pygobject
$ jhbuild build gtk+
$ jhbuild shell
```

Apéndice E

Código fuente del prototipo 10

A continuación se muestra el código fuente desarrollado en el prototipo 10.

- Archivo de configuración JSON config.json).

```
{
certfile: "./Seguridad/server2048.pem",
passwdSSE: "12345678",
passwd: "360_live",
portSmtp: "587",
portPop: "995",
ssl: true,
hostSmtp: "smtp-mail.outlook.com",
user: "jonny.test.arc.99@hotmail.com",
SSE: false,
nombre: "jonathan arcos",
hostPop: "pop3.live.com",
keyfile: "./Seguridad/server2048.key",
delete: 0
}
```

- Interfaz gráfica del cliente de correo electrónico (setup.py).

```
import gi
import os
import email
import json
import SSE
import re
import captchas
import http
import logging

gi.require_version('Gtk', '3.0')
from gi.repository import Gtk, Gio
```

```
from smtp2 import datosPrincipales
from smtp2 import validarSmtp
from listarCorreos import listaCorreosView
from listarCorreos import contarCorreo
from listarCorreos import body
from pop3 import conexionPop3
from pop3 import validarPop
from envios import envios
from salidaSmtp import salida

path = './Usuarios'

def listdirs(folder):
 return [d for d in os.listdir(folder)
        if os.path.isdir(os.path.join(folder, d))]

def listFiles(folder):
 return [d for d in os.listdir(folder)
        if os.path.isfile(os.path.join(folder, d))]

class MyWindow(Gtk.Window):

 user=listdirs(path)
 selectCarpeta=None
 selectUsuario=None
 config={}

 def __init__(self,config):
  self.config = config
  Gtk.Window.__init__(self, title="Cliente de Correos")
  self.set_border_width(4)
  self.set_default_size(800, 600)

  self.notebook = Gtk.Notebook()
  self.add(self.notebook)

  self.page = self.newPage()
  self.page.set_border_width(10)
  self.notebook.append_page(self.page, Gtk.Label('Index'))

 def visorCorreo(self):
  vistaCorre = Gtk.Box(orientation=Gtk.Orientation.VERTICAL,
                       spacing=10)
  self.emisor = Gtk.Label("De: ")
  self.emisor.set_justify(Gtk.Justification.LEFT)
  self.destinatorio = Gtk.Label("Para: ")
```

```
        self.destinatorio.set_justify(Gtk.Justification.LEFT)
        self.asunto = Gtk.Label("Asunto: ")
        self.asunto.set_justify(Gtk.Justification.LEFT)
        descifrado= Gtk.Button(label="Descifrar")
        descifrado.connect("clicked", self.descifrarBody)
        box1 = Gtk.VBox(False,10)
        box1.pack_start(self.emisor,True,True,0)
        box1.pack_end(self.destinatorio,False,True,0)
        box2 = Gtk.HBox(False,0)
        box2.pack_start(self.asunto,False, False, 0)
        box2.pack_end(descifrado,False,False,0)
        self.cuerpo = Gtk.TextView()
        self.cuerpo.set_wrap_mode(Gtk.WrapMode.WORD)
        self.cuerpo.set_editable(False)
        scrol = Gtk.ScrolledWindow()
        scrol.set_policy(Gtk.PolicyType.AUTOMATIC,
                         Gtk.PolicyType.AUTOMATIC)
        scrol.set_vexpand(True)
        scrol.add(self.cuerpo)
        vistaCorre.add(box1)
        vistaCorre.add(box2)
        vistaCorre.add(scrol)
        return vistaCorre

    def visorCorreoNuevo(self):
        vistaCorre = Gtk.Box(orientation=Gtk.Orientation.VERTICAL,
                             spacing=10)

        newDest = Gtk.Entry(name="Destino")
        newDest.set_editable(True)
        newAsunto = Gtk.Entry(name="Asunto")
        newAsunto.set_editable(True)
        destinatorio = Gtk.Label("De: ")
        destinatorio.set_justify(Gtk.Justification.LEFT)
        asunto = Gtk.Label("Asunto: ")
        asunto.set_justify(Gtk.Justification.LEFT)
        Cerrar = Gtk.Button(label="Cerrar")
        Cerrar.connect("clicked", self.cerrarPagina)
        Enviar = Gtk.Button(label="Enviar")
        Enviar.connect("clicked", self.enviarMensage)
        box1 = Gtk.HBox(False,0)
        box1.pack_start(destinatorio,False,False,0)
        box1.pack_start(newDest,True,True,0)
        box2 = Gtk.HBox(False,0)
        box2.pack_start(asunto,False, False, 0)
        box2.pack_start(newAsunto,True, True, 0)
```

```
  box3 = Gtk.HBox(False,0)
  box3.pack_end(Cerrar,False, False, 0)
  box3.pack_end(Enviar,False, False, 0)

  cuerpo = Gtk.TextView(name="cuerpo")
  cuerpo.set_wrap_mode(Gtk.WrapMode.WORD)
  #WRAP_WORD
  scrol = Gtk.ScrolledWindow()
  scrol.set_policy(Gtk.PolicyType.AUTOMATIC,
                   Gtk.PolicyType.AUTOMATIC)
  scrol.set_vexpand(True)
  scrol.add(cuerpo)
  vistaCorre.add(box1)
  vistaCorre.add(box2)
  vistaCorre.add(scrol)
  vistaCorre.add(box3)
  return vistaCorre

 def cerrarPagina(self,button):
  print("cerrarPagina")
  page=self.notebook.get_current_page()
  self.notebook.remove_page(page)
  self.notebook.show_all()

 def enviarMensage(self,button):
  print("enviarMensage")
  page=self.notebook.get_current_page()
  contenedor = self.notebook.get_nth_page(page)
  asunto = ""
  destino = ""
  cuerpo = ""
  for c in contenedor.get_children():
    for x in c.get_children():
     if isinstance(x,Gtk.Entry):
      if x.get_name() == "Destino":
       destino = x.get_text()
      elif x.get_name() == "Asunto":
       asunto = x.get_text()
     if isinstance(x,Gtk.TextView):
      buf = x.get_buffer()
      end_iter = buf.get_end_iter()
      start_iter = buf.get_start_iter()
      cuerpo = x.get_buffer().get_text(start_iter, end_iter, True)
  if not destino.split():
   dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
    Gtk.ButtonsType.CANCEL, "Error al enviar Mensaje")
```

