



INSTITUTO POLITÉCNICO NACIONAL
Escuela Superior de Cómputo

ESCOM

Trabajo Terminal

“Protocolo criptográfico para el almacenamiento
sin duplicados en la nube, resistente a ataques
por fuerza bruta.”

2016-B045

Presentan

Eder Jonathan Aguirre Cruz
Diana Leslie González Olivier
Jhonatan Saulés Cortés

Directora

Dra. Sandra Díaz Santiago

Índice

1. Introducción	1
1.1. Propuesta de solución.	3
1.2. Objetivos.	4
1.2.1. Objetivo General.	4
1.2.2. Objetivos Específicos.	4
1.3. Organización del documento.	5
2. Marco Teórico	6
2.1. Criptografía.	6
2.2. Cifrador por bloques.	9
2.3. Funciones Hash.	10
2.4. RSA	10
2.5. Firma digital	13
2.6. Firma digital con RSA.	14
2.7. Firmas a ciegas.	15
2.8. Firmas a ciegas con RSA.	16
2.9. Cómputo Nube.	17
3. Protocolos criptográficos para evitar duplicados	21
3.1. Estado del Arte.	21
3.2. Protocolo DupLESS.	22
4. Análisis	26
4.1. Estudio de Factibilidad	26
4.1.1. Factibilidad técnica	26
4.1.2. Factibilidad operativa	28
4.1.3. Factibilidad económica	29
4.2. Modelo del Proceso de Desarrollo del Software	29
4.3. Análisis de riesgos	31
4.3.1. Escalas de impacto de un riesgo	31
4.4. Arquitectura del sistema.	34
4.5. Descripción de procesos.	42
4.5.1. Descripción del proceso subir archivo.	42
Participantes	43
4.5.2. Descripción del proceso Descargar archivo.	44
Participantes	44
4.5.3. Descripción del proceso eliminar archivo.	45
Participantes	46
4.6. Modelo de entidades.	47
4.6.1. Diagrama de Entidad Relación.	47
4.6.2. Diagrama de clases.	48

4.7.	Requerimientos Funcionales	49
4.8.	Requerimientos No Funcionales	50
4.9.	Reglas de Negocio	52
5.	Diseño	54
5.1.	Especificación de Plataforma	54
5.2.	Casos de Uso	55
5.2.1.	CUSLL1 Generar las llaves del servidor de llaves	56
Descripción completa	56	
Atributos importantes	56	
Trayectorias del Caso de Uso	56	
5.2.2.	CUSLL2 Generar firma ciega (y)	58
Descripción completa	58	
Atributos importantes	58	
Trayectorias del Caso de Uso	58	
5.2.3.	CUCL1 Subir archivo	59
Descripción completa	59	
Atributos importantes	59	
Trayectorias del Caso de Uso	60	
5.2.4.	CUCL3 Descargar archivos	62
Descripción completa	62	
Atributos importantes	62	
Trayectorias del Caso de Uso	62	
5.2.5.	CUCL4 Eliminar archivos cifrado	64
Descripción completa	64	
Atributos importantes	64	
Trayectorias del Caso de Uso	64	
5.2.6.	CUCL6 Iniciar Sesión	66
Descripción completa	66	
Atributos importantes	66	
Trayectorias del Caso de Uso	66	
5.2.7.	CUCL7 Registrar usuario	69
Descripción completa	69	
Atributos importantes	69	
Trayectorias del Caso de Uso	69	
5.3.	Diagramas de secuencia	71
5.3.1.	Registrar Usuario	71
5.3.2.	Iniciar Sesión	75
5.3.3.	Subir Archivo	79
5.3.4.	Firma a ciegas	82
5.3.5.	Descargar Archivo	83
5.3.6.	Eliminar Archivo	85
5.4.	Mensajes del sistema	86

5.4.1. Mensajes	87
6. Desarrollo	91
6.1. Descripción General del Desarrollo del Protocolo	91
6.2. Servidor de Llaves	91
Objetivo	91
Entradas	92
6.3. Aplicación Criptográfica (Cifrado/Descifrado)	93
6.3.1. Cifrado	93
Objetivo	94
Entradas	94
6.3.2. Descifrado	95
Objetivo	95
Entradas	95
6.4. Interfaz Web	96
6.5. Mapa de Navegación.	97
6.5.1. Registrar un Nuevo Usuario.	98
6.5.2. Iniciar Sesión de un usuario.	99
6.5.3. Subir un archivo.	100
Cifrar un archivo.	101
Duplicación.	104
6.5.4. Descargar un archivo	107
Descifrar un archivo	107
6.5.5. Eliminar un archivo	109
6.6. Funcionamiento de la Interfaz Web	111
6.6.1. Pantalla Principal	111
6.6.2. Pantalla Registro de Usuario	111
6.6.3. Registrandote por primera vez	112
6.6.4. Pantalla de la Base de Datos	112
6.6.5. Pantalla Inicio de Sesión	113
6.6.6. Pantalla Tabla Cipher	113
6.6.7. Pantalla Tabla Hashes	114
6.6.8. Pantalla del Perfil del usuario	114
6.6.9. Pantalla Subir Archivo	115
6.6.10. Pantalla Elegir archivo a Subir	115
6.6.11. Pantalla del primer caso: Cuando Subimos un Archivo Nuevo	116
Pantalla de Perfil Actualizado	116
Pantalla Tabla Hashes Actualizada	117
Pantalla Tabla Cipher Actualizada	117
6.6.12. Pantalla Subir Otro Archivo Nuevo	118
Pantalla Tabla Cipher Actualizada	118
Pantalla Tabla Hashes Actualizada	119

6.6.13. Pantalla segundo caso: Cuando otro usuario sube el mismo archivo con el mismo nombre.	119
Inicio de Sesión con otro usuario	119
Perfil de Usuario	120
Subir Archivo	120
Tabla Cipher actualizada	121
Tabla Hashes actualizada	121
Perfil Actualizado	122
Tabla Cipher actualizada	122
Tabla Hashes actualizada	123
6.6.14. Perfil de usuario1 actualizado con distintos formatos de archivos	123
Perfil de usuario2 actualizado con distintos formatos de archivos	124
Tabla Cipher actualizada	124
Tabla Hashes actualizada	125
6.6.15. Descargar Archivo	125
Seleccionando Archivo	126
Archivo Descargado	126
Archivo Descargado en la Carpeta	127
Archivos Descargados del usuario 1 en la Carpeta	127
Archivos Descargados del usuario 2 en la Carpeta	128
6.6.16. Eliminar Archivo	128
Seleccionando Archivo a eliminar	129
Tabla Cipher con Archivo eliminado de usuario 1	129
Pantalla archivos actualizados	130
Tabla Cipher con Archivo eliminado de usuario 2	130
Pantalla archivos actualizados de usuario 2	131
Tabla Hashes con Archivo eliminado de los dos usuarios	131
7. Lista de acrónimos	132
7.1. Definiciones, acrónimos y abreviaturas	132
Bibliografía	134

Índice de Figuras

1.1.	Crecimiento global del tráfico en la nube	1
1.2.	Duplicación de Archivos en la Nube	2
1.3.	Eliminación de Duplicados	3
2.1.	Diagrama de Criptografía Simétrica.	7
2.2.	Diagrama de Criptografía Asimétrica.	7
2.3.	Diagrama de Cifradores por Bloques	10
3.1.	Servicios de almacenamiento en la nube	23
4.1.	Modelo de proceso de desarrollo de software en Cascada con retroalimentación	29
4.2.	Arquitectura general del Protocolo	34
4.3.	Arquitectura del Protocolo paso 1.	35
4.4.	Arquitectura del Protocolo paso 2.	36
4.5.	Arquitectura del Protocolo paso 3.	37
4.6.	Arquitectura del Protocolo paso 4.	38
4.7.	Arquitectura del Protocolo paso 5.	39
4.8.	Arquitectura del Protocolo paso 6.	40
4.9.	Arquitectura del Protocolo paso 7.	41
4.10.	Arquitectura del Protocolo paso 8.	42
4.11.	BPMN Subir archivo.	43
4.12.	BPMN Descargar archivo.	45
4.13.	BPMN Eliminar archivo.	46
4.14.	Diagrama Entidad relación del sistema.	47
4.15.	Diagrama de clases del sistema.	48
5.1.	Diagrama de Casos de Uso del sistema.	55
5.2.	Diagrama de secuencias de Registrar un usuario nuevo.	71
5.3.	Diagrama de secuencias de Registrar un usuario nuevo con datos incorrectos.	72
5.4.	Diagrama de secuencias de Registrar un usuario nuevo con datos incompletos.	73
5.5.	Diagrama de secuencias de Registrar un usuario nuevo con usuario repetido.	74
5.6.	Diagrama de secuencias de Iniciar sesión un usuario.	75
5.7.	Diagrama de secuencias de Iniciar sesión un usuario con datos incorrectos. .	76
5.8.	Diagrama de secuencias de Registrar un usuario nuevo con datos incompletos.	77

5.9.	Diagrama de secuencias de Iniciar sesión un usuario con usuario repetido.	78
5.10.	Diagrama de secuencias de subir un archivo nuevo.	79
5.11.	Diagrama de secuencias de subir un archivo con carpeta vacía.	80
5.12.	Diagrama de secuencias de subir un archivo incompatible.	80
5.13.	Diagrama de secuencias de subir un archivo existente.	81
5.14.	Diagrama de secuencias de la firma a ciegas del servidor de llaves.	82
5.15.	Diagrama de secuencias de descargar un archivo de la nube.	83
5.16.	Diagrama de secuencias de descargar un archivo de la nube no encontrado. . .	84
5.17.	Diagrama de secuencias de eliminar un archivo de la nube.	85
6.1.	Mapa de navegación	98
6.2.	Pantalla Principal de la interfaz web.	111
6.3.	Pantalla de Registro del usuario.	111
6.4.	Pantalla con el formulario lleno con los datos de un usuario nuevo.	112
6.5.	Tabla de Registro de Usuario en la Base de Datos.	112
6.6.	Pantalla de Inicio de Sesión.	113
6.7.	Tabla que nombramos Cipher donde se almacenan los archivos cifrados del usuario.	113
6.8.	Tabla que nombramos Hashes donde se almacena el hash de los archivos almacenados en la nube.	114
6.9.	Pantalla del perfil de usuario, donde puede ver sus archivos y realizar otras acciones.	114
6.10.	Pantalla al dar clic en la opción Subir Archivo.	115
6.11.	Al dar clic en seleccionar archivo, despliega esta pantalla donde podemos elegir un archivo para almacenarlo en la nube.	115
6.12.	Una vez que elegimos el archivo aparecerá como en esta pantalla.	116
6.13.	En el perfil de usuario aparecerá una lista de los archivos cifrados que tiene almacenados en la nube.	116
6.14.	En esta pantalla mostramos la tabla Hashes actualizada.	117
6.15.	En esta pantalla mostramos que al subir el archivo ya se cifró y almacenó en la base de datos.	117
6.16.	En esta pantalla mostramos que el usuario subió otro archivo y se actualizó su perfil.	118
6.17.	En esta pantalla mostramos la tabla Cipher actualizada.	118
6.18.	En esta pantalla mostramos la tabla Hashes actualizada.	119
6.19.	Pantalla de Inicio de Sesión de un usuario diferente.	119
6.20.	Pantalla de Perfil.	120
6.21.	Pantalla de Perfil actualizada ya que el usuario subió un archivo, como es un duplicado sólo se guarda el c2 de dicho archivo.	120
6.22.	Tabla Cipher actualizada.	121
6.23.	Tabla Hashes actualizada.	121
6.24.	Perfil actualizado ya que el nuevo usuario subió más archivos.	122
6.25.	Tabla Cipher actualizada.	122

6.26. Tabla Hashes actualizada.	123
6.27. Pantalla donde se muestran distintos tipos de archivos cifrados almacenados en la nube.	123
6.28. Pantalla donde se muestran distintos tipos de archivos cifrados almacenados en la nube.	124
6.29. Tabla Cipher actualizada.	124
6.30. Tabla Hashes actualizada.	125
6.31. Pantalla donde se muestran el formulario donde se puede descargar los archivos.	125
6.32. Pantalla donde se muestra el archivo seleccionado que se desea descargar.	126
6.33. Pantalla donde se muestra que se descargo correctamente el archivo.	126
6.34. Pantalla donde se muestra el archivo en la carpeta donde se descargo.	127
6.35. Pantalla donde se muestran los archivos en la carpeta donde se descargo del usuario 1.	127
6.36. Pantalla donde se muestran los archivos en la carpeta donde se descargo del usuario 2.	128
6.37. Pantalla donde se muestra el formulario donde se puede eliminar un archivo.	128
6.38. Pantalla donde se muestra el formulario donde se eligió un archivo a eliminar.	129
6.39. Pantalla donde se muestra la base de datos ya con el archivo eliminado.	129
6.40. Pantalla donde se muestra la lista de archivos actualizada despues de eliminar el archivo.	130
6.41. Pantalla donde se muestra la base de datos ya con el archivo eliminado en los dos usuarios.	130
6.42. Pantalla donde se muestra la lista de archivos actualizada despues de eliminar el archivo del usuario 2.	131
6.43. Pantalla donde se muestra la base de datos en la tabla de Hashes despues de haber eliminado un archivo por completo	131

Índice de Tablas

4.1. Componentes físicos	28
4.2. Componentes Lógicos	28
4.3. Escalas de impacto de un riesgo [10]	31
4.4. Análisis de riesgos del proyecto	32
4.5. Plan de acción ante riesgos en el proyecto	34
4.6. Requerimientos funcionales del servidor de llaves	49
4.7. Requerimientos funcionales del servidor de llaves	49
4.8. Requerimientos funcionales del cliente	50
4.9. Requerimientos funcionales del Servicio de almacenamiento (Nube)	50
4.10. Requerimientos no funcionales del sistema	52

Capítulo 1

Introducción

Hoy en día millones de personas en el mundo tienen la facilidad de acceder a un dispositivo electrónico que les permite manipular información o almacenarla ya sea en un dispositivo físico o en la nube. Debido a los limitados recursos financieros y altos gastos de almacenamiento de datos electrónicos, los usuarios prefieren almacenar sus datos en los entornos de nube provocando enormes demandas a las compañías que ofrecen este servicio. El incremento en el uso de estos servicios implica que los sistemas de almacenamiento necesiten más capacidad para poder cubrir la alta demanda que se presenta en el mercado [2].

Para entender un poco más acerca de la problemática a la que se enfrenta el almacenamiento en la nube, en la figura 1.1 se puede apreciar un estudio realizado por EFE/Cisco, en el año 2014. En su cuarto informe anual Índice Global sobre la nube (2013-2018), se puede observar cómo el almacenamiento se ha ido incrementado.



Figura 1.1: Crecimiento global del tráfico en la nube

Una de las principales razones del incremento en el tamaño en la estructura de almacenamiento de servicios en línea es la duplicación de archivos, es decir, existen muchas copias en la nube de un mismo archivo que se encuentra presente en diferentes cuentas de usuarios, para ello presentamos el siguiente ejemplo ilustrado en la Figura 1.2

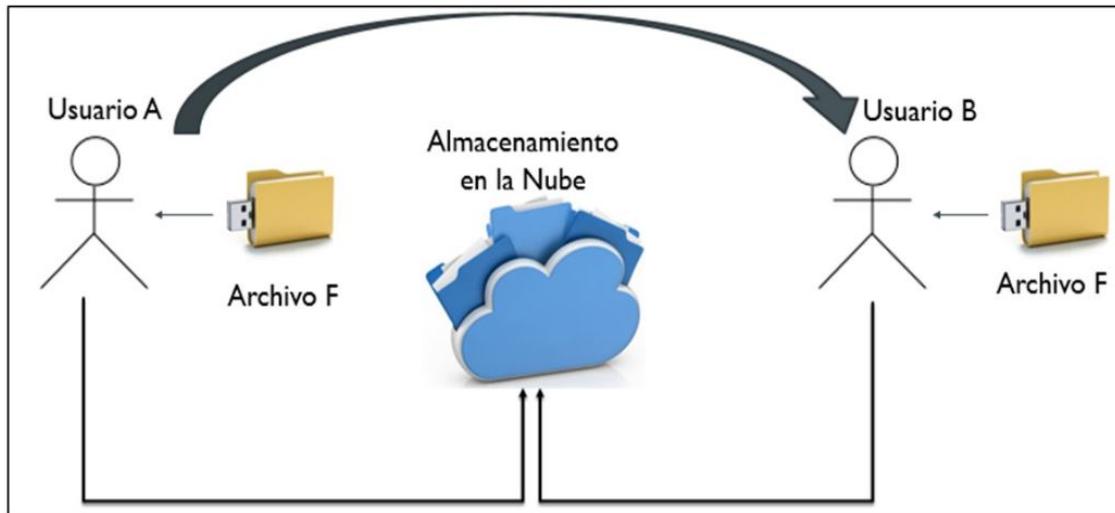


Figura 1.2: Duplicación de Archivos en la Nube

El usuario registrado e identificado dentro de la nube como *Usuario A*, el día de hoy desea almacenar un archivo **Archivo F** que corresponde a la especificación de los requisitos para la obtención de una beca escolar. Este usuario para no extraviar o modificar el archivo lo almacena en la nube y ahí queda disponible para cuando él lo solicite. Ahora este usuario comparte este archivo mediante un dispositivo *USB* a su amigo ya que este también desea conocer los requisitos para solicitar una beca escolar, este amigo el cuál es otro usuario registrado e identificado dentro de la nube como *Usuario B* también quiere almacenar este archivo **Archivo F** en la nube, ya que requiere utilizar el espacio de memoria en su dispositivo *USB* para otras actividades y no desea perder los requisitos para la solicitud de su beca escolar.

Ahora en la nube se encuentran almacenados 2 copias del **Archivo F** por dos diferentes usuarios identificados como *Usuario A* y *Usuario B*, ambos almacenaron el mismo archivo en el mismo lugar, sin darse cuenta que ahora este archivo se encuentra duplicado en la nube.

Sin embargo, en este escenario es fácil detectar los duplicados. Como se observa en la Figura 1.3 varios usuarios realizan peticiones para subir archivos, pero se puede notar que algunos son iguales, por lo que pasan por un proceso de eliminación de duplicados antes descrito y finalmente se almacena una copia de cada uno.

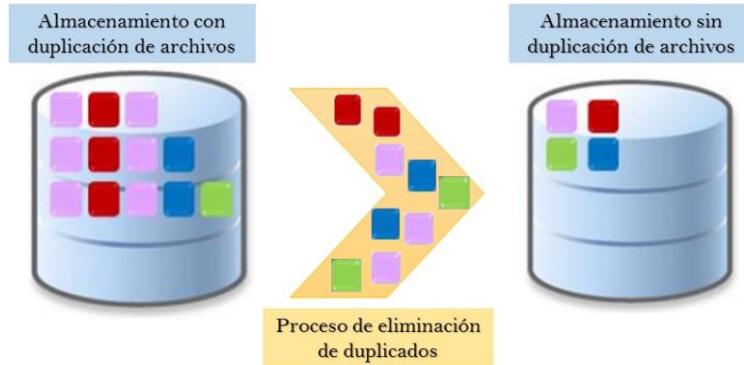


Figura 1.3: Eliminación de Duplicados

Otro punto importante, es la seguridad e integridad de la gran cantidad de información que se almacena en la nube. Cuando un usuario almacena un archivo en la nube, está cediendo su información de manera voluntaria a un tercero, que podría hacer mal uso de la misma. Éste tema se está volviendo cada vez más preocupante sobre todo para instituciones o empresas cuya información puede ser sensible. Para proteger a los archivos que se almacenan en la nube, es posible utilizar mecanismos criptográficos. La criptografía, es una ciencia que se encarga de ocultar información de tal forma que solamente aquellos usuarios autorizados puedan acceder a ella.

Encontrar una solución para ambos problemas, es decir, evitar la duplicación y tener confidencialidad de la información en la nube, no es trivial. Por un lado, si se tiene la información en claro es relativamente fácil poder detectar que dos archivos son los mismos, ya que solo se compara bit a bit, pero si estos mismos archivos cada usuario cifra su archivo el resultado serían dos archivos diferentes por esto es que no es posible detectar archivos duplicados con este método.

1.1. Propuesta de solución.

Como se mencionó anteriormente se tienen dos problemas principales, uno es la confidencialidad de los archivos cuando se almacenan en la nube y el otro es la cantidad de archivos duplicados en la misma. Si se utilizara un algoritmo de cifrado estándar para combatir el problema de la confidencialidad, entonces a cada usuario se le proporcionará una llave con la que puede cifrar y descifrar archivos, pero si dos usuarios cifran el mismo archivo daría como resultado dos archivos cifrados distintos, entonces al hacer la comparación de ambos archivos no habría forma de atacar el problema de la duplicación de archivos. En conclusión se garantiza la privacidad de los archivos, pero no es posible detectar duplicados.

Una alternativa para solucionar ambos problemas es usar un protocolo de *cifrado convergente*, donde la llave de cifrado se deriva del texto en claro, es decir, si dos usuarios poseen el mismo archivo, podrán obtener la misma llave de cifrado. Posteriormente se utiliza un

cifrador por bloques para ocultar la información. Aunque este mecanismo es mejor que el descrito previamente, solamente es seguro si el espacio de mensajes es muy grande, es decir, si un mensaje es *impredecible*. Por el contrario, si el espacio de mensajes es pequeño, entonces un posible atacante puede probar con todos los posibles mensajes, determinar si ya está almacenado en la nube y descubrir la información para un usuario específico.

Bellare, Keelveedhi y Ristenpart, propusieron un protocolo denominado DupLESS [1], el cual ofrece una solución para el escenario descrito anteriormente, es decir, cuando el espacio de mensajes es pequeño. Este protocolo combina varias herramientas criptográficas de tal forma que se pueda tener un control de las llaves relacionadas con los archivos y los usuarios sin comprometer el contenido e integridad de la información, de esta manera si dos usuarios quieren cifrar el mismo archivo obtendrán la misma llave. Por consiguiente, al cifrar los archivos obtendríamos como resultado el mismo archivo cifrado, esta vez al hacer la comparación se detectará la duplicación de archivos y únicamente se guardará una copia del archivo, ahorrando espacio de memoria y costos de infraestructura.

En el presente trabajo terminal, se implementará el protocolo DupLESS, para garantizar la confidencialidad de la información y detectar duplicados en la nube, considerando un espacio de mensajes pequeño.

1.2. Objetivos.

A continuación se describirán los objetivos que se pretende alcanzar, durante el desarrollo del presente trabajo terminal.

1.2.1. Objetivo General.

Desarrollar un protocolo criptográfico para evitar la duplicación de archivos almacenados en la nube, garantizando la privacidad de los usuarios contra adversarios cuando el espacio de mensajes es pequeño, utilizando algoritmos criptográficos para su implementación.

1.2.2. Objetivos Específicos.

- Evitar la duplicación de archivos que sean almacenados por los usuarios de la nube
- Proteger ante los adversarios la información de los usuarios de la nube
- Establecer un esquema de autenticación de usuarios
- Reducir la pérdida y filtración de información de los usuarios de la nube
- Evitar los ataques por fuerza bruta al contenido de los archivos de usuarios en la nube.

1.3. Organización del documento.

El resto del presente documento detalla el proceso de implementación del protocolo criptográfico DupLESS y está conformado como se describe a continuación

- **Capítulo 2 Marco Teórico**

El contenido de este capítulo abordará temas estrechamente relacionados con la criptografía y la seguridad de la información. También, este capítulo contiene información acerca de los 2 tipos de criptografía que existen mencionando los diferentes esquemas de cifrado y los modos de operación que son utilizados por algunos de estos. De igual forma se describe con detalle los servicios que ofrece el cómputo nube.