```
    dialog.format_secondary_text(
     "El correo Destinatario no ha sido ingresado.")
    dialog.run()
    dialog.destroy()
   elif not asunto and not cuerpo:
    dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.WARNING,
     Gtk.ButtonsType.OK_CANCEL, "Mensaje vacio")
    dialog.format_secondary_text(
     "El mensaje de correo esta vacio, decea que se envie?")
    response = dialog.run()
    if response == Gtk.ResponseType.OK:
     print("Mensaje Incompleto")
     t = envios(destino,self.user[0],asunto,cuerpo,[],self.config)
     t.start()
    dialog.destroy()
   else:
    print("Mensaje Completo")
    t = envios(destino,self.user[0],asunto,cuerpo,[],self.config)
    t.start()
   print("Asunto: "+asunto)
   print("Destino: "+destino)
   print("Cuerpo: "+cuerpo)
   print("cerrarPagina")
   page=self.notebook.get_current_page()
   self.notebook.remove_page(page)
   self.notebook.show_all()
  #print(contenedor.query_child_packing())

 def descifrarBody(self, button):
  print "Descifrar body"
  bodyBuffer=self.cuerpo.get_buffer()
  start_iter = bodyBuffer.get_start_iter()
  end_iter = bodyBuffer.get_end_iter()
  text = bodyBuffer.get_text(start_iter, end_iter, True)
  firma=text.find("------SSE Cipher------")
  if(firma>=0):
   text2 = text[firma:]
   m = re.search('\-\n(.+)\n\-',text2)
   if(m!=None):
    textFirma = m.group(1)
    print textFirma
    correoOrigen=self.emisor.get_text()
    correoDestino=self.destinatorio.get_text()
    m = re.search(
     "([(a-z0-9\_\-\.)]+@[(a-z0-9\_\-\.)]+\.[(a-z)]{2,15})"
     ,correoOrigen)
```

```
    correoOriegen = m.group(1)
    m = re.search(
    "([(a-z0-9\_\-\.)]+@[(a-z0-9\_\-\.)]+\.[(a-z)]{2,15})"
    ,correoDestino)

    correoDestino = m.group(1)
    despliegue=captchas.buscarCAPTCHAS(textFirma,
                                       correoDestino,
                                       correoOriegen)
    print(correoOriegen)
    print(correoDestino)
    if len(despliegue)>0:
     ventanaCaptcha(self,despliegue)
    else:
     dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
     Gtk.ButtonsType.CANCEL, "Error al descargar CAPTCHA")
     dialog.format_secondary_text(
      "Ocurrio un error con el servidor, intentarlo mas tarde")
     dialog.run()
     dialog.destroy()

def listaMail(self, usuario, carpeta):

 software_liststore = Gtk.ListStore(str, str, str, str)
 #archivos = listdirs(path+usuario+"/"+carpeta)
 if self.selectCarpeta == None:
  archivos = [("", "",  "", "")]
 else:
  archivos = [("prueba", "prueba",  "prueba", "mail-attachment")]
 for archivo in archivos:
  software_liststore.append(list(archivo))
 lista = software_liststore.filter_new()
 self.listaCar = Gtk.TreeView.new_with_model(lista)
 self.listaCar.connect("row-activated",self.celdasCorreo)
 for i, column_title in enumerate(["Asunto",
                                   "Correo",
                                   "Fecha",
                                   "Adjunto"]):

  if i == 3:
   renderer = Gtk.CellRendererPixbuf()
   column = Gtk.TreeViewColumn(column_title,renderer,icon_name=i)
  else:
   renderer = Gtk.CellRendererText()
   column = Gtk.TreeViewColumn(column_title, renderer, text=i)
  self.listaCar.append_column(column)
```

```
 scrollable_treelist = Gtk.ScrolledWindow()
 scrollable_treelist.set_vexpand(True)
 scrollable_treelist.set_hexpand(True)
 scrollable_treelist.add(self.listaCar)

 return scrollable_treelist

def listaCarpetas(self, usuarios):
 treestore = Gtk.TreeStore(str)
 numCorreo = 0
 for usuario in usuarios:
  carpetas = listdirs(os.path.join(path,usuario))
  piter = treestore.append(None, ['%s' %usuario])
  for carpeta in carpetas:
   numCorreo = contarCorreo(os.path.join(path,usuario,carpeta))
   if carpeta=="Entrada":
    treestore.prepend(piter, ['%s \t %d' %(carpeta, numCorreo)])
   else:
    treestore.append(piter, ['%s \t %d' %(carpeta, numCorreo)])

 treeview = Gtk.TreeView(treestore)
 tvcolumn = Gtk.TreeViewColumn('Cuentas de Correos')
 tvcolumn.set_reorderable(False)
 treeview.append_column(tvcolumn)
 treeview.connect("row-activated",self.celdasCarp)
 cell = Gtk.CellRendererText()
 tvcolumn.pack_start(cell, True)
 tvcolumn.add_attribute(cell, 'text', 0)
 treeview.set_search_column(0)
 tvcolumn.set_sort_column_id(0)
 treeview.set_reorderable(False)
 return treeview

def celdasCorreo(self, treeview, posi, column):
 model = treeview.get_model()
 car = model.get_iter(posi)
 correo = (model.get_value(car, 0),
           model.get_value(car, 1),
           model.get_value(car, 2),
           model.get_value(car, 3))

 for key in self.listaCotejoCorreos:
  aux = self.listaCotejoCorreos[key]
  if ((aux[1]==correo[1]) and (aux[2]==correo[2])):
   archivo=key
   break
```

```
 ruta=os.path.join(path,
                   self.selectUsuario,
                   self.selectCarpeta,
                   archivo)

 fp=open(ruta,"r")
 ms = email.message_from_file(fp)
 fp.close()
 self.destinatorio.set_text("Para: "+ms['To'])
 self.emisor.set_text("De: "+ms['From'])
 self.asunto.set_text("Asunto: "+ms['Subject'])
 textbody = body(ms)
 print(textbody)
 buffered = Gtk.TextBuffer()
 buffered.set_text(textbody.strip())
 self.cuerpo.set_buffer(buffered)
 #print(correo in self.listaCotejoCorreos)

def celdasCarp(self, treeview, posi, column):
 model = treeview.get_model()
 car = model.get_iter(posi)
 carpeta = model.get_value(car, 0)
 if carpeta.find('@')>0:
  return
 carpeta = carpeta.split('\t')[0].strip()
 self.selectCarpeta=carpeta
 usu = model.iter_parent(car)
 usuario = model.get_value(usu, 0)
 self.selectUsuario=usuario
 self.listaCotejoCorreos = listaCorreosView(
                           os.path.join(path,usuario,carpeta))

 software_liststore = Gtk.ListStore(str, str, str, str)
 for reg in self.listaCotejoCorreos:
  software_liststore.append(self.listaCotejoCorreos[reg])
 lista = software_liststore.filter_new()
 self.listaCar.set_model(lista)

def headerMail(self):
 box = Gtk.HBox(False,0)
 botonNewMail = Gtk.Button(label="Nuevo correo")
 botonNewMail.connect("clicked", self.nuevoCorreo,"newMail")
 botonEnviarRecibir = Gtk.Button(label="Enviar y Recibir")
 botonEnviarRecibir.connect("clicked", self.enviarRecibir)
 botonHerramientas = Gtk.Button(label="Herramientas")
 box.pack_start(botonNewMail, False, False, 0)
```