- **Capítulo 3 Protocolos criptográficos para evitar duplicados**

Este capítulo muestra una comparativa entre los distintos protocolos criptográficos que garantizan confidencialidad y la detección de duplicados. También el capítulo contiene la explicación del protocolo criptográfico DupLESS, el cuál provee de toda la investigación y posteriormente el desarrollo de este trabajo terminal.

- **Capítulo 4 Análisis**

Aquí se describe el análisis del protocolo criptográfico, el cual se compone de la realización de un estudio de factibilidad y análisis de riesgos para la realización de este proyecto, asimismo se muestra cuál será la arquitectura del trabajo terminal, la descripción de todos los procesos que involucran la creación de este protocolo, el modelo de entidades que comprende diagrama entidad relación y el diagrama de clases para su implementación. Al final de este capítulo se podrán encontrar los requerimientos de este protocolo criptográfico tanto funcionales como no funcionales y las reglas de negocio que lo componen.

- **Capítulo 5 Diseño**

El capítulo de diseño tiene como contenido la especificación de la plataforma del protocolo, es decir, la especificación de recursos necesarios para poder llevar a cabo su implementación con las herramientas necesarias, y también todos los casos de usos que involucran el funcionamiento de dicho protocolo.

Capítulo 2

Marco Teórico

El contenido de este capítulo abordará temas estrechamente relacionados con la criptografía. Se abordará información acerca de los 2 tipos de criptografía que existen, criptografía simétrica y asimétrica. En el caso de la criptografía simétrica se describirán dos primitivas criptográficas, los cifradores por bloque y las funciones hash. Respecto a la criptografía asimétrica o de clave pública, se hablará sobre el algoritmo de cifrado RSA, firmas digitales y firmas a ciegas, todas éstas primitivas juegan un papel fundamental para el funcionamiento de este proyecto. De igual forma se describe con detalle los servicios que ofrece el cómputo nube, haciendo énfasis en un servicio en particular que es el almacenamiento que se utilizará para la implementación de este protocolo criptográfico.

2.1. Criptografía.

La *criptografía* tiene como objetivo fundamental habilitar a dos personas usualmente referidas como Alice y Bob para que se comuniquen sobre un canal inseguro de tal manera que un adversario, Oscar, no pueda entender que es lo que están diciendo, éste canal podría ser una línea telefónica o una red de computadora, por ejemplo. La información que quiere enviar Alice a Bob que nosotros llamaremos “texto en claro”, puede ser un texto en inglés, datos numéricos, o cualquier cosa—esta estructura es completamente arbitraria. Alice cifra el texto en claro usando una llave predeterminada y envía el texto cifrado resultante sobre el canal. Oscar, al ver el texto cifrado en el canal que está escuchando a escondidas, no puede determinar qué es el texto en claro; pero Bob, quien sabe la llave con la que se cifró el archivo, puede descifrar el texto y obtener el texto claro [12].

Esta ciencia que mantiene la información segura se encuentra dividida en dos tipos: *Criptografía Simétrica* y *Criptografía Asimétrica*. La *criptografía simétrica* o también llamada criptografía de llave secreta, basa su seguridad en una sola llave que se comparte entre dos entidades que quieren compartir información, dicha llave es utilizada para cifrar un archivo al ser enviado a la otra entidad y este utilizará la misma llave para descifrarlo cuando lo reciba. El esquema de cifrado simétrico se puede representar a través de la siguiente figura 2.1.

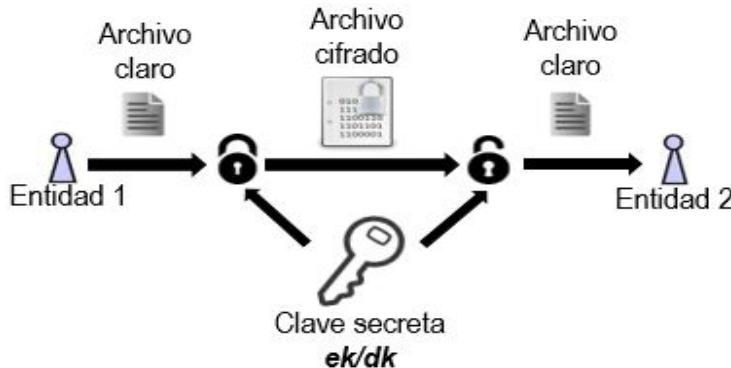


Figura 2.1: Diagrama de Criptografía Simétrica.

Los algoritmos criptográficos simétricos tienen dos versiones: cifrador en bloque y cifrador de flujo. El beneficio en cuanto al uso de un algoritmo simétrico se encuentra en el procesamiento rápido para cifrar y descifrar un alto volúmen de datos. El cifrado simétrico es una práctica eficaz de almacenamiento de información sensible en una base de datos, un registro o archivo [8].

La *criptografía asimétrica* o criptografía de llave pública involucra el uso de un par de llaves para cada entidad que desea comunicarse, estas llaves llamadas pública y privada. Para que una entidad envíe un archivo a otra, necesita cifrar el archivo con la llave pública de esa entidad a la que se desea enviar, y cuando lo reciba esa entidad lo deberá descifrar con su llave privada o secreta. De esta manera se evita el compartir llaves para cifrar y descifrar como sucede en la criptografía simétrica y reduce los riesgos de un ataque de adversarios. El cifrado asimétrico puede ser representado como aparece en la figura 2.2.

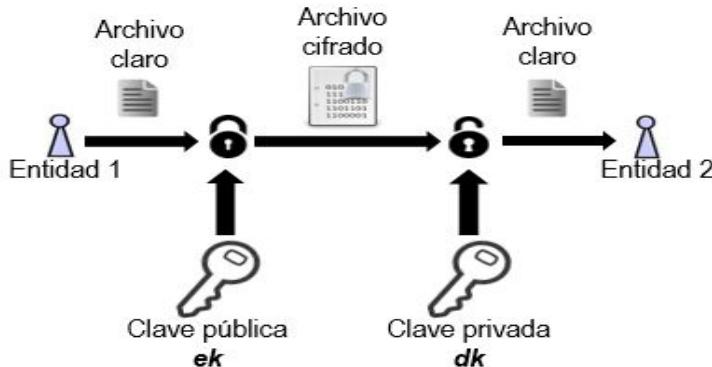


Figura 2.2: Diagrama de Criptografía Asimétrica.

Los beneficios de la criptografía asimétrica son que las claves públicas pueden ser distribuidas con toda tranquilidad, pues no puedes hacer uso de las llaves públicas sin las llaves privadas. Asimismo, el cifrado asimétrico se emplea frecuentemente para elaborar firmas digitales, un mecanismo que permite al receptor de un mensaje firmado digitalmente poder

identificar a la entidad que origino ese mensaje y de esa manera confirmar que el mensaje no ha sido alterado.

Los servicios criptográficos son aquellos que garantizan en un sistema de información la adquisición, almacenamiento, procesamiento y transmisión de la información y para lograrlo se valen de uno o más objetivos fundamentales.

- **Confidencialidad.** Es un servicio utilizado para garantizar que solo las personas autorizadas tienen acceso a la información.
- **Autenticación.** Esta función se aplica tanto a las entidades como a la propia información. Dos entidades que participan en una comunicación deben identificarse entre sí. La información entregada a través de un canal debe ser autenticada en cuanto al origen, fecha de origen, contenido de los datos, tiempo enviado, etc.
- **Integridad.** Es un servicio que se ocupa de la alteración no autorizada de los datos. Para asegurar la integridad de los datos, se debe tener la capacidad de detectar la manipulación de datos por parte de algún adversario. La manipulación de datos incluye inserción, supresión y sustitución, entre otros.
- **No repudio.** Es un servicio que impide a una entidad negar compromisos o acciones anteriores. Cuando surgen disputas debido a que una entidad niega que se tomaron ciertas acciones, es necesario un medio para resolver la situación. Por ejemplo, una entidad puede autorizar la compra de una propiedad por otra entidad y posteriormente negar que se concedió dicha autorización [4].

El *Criptoanálisis* es la ciencia que se ocupa del análisis de un texto cifrado para obtener la información original sin conocimiento de la clave secreta, esto es, de forma ilícita rompiendo así los procedimientos de cifrado establecidos por la Criptografía, por lo que se dice que Criptoanálisis y Criptografía son ciencias complementarias pero contrarias [14].

Los *ataques a servicios criptográficos* son una violación a la seguridad de la información realizada por intrusos que tienen acceso físico al sistema sin ningún tipo de restricción, su objetivo es robar la información o hacer que ésta pierda valor relativo, o que disminuyan las posibilidades de su supervivencia a largo plazo.

- Ataque sólo con texto cifrado. Este caso es cuando el criptoanalista sólo conoce el criptograma y el algoritmo con que fue generado; con esta información pretende obtener el texto en claro.
- Ataque con texto en claro conocido. En esta situación el criptoanalista conoce mensajes en claro seleccionados por él mismo y sus correspondientes criptogramas, así como el algoritmo con que éstos fueron generados; aquí el objetivo es conocer la clave secreta y poder descifrar libremente cualquier texto.

- Ataque con texto cifrado escogido. El criptoanalista conoce el algoritmo de cifrado, así como un criptograma seleccionado por él mismo y su correspondiente texto en claro, su objetivo es obtener el mensaje en claro de todo criptograma que intercepte.
- Ataque con texto en claro escogido. En este caso el criptoanalista además de conocer el algoritmo de cifrado y el criptograma que quiere descifrar, también conoce el criptograma de un texto en claro [15].

2.2. Cifrador por bloques.

Un cifrador de bloques es una función que convierte bloques de texto de n bits a bloques de texto cifrado de n bits; a n se denomina longitud de bloque. La función que convierte los bloques de texto simple está parametrizada por una clave K de k -bits, tomando valores de un subconjunto \mathcal{K} (el espacio de la llave) del conjunto de todas las palabras numéricas de $\{0, 1\}^k$. Generalmente la clave K se elige al azar. El uso de bloques de texto claro y texto cifrado de igual tamaño evita la expansión de datos [4].

Para los bloques de texto de n -bit, texto cifrado de n -bit y una clave fija de n -bit, la función de cifrado es una biyección, es decir que el texto cifrado debe ser siempre diferente pero cuando se descifre este debe corresponder al texto en claro, definiendo una permutación de palabras numéricas de n -bits. Cada clave potencial define una biyección diferente. El número de llaves es de longitud $|\mathcal{K}|$, y el tamaño efectivo de la clave es de longitud $|\mathcal{K}|$. Esto es igual a la longitud de la clave si todos las palabras numéricas de k -bits son claves válidas ($\mathcal{K} = V_k$). Si las llaves son equiprobables (misma probabilidad) y cada una define una biyección diferente, la entropía (medida de incertidumbre) del espacio clave es también de longitud $|\mathcal{K}|$ [4].

Los cifradores por bloques más usados son AES (*Advanced Encryption Standard*, por sus siglas en inglés) y DES (*Data Encryption Standard*, por sus siglas en inglés) [9].

Los cifradores por bloques pueden ser representados como se ve en la figura 2.3.

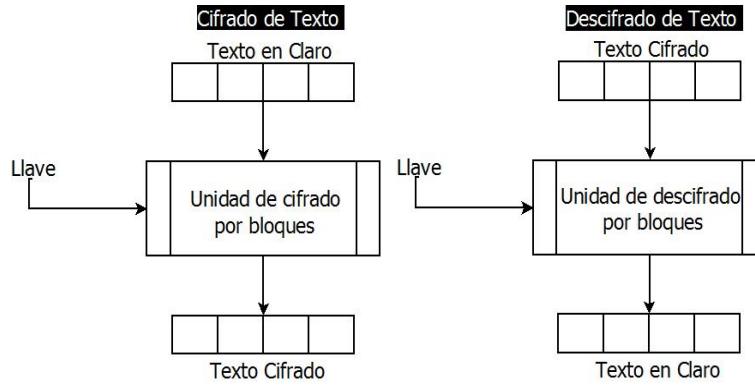


Figura 2.3: Diagrama de Cifradores por Bloques

2.3. 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 solo 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 este tipo y probar que lo son.

Definición 2.1 Una función hash, es una función de un solo 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 propiedad de las funciones hash es que sea imposible que se produzca una colisión ya sea débil o fuerte, además de resistir ataques de pre imagen.

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.

Dentro de SHA-2 encontramos varios tipos, el SHA-224, SHA-256, SHA-384 y SHA-512. El más seguro, es el que da mayor salida de bits que es el SHA-512, con una salida de 512 bits, que tiene 80 rondas (pasos). En la SHA-256, tenemos una longitud de palabra de 323 bits, con 64 rondas, tamaño del bloque de 512 bits, y con una salida de 256 bits. Como ocurre con todos los cifrados y hash, cuanto más seguro, más lento su procesamiento y uso, debemos encontrar un equilibrio entre seguridad y velocidad.

2.4. RSA

El esquema de criptografía RSA se llama así por los nombres de los autores Ron Rivest, Adi Shamir y Leonard Adleman, es actualmente el esquema criptográfico asimétrico más

utilizado. RSA fue patentado en los Estados Unidos (pero no en el resto del mundo) hasta el 2000. Utiliza una clave pública, la cual se distribuye (en forma autenticada preferentemente), y otra privada, la cual es guardada en secreto por su propietario. Cuando se envía un mensaje, el emisor busca la clave pública de cifrado del receptor y una vez que dicho mensaje llega al receptor, éste se ocupa de descifrarlo usando su clave oculta. Los mensajes enviados usando el algoritmo RSA se representan mediante números y el funcionamiento se basa en el producto de dos números primos grandes (mayores que 10100) elegidos al azar para conformar la clave de descifrado. Emplea expresiones exponenciales en aritmética modular. La seguridad de este algoritmo radica en que no hay maneras rápidas conocidas de factorizar un número grande en sus factores primos utilizando computadoras tradicionales. La computación cuántica podría proveer una solución a este problema de factorización. Este popular sistema se basa en el problema matemático de la factorización de números grandes. El algoritmo RSA funciona de la siguiente manera:

Generación de llaves. Estos son los pasos involucrados en el cálculo de la clave pública y privada para un criptosistema RSA.

- Inicialmente es necesario generar aleatoriamente dos números primos grandes, a los que llamaremos p y q .
- A continuación calcularemos n como producto de p y q : $n = p * q$
- Se calcula ϕ : $\phi(n) = (p - 1)(q - 1)$
- Se elige al azar un entero e , tal que $mcd(e, \phi(n)) = 1$, es decir e es primo relativo con $\phi(n)$.
- Se calcula d tal que $ed \bmod \phi(n) = 1$.
- El par de números (e, n) son la clave pública.
- El par de números (d, n) son la clave privada.

Cifrado y descifrado. Después de que un usuario tenga su par de llaves, será posible realizar las operaciones para cifrar y descifrar como se muestra a continuación

- Cifrado: la función de cifrado es $C = M^e \bmod n$.
- Descifrado: la función de descifrado es $M = C^d \bmod n$.

Ejemplo. El primer paso es calcular las llaves pública y privada. Si se consideran los primos $p = 11$ y $q = 23$ entonces $n = 11 * 23 = 253$ y $\phi = (11 - 1) * (23 - 1) = 220$. Si se escoge $e = 3$, es fácil verificar que $mcd(220, 3) = 1$. En tanto que $d = 147$ que $3 * 147 \bmod 220 = 1$. Por lo tanto, la clave pública será $(e, n) = (3, 253)$ y la clave privada $(d, n) = (147, 253)$.

Cifrado:

Cifraremos la palabra seguridad y para esto le asignaremos números a todas las letras del abecedario dando como resultado para nuestra palabra los siguientes valores:

S	E	G	U	R	I	D	A	D
18	4	6	20	17	8	3	0	3

Tenemos el mensaje $M = 18 \ 4 \ 6 \ 20 \ 17 \ 8 \ 3 \ 0 \ 3$ a cifrar

$$18^3 = 5832 \text{ mód } 253 = 13$$

$$4^3 = 64 \text{ mód } 253 = 64$$

$$6^3 = 216 \text{ mód } 253 = 216$$

$$20^3 = 8000 \text{ mód } 253 = 157$$

$$17^3 = 4913 \text{ mód } 253 = 106$$

$$8^3 = 512 \text{ mód } 253 = 6$$

$$3^3 = 27 \text{ mód } 253 = 27$$

$$0^3 = 0 \text{ mód } 253 = 0$$

$$3^3 = 27 \text{ mód } 253 = 27$$

Obtenemos el mensaje cifrado $C = 13 \ 64 \ 216 \ 157 \ 106 \ 6 \ 27 \ 0 \ 27$

Descifrado:

Tenemos el texto cifrado, $C = 13 \ 64 \ 216 \ 157 \ 106 \ 6 \ 27 \ 0 \ 27$

$$13^{147} \text{ mód } 253 = 18$$

$$64^{147} \text{ mód } 253 = 4$$

$$216^{147} \text{ mód } 253 = 6$$

$$157^{147} \text{ mód } 253 = 20$$

$$106^{147} \bmod 253 = 17$$

$$6^{147} \bmod 253 = 8$$

$$27^{147} \bmod 253 = 3$$

$$0^{147} \bmod 253 = 0$$

$$27^{147} \bmod 253 = 3$$

Obtenemos el mensaje descifrado, $M = 18 \ 4 \ 6 \ 20 \ 17 \ 8 \ 3 \ 0 \ 3$

2.5. Firma digital

Una *firma digital* es un mecanismo de autenticación que permite al creador de un mensaje fijar un código que actúa como una firma. La firma es formada tomando el hash del mensaje y cifrar el mensaje con clave privada del creador. La firma garantiza el origen y la integridad del mensaje [11].

El estándar de la firma digital (*Digital Signature Standard*) es un estándar *NIST (National Institute Standards of Technology)* que utiliza el algoritmo de hash seguro (SHA). El desarrollo más importante del trabajo sobre criptografía de clave pública es la firma digital. La firma digital proporciona un conjunto de capacidades de seguridad que sería difícil de aplicar en cualquier otra forma [11].

La autenticación de mensajes protege dos partes que intercambian mensajes de terceros. Sin embargo, no protege a los dos partidos uno contra el otro. Varias formas de disputa entre los dos son posibles [11].

Los algoritmos de firma son los siguientes *DSA*, *ECDSA* y *firmas basadas con RSA*. En situaciones donde no existe una completa confianza entre el emisor y el receptor, se necesita algo más que la autenticación. La solución más atractiva para este problema es la firma digital. La firma digital es análoga a la firma manuscrita. Debe tener las siguientes propiedades:

- Debe verificar el autor, la fecha y hora de la firma.
- Debe autenticar el contenido en el momento de la firma.
- Debe ser verificable por terceras personas, para resolver los conflictos.

Así, la función de firma digital incluye la función de autenticación [11].

Bob quiere enviar un mensaje a Alice y ella quiere estar segura de que realmente el mensaje provenga de Bob, por lo tanto, Bob debe firmar el mensaje y para lograr este objetivo se realizan 3 procesos, el primero es el de generación de llaves donde Bob obtiene su par de

llaves pública (pk) y privada (sk), en el segundo proceso Bob firma el mensaje con la siguiente función: $s = sig_{sk}(x)$ donde s es el mensaje firmado, x es el mensaje y sig es la función para firmar, Bob envía pk y s a Alice, el tercer proceso consiste en la verificación de la firma, Alice recibe los archivos de Bob y realiza las siguientes operaciones: $ver_{pk}(x, s) = true/false$ donde ver es la función para verificar [5].

2.6. Firma digital con RSA.

El esquema de firma con RSA se basa en el cifrado RSA. Su seguridad se basa en la dificultad de factorizar un producto de dos grandes primos. Desde su primera descripción en 1978, el esquema de firma con RSA ha surgido como el esquema de firmas digitales más utilizado en la práctica [5]. El protocolo para firmar y verificar es el siguiente:

Bob

Calcular las claves con RSA

$sk = d$ clave privada.

$pk = (n, e)$ clave pública.

Firmar

$$s = sig_{sk}(x) \equiv x^d \pmod{n}$$

Alice

Verificar

$$ver_{pk}(x, s)$$

$$x' \equiv s^e \pmod{n}$$

Si $x' \equiv x \pmod{n}$ la firma es válida

Si $x' \not\equiv x \pmod{n}$ la firma es invalida

Como puede verse en el protocolo, Bob calcula la firma s para un mensaje x que ha sido cifrado con RSA usando su sk (clave privada). Bob es la única persona que puede tener la sk , y por lo tanto la propiedad de sk lo autentica como el autor del mensaje firmado. Bob agrega la firma s al mensaje x y envía ambos a Alice. Alice recibe el mensaje firmado y descifra con RSA usando la pk (clave pública) de Bob, produciendo x_2 . Si x y x_2 coinciden, Alice sabe dos cosas importantes: Primero, el autor del mensaje estaba en posesión de la sk de Bob, y si Bob hubiera tenido acceso a la sk , fue Bob quien firmó el mensaje. Esto se llama autenticación de mensajes. En segundo lugar, el mensaje no se ha cambiado durante

el camino, por lo que se da la integridad del mensaje [5], por ejemplo.

Bob

Calcular las claves con RSA

Elije $p=3$ y $q=11$ y calcula sus llaves sk y pk que dan como resultado:

$$sk = 7 \text{ y } pk = 3 \text{ con } n = 33$$

Firmar

Bob quiere firmar el archivo x que tiene un valor de 4

$$s = x^7 \equiv 4^7 \equiv 16 \pmod{33}$$

Se envian los siguientes archivos a Alice: $x = 4$ y $s = 16$

Alice

Verificar

$$x' = s^e \equiv 16^3 \equiv 4 \pmod{33}$$

Como $x' \equiv x \pmod{33}$ la firma es válida.

2.7. Firmas a ciegas.

Las *firmas a ciegas* son un tipo especial de firmas digitales en las que se firma algo que no se conoce. Para hacer firmas a ciegas se utilizan factores de opacidad, para ocultar el mensaje original que se requiere que esté firmado, y así la autoridad no pueda conocer lo que está firmando. Por lo tanto, el propósito de una firma a ciegas es evitar que el firmante B conozca el mensaje que firma; y así posteriormente, sea incapaz de asociar el mensaje que firmó con el remitente A . Entonces, las firmas a ciegas tienen aplicación en varias situaciones, por ejemplo, en las elecciones electrónicas también pueden utilizarse las firmas a ciegas, ya que se requiere que B (una autoridad electoral) no conozca la identidad de A (el votante) debido a que el voto debe efectuarse de manera anónima. Sin embargo, es necesario que A demuestre que su voto m es válido. Lo cual se logra cuando A presenta ante B la firma $s(m)$. Y se sabe de antemano que B no puede asociar $s(m)$ a A , debido a que el votante previamente le envió a B su voto m pero de forma oculta para que se lo firmara. [6]

2.8. Firmas a ciegas con RSA.

Un esquema de firma a ciegas es un protocolo que involucra un remitente A y un firmante B . La idea básica en un esquema basado en RSA es la siguiente: A le envía cierta información z a B , donde z está compuesto por el mensaje que se desea que firme B y por un factor de ocultamiento cifrado con la llave pública de B , es decir, $z = (m * b^e) \bmod n$. B firma dicha información $s(z)$ y se la regresa a A . De la firma $s(z)$, A puede obtener la firma de B para el mensaje m , quitando el factor de ocultamiento b a $s(z)$. Pues:

$$s(z) = (m * b^e) \bmod n = (m^d * b^{ed}) \bmod n = (m^d \bmod n) * b$$

Ahora bien, al dividir $s(z)$ entre b , obtendremos $s(m)$:

$$(m) = s(z)/b = ((m^d \bmod n) * b)/b = m^d \bmod n$$

Al finalizar el protocolo, B no conoce el mensaje m ni la firma asociada a él $s(m)$ que ahora posee A [6].

A continuación, se presenta un ejemplo.

- Alice quiere que Bob le firme un mensaje con un valor de $m = 65$
- Para ello conoce la llave pública de Bob con los valores de $e = 7$ y $n = 299$
- Alice calcula un factor de ocultamiento que cumpla que su mcd sea igual con 1
- Para ello emplea el algoritmo de Euclides del cuál obtenemos lo siguiente:
 - $299 = 60 * 4 + 59$
 - $60 = 59 * 1 + 1$
 - $59 = 1 * 59 + 0$

Recordemos que cuando el número a sumar sea 0 el resto anterior será el máximo común divisor de esos dos números, por lo tanto, el factor de ocultamiento tiene un valor de 60

- Alice emplea el algoritmo extendido de Euclides puede encontrar el inverso del factor de ocultamiento de la siguiente forma:
 - $59 = 299 - 60(4)$

- $1 = 60 - 59(1)$
- $1 = 60 - [299 - 60(4)]$
- $1 = 60(5) + 299(-1) \rightarrow 1 = ax + by$
- $a^{-1} \bmod b = x \rightarrow 60^{-1} \bmod 299 = 5$

El inverso del factor de ocultamiento tiene un valor de 5.

- Una vez obtenido este resultado puede ocultar este mensaje y posteriormente se lo va a enviar a Bob
 - $z = (65 * 60^7) \bmod 7$
 - $z = (65 * 226) \bmod 7$
 - $z = 39$
- Bob recibe el mensaje oculto, y lo cifrará con su llave privada que tiene un valor de 151
 - $s(z) = 39^{151} \bmod 299$
 - $s(z) = 104$
- Se lo envía a Alice, se podría decir que en este paso ha firmado ciegamente.
- Alice quita el factor de ocultamiento y ahora obtiene su documento firmado.
 - $S = (104 * 5) \bmod 299$
 - $S = 221$

Este valor obtenido es el mismo que si Bob firmará un mensaje con su llave privada.

- $65^{151} \bmod 299 = 221$

2.9. Cómputo Nube.

El cómputo nube definido así por el NIST, es un modelo para permitir un acceso a la red ubicuo, es decir, que se encuentra presente en todas partes al mismo tiempo y conveniente a un conjunto de recursos informáticos configurables (por ejemplo, redes, servidores, almacenamiento, aplicaciones y servicios) que se puede aprovisionar y liberar rápidamente con un esfuerzo mínimo de gestión o una interacción entre el proveedor de servicios. Este modelo de cómputo nube se compone de 5 características esenciales, 3 modelos de servicio y 4 modelos de despliegue.

Características:

- **Auto-servicio bajo demanda.**

Un consumidor puede proporcionar unilateralmente capacidades del tiempo del servidor y el almacenamiento en red, según se necesite automáticamente sin interacción con cada proveedor de servicios.

- **Amplio acceso a la red.**

Las capacidades están disponibles a través de la red y se accede a través de mecanismos que promueven el uso por plataformas de cliente heterogéneas finas o gruesas (por ejemplo, teléfonos móviles, tablets, computadoras portátiles y estaciones de trabajo)

- **Agrupación de recursos.**

Los recursos informáticos del proveedor se agrupan para servir a múltiples consumidores utilizando un modelo de múlti-usuario, con diferentes recursos físicos y virtuales asignados dinámicamente y reasignados de acuerdo con la demanda del consumidor. Hay una sensación de independencia de ubicación en que el cliente generalmente no tiene control o conocimiento sobre la ubicación exacta de los recursos proporcionados, pero puede especificar la ubicación en un nivel superior de abstracción (por ejemplo, país, estado o centro de datos). Ejemplos de recursos incluyen almacenamiento, procesamiento, memoria y ancho de banda de la red.

- **Elasticidad rápida.**

Las capacidades pueden ser suministradas elásticamente y liberadas, en algunos casos de forma automática, para escalar rápidamente hacia fuera y hacia adentro proporcional a la demanda. Para el consumidor, las capacidades disponibles para la provisión a menudo parecen ser ilimitadas y pueden ser apropiadas en cualquier cantidad en cualquier momento.

- **Servicio medido.**

Los sistemas de cómputo nube controlan y optimizan automáticamente el uso de recursos aprovechando una capacidad de medición en algún nivel de abstracción apropiado al tipo de servicio (por ejemplo, almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas). El uso de recursos puede ser monitoreado, controlado y reportado, proporcionando transparencia tanto para el proveedor como para el consumidor del servicio utilizado.

Modelos de servicio. Entre los diversos tipos de servicios de cómputo en la nube proporcionados internamente o por proveedores de servicios de terceros, los más habituales son:

- **Software como Servicio (SaaS).**

La capacidad proporcionada al consumidor es utilizar las aplicaciones del proveedor que se ejecutan en una infraestructura en la nube. Las aplicaciones son accesibles desde varios dispositivos cliente a través de una interfaz de cliente ligero, como un navegador web (por ejemplo, correo electrónico basado en web) o una interfaz de programa. El consumidor no gestiona ni controla la infraestructura oculta de la nube, incluyendo la

red, los servidores, los sistemas operativos, el almacenamiento o incluso las capacidades de las aplicaciones individuales, con la posible excepción de las limitadas configuraciones específicas de la configuración de la aplicación.

- **Plataforma como Servicio (PaaS).**
- **Infraestructura como Servicio (IaaS).**

Modelos de despliegue.

- **Nube privada.**

La infraestructura de la nube está preparada para el uso exclusivo de una sola organización que comprende varios consumidores (por ejemplo, unidades de negocio). Puede ser propiedad, administrado y operado por el órgano.

- **Nube de la comunidad.**

La infraestructura de la nube está preparada para uso exclusivo por una comunidad específica de consumidores de organizaciones que tienen preocupaciones compartidas (por ejemplo, misión, requisitos de seguridad, política y consideraciones de cumplimiento). Puede ser propiedad, administrado y operado por una o más de las organizaciones de la comunidad, un tercero, o una combinación de ellos, y puede existir dentro o fuera de las instalaciones.

- **Nube pública.**

La infraestructura de la nube está preparada para el uso abierto por el público en general. Puede ser propiedad, administrado y operado por una organización comercial, académica u gubernamental, o alguna combinación de ellos. Existe en las instalaciones del proveedor de la nube.

- **Nube híbrida.**

La infraestructura de la nube es una composición de dos o más infraestructuras de nube distintas (privadas, comunitarias o públicas) que siguen siendo entidades únicas, pero están unidas por una tecnología estandarizada o propietaria que permite la portabilidad de datos y aplicaciones (por ejemplo, burbujas de nube para equilibrar la carga entre Nubes).

Problemas en Cómputo Nube.

- **Interfaces y API poco seguros.**

Generalmente los proveedores de servicios en la nube ofrecen una serie de interfaces y API (del inglés, Application Programming Interface) para controlar e interactuar con los recursos. De este modo, toda la organización, el control, la provisión y la monitorización de los servicios cloud se realiza a través de estos API o interfaces. Dado que todo (autenticación, acceso, cifrado de datos, etc.) se realiza a través de estas herramientas, se hace necesario que los interfaces estén diseñados de forma segura, evitando así los problemas de seguridad, tanto los que son intencionados como los que se producen de forma accidental.

- **Pérdida o fuga de información.**

Existen muchas formas en las que los datos se pueden ver comprometidos. Por ejemplo, el borrado o modificación de datos sin tener una copia de seguridad de los originales, supone una pérdida de datos. En la nube, aumenta el riesgo de que los datos se vean comprometidos ya que el número de interacciones entre ellos se multiplica debido a la propia arquitectura de la misma. Esto deriva en pérdida de imagen de la compañía, daños económicos y, si se trata de fugas, problemas legales, infracciones de normas, etc.

- **Secuestro de sesión o servicio.**

En un entorno en la nube, si un atacante obtiene las credenciales de un usuario del entorno puede acceder a actividades y transacciones, manipular datos, devolver información falsificada o redirigir a los clientes a sitios maliciosos.

Capítulo 3

Protocolos criptográficos para evitar duplicados

En este capítulo se pretende establecer mediante un análisis, la comparativa realizada entre sistemas o prototipos existentes que hacen uso de herramientas criptográficas para proveer un servicio similar al protocolo criptográfico que se propone en este trabajo terminal. Dicho análisis contendrá detalladamente aquellos elementos técnicos que tienen relación con la criptografía y de qué manera se están utilizando. De igual forma, el capítulo contiene la explicación y el funcionamiento del protocolo criptográfico Dupless, dicho protocolo es la base de la investigación, de la cual surge el desarrollo del protocolo criptográfico que se presenta para este trabajo terminal.

3.1. Estado del Arte.

Una solución para tener privacidad y evitar duplicación es el *cifrado convergente* que la proporcionó John R. Douceur, la cual dice que teniendo a M que será el contenido de un archivo de aquí en adelante denominado el mensaje, el cliente primero calcula una clave $K, H(M)$ mediante la aplicación de una función de hash criptográfica H al mensaje y luego calcula el texto cifrado $C, E(K, M)$ a través de un esquema de cifrado simétrico determinista. El derivado del mensaje K se almacena por separado cifrándolo con una llave por cliente. Un segundo cliente B cifra el mismo archivo M que producirá el mismo C , evitando la duplicación [7]. Existen sistemas que aplican esta herramienta de cifrado para proteger la información, a continuación, se presenta una tabla comparativa de algunos de estos.

TahoeFS	ABS: The Apportioned Backup System
Firmas Digitales	Firmas Digitales
SHA256	SHA256
AES-128	Criptografía asimétrica
Cifrado convergente	Cifrado convergente
RSA de 2048 bits (256 bytes)	rsync

- TahoeFS

Tahoe es un sistema para el almacenamiento seguro distribuido. Usa las funciones de control de acceso, criptografía, confidencialidad, integridad y eliminación para tolerancia a fallos. Se ha desplegado en un servicio de copia de seguridad comercial y es actualmente operacional. La aplicación es de código abierto. Se basa en la restricción de permisos a la información y clasifica a los archivos en mutables e inmutables. Un archivo inmutable se crea exactamente una vez y se puede leer varias veces. Para crear un archivo inmutable, un cliente elige una clave de cifrado simétrico, utiliza esa clave para cifrar el archivo. Usa la técnica de cifrado convergente. Un archivo mutable permite las operaciones de leer y escribir, para esto utiliza RSA de 2048 bits para la generación de llaves, cada archivo mutable se asocia con un único par de llaves. Escritores autorizados tienen acceso a la llave privada, por lo que hacen las firmas digitales en las nuevas versiones del archivo que escriben, esta firma digital nos ayuda a saber quién hizo la última modificación al archivo. Para ambos tipos de archivos se utiliza un SHA 256 para generar un Hash del archivo en cuestión, y se utiliza un cifrador por bloques AES 128 bits [13].

- ABS

Proporciona un recurso fiable de copia de seguridad de colaboración, aprovechando estos recursos independientes distribuidos. Con ABS, la adquisición y mantenimiento de hardware especializado de copia de seguridad es innecesaria. ABS hace un uso eficiente de los recursos de red y almacenamiento de información mediante el uso de técnicas, como cifrado convergente, almacenamiento, procesos de verificación y control de versiones eficientes de codificación, para lograr esto hace uso de herramientas criptográficas como firmas digitales, SHA 256 para calcular el Hash y para el cifrado se ocupa criptografía asimétrica [3].

3.2. Protocolo DupLESS.

Los proveedores de servicios de almacenamiento en la nube como lo son: Dropbox, Mozy, Mega y otros (*Figura 3.1*), realizan la eliminación de archivos duplicados para ahorrar espacio en el servidor, almacenando sólo una copia de cada archivo cargado.



Figura 3.1: Servicios de almacenamiento en la nube

Ahora bien, si los clientes cifran convencionalmente sus archivos es decir usando la criptografía estándar (Simétrica y Asimétrica), se pierde ese ahorro de espacio. El cifrado convergente en adelante (*CE*) resuelve esta problemática. Sin embargo, está inherentemente sujeto a ataques de fuerza bruta.

El protocolo criptográfico *DupLESS* contiene una arquitectura que proporciona almacenamiento sin duplicados, seguro y resistente a ataques por fuerza bruta. En DupLESS, los clientes cifran sus archivos con claves basadas en mensajes, obtenidos desde un servidor de llaves a través de una función llamada *OPRF*[(*Oblivious Pseudorandom Function*) Función *Pseudoaleatoria de Poca Memoria*]. Esto permite a los clientes almacenar datos cifrados en un servicio de almacenamiento, y en este servicio se realiza la eliminación de archivos duplicados garantizando la confidencialidad del cliente. Dupless demuestra que el cifrado para el almacenamiento sin duplicados puede lograr ahorros de rendimiento y espacio [1].

Funcionamiento

DupLESS contempla que un ataque por fuerza bruta a un texto cifrado dentro de un esquema de tipo CE puede tratarse utilizando un servidor de llaves *KS* para derivar llaves a partir de un archivo, en lugar de establecer llaves con funciones hash de los archivos. Para poder acceder al *KS*, se debe de realizar el procedimiento de autenticación a los usuarios, esto detiene a los atacantes externos. El aumento de los costos a los ataques, disminuye los ataques por fuerza bruta hacia los usuarios.

DupLESS implementa un esquema que consta de cinco algoritmos (*Kg*, *EvC*, *EvS*, *Vf*, *Ev*), los últimos dos deterministas:

- *Generación de claves Kg*:
 $(pk, sk) \leftarrow Kg$. Este algoritmo genera una llave pública $pk = (e, n)$ que puede distribuirse libremente entre varios usuarios, y una clave privada $sk = (d, n)$ que permanece sólo en el servidor de llaves. Dichas claves serán parte del servidor, son generadas a través del algoritmo de clave pública RSA.
- *EvC*:
La evaluación de este protocolo comienza del lado del cliente:

1. Se obtiene del servidor la clave pública e y se realiza una comparación con $e \leq N$, N fue el producto de 2 números primos utilizado en la generación de llaves del servidor. Si la comparación no se cumple, el protocolo envía un error.
2. Se elige un número aleatorio entero $r < N$.
3. Se realiza una función hash al archivo que se desea almacenar en Nube, es decir $h \leftarrow H(M)$
4. Ahora se multiplica la función hash del mensaje M por el número aleatorio elevado a la llave pública del servidor (r^e) y se almacena en \mathbf{x} , es decir $x \leftarrow h \cdot r^e \pmod{N}$
5. Esta \mathbf{x} es enviada al servidor de llaves, en donde será firmada por éste sin saber su contenido ni procedencia, es decir realizar una firma a ciegas del mensaje a cargar.
6. La firma realizada por el servidor es recibida en el cliente y se realiza la siguiente operación $z \leftarrow y \cdot r^{-1} \pmod{N}$

■ *EvS:*

Del lado del servidor, se recibe la entrada x que corresponde a la función hash del archivo enviada por el lado del cliente. El servidor sólo realiza una operación la cuál es la firma con la llave privada del servidor. Dicha firma es a ciegas ya que el servidor no conoce el contenido del archivo ni la procedencia de quien lo está enviando. Dicha firma se realiza de la siguiente manera: $y \leftarrow x^d \pmod{N}$

■ *Vf:*

Este algoritmo de verificación se encarga de corroborar la firma realizada por el servidor, es decir, verifica que efectivamente sea el servidor de llaves KS la entidad que realizó la firma a ciegas que el cliente estaba solicitando. Además de que también este algoritmo asegura que el archivo firmado por el servidor KS sea el que el cliente envió para su solicitud de almacenamiento. Dicho algoritmo opera de la siguiente manera:

- Realiza la operación $z^e \pmod{N}$. Dicha operación se realiza para eliminar el factor de ocultamiento realizado por el cliente. De esta manera se puede llegar hasta la comprobación de la misma función hash generada en el cliente, si ésta es igual a la que estaba bajo el factor de ocultamiento, el algoritmo comprueba que el KS firmó correctamente y envía un 1.
- Si el algoritmo realizó la operación $z^e \pmod{N}$ y el resultado en la comparación de las funciones hash no es el mismo, entonces el servidor no llevó a cabo correctamente el proceso de firmado y arroja un error 0.

■ *Ev:*

Como último algoritmo se encuentra Ev, dicho algoritmo se encarga del proceso de cifrado $C1$ del archivo original M junto con la clave generada z a través del proceso entre el cliente y el servidor KS . También realiza el cifrado $C2$ de la clave z junto con

la clave privada del usuario que desea almacenar un archivo. Dichas operaciones son las siguientes:

- $C1 \leftarrow E_z(M)$
- $C2 \leftarrow E_{pk}(z)$

Capítulo 4

Análisis

El capítulo de análisis dentro de este documento, contiene todo el proceso que corresponde a la etapa de análisis de la modelo de desarrollo de software en cascada. Dicho análisis comprende la presentación del estudio de factibilidad de este proyecto, factibilidad técnica, factibilidad operativa y económica, además de un análisis de riesgos posibles en la realización de este proyecto. Asimismo, se presenta la arquitectura que tendrá el protocolo criptográfico, la descripción de todos los procesos involucrados en el funcionamiento de este protocolo junto con los actores correspondientes a cada proceso. Como parte final de este capítulo, se encontrarán los requerimientos funcionales que este protocolo debe cubrir y de igual forma los requerimientos funcionales que estarán presentes dentro del proyecto.

4.1. Estudio de Factibilidad

Como se expuso anteriormente, el protocolo criptográfico que se presenta en este trabajo terminal da como resultados grandes e importantes beneficios en cuanto a la optimización de espacio de memoria en la nube y garantiza la seguridad e integridad de los datos de usuarios registrados en la nube. Hasta ahora se tiene muy claro que este trabajo terminal, evitará la duplicación de archivos en la nube garantizando la integridad y confidencialidad tanto de los archivos almacenados como de los usuarios de la nube, pero aún queda por establecer cuán viable es la realización de este protocolo en cuanto a facilidades técnicas y operativas.

4.1.1. Factibilidad técnica

Para el desarrollo de este trabajo terminal, se cuenta con el software necesario para la implementación del protocolo criptográfico propuesto, ya que este existe y está disponible para su uso y disponibilidad inmediata al desarrollo del protocolo. Cabe mencionar que los insumos que involucran todo el desarrollo del protocolo, se encuentran completas en tanto a funcionalidades, soportes, documentación, seguridad, etc. En cuanto a los conocimientos necesarios al protocolo, se cuentan con los suficientes que este protocolo necesita para su aplicación. Dichos conocimientos se han adquirido a lo largo de la estadía en la ingeniería mediante las clases recibidas en las instalaciones de la escuela o por la adquisición por

medios externos. Ahora bien, un pequeño inconveniente para la aplicación del protocolo, es el desarrollo del software, ya que existen partes fundamentales del protocolo que requieren de nuevo conocimiento que aun no se adquiere y es necesario aprender, por tanto, existe una pequeña posibilidad de que esto signifique un retraso en la producción del protocolo.

- **Lenguaje de programación:** para el desarrollo del sistema se estará trabajando con varios lenguajes de programación, la base de este desarrollo será Python, ya que es un lenguaje de fácil comprensión y con una lógica bastante familiar a lenguajes que anteriormente se habían trabajado, además de que Python nos permite trabajar primitivas criptográficas con una mayor facilidad.
- **Sistema gestor de base de datos:** Se implementará MySQL como un manejador de base de datos, ya que es software libre y además se cuenta con una mayor experiencia de manejo y entendimiento.
De acuerdo con los requerimientos del sistema, los componentes necesarios para la implementación de este protocolo son:
 - **Hardware** Se requiere de equipo de cómputo para poder llevar a cabo la codificación del protocolo y la configuración de los servidores que les darán soporte a las actividades realizadas en la interacción del usuario con el sistema. El equipo de trabajo cuenta con 3 computadoras personales (LAPTOP) que son las siguientes:

Componentes Físicos (Hardware)	
Componente	Características
Laptop HP Pavilion g4	<ul style="list-style-type: none">● Procesador: AMD A6-4400M APU 2.70Hz● Memoria RAM: 8.00GB● Disco Duro: 750GB● Tipo Sistema: 64bits x64
Laptop Acer Aspire V5	<ul style="list-style-type: none">● Procesador: Intel(R) Celeron(R) 1.50GHz● Memoria RAM: 2.00GB● Disco Duro: 250GB● Tipo Sistema: 64bits x64

Continúa en la siguiente página.

Componente	Características
Laptop HP probook	<ul style="list-style-type: none"> • Procesador: AMD Phenom(tm) ll X2 545 3.00GHz • Memoria RAM: 4.00GB • Disco Duro: 350GB • Tipo Sistema: 64bits x64
Servidor ownCloud	<ul style="list-style-type: none"> • Servicio de alojamiento de archivos con almacenamiento en la nube • Servidores de instalación: PHP, SQLite, MySQL, PostgreSQL • Servidor de archivos: WebDAV • Calendario de sincronización: CardDAV • Sistema operativo: Multiplataforma

Tabla 4.1: Componentes físicos

- **Software** Con respecto al software, no se pretende usar un sistema operativo específico ya que el servicio de la nube es compatible con cualquier sistema operativo en que se esté desarrollando. Sin embargo, se cuenta con los siguientes sistemas operativos:

Software	
Tipo	Cantidad
S.O. Windows 8.1 Pro	2
S.O. Windows 10 Pro	1

Tabla 4.2: Componentes Lógicos

4.1.2. Factibilidad operativa

El uso e implantación de este protocolo criptográfico, tiene como principal objetivo el eliminar la duplicación de archivos que se almacenan en la nube y de igual manera el evitar los ataques de adversarios por fuerza bruta a la información de los usuarios en la nube, por lo cual resulta factible la operación del proyecto, ya que se está proponiendo una solución de bajo costo con grandes beneficios.

Hoy en día va en aumento el número de usuarios de la nube, ya sean empresas, organizaciones, personas, etc. todos estos usuarios buscan su información de manera rápida y de fácil acceso en cualquier lugar con una conexión a internet, y esto hace que también aumente la cantidad

de información almacenada que en ocasiones está replicada en uno o más usuarios. Este protocolo pretende atender al problema del crecimiento en la infraestructura de la nube ofreciendo una solución para contrarrestarlo y de esa manera más usuarios podrán tener acceso a la nube sin necesidad de aumentar la infraestructura de ésta.

4.1.3. Factibilidad económica

El estudio de factibilidad económica nos permite analizar los costo - beneficios monetarios que se obtendrán con el desarrollo del proyecto, ya que dicho protocolo en un largo plazo podría comenzar en producción industrial para su comercialización y distribución en el mercado de las tecnologías de la información, ya que proveerá de servicios en el cómputo nube que hoy en día las empresas están optando por invertir en él y optimizar sus recursos y procesos del negocio.

En cuanto al análisis del gasto económico que se requiere para la implementación y la realización de pruebas del protocolo, existe un beneficio ya que todos los componentes para el desarrollo del sistema son libres, es decir que no se realizará la adquisición de licencias de desarrollo ni programas para la implementación.

El servidor que nos proveerá del almacenamiento en la nube de nombre **ownCloud.org** es de libre acceso y no existe restricción alguna para la utilización de éste. Nos permite una libre manipulación y configuración de tal modo que podamos cumplir nuestros objetivos.

4.2. Modelo del Proceso de Desarrollo del Software

Para desarrollo de este trabajo terminal, se tomó como referencia el **Modelo de proceso de desarrollo en cascada con retroalimentación** ver Figura 4.1 .