```
 box.pack_start(botonEnviarRecibir, False, False, 0)
 #box.pack_end(botonHerramientas, False, False, 0)
 return box

def nuevoCorreo(self, button,name):
 print(name)
 self.pageNuevoCorreo = self.visorCorreoNuevo()
 self.pageNuevoCorreo.set_border_width(10)
 self.notebook.insert_page(self.pageNuevoCorreo,
                           Gtk.Label("Nuevo Correo"),1)
 self.notebook.show_all()

def enviarRecibir(self, button):
 print(self.config)
 host = self.config["hostPop"]
 port = self.config["portPop"]
 keyfile = self.config["keyfile"]
 certfile = self.config["certfile"]
 user = self.config["user"]
 passwd = self.config["passwd"]
 ssl = self.config["ssl"]
 delete = self.config["delete"]
 ruta = os.path.join(path,user,"Entrada")
 t=conexionPop3(host,
                port,
                keyfile,
                certfile,
                user,
                passwd,
                ssl,
                delete,
                ruta)
 t.start()
 t2=salida(os.path.join(path,user),self.config)
 t2.start()

def newPage(self):
 marco = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=10)
 barra = self.headerMail()
 areaCorreo = Gtk.Box(spacing=10)
 listaCap = Gtk.Box(orientation=Gtk.Orientation.VERTICAL,
                    spacing=10)

 listaCap.add(self.listaCarpetas(self.user))
 areaViewCorreo = Gtk.Box(orientation=Gtk.Orientation.VERTICAL,
                          spacing=10)
```

```
  listaCorreo = Gtk.Box(spacing=10)
  self.listaM = self.listaMail(self.user[0],'Entrada')
  listaCorreo.add(self.listaM)

  viewCorreo = Gtk.Box(spacing=10)
  self.visorCo = self.visorCorreo()
  viewCorreo.add(self.visorCo)

  areaViewCorreo.pack_start(listaCorreo, False, True, 0)
  areaViewCorreo.pack_start(viewCorreo, True, True, 0)
  areaCorreo.add(listaCap)
  areaCorreo.add(areaViewCorreo)

  marco.pack_start(barra, False, False, 0)
  marco.pack_end(areaCorreo, True, True, 0)
  return marco

def cuerpoDk(self,valores,op):
  print(valores)
  bodyBuffer=self.cuerpo.get_buffer()
  start_iter = bodyBuffer.get_start_iter()

  end_iter = bodyBuffer.get_end_iter()
  text = bodyBuffer.get_text(start_iter, end_iter, True)
  firma=text.find("------SSE Cipher------")

  text= text[:firma]
  descifrado=SSE.Ek_din.descifrar(text,valores,op)

  try:
   aux=descifrado.decode("utf8")
   buf = Gtk.TextBuffer()
   print("utf-8 encode")
   buf.set_text(aux.encode("utf8"))
   self.cuerpo.set_buffer(buf)

  except Exception, e:

   print("Error al descifrar")
   dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
    Gtk.ButtonsType.CANCEL, "Error al descifrar")

   dialog.format_secondary_text(
    "El CAPTCHA fue ingresado incorrectamente")
   dialog.run()
   dialog.destroy()
```

```
class ventanaCaptcha(Gtk.Window):

 def __init__(self,ventana,despliegue):
  self.ventana = ventana
  self.ruta=despliegue[0]
  self.archivos=despliegue[1]
  self.op=despliegue[2]
  Gtk.Window.__init__(self, title="Cliente de Correos")
  self.set_border_width(4)
  self.set_default_size(500,300)
  self.add(self.viewCAPTCHAS())
  self.show_all()

 def descifrado(self, button):
  for c in self.box1.get_children():
   for x in c.get_children():
    if isinstance(x,Gtk.Entry):
     print(x.get_name())
     valor=x.get_text()
     print(valor)
  if valor=="":
   dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
    Gtk.ButtonsType.CANCEL, "Error en el CAPTCHA")
   dialog.format_secondary_text(
    "Resolver el CAPTCHA")
   dialog.run()
   dialog.destroy()
  else:
   self.ventana.cuerpoDk(valor,self.op)

 def descifrado2(self, button):
  valor=[]
  for c in self.box1.get_children():
   for x in c.get_children():
    if isinstance(x,Gtk.Entry):
     print(x.get_name())
     aux=x.get_text()
     if aux!="":
      valor.append([self.archivos[x.get_name()],x.get_text()])
      print(valor)
  for c in self.box2.get_children():
   for x in c.get_children():
    if isinstance(x,Gtk.Entry):
     print(x.get_name())
     aux=x.get_text()
     if aux!="":
```

```
            valor.append([self.archivos[x.get_name()],x.get_text()])
            print(valor)
        for c in self.box3.get_children():
         for x in c.get_children():
          if isinstance(x,Gtk.Entry):
           print(x.get_name())
           aux=x.get_text()
           if aux!="":
            valor.append([self.archivos[x.get_name()],x.get_text()])
            print(valor)
        if len(valor)==0:
         dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
          Gtk.ButtonsType.CANCEL, "Error en el CAPTCHA")
         dialog.format_secondary_text(
          "Resolver el CAPTCHA")
         dialog.run()
         dialog.destroy()
        else:
         self.ventana.cuerpoDk(valor,self.op)

    def viewCAPTCHAS(self):
        marco = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=10)
        scrol = Gtk.ScrolledWindow()
        scrol.set_hexpand(True)
        boxGen=Gtk.VBox(False,0)
        boxGen.set_spacing(10)
        boxGen.set_border_width(10)
        separator = Gtk.HSeparator()
        separator.set_size_request(400, 5)
        separator2 = Gtk.HSeparator()
        separator2.set_size_request(400, 5)
        self.box1=Gtk.HBox(False,0)
        boxGen.pack_start(self.box1,False,False,0)
        boxGen.pack_start(separator, False, True, 5)
        self.box2=Gtk.HBox(False,0)
        boxGen.pack_start(self.box2,False,False,0)
        boxGen.pack_start(separator2, False, True, 5)
        self.box3=Gtk.HBox(False,0)
        boxGen.pack_start(self.box3,False,False,0)
        box4=Gtk.HBox(False,0)
        descifrado= Gtk.Button(label="Descifrar")
        if isinstance(self.archivos,dict):
         index=0
         for img in self.archivos.keys():
          self.set_default_size(700,400)
          aux = Gtk.VBox(False,0)
```