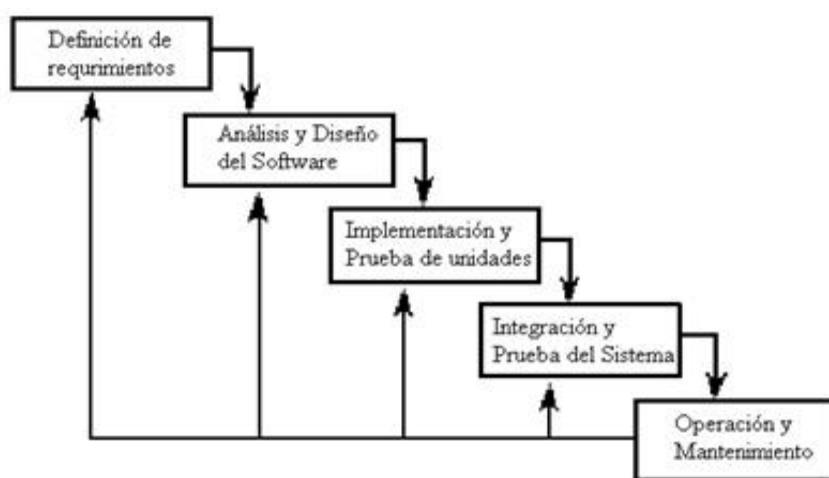


Figura 4.1: Modelo de proceso de desarrollo de software en Cascada con retroalimentación

Dicho modelo fué tomado como desarrollo para este protocolo ya que:

- Es un modelo lineal con retroalimentación, por tanto es más fácil implementar dada la complejidad que involucra el desarrollo de este protocolo, ya que se puede retroceder a la etapa anterior para corregir errores y optimizar los procesos.
- La cantidad de recursos que se necesitan para implementar este modelo en el protocolo es mínimo.
- La documentación del protocolo se produce en cada etapa del desarrollo del modelo de cascada, y debido a que este modelo es con retroalimentación, hace que la comprensión y el desarrollo de este proyecto sea más fácil de implementar con un procedimiento sencillo.
- Después de la etapa de codificación del protocolo, las pruebas se realizan para comprobar el correcto funcionamiento de éste.
- Es menos probable que se presenten errores al momento de avanzar a la siguiente etapa, ya que la fase posterior comienza hasta el término de la fase anterior inmediata [10].

Este modelo comprende 5 fases primordiales para el proceso del desarrollo de software, sin embargo en este trabajo terminal se hará enfasis en las fases 1 a 4, y este protocolo será presentado hasta la integración y primeras fases de funcionamiento, y la última fase de operación y mantenimiento se contempla después de un periodo de tiempo en el que el sistema se pone en producción .

Dichas fases son las siguientes:

1. **Análisis y definición de requerimientos.** Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces, se definen en detalle y sirven como una especificación del sistema.
2. **Diseño del sistema del software.** El proceso de diseño del sistema divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.
3. **Implementación y prueba de unidades.** Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
4. **Integración y pruebas del sistema.** Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las puebas, el sistema software se entrega al cliente.

5. **Funcionamiento y mantenimiento.** Por lo general, esta fase es la más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos [10].

4.3. Análisis de riesgos

El análisis de riesgos consiste en identificar los riesgos que este sistema puede tener en su implementación y desarrollo. Este análisis realizado a este trabajo terminal, se llevó a cabo bajo la metodología de análisis de riesgos presentada por Ian Sommerville en el libro Ingeniería del software [10], dicho autor presenta los posibles riesgos identificados en un sistema y 3 indicadores (Prioridad, Probabilidad e Impacto) fundamentales para llevar a cabo la gestión de riesgos y así tener presente cuales son los riesgos más destacados en el proyecto para dimensionar las consecuencias que estos podrían traer consigo si llegaran a ocurrir. Para elaborar este análisis de riesgos, se evaluaron los impactos que pueden traer, mediante 5 escalas de impacto de un riesgo, dichas escalas son:

4.3.1. Escalas de impacto de un riesgo

Escalas de impacto de un riesgo					
<i>Objetivo del proyecto</i>	Muy bajo/ 0.05	Bajo/ 0.10	Moderado/ 0.20	Alto / 0.40	Muy alto/ 0.80
Costo	Aumento insignificante de costo	Aumento del costo <10 %	Aumento del costo 10 - 20 %	Aumento del costo 20 - 40 %	Aumento del costo >40 %
Tiempo	Aumento insignificante de tiempo	Aumento del tiempo <5 %	Aumento del tiempo 5 - 10 %	Aumento del tiempo 10 - 20 %	Aumento del tiempo >20 %
Alcance	Disminución del alcance apenas perceptible	Áreas de alcance secundarias afectadas	Áreas de alcance principales afectadas	Reducción del alcance inaceptable para los objetivos	El producto terminado es inservible
Calidad	Degradación de calidad apenas perceptible	Sólo las aplicaciones muy exigentes se ven afectadas	La reducción de la calidad requiere aprobación externa	Reducción de calidad inaceptable	El producto terminado es inservible

Tabla 4.3: Escalas de impacto de un riesgo [10]

Análisis de riesgos					
ID	Riesgo	Prioridad	Probabilidad	Impacto	Causa
R1	Modificar requerimientos	ALTA	MEDIA	ALTO	Se agregan nuevos o modificación de requerimientos actuales
R2	Cambios en tecnología	BAJA	MEDIA	BAJO	La tecnología usada es menos eficiente y causa conflictos
R3	Falta de usuarios y peticiones	ALTA	BAJA	BAJO	No se localizaron todos los involucrados en el protocolo
R4	Equipo de cómputo	ALTA	BAJA	MUY BAJO	El equipo de cómputo empleado falla o no se encuentra funcionando
R5	Incumplimiento de acuerdos	ALTA	ALTA	MODERAD	Retraso en actividades programadas
R6	Problemas de planeación	MEDIA	ALTA	ALTO	Falta de comunicación por parte del equipo
R7	Falta de recursos	ALTA	MEDIA	MUY ALTO	No se cuenta con el poder adquisitivo de recursos e insuimos
R8	Falta de conocimientos	MEDIA	MEDIA	MUY ALTO	El equipo no se encuentra capacitado para el desarrollo del sistema.
R9	Sobre estimación de insumos	MEDIA	MEDIA	MODERAD	Los insumos para la implementación del sistema no son los adecuados para el desarrollo.

Tabla 4.4: Análisis de riesgos del proyecto

Dichos riesgos fueron considerados durante la realización de este proyecto, ya que se tomaron en cuenta todos los factores que se ven involucrados en el ambiente que se encuentra este trabajo terminal. Principalmente se tienen muy en cuenta el factor humano, es decir, los errores, modificaciones, alteraciones que el equipo de trabajo de este trabajo terminal

pudiera cometer o que se encuentre en una situación que pueda comprometer el avance en tiempo y forma de este proyecto.

Asimismo, el riesgo no puede ser mitigado en su totalidad y mucho menos evitar una afectación al desarrollo del trabajo terminal, pero con un plan de acción detallado para cada riesgo identificado se puede disminuir en gran medida las consecuencias que estos riesgos podrían generar, es por ello que se propone el siguiente plan de acción para cada riesgo:

Plan de acción para riesgos del proyecto		
ID	Riesgo	Acción a realizar
R1	Modificar requerimientos	Se cuenta como un requerimiento no funcional, la mantenibilidad, en la cuál dice que cada nuevo requerimiento tendrá que ser analizado para cuantificar las implicaciones de éste y su efectividad
R2	Cambios en tecnología	En cuanto exista una nueva tecnología, se analizarán las ventajas y desventajas de ésta para posteriormente tomar una decisión para actualizar o conservar dicha tecnología
R3	Falta de usuarios y peticiones	El sistema tendrá como máximo 118ms de latencia, en caso de persistir el problema se volverá a realizar la petición y en el peor escenario se notificará del error al administrador del sistema para su corrección
R4	Equipo de cómputo	Cuando exista una falla en el equipo de cómputo que opera el sistema, se procederá a poner fuera de línea el sistema para poder cambiar o reparar el equipo que se encuentra dañado
R5	Incumplimiento de acuerdos	En caso de retraso en actividades programadas, se procederá a una reunión con el equipo de trabajo para dar prioridades a las actividades y avanzar en un mínimo período de tiempo
R6	Problemas de planeación	En caso de presentarse la falta o errónea comunicación en el equipo, se procederá a una reunión con todo el equipo para revisar los incidentes y replantear las actividades
R7	Falta de recursos	A falta de recursos, se buscará la adquisición de estos a costos más bajos a los actuales o se tendrá que adquirir una deuda con un acreedor ajeno al proyecto
R8	Falta de conocimientos	Cuando se presente este problema, el equipo debe inmediatamente reunirse para poder planificar cursos de capacitación, talleres, conferencias, etc.

Continúa en la siguiente página.

ID	Riesgo	Acción a realizar
R9	Sobreestimación de insumos	Los insumos utilizados en el proyecto están especificados desde el inicio del proyecto, en caso de que esta especificación falle se tendrá que realizar una nueva valoración de los insumos que sean necesarios.

Tabla 4.5: Plan de acción ante riesgos en el proyecto

4.4. Arquitectura del sistema.

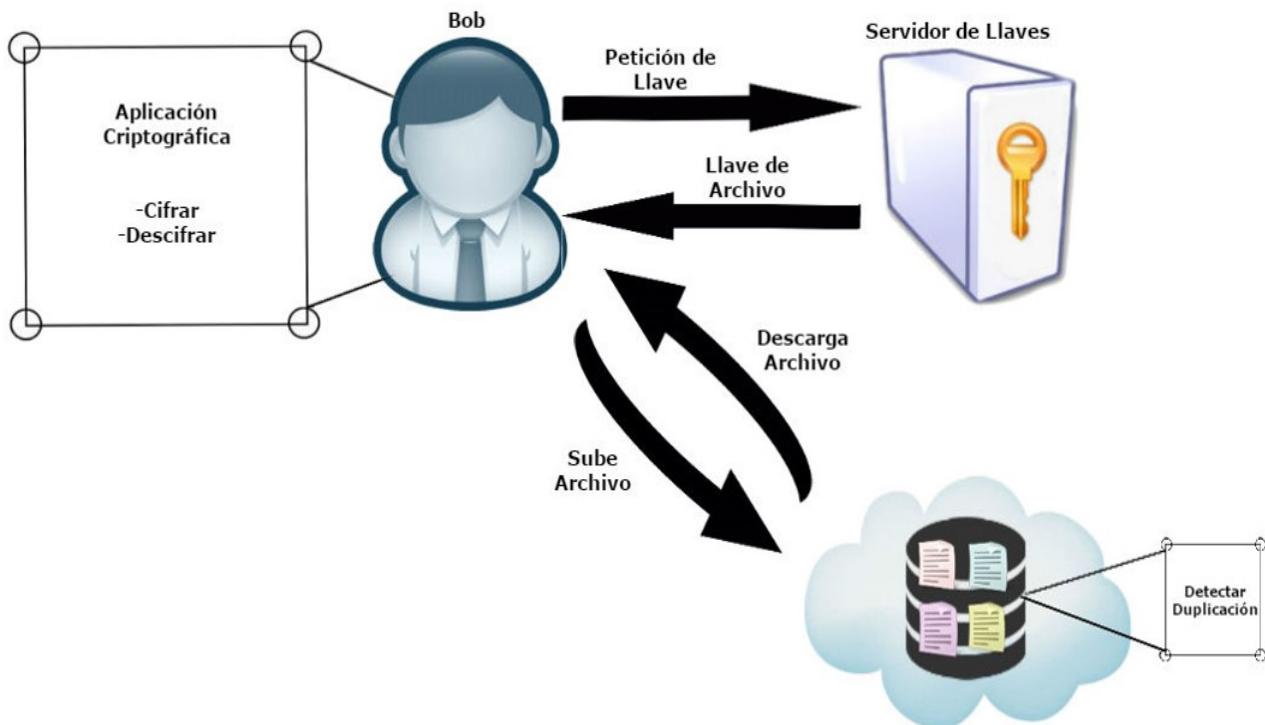


Figura 4.2: Arquitectura general del Protocolo

En este apartado se presenta la arquitectura general del protocolo criptográfico, dicha arquitectura tiene como base de desarrollo al protocolo DupLESS que se menciona en este documento. DupLESS provee a esta arquitectura la implementación de un servidor de llaves *KS* para evitar los duplicados en la Nube garantizando la confidencialidad e integridad de toda la información almacenada por los usuarios del protocolo. Esta arquitectura se compone de 3 importantes entidades que son:

- **Servidor de Llaves *KS*:** Dicha entidad será la responsable de proveer firmas para generar claves específicas para cifrar cada archivo diferente *M* a almacenar por un

usuario en la Nube.

- **Cliente:** Esta entidad corresponde al lado del usuario, el cuál almacenará, descargará y consultará archivos en la Nube, además de realizar procedimientos necesarios para la gestión de un archivo M .
- **La Nube:** La entidad Nube, es el almacenamiento en línea en el cuál se estarán gestionando archivos por diferentes usuarios que tienen acceso a este servicio. De igual manera, esta entidad será la encargada de detectar los duplicados generados por usuarios.



Figura 4.3: Arquitectura del Protocolo paso 1.

- **Generación de claves de Usuario:** El protocolo genera al usuario a través del algoritmo de clave pública RSA un par de claves: *pública* (pk) y *privada* (sk) las cuales serán almacenadas en un certificado digital para su uso posterior. Dicha generación se realiza cuando un usuario se registra por primera vez al protocolo criptográfico. Estas claves servirán a dicho usuario para poder realizar cualquier procedimiento de gestión de archivos en la Nube.
- **Generación de claves del Servidor KS :** El protocolo genera al servidor de llaves su propio par de claves, una *pública* (e) que puede distribuirse libremente entre varios usuarios y una *privada* (d) que permanece sólo en el servidor de llaves. Dichas claves son generadas a través del algoritmo de clave pública RSA.
- Posteriormente, cuando el usuario desea almacenar un archivo M en la nube comienza un proceso el cuál se muestra a continuación:

1. El protocolo obtiene del servidor KS la clave pública e y se realiza una comparación con $e \leq N$, N fue el producto de los 2 números primos utilizados en la generación de llaves del servidor. Si la comparación no se cumple, el protocolo envía un error.
2. Se elige un número entero aleatorio, tal que $r < N$.
3. El usuario genera la función hash $H(M)$ del archivo M
4. El cliente realiza la operación $x \leftarrow h \cdot r^e \pmod{N}$, ya que así se mantiene oculto el mensaje que firmará el servidor.
5. La x obtenida de la operación anterior, es enviada al servidor KS para realizar la firma a ciegas.



Figura 4.4: Arquitectura del Protocolo paso 2.

El cliente envía al servidor KS el archivo x que se requiere firmar para continuar con el proceso de almacenamiento de archivo.

El servidor recibe a x que corresponde a la función hash del archivo M . Para poder realizar la firma, es necesario que el servidor realice una operación con su llave privada d . Dicha firma es a ciegas ya que el servidor no conoce el contenido del archivo M ni el usuario que esta solicitando dicha firma.

El proceso se puede representar de la siguiente manera:

$$y \leftarrow x^d \text{ mód } N$$

Posteriormente, el servidor envía al cliente la firma obtenida y

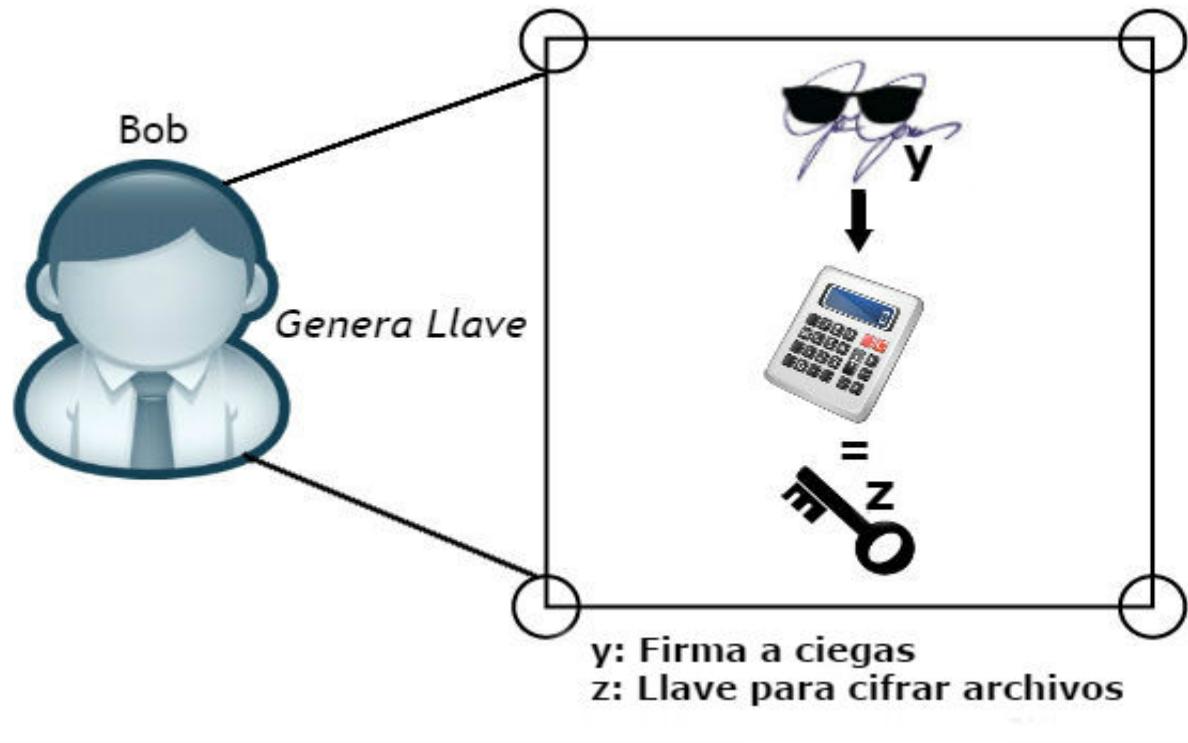


Figura 4.5: Arquitectura del Protocolo paso 3.

Ahora que el cliente recibió la firma y generada por el servidor KS puede continuar con el proceso de almacenamiento de archivo.

Lo que hace el cliente es obtener el valor de z , y se lleva a cabo de la siguiente manera: $z \leftarrow y \cdot r^{-1} \text{ mód } N$.

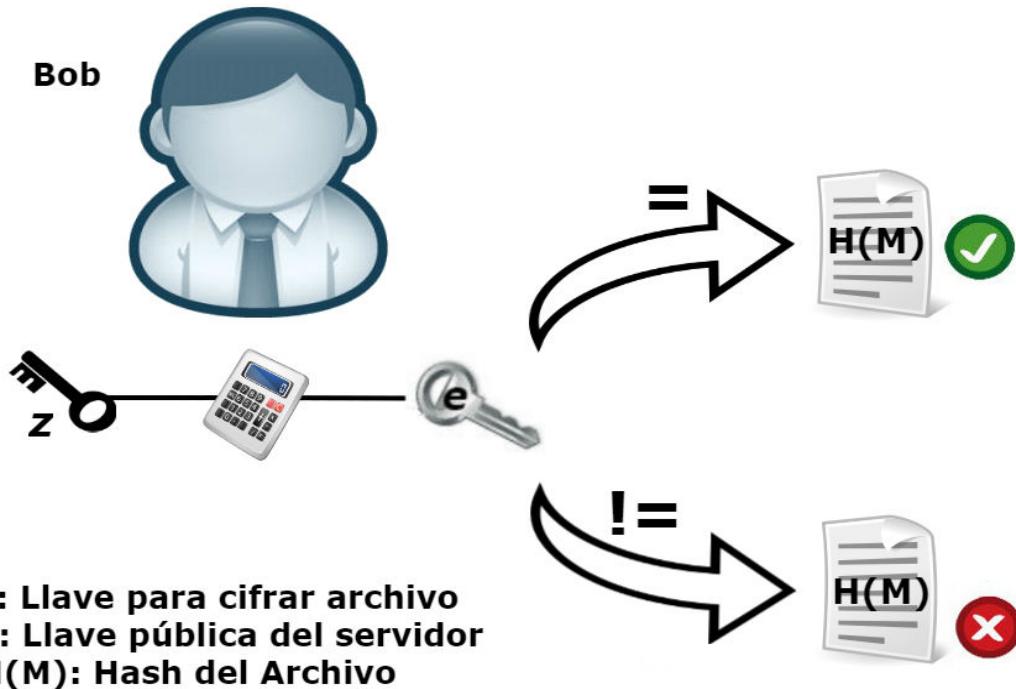


Figura 4.6: Arquitectura del Protocolo paso 4.

Para continuar con el almacenamiento de un archivo M , es necesario realizar una verificación de la procedencia de la firma realizada en el servidor KS . En este paso se corrobora la firma, es decir, verifica que efectivamente sea el servidor de llaves KS la entidad que realizó la firma a ciegas al archivo M que el cliente estaba solicitando. También en este paso se asegura que el archivo firmado por el servidor KS sea el que el cliente envió para su solicitud de almacenamiento.

La verificación se lleva a cabo de la siguiente manera:

- Se realiza la operación $z^e \text{ mód } N$, para eliminar el factor de ocultamiento realizado por el cliente. Así se comprueba que la misma función hash generada en el cliente es igual a la que estaba bajo el factor de ocultamiento, si estas son iguales entonces se verifica que efectivamente el servidor KS fue el responsable de la firma y obtenida.
- Si se realiza la operación $z^e \text{ mód } N$ y el resultado en la comparación de las funciones hash no es el mismo, entonces el servidor no llevó a cabo correctamente el proceso de firmado.

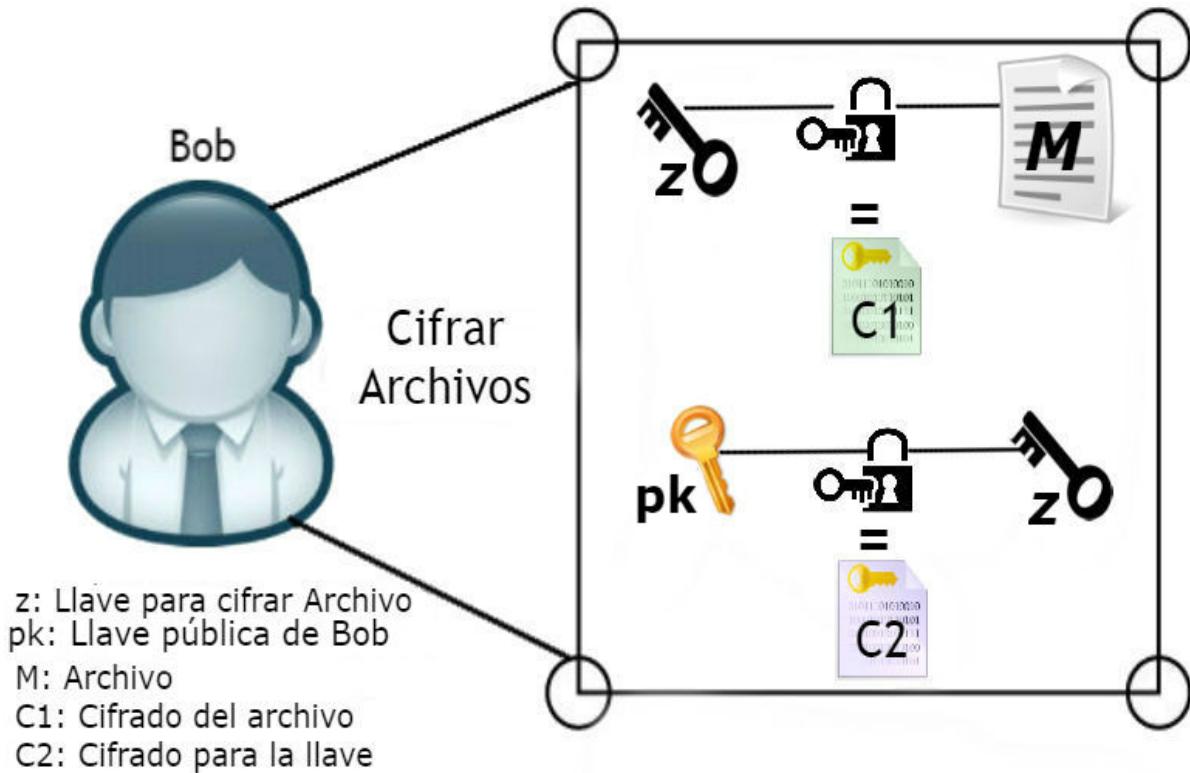


Figura 4.7: Arquitectura del Protocolo paso 5.

Para continuar, el cliente ya verificó la procedencia de la firma y obtuvo la clave z que será la utilizada para llevar a cabo el proceso de cifrado.

En este paso, el cliente se encarga del cifrado llamado $C1$ del archivo original M junto con la clave generada z a través del proceso entre el cliente y el servidor KS . De igual manera, realiza el cifrado llamado $C2$ de la clave z junto con la clave privada pk del usuario que desea almacenar un archivo. Dichas operaciones se representan de la siguiente manera:

- $C1 \leftarrow E_z(M)$
- $C2 \leftarrow E_{pk}(z)$

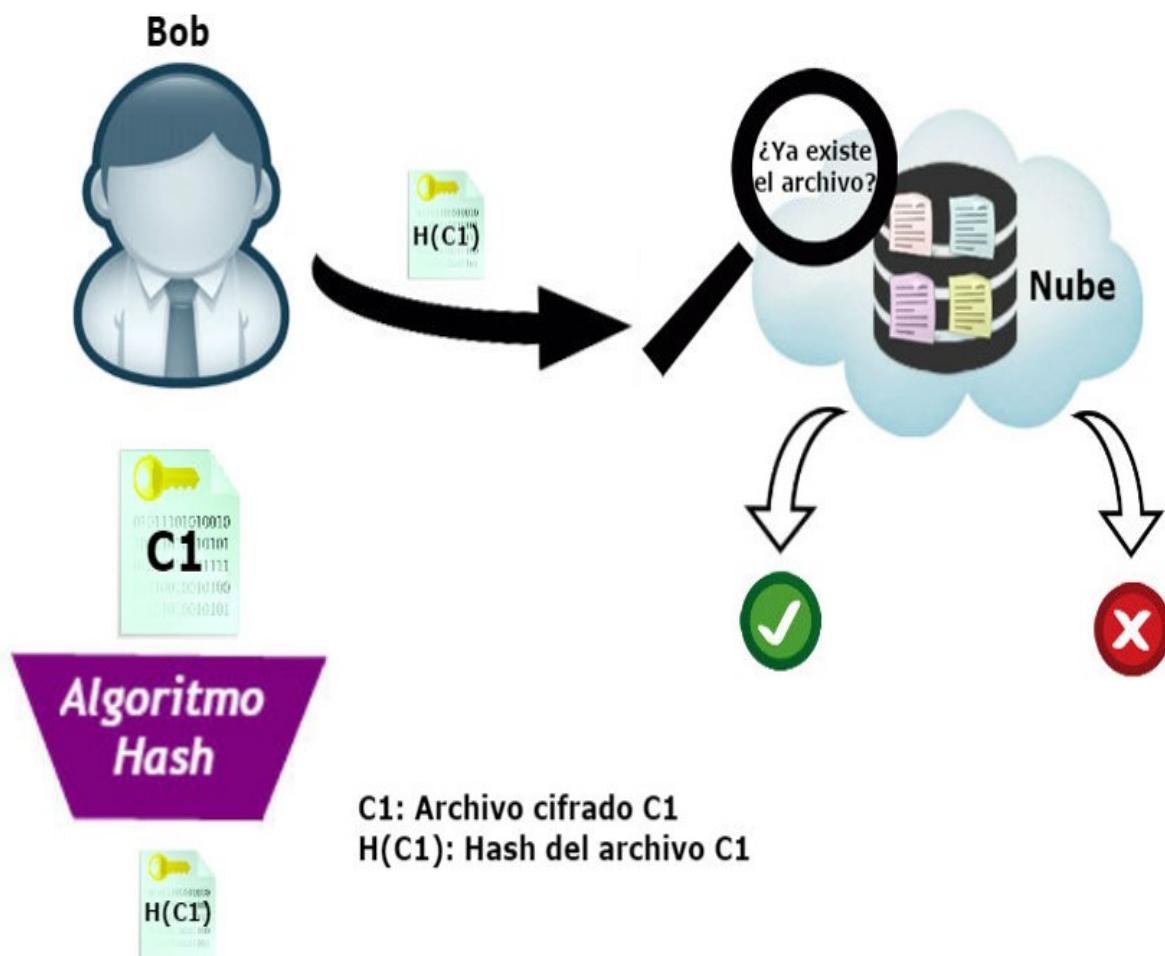


Figura 4.8: Arquitectura del Protocolo paso 6.

El siguiente paso es realizar una función hash al archivo cifrado que se obtuvo en el paso anterior.

Dicha función hash $H(C1)$ corresponde al cifrado del archivo M , con esta función se obtiene un identificador del archivo M , el cuál servirá para detectar los posibles duplicados del archivo M en la Nube.

- Si dicho identificador se encuentra ya almacenado en la Nube: Se procede al almacenamiento del cifrado $C2$ correspondiente a la clave privada del usuario que llevó a cabo todo el proceso, actualizando y asociando la lista de archivos que ahora el usuario tiene registrados en la Nube.
- Si el identificador no se encontró en el almacenamiento de la Nube: Se procede a almacenar el cifrado $C1$ y cifrado $C2$, actualizando y asociando la lista de archivos que ahora el usuario tiene registrados en la Nube.

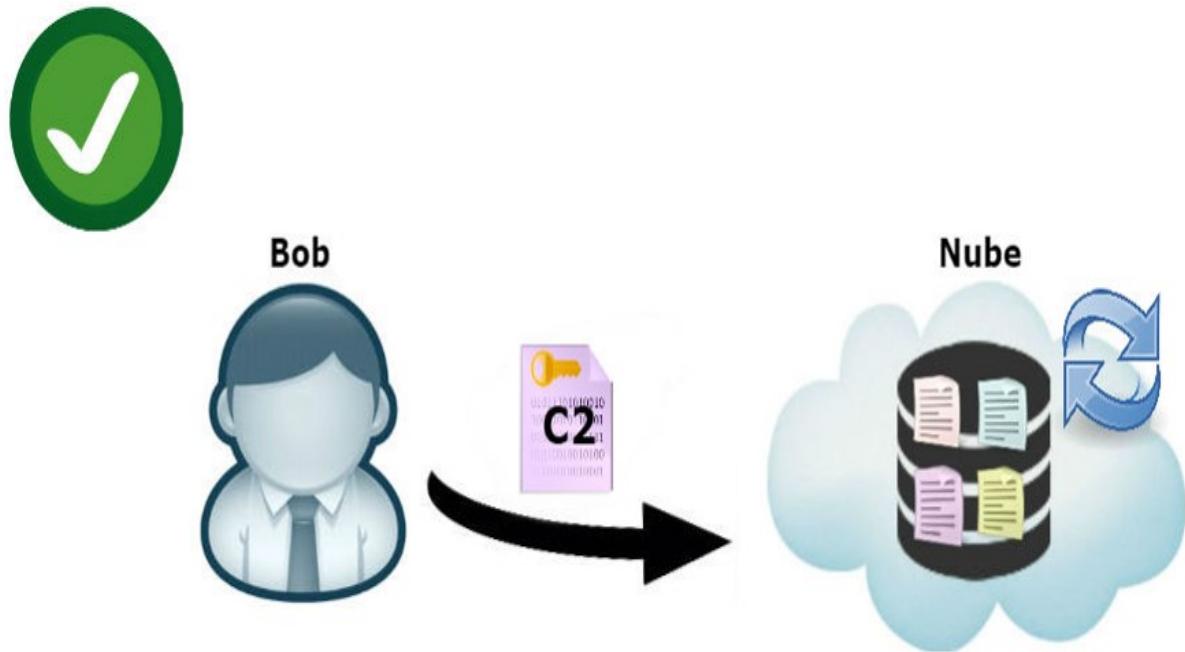


Figura 4.9: Arquitectura del Protocolo paso 7.

En este caso de la detección de duplicados, se encontró una copia almacenada del archivo M por lo que sólo se almacena el cifrado $C2$ correspondiente a la clave privada del usuario que llevó a cabo todo el proceso y se actualiza la lista de archivos que ahora el usuario tiene registrados en la Nube.

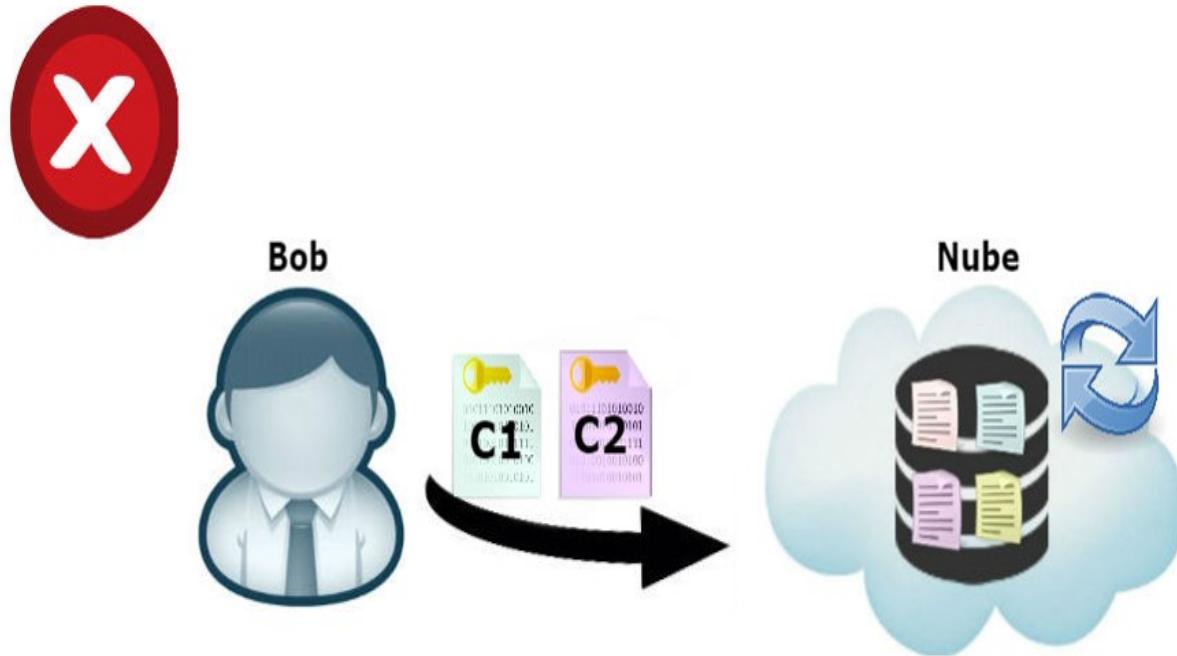


Figura 4.10: Arquitectura del Protocolo paso 8.

En este caso de la detección de duplicados, no se encontró ninguna almacenada del archivo M por lo que se almacena el cifrado $C1$ y el cifrado $C2$ y se actualiza la lista de archivos que ahora el usuario tiene registrados en la Nube.

4.5. Descripción de procesos.

4.5.1. Descripción del proceso subir archivo.

El proceso inicia cuando el cliente desea subir un archivo nuevo, el cliente debe dar clic en la opción de subir archivo y seleccionar el archivo que deseé subir, el sistema va a calcular el hash del archivo elegido, después hará unas operaciones aritméticas con el hash para generar una x que se enviará al servidor para que realice una firma a ciegas, y con esta firma que se le regresará al cliente, se va a generar del lado del cliente su llave z que será la llave con la cual cifrará el archivo, y así si otro archivo que se quiera subir es igual a éste tendrá la misma z y podrá el sistema detectar que son duplicados, también el sistema va a cifrar la llave z por si se le llega a perder al cliente, para poder almacenarlo en la nube el sistema mandará el hash del archivo cifrado para ver si ya está registrado en la base de datos, si es así solo guarda la llave y actualiza la lista de los usuarios que comparten el archivo, de lo contrario solicita la llave cifrada y el archivo cifrado para almacenarlos y actualiza su lista de usuarios.

añadiendo un archivo en ella, y por último se le notificará al cliente que su archivo ha sido almacenado.

Participantes

Participantes		
Nombre	Descripción	Responsabilidades
Servidor	Actor que realiza la firma a ciegas del archivo.	<ul style="list-style-type: none"> ■ Firma a ciegas.
Cliente	Actor que sube archivos a la Nube.	<ul style="list-style-type: none"> ■ Selecciona archivo a subir. ■ Genera hash del archivo. ■ Calcula la llave z. ■ Cifra los archivos a subir.
Nube	Actor que almacena los archivos.	<ul style="list-style-type: none"> ■ Almacena los archivos seleccionados. ■ Genera lista de usuarios relacionados.

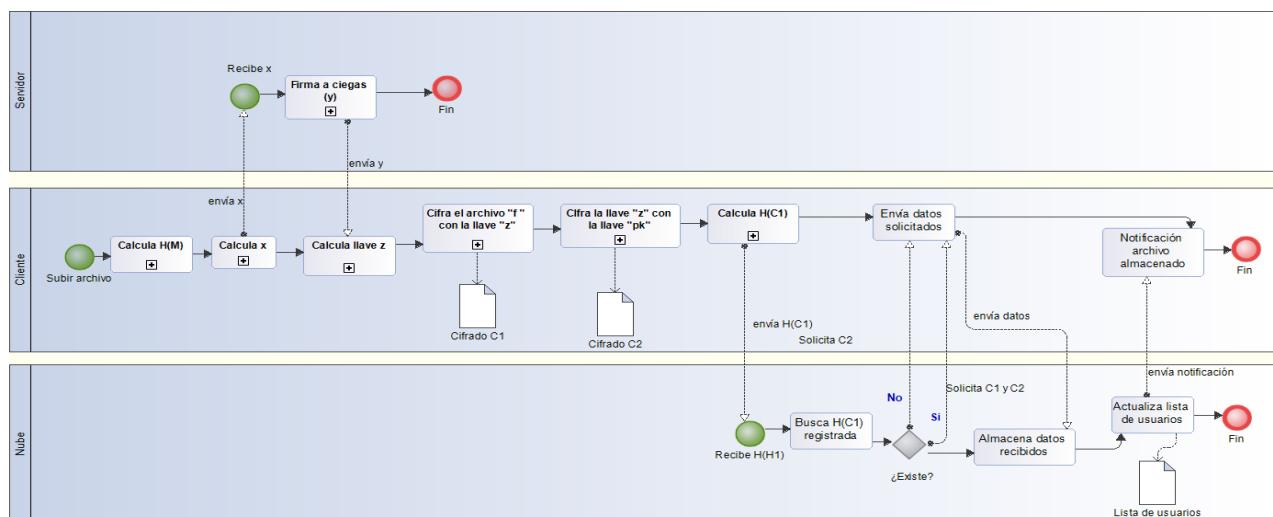


Figura 4.11: BPMMN Subir archivo.

4.5.2. Descripción del proceso Descargar archivo.

El proceso inicia cuando el cliente da clic en la opción de descargar archivo y seleccionar el archivo a descargar, el sistema va a mandar el nombre del archivo a la nube para que busque en su base de datos los archivos correspondientes al usuario y nombre del archivo, se le regresará al cliente y el sistema en el lado del cliente deberá descifrar el archivo C_2 que contiene la llave z para poder descifrar el otro archivo C_1 donde se encuentra el archivo original, el sistema notificara al cliente que su archivo se ha descargado con éxito y este podrá abrirlo.

Participantes

Participantes		
Nombre	Descripción	Responsabilidades
Cliente	Actor que descarga archivos de la Nube.	<ul style="list-style-type: none">■ Selecciona archivo a descargar.■ Descifra los archivos descargados.
Nube	Actor que almacena los archivos.	<ul style="list-style-type: none">■ Almacena los archivos seleccionados.■ Genera lista de usuarios relacionados.■ Enviar los archivos a descargar.

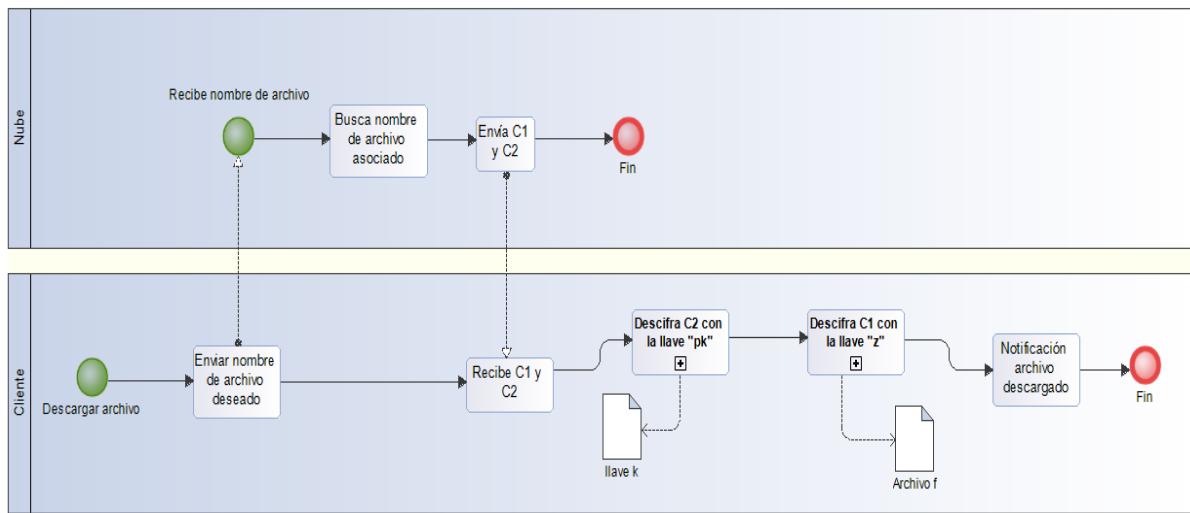


Figura 4.12: BPMN Descargar archivo.

4.5.3. Descripción del proceso eliminar archivo.

El proceso inicia cuando el cliente desea eliminar un archivo nuevo, el cliente debe dar clic en la opción de eliminar archivo y seleccionar el archivo que desee eliminar, el sistema va a enviar el nombre del archivo a la nube donde este buscará en su base de datos los archivos que corresponden al usuario y nombre del archivo, los va a eliminar de su base de datos y actualizará su lista de usuarios eliminando de ella los datos del archivo y usuario que coinciden con el archivo eliminado, se le enviará una notificación al cliente que su archivo ha sido eliminado con éxito de la nube.

Participantes

Participantes		
Nombre	Descripción	Responsabilidades
Cliente	Actor que elimina archivos de la Nube.	<ul style="list-style-type: none"> ■ Selecciona archivo a eliminar.
Nube	Actor que almacena los archivos.	<ul style="list-style-type: none"> ■ Elimina los archivos seleccionados. ■ Genera lista de usuarios relacionados.

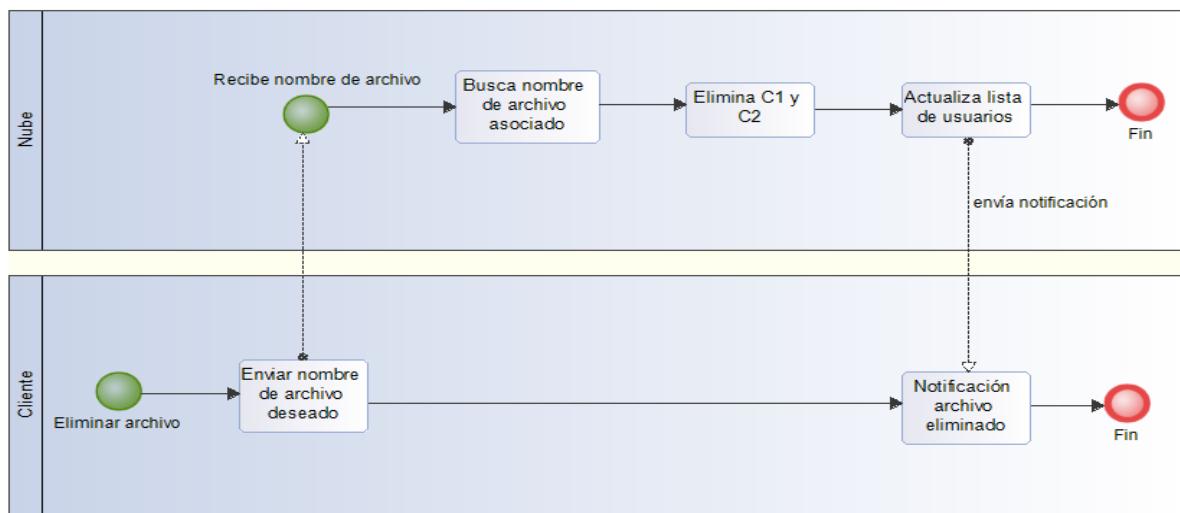


Figura 4.13: BPMN Eliminar archivo.

4.6. Modelo de entidades.

4.6.1. Diagrama de Entidad Relación.

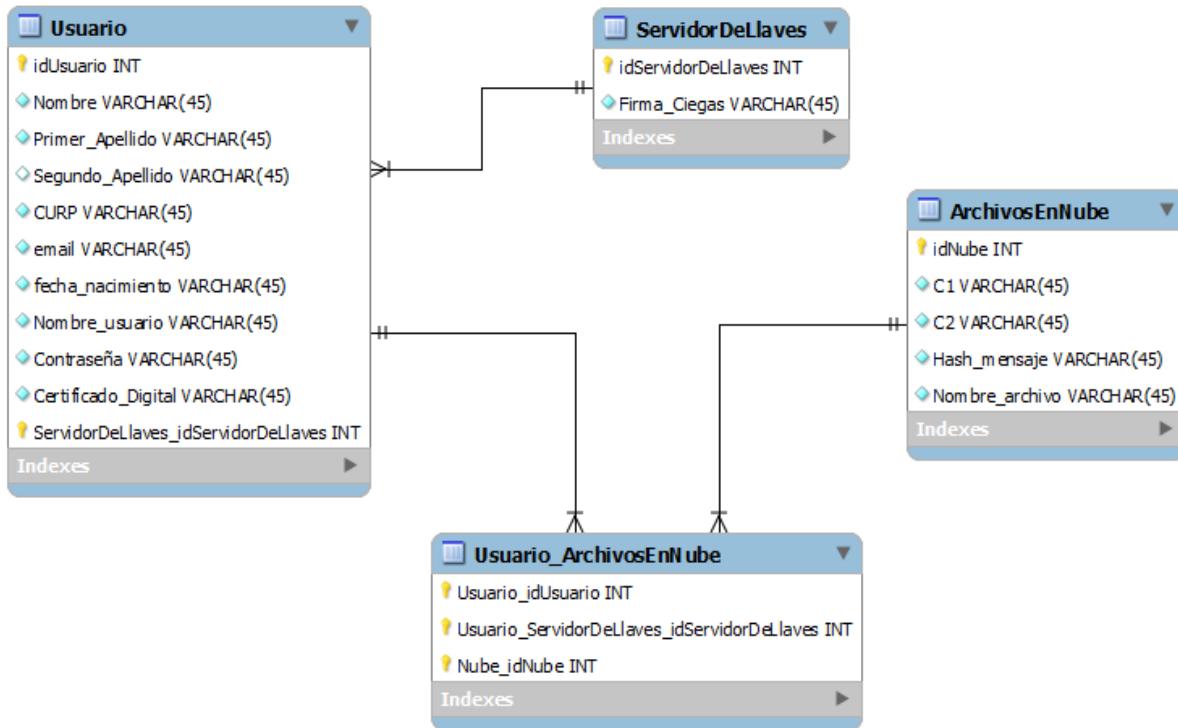


Figura 4.14: Diagrama Entidad relación del sistema.