```
      texto = Gtk.Entry(name=img)
      image = Gtk.Image()
      rutaImg=os.path.join(self.ruta,img)
      print(rutaImg)
      image.set_from_file(rutaImg)
      image.show()
      aux.pack_start(image,False,False,0)
      aux.pack_end(texto,False,False,0)
      if index<2:
       self.box1.pack_start(aux,False,False,0)
      if index<4:
       self.box2.pack_start(aux,False,False,0)
      else:
       self.box3.pack_start(aux,False,False,0)
      index+=1
     descifrado.connect("clicked", self.descifrado2)

    else:
     for img in self.archivos:
      aux = Gtk.VBox(False,0)
      texto = Gtk.Entry(name=img)
      image = Gtk.Image()

      rutaImg=os.path.join(self.ruta,img)
      print(rutaImg)
      image.set_from_file(rutaImg)
      image.show()
      aux.pack_start(image,False,False,0)
      aux.pack_end(texto,False,False,0)
      self.box1.pack_start(aux,False,False,0)
     descifrado.connect("clicked", self.descifrado)

    box4.pack_end(descifrado,False,False,0)
    marco.pack_start(boxGen,False,False,0)
    marco.pack_start(box4,False,False,0)
    scrol.add(marco)
    return scrol

class configView(Gtk.Window):

 def __init__(self):
  Gtk.Window.__init__(self, title="Configuracion")
  self.set_border_width(4)
  self.set_default_size(500, 600)
  self.add(self.viewConfig())
  self.show_all()
```

```
def viewConfig(self):

 marco = Gtk.Box(orientation=Gtk.Orientation.VERTICAL, spacing=10)
 marco.pack_start(Gtk.Label("Servidor Smtp"),False,False,0)

 self.servidorSmtp=Gtk.Entry()
 marco.pack_start(self.servidorSmtp,False,False,0)
 marco.pack_start(Gtk.Label("Puerto Smtp"),False,False,0)

 self.puertoSmtp=Gtk.Entry()
 marco.pack_start(self.puertoSmtp,False,False,0)
 marco.pack_start(Gtk.Label("Servidor Pop"),False,False,0)

 self.servidorPop=Gtk.Entry()
 marco.pack_start(self.servidorPop,False,False,0)
 marco.pack_start(Gtk.Label("Puerto Pop"),False,False,0)

 self.puertoPop=Gtk.Entry()
 marco.pack_start(self.puertoPop,False,False,0)
 marco.pack_start(Gtk.Label("Usuario de Correo Electronico")
                              ,False,False,0)

 self.usuCorreoElec=Gtk.Entry()
 marco.pack_start(self.usuCorreoElec,False,False,0)
 marco.pack_start(Gtk.Label("Contrasena de Correo Elecctronico")
                              ,False,False,0)

 self.contraCorreoElec=Gtk.Entry()
 self.contraCorreoElec.set_visibility(False)
 marco.pack_start(self.contraCorreoElec,False,False,0)
 marco.pack_start(Gtk.Label("Conexion POP SSL"),False,False,0)

 self.conexSSL=Gtk.Switch()
 self.conexSSL.set_active(False)
 marco.pack_start(self.conexSSL,False,False,0)
 marco.pack_start(Gtk.Label("Usuario del Servidor de CAPTCHAS")
                              ,False,False,0)

 self.usuSerCAPTCHA=Gtk.Entry()
 marco.pack_start(self.usuSerCAPTCHA,False,False,0)
 marco.pack_start(Gtk.Label("Contrasena del Servidor de CAPTCHAS")
                              ,False,False,0)

 self.contraSerCAPTCHA=Gtk.Entry()
 self.contraSerCAPTCHA.set_visibility(False)
 marco.pack_start(self.contraSerCAPTCHA,False,False,0)
```

```
 marco.pack_start(Gtk.Label(
                  "Activar Esquema de Secreto Compartido"),
                  False,False,0)

 self.SSE=Gtk.Switch()
 self.SSE.set_active(False)
 marco.pack_start(self.SSE,False,False,0)
 boton = Gtk.Button(label="Activar")

 boton.connect("clicked", self.Activar)
 marco.pack_start(boton,False,False,0)
 return marco

def Activar(self,button):

 dic={}
 dic["nombre"]=self.usuCorreoElec.get_text()
 dic["contrasena"]=self.contraSerCAPTCHA.get_text()
 dic["correo_electronico"]=self.usuCorreoElec.get_text()

 if http.httpAltaUsu(dic):
  disc={}
  disc["host"]=self.servidorPop.get_text()
  disc["port"]=self.puertoPop.get_text()
  disc["keyfile"]="./Seguridad/server2048.key"
  disc["certfile"]="./Seguridad/server2048.pem"
  disc["user"]=self.usuCorreoElec.get_text()
  disc["passwd"]=self.contraCorreoElec.get_text()
  disc["ssl"]=self.conexSSL.get_active()

  if validarPop(disc):
   disc["host"]=self.servidorSmtp.get_text()
   disc["port"]=self.puertoSmtp.get_text()
   disc["ssl"]=False

   if validarSmtp(disc):
    disc={}
    disc["hostSmtp"]=self.servidorSmtp.get_text()
    disc["portSmtp"]=self.puertoSmtp.get_text()
    disc["hostPop"]=self.servidorPop.get_text()
    disc["portPop"]=self.puertoPop.get_text()
    disc["keyfile"]="./Seguridad/server2048.key"
    disc["certfile"]="./Seguridad/server2048.pem"
    disc["user"]=self.usuCorreoElec.get_text()
    disc["passwd"]=self.contraCorreoElec.get_text()
    disc["ssl"]=self.conexSSL.get_active()
```

```
      disc["delete"]=0
      disc["SSE"]=self.SSE.get_active()
      disc["nombre"]=self.usuSerCAPTCHA.get_text()
      disc["passwdSSE"]=self.contraSerCAPTCHA.get_text()

      lista=open("config.json","w")
      lista.write(json.dumps(disc))
      lista.close

      win = MyWindow(disc)
      win.connect("delete-event", Gtk.main_quit)
      win.show_all()

     else:
      dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
      Gtk.ButtonsType.CANCEL, "Error en el servidor SMTP")
      dialog.format_secondary_text(
       "No se logro estableser comunicacion con el
       servidor SMTP, verificar los datos ingresados")
      dialog.run()
      dialog.destroy()

    else :
     dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
     Gtk.ButtonsType.CANCEL, "Error en el servidor POP")
     dialog.format_secondary_text(
      "No se logro estableser comunicacion con el servidor POP,
      verificar los datos ingresados")
     dialog.run()
     dialog.destroy()

   else:
    dialog = Gtk.MessageDialog(self, 0, Gtk.MessageType.ERROR,
    Gtk.ButtonsType.CANCEL, "Error en el servidor de CAPTCHAS")
    dialog.format_secondary_text(
     "Ocurrio un error en el registro como
     usuario en el servidor de CAPTCHAS")
    dialog.run()
    dialog.destroy()

def setup_logger(logger_name, log_file, level=logging.INFO):

 l = logging.getLogger(logger_name)
 formatter = logging.Formatter('%(asctime)s : %(message)s')
 fileHandler = logging.FileHandler(log_file, mode='w')
 fileHandler.setFormatter(formatter)
```