4.6.2. Diagrama de clases.

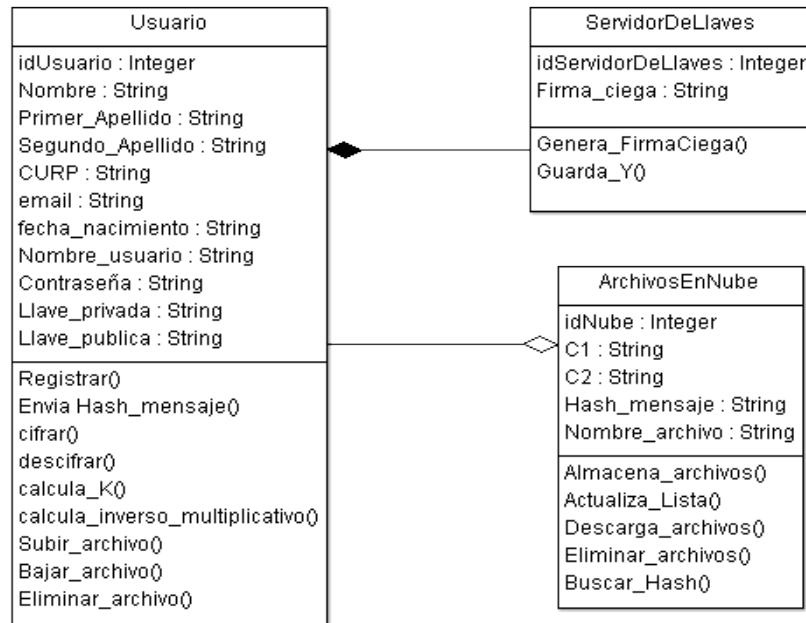


Figura 4.15: Diagrama de clases del sistema.

4.7. Requerimientos Funcionales.

Requerimientos funcionales del protocolo	
ID	Descripción
RF1	El sistema permitirá el registro de un nuevo usuario
RF2	El sistema permitirá al usuario iniciar sesión para comenzar a manipular su información
RF3	El sistema permitirá al usuario gestionar su perfil para la visualización, actualización y configuración de su información
RF4	El sistema permitirá al usuario gestionar los archivos que dicho usuario tiene registrado en su perfil

Tabla 4.6: Requerimientos funcionales del servidor de llaves

Servidor de Llaves	
ID	Descripción
RF – SLL1	El sistema permitirá la generación de llaves de usuario a través de las llaves pública y privada del servidor de llaves
RF – SLL2	El sistema permitirá la firma a ciegas (y) de cualquier archivo que se desee almacenar

Tabla 4.7: Requerimientos funcionales del servidor de llaves

Cliente	
ID	Descripción
RF – CL1	El sistema permitirá al usuario gestionar archivos: Subir, Descargar, Eliminar
RF – CL2	El sistema permitirá al usuario subir un archivo (M) cifrado al servicio de almacenamiento
RF – CL3	El sistema permitirá al usuario descargar un archivo (M) descifrado elegido de su lista de archivos en el servicio de almacenamiento.
RF – CL4	El sistema permitirá al usuario eliminar un archivo (M) cuando el usuario elige alguno de su lista de archivos cargados en el servicio de almacenamiento.

Tabla 4.8: Requerimientos funcionales del cliente

Servicio de almacenamiento (Nube)	
ID	Descripción
RF – N1	El sistema permitirá al servicio de almacenamiento llevar a cabo un proceso para la detección de archivos duplicados.

Tabla 4.9: Requerimientos funcionales del Servicio de almacenamiento (Nube)

4.8. Requerimientos No Funcionales

Requerimientos No Funcionales		
ID	Atributo	Descripción
RNF1	Eficiencia	<ul style="list-style-type: none"> ■ El servidor de llaves tendrá la capacidad de realizar 1000 peticiones de gestión de almacenamiento de archivos por segundo. ■ El sistema podrá funcionar de forma correcta con usuarios conectados de manera concurrente. ■ Los archivos que sean gestionados dentro del servidor de almacenamiento, deben ser actualizados en la base datos y la visualización de cada cliente de manera casi inmediata.
Continúa en la siguiente página		

ID	Atributo	Descripción
RNF2	Fiabilidad	<ul style="list-style-type: none"> ■ La pérdida de consultas en el servidor de llaves es menor a 3 veces el máximo de consultas realizadas. ■ Los archivos almacenados en el servidor de almacenamiento deben ser recuperados por el usuario al instante en que este lo solicite. ■ El tiempo de latencia que existe entre el servidor de llaves y el cliente será de máximo 118ms.
RNF3	Seguridad	<ul style="list-style-type: none"> ■ El sistema almacenará los datos de los usuarios y sus contraseñas en una base de datos MySQL, dichos datos serán modificados mínimo 2 veces al año. ■ Se autenticarán los clientes antes de comenzar el proceso de generación de llaves de archivo. ■ El servidor de llaves firmará claves para un sólo mensaje a la vez sin saber el contenido de éste. ■ El inicio de sesión de usuarios estará protegido en un canal seguro utilizando algoritmos criptográficos. ■ Las funciones hash de archivos a almacenar utilizarán la función criptográfica SHA-(256) ■ Los formularios para ingresar datos al sistema estarán validados por tipo de dato, longitud e internamente se utilizará un ORM (Object Relational Mapping) para evitar inyecciones SQL.

Continúa en la siguiente página

ID	Atributo	Descripción
RNF4	Mantenibilidad	<ul style="list-style-type: none"> ■ Cualquier nuevo requerimiento funcional o no funcional tendrá que ser analizado y diseñado para poder cuantificar las implicaciones que este tendrá sobre el funcionamiento del sistema. ■ El sistema contará con un plan de pruebas que facilitará la identificación de posibles fallas existentes en el funcionamiento de este.
RNF5	Usabilidad	<ul style="list-style-type: none"> ■ El tiempo de aprendizaje del sistema por un usuario deberá ser menor a 15 días. ■ El sistema debe proporcionar mensajes de error que sean informativos y orientados al usuario final. ■ El sistema debe poseer interfaces gráficas bien formadas.
RNF6	Extensibilidad	<ul style="list-style-type: none"> ■ El sistema podrá tener un crecimiento a futuro ya que este será programado por módulos lo cual hará más fácil su crecimiento.

Tabla 4.10: Requerimientos no funcionales del sistema

4.9. Reglas de Negocio

Regla de Negocio: RN1 Datos requeridos

Descripción: El usuario debe ingresar toda la información marcada como requerida en el modelo conceptual del negocio.

Tipo: Restricción de operación.

Regla de Negocio: RN2 Datos correctos

Descripción: La información que el usuario proporcione, debe ser del tipo y longitud definida en el modelo conceptual del negocio.

Tipo: Restricción de operación.

Regla de Negocio: RN3 Unicidad de elementos

Descripción: Hay ciertos elementos que no pueden repetirse, ya sea por ser idénticos o por coincidir en uno o más datos. Esto se define como dato único en la tabla de atributos del modelo conceptual del negocio para cada entidad.

Tipo: Restricción de operación.

Regla de Negocio: RN4 Usuario registrado

Descripción: El usuario debe tener una cuenta activa en el sistema.

Tipo: Hecho

Capítulo 5

Diseño

El capítulo de diseño, está conformado por la especificación de cómo está conformado el protocolo criptográfico. Dicho capítulo contiene la especificación de la plataforma del protocolo, es decir, los recursos tanto de hardware como de software necesarios para poder desarrollar el proyecto con las herramientas tecnológicas necesarias. El capítulo muestra los modelos de entidades del protocolo, dichos modelos comprenden el modelo entidad relación que corresponde a la base de datos creada para la persistencia de los datos involucrados en el protocolo, y el diagrama de clases que brinda un bosquejo del funcionamiento e interacción de atributos y operaciones del protocolo. Por último, se muestra la descripción de todos los casos de uso que conforman al proyecto.

5.1. Especificación de Plataforma

Estación de trabajo y computadores personales

1. Hardware

- Procesador: Intel Core i3 o superior.
- RAM: 2 GB o superior

2. Software

- Visual Paradigm Community Edition 13.2.
- ArgoUML 0.34.
- Inkscape 0.92.
- Python 3.6.1.
- Photoshop Online.
- LaTeX.
- MySQL Community Edition.
- MySQL Workbench.

- ownCloud.

3. Red

- Conexión a internet de 2 Mb/s

5.2. Casos de Uso

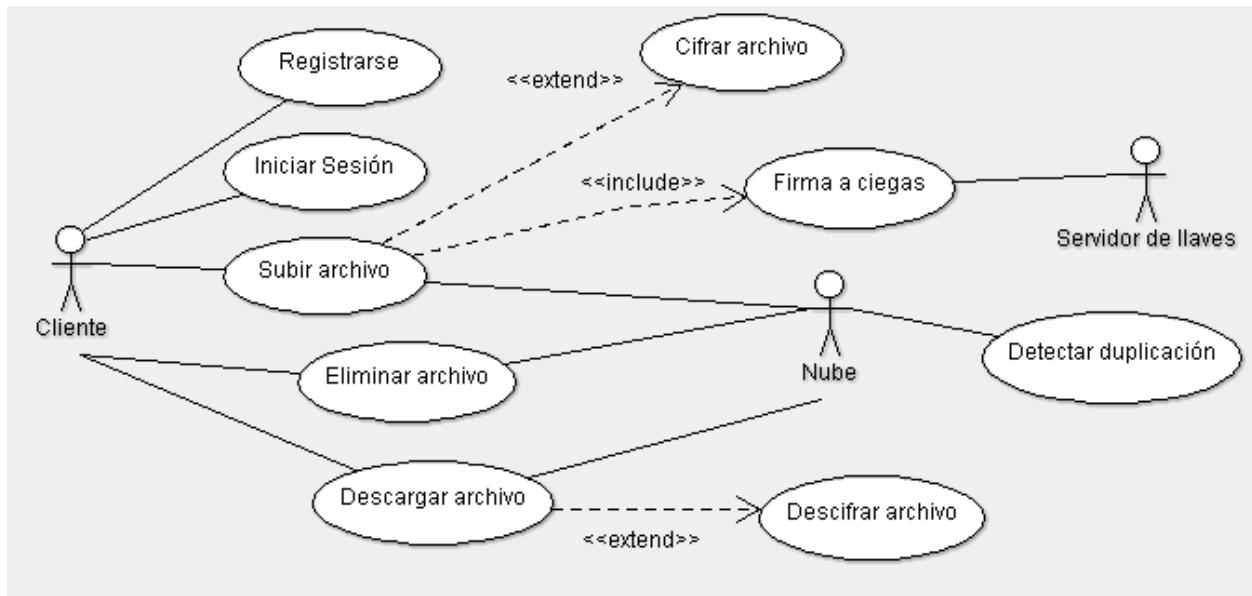


Figura 5.1: Diagrama de Casos de Uso del sistema.

5.2.1. CUSLL1 Generar las llaves del servidor de llaves

Descripción completa

El servidor de llaves realizará un proceso el cuál involucra la implementación de algoritmos criptográficos de clave pública, dichos algoritmos crearán la llave pública e y la llave privada d , la cuál servirá para la firma a ciegas de archivos que se almacenarán en la nube.

Atributos importantes

Caso de Uso:	CUSLL1 Generar las llaves del servidor de llaves
Versión:	1.0 - 15/04/17
Autor:	Eder Jonathan Aguirre Cruz
Prioridad:	Alta
Módulo:	Servidor de Llaves
Actor:	Servidor
Propósito:	Tener las llaves del servidor para poder comenzar el proceso de firma a ciegas de un archivo
Entradas:	
Salidas:	<ul style="list-style-type: none">▪ Llave pública e▪ Llave privada d
Precondiciones:	
Postcondiciones:	El servidor de llaves esta listo para realizar formas a ciegas de archivos a almacenar
Reglas del negocio:	
Mensajes:	<ul style="list-style-type: none">▪ MSG-SLL1 Generación de llaves

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1  Seleccionar dos números primos aleatorios. [Trayectoria A]
- 2  Encontrar el producto de esos números primos denominado N .
- 3  Calcular la función de euler $\varphi(N)$.

- 4 Elegir un número aleatorio e menor a $\varphi(N)$, tal que ese número sea $\gcd(e, \varphi(N)) = 1$. [Trayectoria B]
 - 5 Elegir un número aleatorio d , tal que cumpla con la congruencia $e \cdot d \equiv 1 \pmod{\varphi(N)}$. [Trayectoria C]
 - 6 Se generan las llaves pública e y d y muestra un mensaje MSG-SLL1 Generación de llaves
- - - - *Fin del caso de uso.*

Trayectoria alternativa A:

Condición: Numeros aleatorios iguales

- A1** Seleccionar números aleatorios iguales o no primos.
 - A2** Muestra el Mensaje MSG-SLL2 Números Iguales.
 - A3** Continúa en el paso 1 del CUSLL1.
- - - - *Fin de la trayectoria.*

Trayectoria alternativa B:

Condición: Número aleatorio menor

- B1** Elegir un número aleatorio menor al tamaño establecido de $\varphi(N)$.
 - B2** Muestra el Mensaje MSG-SLL3 Número incorrecto
 - B3** Continúa en el paso 4 del CUSLL1.
- - - - *Fin de la trayectoria.*

Trayectoria alternativa C:

Condición: Número aleatorio incorrecto

- C1** Elegir un número aleatorio incongruente con $e \cdot d \equiv 1 \pmod{\varphi(N)}$
 - C2** Muestra el Mensaje MSG-SLL3 Número incorrecto
 - C3** Continúa en el paso 5 del CUSLL1.
- - - - *Fin de la trayectoria.*

5.2.2. CUSLL2 Generar firma ciega (y).

Descripción completa

El servidor realizará una firma a ciegas de un archivo solicitado, este archivo ha sido oculto para que el servidor no sepa de donde proviene o que contiene, esta firma servirá para la generación de una llave para cifrar el archivo solicitado.

Atributos importantes

Caso de Uso: CUSLL2 Generar firma ciega (y).	
Versión:	1.0 - 16/04/17
Autor:	Diana Leslie González Olivier
Prioridad:	Alta
Módulo:	Servidor de Llaves
Actor:	Servidor
Propósito:	Que el servidor firme el archivo solicitado sin saber a que cliente corresponde.
Entradas:	Archivo oculto x
Salidas:	Firma a ciegas y
Precondiciones:	
Postcondiciones:	
Reglas del negocio:	
Mensajes:	

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1  Recibe el archivo oculto x .
 - 2  Firma el archivo generando un nuevo archivo y .
 - 3  Guarda en la base de datos el archivo y .
 - 4  Envía al cliente el archivo y .
- - - - *Fin del caso de uso.*

5.2.3. CUCL1 Subir archivo

Descripción completa

El usuario seleccionará el archivo que desea que sea almacenado en la nube, este archivo será cifrado y enviado de manera transparente para el usuario, y dependiendo si el archivo se detecta duplicado se enviarán distintos archivos.

Atributos importantes

Caso de Uso: CUCL1 Subir archivo	
Versión:	1.0 - 19/04/17
Autor:	Jhonatan Saulés Cortés
Prioridad:	Alta
Módulo:	Cliente
Actor:	Usuario
Propósito:	Almacenar un archivo en la nube, el cual debe estar cifrado para que no lo pueda entender el servicio de almacenamiento.
Entradas:	<ul style="list-style-type: none">■ Archivo a almacenar M■ Llave pública del servidor d
Salidas:	Archivo X a firmar por el servidor de llaves
Precondiciones:	El servidor de llaves debe tener asignada tanto su llave pública como llave privada.
Postcondiciones:	El archivo del usuario estará listo para ser firmado por el servidor de llaves.
Reglas del negocio:	
Mensajes:	<ul style="list-style-type: none">■ MSG-CL1 Carpeta vacía■ MSG-CL2 Archivo incomplatable■ MSG-CL3 Número incorrecto■ MSG-CL4 Error al generar la llave.■ MSG-CL5 Archivo almacenado.

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1 Da un clic en la opción *Subir Archivo* en la pantalla .
- 2 Despliega una ventana con la carpeta personal que muestra los archivos del usuario. [Trayectoria A]
- 3 Selecciona el archivo (M) que va a subir y da un clic en el botón en la pantalla . [Trayectoria B]
- 4 Elige un número aleatorio r dentro del campo de números primos de igual o menor tañano a las llaves del servidor de llaves. [Trayectoria C]
- 5 Calcula una función hash del archivo seleccionado $H(M)$.
- 6 Eleva el número aleatorio r a la potencia llave pública del servidor de llaves, r^e .
- 7 Multiplica $H(M)$ por r^e , $H(M) \cdot r^e$.
- 8 Obtiene Archivo X a firmar y lo envía al servidor de llaves.
- 9 Recibe la firma a ciegas Y del servidor.
- 10 Calcula el inverso multiplikcativo del numero aleatorio r .
- 11 Multiplica Y por r^{-1} , $Y \cdot r^{-1}$.
- 12 Verifica que k^e sea igual a $H(M)$. [Trayectoria D]
- 13 Obtiene llave k y la almacena, junto con el nombre del archivo al que le corresponde.
- 14 Cifra el archivo (M) con su llave k .
- 15 Obtiene el archivo $C1$.
- 16 Cifra el archivo k con su llave publica del usuario ka .
- 17 Obtiene el archivo $C2$.
- 18 Envia a la nube el hash del archivo $C1$, $H(C1)$.
- 19 Recibe solicitud de archivos. [Trayectoria E]
- 20 Envía los archivos $C1$ y $C2$.
- 21 Muestra el Mensaje MSG-CL5 Archivo almacenado.
- - - *Fin del caso de uso.*

Trayectoria alternativa A:

Condición: Archivos inexistentes

- A1 Despliega una ventana con la carpeta personal del usuario sin archivos existentes.
- A2 Muestra el Mensaje MSG-CL1 Carpeta vacía.
- A3 Termina el caso de uso.
- - - *Fin de la trayectoria.*

Trayectoria alternativa B:

Condición: Archivo incompatible

- B1 Selecciona el archivo (M) en un formato incompatible para el protocolo y su almacenamiento
- B2 Muestra el Mensaje MSG-CL2 Archivo incompliable.
- B3 Termina el caso de uso.
- - - *Fin de la trayectoria.*

Trayectoria alternativa C:

Condición: Número aleatorio incorrecto

- C1 Elegir un número aleatorio no primo o mayor al tamaño de las llaves del servidor de llaves.
- C2 Muestra el Mensaje MSG-CL3 Número incorrecto.
- C3 Continúa en el paso 4 del CUCL2.
- - - *Fin de la trayectoria.*

Trayectoria alternativa D:

Condición: Comparacion es diferente

- D1 Detecta que k^e y $H(M)$ son diferentes.
- D2 Muestra el Mensaje MSG-CL4 Error al generar la llave.
- D3 Continúa en el paso 4 del CUCL2.
- - - *Fin de la trayectoria.*

Trayectoria alternativa E:

Condición: Archivo duplicado

- E1 Detecta que el archivo $H(C1)$ ya ha sido almacenado.
- E2 Envía el archivo $C2$.
- E3 Continúa en el paso 21 del CUCL2.
- - - *Fin de la trayectoria.*

5.2.4. CUCL3 Descargar archivos.

Descripción completa

El cliente podrá descargar su archivo y descifrarlo.

Atributos importantes

Caso de Uso:	CUCL3 Descargar archivos.
Versión:	1.0 - 16/04/17
Autor:	Diana Leslie González Olivier
Prioridad:	Alta
Módulo:	Cliente
Actor:	Cliente
Propósito:	Que el cliente pueda obtener su archivo con texto en claro
Entradas:	C1 y C2
Salidas:	Archivo descargado
Precondiciones:	El archivo debe existir en la Nube
Postcondiciones:	
Reglas del negocio:	
Mensajes:	<ul style="list-style-type: none">■ MSG1 Operación exitosa■ MSG-CL6 Archivo inexistente

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1 Selecciona el archivo a descargar y da clic en la opcion de descargar archivo.
- 2 Envía a la nube una petición con el nombre del archivo que desea descargar.
- 3 Recibe los archivos *C1* y *C2* asociados al nombre que envió.
- 4 Descifra *C2* con la llave *Ka* del cliente.
- 5 Obtiene un archivo con la llave *K*.
- 6 Descifra *C1* con la llave *K*.
- 7 Obtiene su archivo *M* con su informacion visible.
- 8 Muestra el mensaje MSG1 Operación exitosa.
- - - Fin del caso de uso.

Trayectoria: Trayectoria Alternativa

- 1  Envía a la nube una petición con el nombre del archivo que desea descargar.
- 2  Muestra el mensaje MSG-CL4 Archivo inexistente.

- - - - *Fin del caso de uso.*

5.2.5. CUCL4 Eliminar archivos cifrado.

Descripción completa

El cliente podrá elegir la opción de eliminar un archivo cifrado del servicio de almacenamiento en la nube.

Atributos importantes

Caso de Uso: CUCL4 Eliminar archivos cifrado.	
Versión:	1.0 - 16/04/17
Autor:	Diana Leslie González Olivier
Prioridad:	Alta
Módulo:	Cliente
Actor:	Cliente
Propósito:	Que el cliente pueda eliminar un archivo
Entradas:	
Salidas:	Archivo eliminado
Precondiciones:	El archivo debe existir en la Nube
Postcondiciones:	
Reglas del negocio:	
Mensajes:	<ul style="list-style-type: none">■ MSG1 Operación exitosa

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1 El cliente da clic en el botón eliminar archivo.
- 2 El sistema despliega la pantalla para seleccionar el archivo que se desea eliminar.
- 3 El cliente selecciona el archivo que desea eliminar.
- 4 El sistema recibe petición para eliminar archivo.
- 5 El sistema busca el nombre de archivo asociado en su base de datos.[Trayectoria A]
- 6 El sistema elimina C1 y C2.
- 7 El sistema despliega la lista de usuarios en la base de datos.
- 8 Muestra el mensaje MSG1 Operación exitosa.
- - - - Fin del caso de uso.

Trayectoria alternativa A:

Condición: Archivo inexistente

A1  El cliente da clic en el botón .

A2  El sistema despliega la pantalla principal.

- - - - *Fin de la trayectoria.*

5.2.6. CUCL6 Iniciar Sesión.

Descripción completa

Permitir el acceso al sistema con su usuario y contraseña correspondientes, el cual es autenticado y autorizado para la utilización del sistema.

Atributos importantes

Caso de Uso: CUCL6 Iniciar Sesión.	
Versión:	1.0 - 19/04/17
Autor:	Jhonatan Saulés Cortés.
Prioridad:	Alta
Módulo:	Cliente
Actor:	Cliente
Propósito:	Dar acceso al usuario al sistema para poder realizar sus actividades.
Entradas:	Nombre de usuario, Contraseña.
Salidas:	Página principal del usuario que inicio sesión
Precondiciones:	Estar registrado en el sistema.
Postcondiciones:	
Reglas del negocio:	<ul style="list-style-type: none">▪ RN4 Usuario registrado
Mensajes:	<ul style="list-style-type: none">▪ MSG1 Operación exitosa.▪ MSG5 Dato incorrecto.▪ MSG6 Longitud inválida.▪ MSG9 Dato requerido.▪ MSG10 No existe información.▪ MSG11 Contraseña incorrecta

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1  Da clic en la opción *Iniciar sesión*.
- 2  Despliega los campos para introducir nombre de usuario y contraseña.