```
 streamHandler = logging.StreamHandler()
 streamHandler.setFormatter(formatter)

 l.setLevel(level)
 l.addHandler(fileHandler)
 l.addHandler(streamHandler)

setup_logger('debug', r'./logs/debug.log')
setup_logger('errorLog', r'./logs/error.log')
debug = logging.getLogger('debug')
errorLog = logging.getLogger('errorLog')

win=None
if not os.path.exists("config.json"):
 debug.info('Inicia Aplicacion')
 win = configView()
 win.connect("delete-event", Gtk.main_quit)
 win.show_all()
 Gtk.main()

else:
 jsonCorreo = open("config.json","r")
 jsonLectura = jsonCorreo.readline()
 jsonCorreo.close()
 configuracion = json.loads(jsonLectura)
 win = MyWindow(configuracion)
 win.connect("delete-event", Gtk.main_quit)
 win.show_all()
 Gtk.main()
```

- Conexión SMTP (smtp2.py).

```
import os
import smtplib
import mimetypes
import hashlib
import time
import email
# For guessing MIME type based on file name extension
from email import encoders
from email.message import Message
from email.mime.audio import MIMEAudio
from email.mime.base import MIMEBase
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
```

```
def validarSmtp(dat):
 print("abriendo conexion")
 try:
  if dat["ssl"]:
   M = smtplib.SMTP_SSL(host=dat["host"], port=dat["port"],
                        keyfile=dat["keyfile"],
                        certfile=dat["certfile"])
  else:
   M = smtplib.SMTP(host=dat["host"], port=dat["port"])
  #M.set_debuglevel(True)
 except Exception, e:
  print(e)
  print("Error de conexion")
  return False
 print("Validando usuario")
 try:
  M.ehlo()
  M.starttls()
  M.ehlo()
  M.login(dat["user"], dat["passwd"])
  M.close()
  return True
 except Exception, e:
  print("Invalid credentials")
  return False

def smtpOneMensaje(to, subject, fromUser, text, attach, passwd
                   , server, port, op, ssl):
 outer = MIMEMultipart()
 if to == None:
  return 0
 elif fromUser == None:
  return 1
 elif (subject == None) and (text == None):
  return 2
 elif passwd == None:
  return 3
 elif server == None:
  return 4
 elif port == None:
  return 5

 outer['From'] = fromUser
 outer['To'] = to
 outer['Subject'] = subject
 outer['Date'] = time.asctime(time.localtime(time.time()))
```

```
outer.attach(MIMEText(text))
for path in attach:
 if not os.path.isfile(path):
  continue

 ctype, encoding = mimetypes.guess_type(path)
 if ctype is None or encoding is not None:
  ctype = 'application/octet-stream'
 maintype, subtype = ctype.split('/', 1)
 if maintype == 'text':
  fp = open(path)
  msg = MIMEText(fp.read(), _subtype=subtype)
  fp.close()

 elif maintype == 'image':
  fp = open(path, 'rb')
  msg = MIMEImage(fp.read(), _subtype=subtype)
  fp.close()

 elif maintype == 'audio':
  fp = open(path, 'rb')
  msg = MIMEAudio(fp.read(), _subtype=subtype)
  fp.close()
 else:
  fp = open(path, 'rb')
  msg = MIMEBase(maintype, subtype)
  msg.set_payload(fp.read())
  fp.close()
  encoders.encode_base64(msg)
 msg.add_header('Content-Disposition',
                'attachment',
                filename=os.path.basename(path))

 outer.attach(msg)
composed = outer.as_string()
if op:
 m = hashlib.md5()
 m.update(time.asctime(time.localtime(time.time())))
 aux = m.hexdigest()+".txt"
 fp = open(aux, 'w')
 fp.write(composed)
 fp.close()
 return 6
else:
 print("abriendo conexion")
 try:
```

```
    if ssl:
     s = smtplib.SMTP_SSL(server,port)
    else:
     s = smtplib.SMTP(server,port)
    #s.set_debuglevel(2)
    s.ehlo()
    print("tls")
    s.starttls()
    s.ehlo()
    print("login")
    print(fromUser,passwd)
    s.login(fromUser, passwd)
    print("enviando correo")
    s.sendmail(fromUser, to, composed)
    print("fin")
    s.close()
    return 6
   except Exception, e:
    m = hashlib.md5()
    m.update(time.asctime(time.localtime(time.time())))
    aux = m.hexdigest()+".txt"
    fp = open(aux, 'w')
    fp.write(composed)
    fp.close()
    return 8

def smtpAllMensaje(correos, user, passwd, server, port, ssl):
 if correos == None:
  return 0
 if user == None:
  return 1
 elif passwd == None:
  return 3
 elif server == None:
  return 4
 elif port == None:
  return 5

 print("abriendo conexion")
 try:
  if ssl:
   s = smtplib.SMTP_SSL(server,port)
  else:
   s = smtplib.SMTP(server,port)
  #s.set_debuglevel(2)
  s.ehlo()
```

```
  print("tls")
  s.starttls()
  s.ehlo()
  print("login")
  s.login(user, passwd)
  print("enviando correo")
  for ms in correos:
   try:
    composed = ms.as_string()
    time.sleep(1)
    s.sendmail(ms['From'], ms['To'], composed)
   except Exception, e:
    m = hashlib.md5()
    m.update(time.asctime(time.localtime(time.time())))
    aux = m.hexdigest()+".txt"
    fp = open(aux, 'w')
    fp.write(composed)
    fp.close()
  print("fin")
  s.close()
  return 6

 except Exception, e:
  return 8

def smtpEnpaquetar(to, subject, fromUser, text, attach):
 outer = MIMEMultipart()
 if to == None:
  return 0
 elif fromUser == None:
  return 1
 elif (subject == None) and (text == None):
  return 2

 outer['From'] = fromUser
 outer['To'] = to
 outer['Subject'] = subject
 outer['Date'] = time.asctime(time.localtime(time.time()))

 outer.attach(MIMEText(text))
 for path in attach:
  if not os.path.isfile(path):
   continue
  ctype, encoding = mimetypes.guess_type(path)
  if ctype is None or encoding is not None:
   ctype = 'application/octet-stream'
```