- 3 Ingresa su nombre de usuario y contraseña en los campos mostrados.
- 4 Da clic en el botón *Ingresar*.
- 5 Autentica y autoriza el nombre usuario y contraseña con base en la regla de negocio RN4 Usuario registrado. [Trayectoria A] [Trayectoria B] [Trayectoria C] [Trayectoria D] [Trayectoria E]
- 6 Solicita las llaves privada y pública del usuario.
- 7 Almacena las llaves en el dispositivo actual.
- 8 Muestra el mensaje MSG1 Operación exitosa.
- 9 Muestra el menú principal del usuario.
- 10 Fin del caso de uso.
- - - - *Fin del caso de uso.*

Trayectoria alternativa A:

Condición: Datos incorrectos

- A1 Muestra el mensaje MSG5 Dato incorrecto.
- A2 Da clic en el botón *Cerrar*.
- A3 Continúa en el paso 3 del CUCL6
- - - - *Fin de la trayectoria.*

Trayectoria alternativa B:

Condición: Longitud inválida

- B1 Muestra el mensaje MSG6 Longitud inválida.
- B2 Da clic en el botón *Cerrar*.
- B3 Continúa en el paso 3 del CUCL6
- - - - *Fin de la trayectoria.*

Trayectoria alternativa C:

Condición: Datos requeridos

- C1 Muestra el mensaje MSG9 Dato requerido.
- C2 Da clic en el botón *Cerrar*.
- C3 Continúa en el paso 3 del CUCL6
- - - - *Fin de la trayectoria.*

Trayectoria alternativa D:

Condición: No existe información

- D1 Muestra el mensaje MSG10 No existe información.

D2  Da clic en el botón *Cerrar*.

D3  Continúa en el paso 3 del CUCL6

- - - *Fin de la trayectoria.*

Trayectoria alternativa E:

Condición: Contraseña incorrecta

E1  Muestra el mensaje MSG11 Contraseña incorrecta

E2  Da clic en el botón *Cerrar*.

E3  Continúa en el paso 3 del CUCL6

- - - *Fin de la trayectoria.*

5.2.7. CUCL7 Registrar usuario.

Descripción completa

Solicitar los datos importantes de un usuario nuevo, generar sus llaves pública y privada para darlo de alta en el sistema.

Atributos importantes

Caso de Uso: CUCL7 Registrar usuario.	
Versión:	1.0 - 19/04/17
Autor:	Jhonatan Saulés Cortés.
Prioridad:	Alta
Módulo:	Cliente
Actor:	Cliente
Propósito:	Habilitar un nuevo usuario generandole sus llaves privada y pública.
Entradas:	Datos del usuario.
Salidas:	Llaves del usuario
Precondiciones:	
Postcondiciones:	
Reglas del negocio:	<ul style="list-style-type: none">■ RN1 Datos requeridos■ RN2 Datos correctos■ RN3 Unicidad de elementos
Mensajes:	<ul style="list-style-type: none">■ MSG1 Operación exitosa.■ MSG4 Registro repetido■ MSG5 Dato incorrecto.■ MSG6 Longitud inválida.■ MSG9 Dato requerido.

Trayectorias del Caso de Uso

Trayectoria: Principal

- 1  Da clic en la opción *Registrarse*.

- 2 Despliega los campos para introducir nombre, primer apellido, segundo apellido, nombre de usuario, contraseña, fecha de nacimiento, CURP.
- 3 Ingresa sus datos que han sido solicitados.
- 4 Da clic en el botón *Registrar*.
- 5 Verifica los datos proporcionados por el usuario con base en las reglas de negocios RN1 Datos requeridos, RN2 Datos correctos, RN3 Unicidad de elementos . [Trayectoria A] [Trayectoria B] [Trayectoria C] [Trayectoria D]
- 6 Genera su llave privada y pública del usuario con RSA.
- 7 Almacena sus llaves en el servidor y en el dispositivo actual.
- 8 Muestra el mensaje MSG1 Operación exitosa.
- 9 Muestra el menú principal del usuario.
- 10 Fin del caso de uso.
- - - - *Fin del caso de uso.*

Trayectoria alternativa A:

Condición: Datos incorrectos

- A1 Muestra el mensaje MSG5 Dato incorrecto.
- A2 Da clic en el botón *Cerrar*.
- A3 Continúa en el paso 3 del CUCL7
- - - - *Fin de la trayectoria.*

Trayectoria alternativa B:

Condición: Longitud inválida

- B1 Muestra el mensaje MSG6 Longitud inválida.
- B2 Da clic en el botón *Cerrar*.
- B3 Continúa en el paso 3 del CUCL7
- - - - *Fin de la trayectoria.*

Trayectoria alternativa C:

Condición: Datos requeridos

- C1 Muestra el mensaje MSG9 Dato requerido.
- C2 Da clic en el botón *Cerrar*.
- C3 Continúa en el paso 3 del CUCL7
- - - - *Fin de la trayectoria.*

Trayectoria alternativa D:

Condición: Registro repetido

- D1 Muestra el mensaje MSG4 Registro repetido
- D2 Da clic en el botón *Cerrar*.
- D3 Continúa en el paso 3 del CUCL7
- - - - *Fin de la trayectoria.*

5.3. Diagramas de secuencia

5.3.1. Registrar Usuario

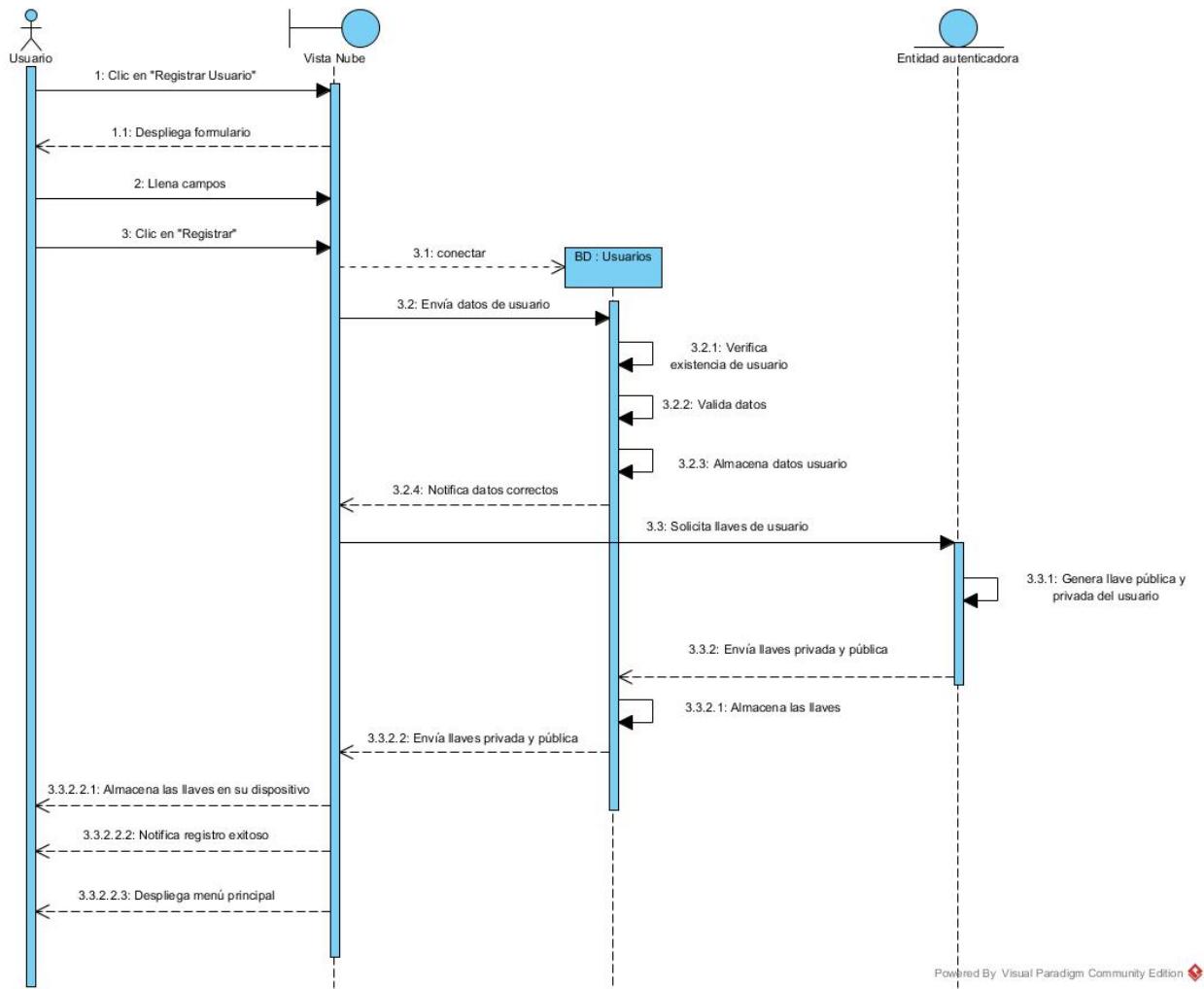


Figura 5.2: Diagrama de secuencias de Registrar un usuario nuevo.

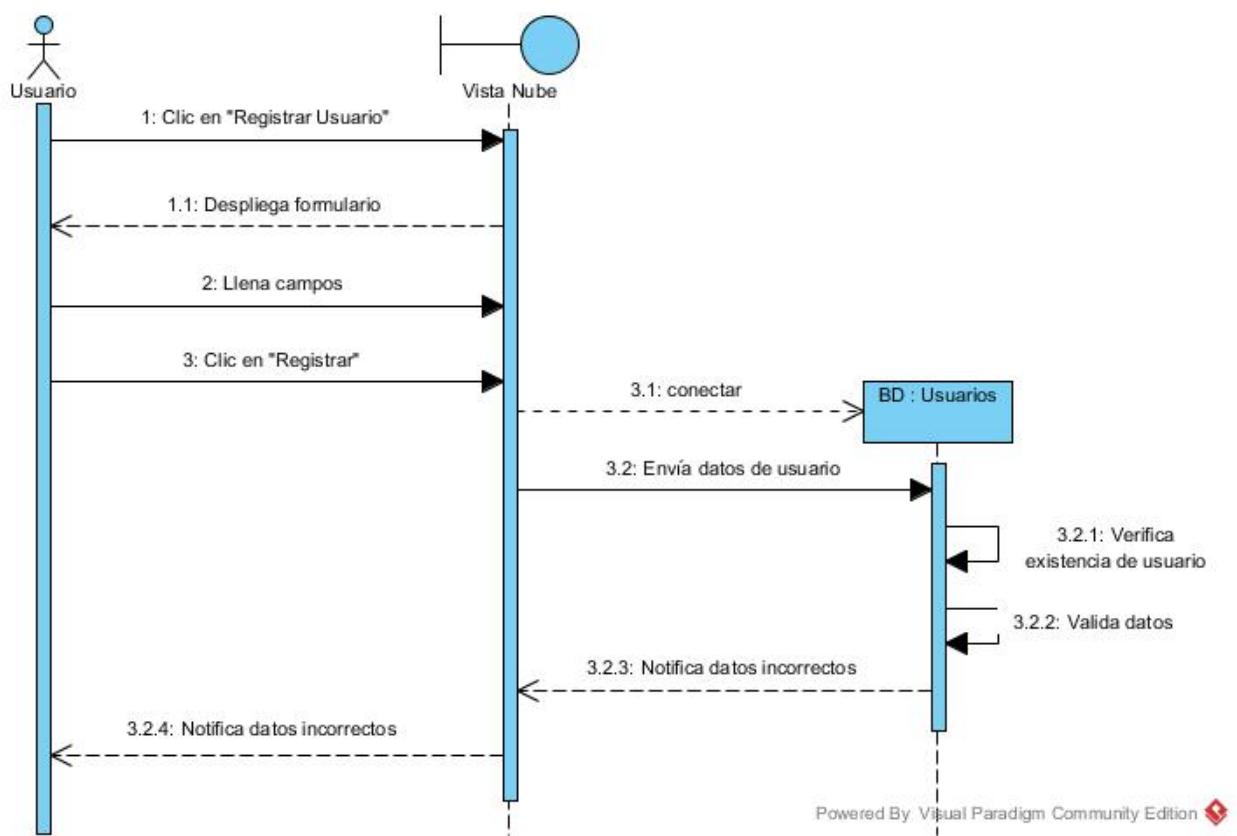


Figura 5.3: Diagrama de secuencias de Registrar un usuario nuevo con datos incorrectos.

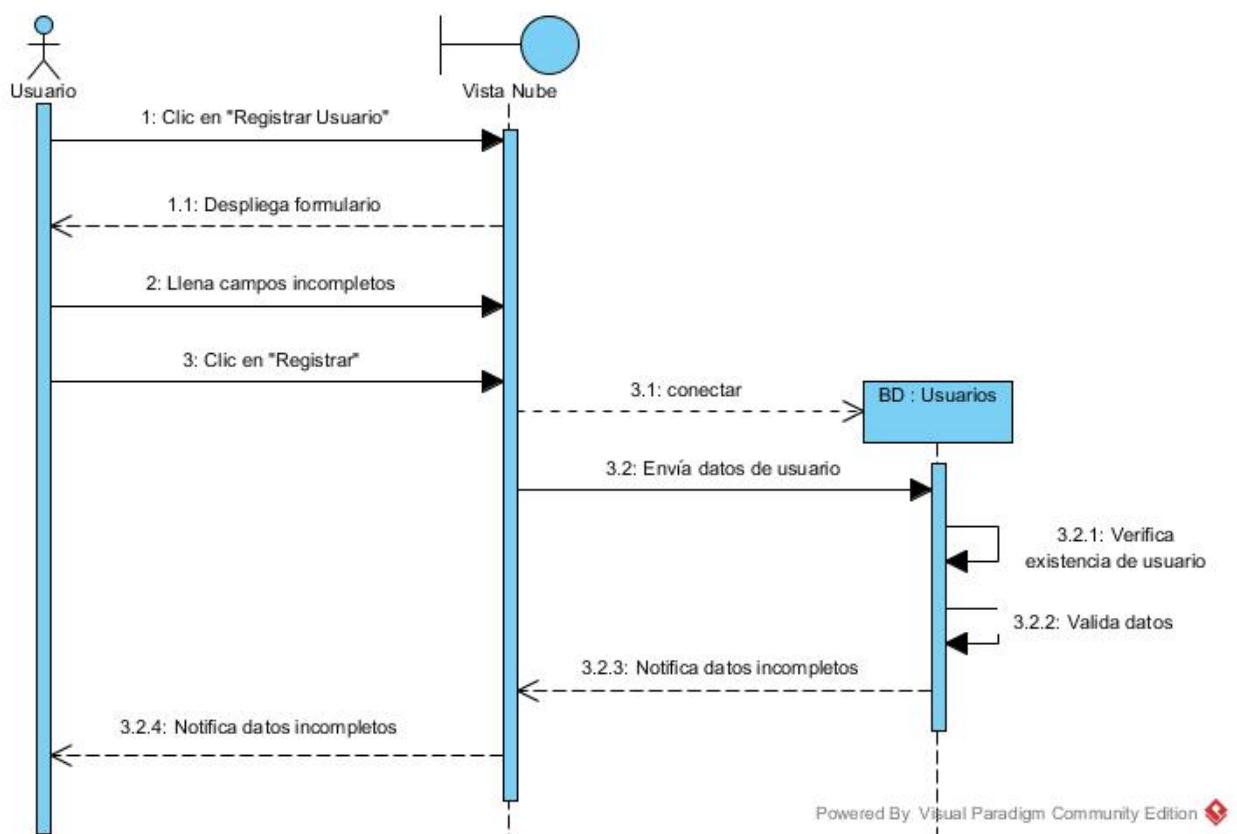


Figura 5.4: Diagrama de secuencias de Registrar un usuario nuevo con datos incompletos.

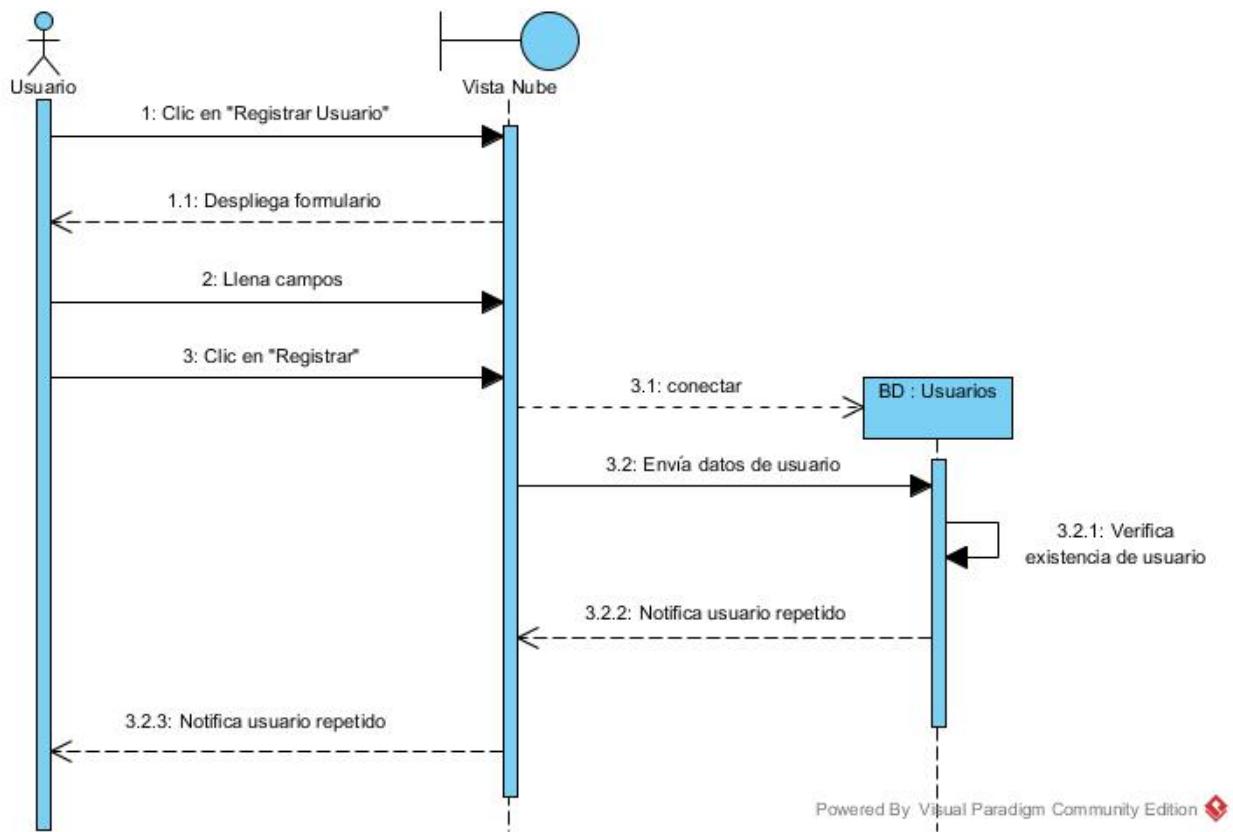


Figura 5.5: Diagrama de secuencias de Registrar un usuario nuevo con usuario repetido.

5.3.2. Iniciar Sesión

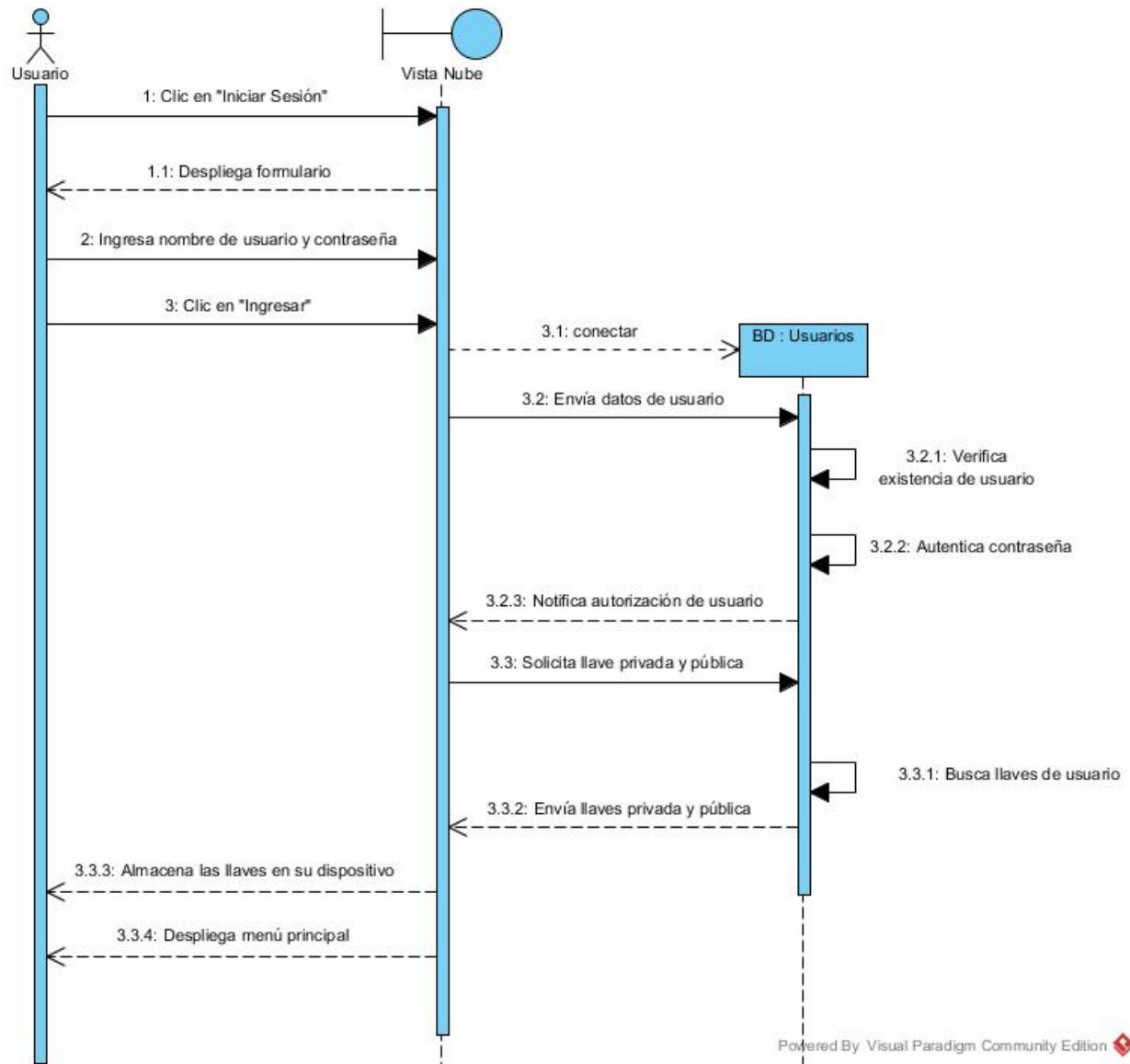


Figura 5.6: Diagrama de secuencias de Iniciar sesión un usuario.