```
  maintype, subtype = ctype.split('/', 1)
  if maintype == 'text':
   fp = open(path)
   msg = MIMEText(fp.read(), _subtype=subtype)
   fp.close()

  elif maintype == 'image':
   fp = open(path, 'rb')
   msg = MIMEImage(fp.read(), _subtype=subtype)
   fp.close()

  elif maintype == 'audio':
   fp = open(path, 'rb')
   msg = MIMEAudio(fp.read(), _subtype=subtype)
   fp.close()

  else:
   fp = open(path, 'rb')
   msg = MIMEBase(maintype, subtype)
   msg.set_payload(fp.read())
   fp.close()
   encoders.encode_base64(msg)
  msg.add_header('Content-Disposition',
                 'attachment',
                 filename=os.path.basename(path))

  outer.attach(msg)
 return outer

def smtpEnvio(ms,server,port,passwd,ssl):
 print("enviando smtp")
 fromUser = ms['From']
 to = ms['To']
 composed = ms.as_string()
 print("Datos del mensaje")
 print("Servidor: "+server)
 print("Port: "+str(port))
 print("Pass: "+str(passwd))
 try:
  s = smtplib.SMTP(server,port)
  s.ehlo()
  print("tls")
  s.starttls()
  s.ehlo()
  print("login")
  s.login(fromUser, passwd)
```

```
  print("enviando correo")
  s.sendmail(fromUser, to, composed)
  print("fin")
  s.close()
  return 0
 except Exception, e:
  return 1

def datosPrincipales(arc):
 fp = open(arc, 'r')
 ms = email.message_from_file(fp)
 fp.close()
 if len(ms.get_payload())>1:
  res =(arc,ms['Subject'],ms['From'],ms['Date']," mail-attachment")
 else:
  res =(ms['Subject'],ms['From'],ms['Date'],"")
 return list(res)
```

- Conexión POP3 (pop3.py).

```
import os
import poplib
import string
import StringIO
import email
import hashlib
import listarCorreos
import threading
from listarCorreos import listaCorreos
# For guessing MIME type based on file name extension
from email import encoders
from email.message import Message
from email.mime.audio import MIMEAudio
from email.mime.base import MIMEBase
from email.mime.image import MIMEImage
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

#path="/home/jonnytest/Documentos/Usuarios/"
class conexionPop3(threading.Thread):

 def __init__(self, host, port, keyfile, certfile, user, passwd,
              ssl, delete, path):
  threading.Thread.__init__(self)
  self.host = host
  self.port = port
```

```
  self.keyfile = keyfile
  self.certfile = certfile
  self.user = user
  self.passwd = passwd
  self.ssl = ssl
  self.delete = delete
  self.path = path

def run(self):
  print("abriendo conexion")
  try:
   if self.ssl:
    M = poplib.POP3_SSL(host=self.host, port=self.port,
                        keyfile=self.keyfile,
                        certfile=self.certfile)
   else:
    M = poplib.POP3(host=self.host, port=self.port)
   #M.set_debuglevel(2)
  except poplib.error_proto, e:
   print(e)
   print("Error de conexion")
   return 0;
  success = False
  print("Validando usuario")
  while success == False:
   lista={}
   try:
    M.user(self.user)
    M.pass_(self.passwd)
    numMesanjes = len(M.list()[1])
    for id in range(1,(numMesanjes+1)):
     resp, text, octets = M.retr(id)
     text = string.join(text, "\n")
     ms = email.message_from_string(text)
     print(ms["Date"])
     m = hashlib.md5()
     m.update(ms["Date"])
     aux = m.hexdigest()+".txt"
     file=os.path.join(self.path,aux)
     if os.path.exists(file):
      if self.delete:
       m.dele(id)
     else:
      composed = ms.as_string()
      fp = open(file, 'w')
      fp.write(composed)
```

```
      fp.close()
      if self.delete:
       m.dele(id)
    listaCorreos(self.path)
   except poplib.error_proto:
    print("Invalid credentials")
   else:
    print("Successful login")
    success = True
   finally:
    if M:
     M.quit()

def validarPop(dat):
 print("abriendo conexion")
 try:
  if dat["ssl"]:
   M = poplib.POP3_SSL(host=dat["host"], port=dat["port"],
                       keyfile=dat["keyfile"],
                       certfile=dat["certfile"])
  else:
   M = poplib.POP3(host=dat["host"], port=dat["port"])
  #M.set_debuglevel(2)
 except Exception, e:
  print(e)
  print("Error de conexion")
  return False
 print("Validando usuario")
 try:
  M.user(dat["user"])
  M.pass_(dat["passwd"])
  assert M.noop() == '+OK'
 except poplib.error_proto:
  print("Invalid credentials")
  return False
 else:
  print("Successful login")
  success = True
  return True
 finally:
  if M: M.quit()
```

- Conexión con el servidor de CAPTCHAS (http.py).

```
import urllib
import urllib2
import re
from poster.encode import multipart_encode
from poster.streaminghttp import register_openers

def httpEnvio(data):
 register_openers()
 datagen, headers = multipart_encode(data)
 request = urllib2.Request(
           "http://correocifrado.esy.es/AltaMensage.php",
           datagen,
           headers)
 found=''
 try:
  for line in urllib2.urlopen(request):
   print(line)
   if line.find("name=\"respuesta\"") >= 0:
    m = re.search('>([0-9]+)<',line)
    if m:
     found = m.group(1)
     print(found)
     if found=="5":
      print("correcto")
      return True
     else:
      return False
    else:
     return False
 except Exception, e:
  return False

def httpDescarga(data,nomArc):
 register_openers()
 datagen, headers = multipart_encode(data)
 print("peticion Server")
 try:
  request = urllib2.Request(
            "http://correocifrado.esy.es/BusquedaArchivo.php",
            datagen,
            headers)
  found=''
  for line in urllib2.urlopen(request):
   if line.find("name=\"respuesta\"") >= 0:
    m = re.search('>(http.+zip)<',line)
```

```
    if m:
     found = m.group(1)
     print(found)
     break
    else:
     return False

 except urllib2.HTTPError, e:
  print e.code
  return False

 except urllib2.URLError, e:
  print e.args
  return False

 try:
  res=urllib2.urlopen(found)
  arc=open(nomArc,"w")
  arc.write(res.read())
  arc.close()
  return True

 except urllib2.HTTPError, e:
  print e.code
  return False

 except urllib2.URLError, e:
  print e.args
  return False

def httpAltaUsu(data):
 register_openers()
 datagen, headers = multipart_encode(data)
 request = urllib2.Request(
           "http://correocifrado.esy.es/AltaUsuario.php",
           datagen,
           headers)

 found=''
 try:
  for line in urllib2.urlopen(request):
   print(line)
   if line.find("name=\"respuesta\"") >= 0:
    m = re.search('>([0-9]+)<',line)
    if m:
     found = m.group(1)
```

```
      print(found)
      if found=="5":
       print("correcto")
       return True
      else:
       return False
     else:
      return False
 except Exception, e:
  return False
```

- Envío de CAPTCHAS (envios.py).

```
import threading
import hashlib
import time
import os
import re
import urllib2
import email
import SSE
import http
from subprocess import call
from SSE import empaquetar
#from empaquetar import empaquetar
from subprocess import call

from smtp2 import smtpEnpaquetar, smtpEnvio
from listarCorreos import listaCorreos, listaCorreosView

class envios(threading.Thread):

 def __init__(self, correoDes, correoOri, asunto, body,
              attach, config):

  threading.Thread.__init__(self)
  self.correoD = correoDes
  self.correoO = correoOri
  self.asunto = asunto
  self.cuerpo = body
  self.attachment = attach
  self.configuracion=config

 def run(self):
  print("envios")
  firma =self.firma()
```

```
 body,ruta = empaquetar(self.cuerpo,
                        firma,
                        self.configuracion["SSE"])

 body += "\n-------SSE Cipher-------\n"
 body += firma
 body += "\n-------SSE Cipher-------\n"
 print(body)
 mv=call("mv "+ruta+" ./CAPTCHAS/"+firma+".zip", shell=True)
 ms=smtpEnpaquetar(self.correoD,
                   self.asunto,
                   self.correoO,
                   body,
                   self.attachment)

 path=os.path.join("./Usuarios",self.correoO,
                   "Salida")
 print(ms.as_string())
 print(os.path.join(path,firma+".txt"))
 fp = open(os.path.join(path,firma+".txt"), 'w')
 fp.write(ms.as_string())
 fp.close()
 print(path)
 listaCorreos(path)
 self.enviarCorreos(os.path.join("./Usuarios",self.correoO),
                    "Salida")

def firma(self):
 m = hashlib.md5()
 localtime = time.asctime( time.localtime(time.time()) )
 m.update(self.correoD+localtime+self.correoO)
 return m.hexdigest()

def enviarCorreos(self,path,carpeta):
 try:
  response=urllib2.urlopen('http://correocifrado.esy.es',
                           timeout=1)

  print("Conexion al servidor")
  salida=os.path.join(path,carpeta)
  dic=listaCorreosView(salida)
  print("Diccionario")
  httpPar={}
  httpPar["nombre"]=self.configuracion["nombre"]
  httpPar["contrasena"]=self.configuracion["passwdSSE"]
  for arc in dic.keys():
```

```
    try:
     m = re.search('^(.+)\.txt$',arc)
     firma = m.group(1)
     print("Firma:    "+firma)
     fp = open(os.path.join(salida,arc), 'r')
     ms = email.message_from_file(fp)
     fp.close()
     httpPar["correo_electronico"]=ms["From"]
     httpPar["correo_destino"]=ms["To"]
     httpPar["firma"]=firma
     httpPar["archivo"]=open(
                          os.path.join("./CAPTCHAS",firma+".zip"))

     print(os.path.join("./CAPTCHAS",firma+".zip"))
     print(httpPar)
     if http.httpEnvio(httpPar):
      smtpEnvio(ms,self.configuracion["hostSmtp"],
                self.configuracion["portSmtp"],
                self.configuracion["passwd"],
                False)

      origen = os.path.join(salida,arc)
      destino =os.path.join(path,"Enviados",arc)
      instruc = "mv "+origen+" "+destino
      call(instruc, shell=True)
    except Exception, e:
     print("Error al enviar Correo Electronico")
   listaCorreos(os.path.join(path,"Enviados"))
   listaCorreos(salida)
  except urllib2.URLError as err:
   print("Sin conexion")
#aux = envios("sdfg", "dsfg","asdfd","dfg",[])
#aux.run()
```

- Busqueda de CAPTCHAS (captchas.py).

```
import os
from os import path
import json
import http
from subprocess import call

def listFiles(folder):
 return [d for d in os.listdir(folder)
         if path.isfile(path.join(folder, d))]
```

```
def buscarCAPTCHAS(firma, correoDes, correoOri):
 ruta=path.join("./CAPTCHAS",firma)
 if path.exists(ruta):
  return listarCaptchas(ruta)
 elif path.exists(ruta+".zip"):
  unzip="unzip "+ruta+".zip -d ./CAPTCHAS"
  print(unzip)
  unz=call(unzip, shell=True)
  return listarCaptchas(ruta)
 else:
  datos={}
  datos["correo_electronico"]=correoOri
  datos["correo_destino"]=correoDes
  datos["firma"]=firma
  if http.httpDescarga(datos,ruta+".zip"):
   unzip="unzip "+ruta+".zip -d ./CAPTCHAS"
   print(unzip)
   unz=call(unzip, shell=True)
   rm=call("rm "+ruta+".zip", shell=True)
   return listarCaptchas(ruta)
  else:
   return []

#unzip archivo
def listarCaptchas(ruta):
 if path.exists(path.join(ruta,"lista.json")):
  op=1
  jsonCorreo = open(path.join(ruta,"lista.json"),"r")
  jsonLectura = jsonCorreo.readline()
  jsonCorreo.close()
  archivos = json.loads(jsonLectura)
  print(archivos)
 else:
  archivos = listFiles(ruta)
  print("NumArchivos: "+str(len(archivos)))
  print(archivos)
  op=0
 print([ruta,archivos,op])
 return [ruta,archivos,op]
```

- Listado de mensajes de correo electrónico (listarCorreos.py)

```
import json
import os
import email
import quopri

from os import path

def listFiles(folder):
 return [d for d in os.listdir(folder)
         if path.isfile(path.join(folder, d))]

def listaCorreos(ruta):
 print("listaCorreos")
 archivos = listFiles(ruta)
 disc={}
 for archivo in archivos:
  if archivo.find(".txt")>0:
   fp = open(path.join(ruta,archivo), 'r')
   ms = email.message_from_file(fp)
   fp.close()
   if len(ms.get_payload())>1:
    disc[archivo]=(ms['Subject'],
                   ms['From'],
                   ms['Date'],
                   "mail-attachment")
   else:
    disc[archivo]=(ms['Subject'],ms['From'],ms['Date'],"")
 lista=open(path.join(ruta,"lista.json"),"w")
 lista.write(json.dumps(disc))
 lista.close
 print("Fin listaCorreos")

def listaCorreosView(ruta):
 print("listaCorreosView")
 disc={}
 jsonCorreo = open(os.path.join(ruta,"lista.json"),"r")
 if jsonCorreo:
  jsonLectura = jsonCorreo.readline()
  jsonCorreo.close()
  dic = json.loads(jsonLectura)
  return dic
 else:
  return None
 print("Fin listaCorreosView")
```

```
def contarCorreo(directorio):
 print("contarCorreo")
 if directorio==None:
  return -1
 for files in os.walk(directorio):
  num=0
  for file in files[2]:
   if file.find(".txt")>0:
    num+=1
 print("Fin contarCorreo")
 return num

def body(ms):
 print("body")
 charset = ms.get_content_charset()
 if ms.is_multipart():
  for payload in ms.get_payload():
   ctype=payload.get_content_type()
   cdispo = str(payload.get('Content-Disposition'))
   if payload.is_multipart():
    return body(payload)
    break
   elif ctype=='text/plain' and 'attachment' not in cdispo:
    charset = payload.get_content_charset()
    aux = payload.get_payload(decode=True)
    print("Fin body")
    return aux.decode(charset)
 else:
  aux = ms.get_payload(decode=True)
  print("Fin body")
  return aux.decode(charset)
```

Referencias

[1] Cifrado simetrico. Guía de Gnu Privacy Guard, 2015. https://www.gnupg.org/gph/es/manual/c190.html#AEN201.

[2] em client. eM Client web page, 2015. http://www.emclient.com/.

[3] Opera mail. Opera Mail web pages, 2015. http://www.opera.com/es-419/computer/mail.

[4] Post box. Post Box web page, 2015. https://www.postbox-inc.com/.

[5] Thunderbird. Thunderbird web pages, 2015. https://www.mozilla.org/es-ES/thunderbird/.

[6] Zimbra. Zimbra web pages, 2015. https://www.zimbra.com/.

[7] R. Allenby. *Rings, fields, and groups: an introduction to abstract algebra*. E. Arnold, 1983.

[8] Alonsojpd. Montar un servidor de correo electrónico mail en linux ubuntu. AJPDsoft, 2015. http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=506.

[9] T. Brehm. The perfect server - ubuntu 15.10 (wily werewolf) with apache, php, mysql, pureftpd, bind, postfix, dovecot and ispconfig 3. How to Forge, 2015. https://www.howtoforge.com/tutorial/ubuntu-perfect-server-with-apache-php-myqsl-pureftpd-bind-postfix-doveot-and-ispc

[10] M. Brodsky. Reflexiones jurídicas sobre el e-marketing en chile. Interactive Advertising Bureau, 2015. http://www.iab.cl/reflexiones-juridicas-sobre-el-e-marketing-en-chile/.

[11] D. Chakraborty and F. Rodríguez-Henríquez. Block cipher modes of operation from a hardware implementation perspective. In Ç. K. Koç, editor, *Cryptographic Engineering*, pages 321–363. Springer, 2009.

[12] S. Diaz-Santiago and D. Chakraborty. On securing communication from profilers. In P. Samarati, W. Lou, and J. Zhou, editors, *SECRYPT 2012 - Proceedings of the International Conference on Security and Cryptography, Rome, Italy, 24-27 July, 2012, SECRYPT is part of ICETE - The International Joint Conference on e-Business and Telecommunications*, pages 154–162. SciTePress, 2012.

[13] P. Golle and A. Farahat. Defending email communication against profiling attacks. In *Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society, WPES 2004, Washington, DC, USA, October 28, 2004*, pages 39–40, 2004.

[14] A. Gulbrandsen and N. Freed. Internet Message Access Protocol (IMAP) - MOVE Extension. RFC 6851, 2015.

[15] D. Jurafsky and J. H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[16] D. J. C. Klensin. Simple Mail Transfer Protocol. RFC 5321, 2015.

[17] J. Klensin. Simple Mail Transfer Protocol. RFC 2821 (Proposed Standard), April 2001. Obsoleted by RFC 5321, updated by RFC 5336.

[18] W. Koch. The gnu privacy guard. GnuPG web page, 2016. `https://www.gnupg.org/index.html`.

[19] D. P. Martínez. Postgresql vs. mysql. geekWare, 2015. `https://danielpecos.com/documents/postgresql-vs-mysql/`.

[20] J. Myers and M. Rose. Post Office Protocol - Version 3. RFC 1939 (Standard), 1996. Updated by RFCs 1957, 2449.

[21] J. Peralta. Anillos y cuerpos. Campus Virtual Univesidad de Almería, 2016. `http://www.ual.es/personal/jperalta/anilloscuerpos.pdf`.

[22] N. Roshanbin and J. Miller. A survey and analysis of current captcha approaches. *J. Web Eng.*, 12(1-2):1–40, 2013.

[23] sawiyati. How to install apache, php and mariadb on ubuntu 15.04. Server Mom, 2015. `http://www.servermom.org/install-apache-php-mariadb-ubuntu-15-04/2208/`.

[24] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[25] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 5a edition, 2002.

[26] D. R. Stinson. *Cryptography Theory and Practice*. Chapman & Hall/CRC, 3a edition, 2006.

[27] O. Tezer. Sqlite vs mysql vs postgresql: A comparison of relational database management systems. Digital Ocean, 2014. `https://www.digitalocean.com/community/tutorials/sqlite-vs-mysql-vs-postgresql-a-comparison-of-relational-database-management-syst`

[28] M. R. S. Villanueva. Aritmética del reloj. Departamento de Matemáticas de la Universidad de Puerto Rico en Aguadilla, 2016. `http://math.uprag.edu/milena/4.5%20ARITMETICA%20DEL%20RELOJ.pdf`.

[29] Wikipedia. Ciphertext-only attack — Wikipedia, the free encyclopedia, 2015. https://en.wikipedia.org/wiki/Ciphertext-only_attack.

[30] Wikipedia. Email — Wikipedia, the free encyclopedia, 2015. http://en.wikipedia.org/wiki/Email.

[31] Wikipedia. Pretty good privacy — Wikipedia, the free encyclopedia, 2015. https://es.wikipedia.org/wiki/Pretty_Good_Privacy.

[32] L. G. G. y Dr. Sergio Rajsbaum. Critofrafía. Temas selectos de la web, 2015. http://www.matem.unam.mx/rajsbaum/cursos/web/presentacion_seguridad_1.pdf.

[33] E. A. M. y M.C. Ma. Jaquelina López Barrientos. Fundamentos de critofrafía. Universidad Nacional Autonoma de México Facultad de Ingenieria, 2015. http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/.