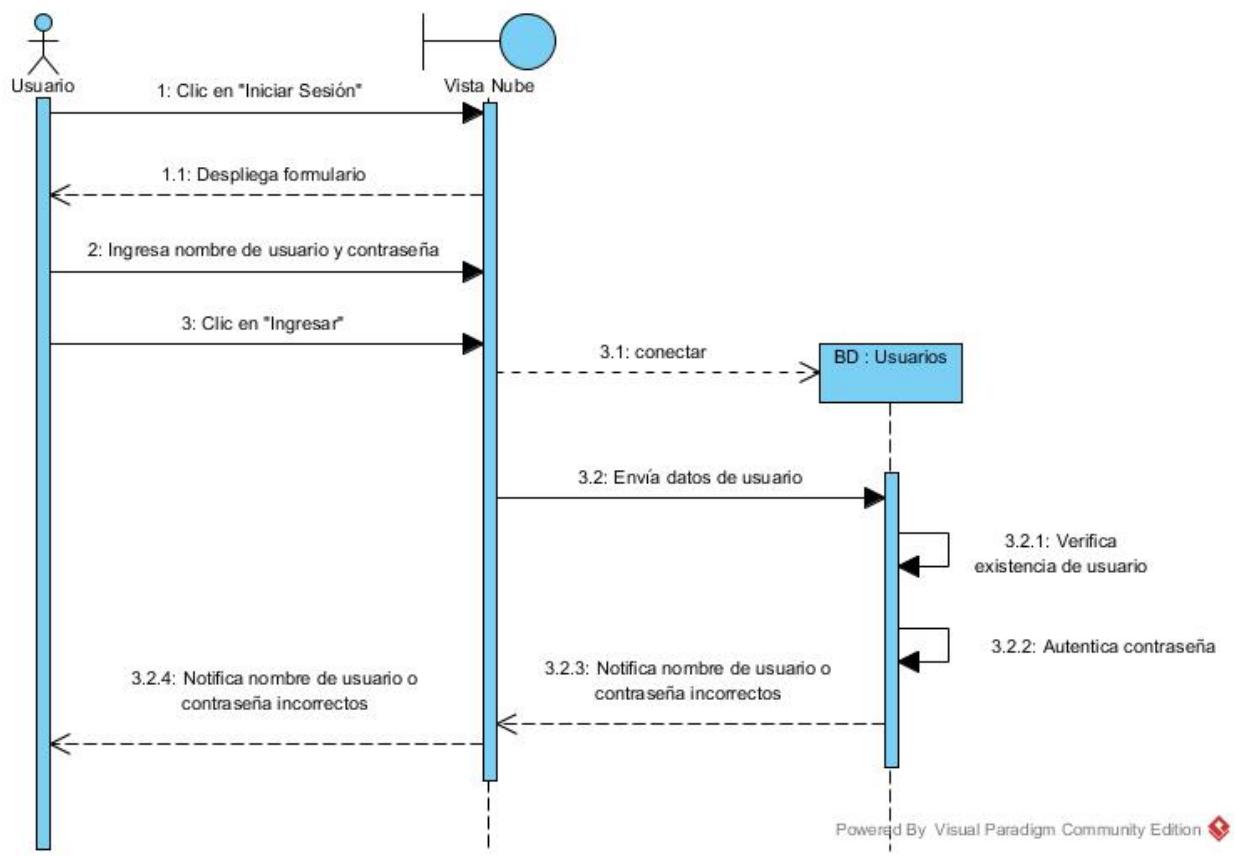


Figura 5.7: Diagrama de secuencias de Iniciar sesión un usuario con datos incorrectos.

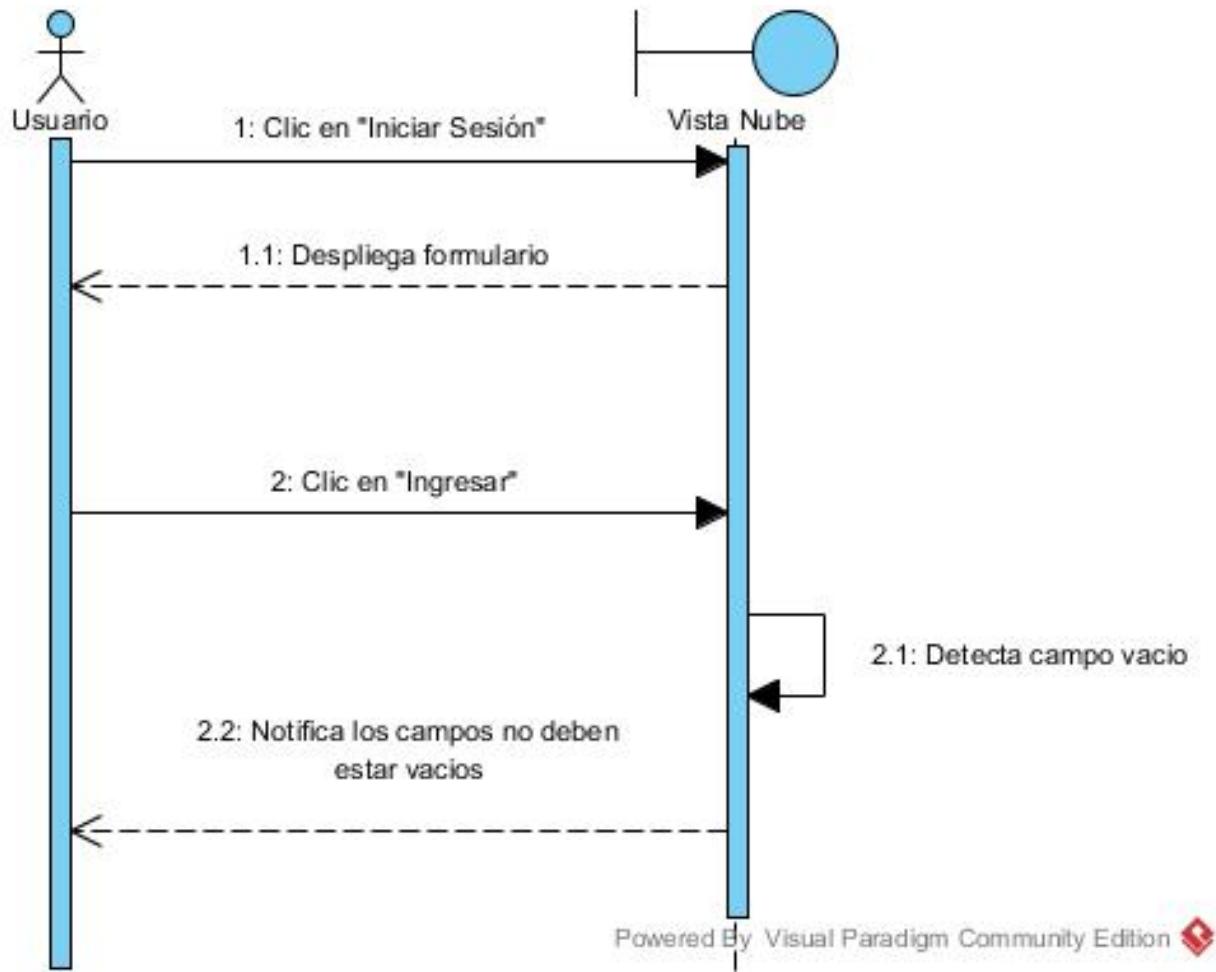


Figura 5.8: Diagrama de secuencias de Registrar un usuario nuevo con datos incompletos.

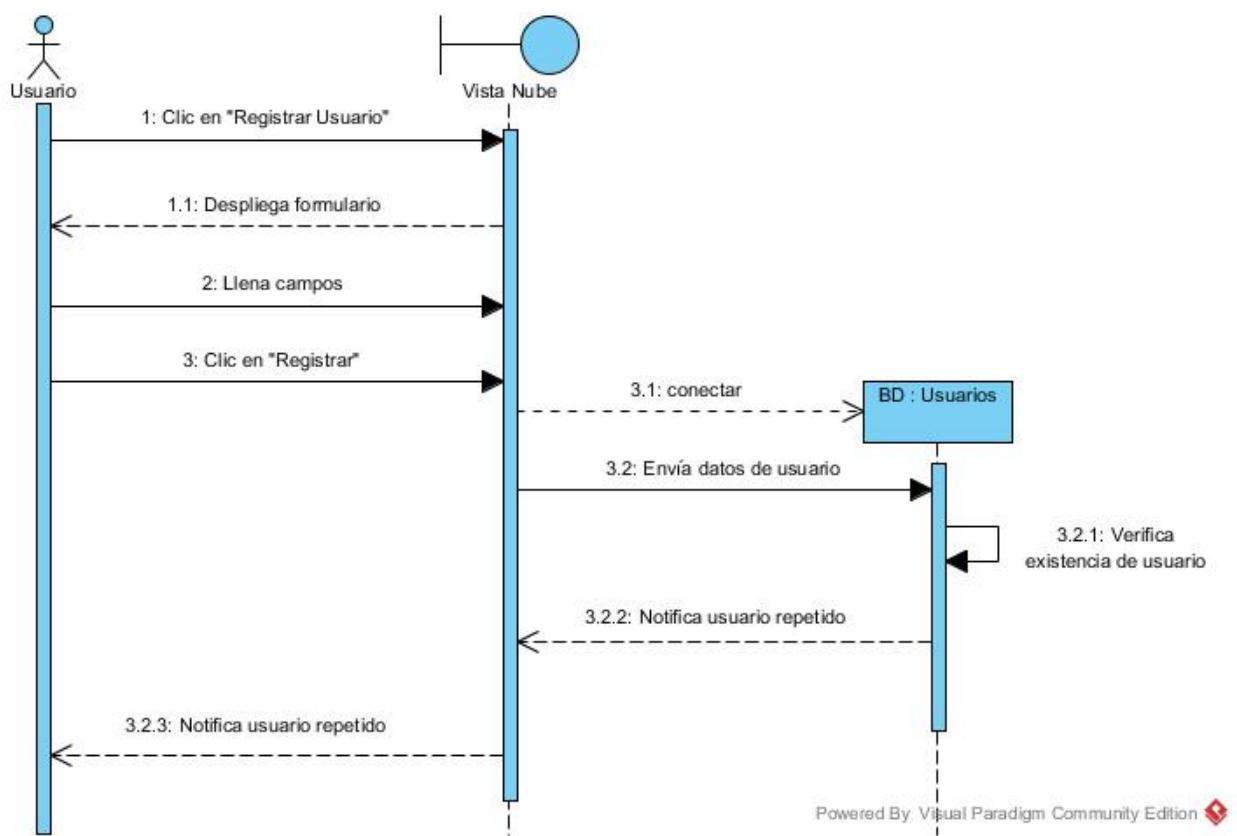


Figura 5.9: Diagrama de secuencias de Iniciar sesión un usuario con usuario repetido.

5.3.3. Subir Archivo

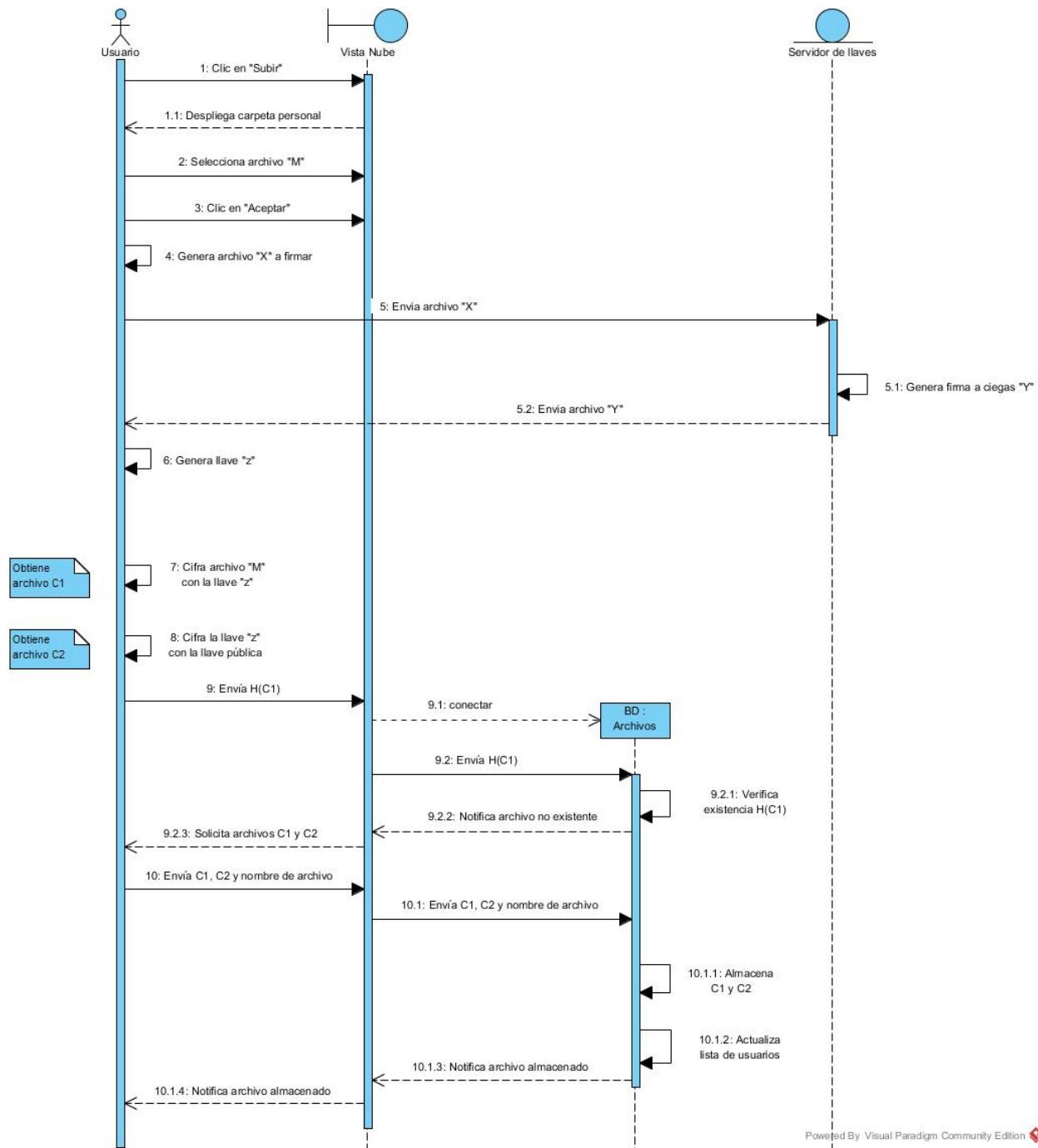


Figura 5.10: Diagrama de secuencias de subir un archivo nuevo.

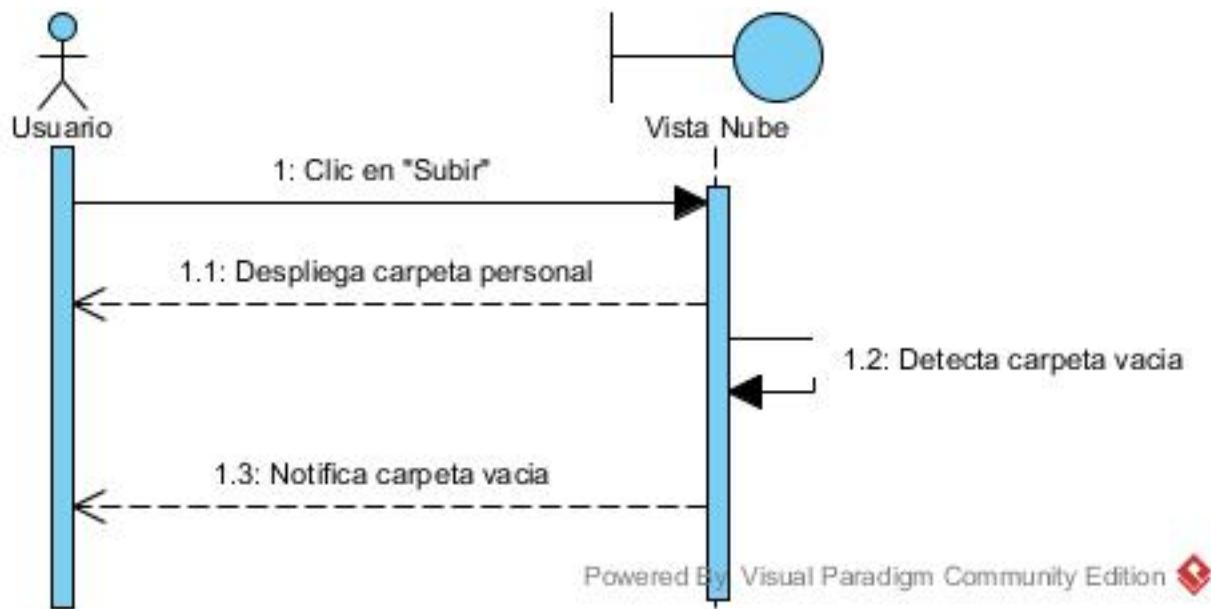


Figura 5.11: Diagrama de secuencias de subir un archivo con carpeta vacía.

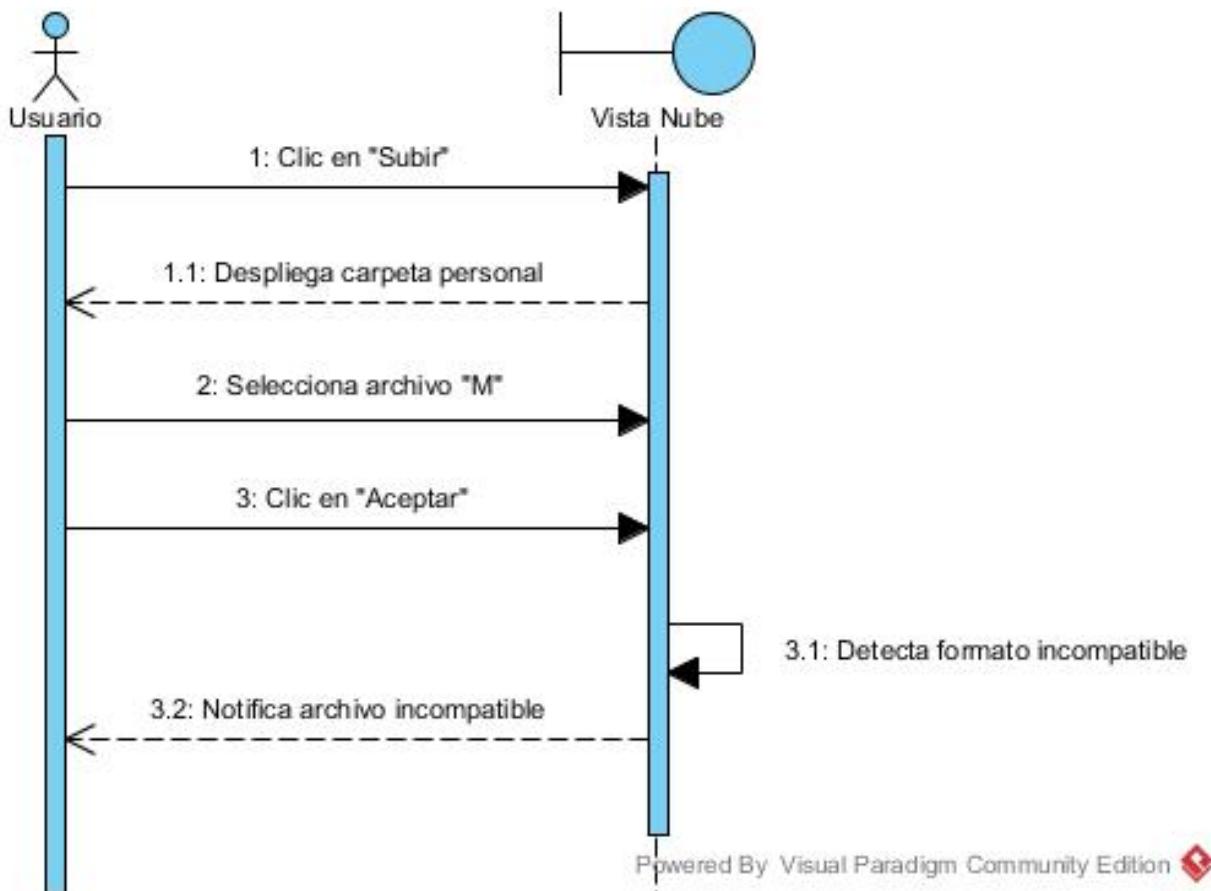


Figura 5.12: Diagrama de secuencias de subir un archivo incompatible.

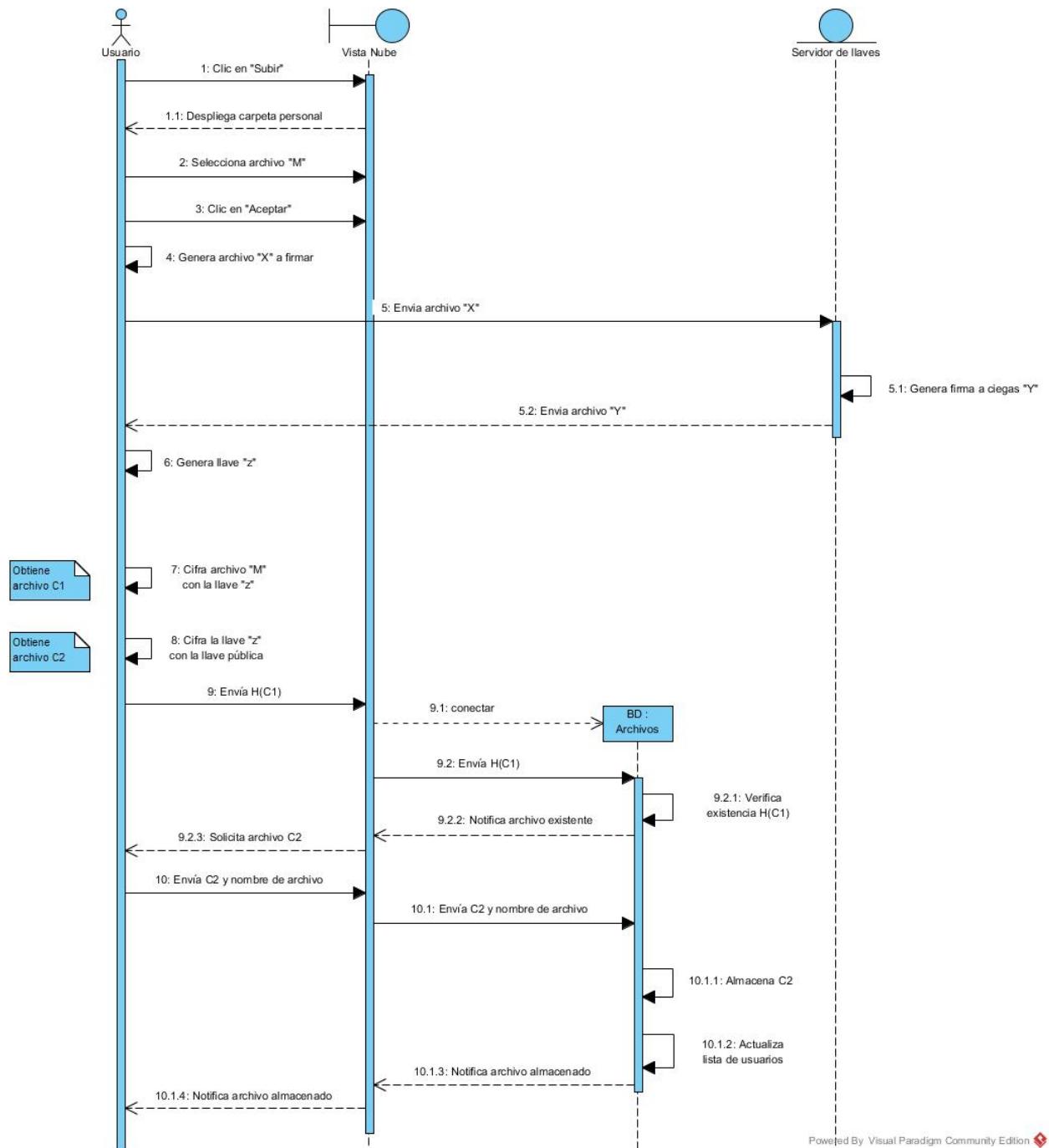


Figura 5.13: Diagrama de secuencias de subir un archivo existente.

5.3.4. Firma a ciegas

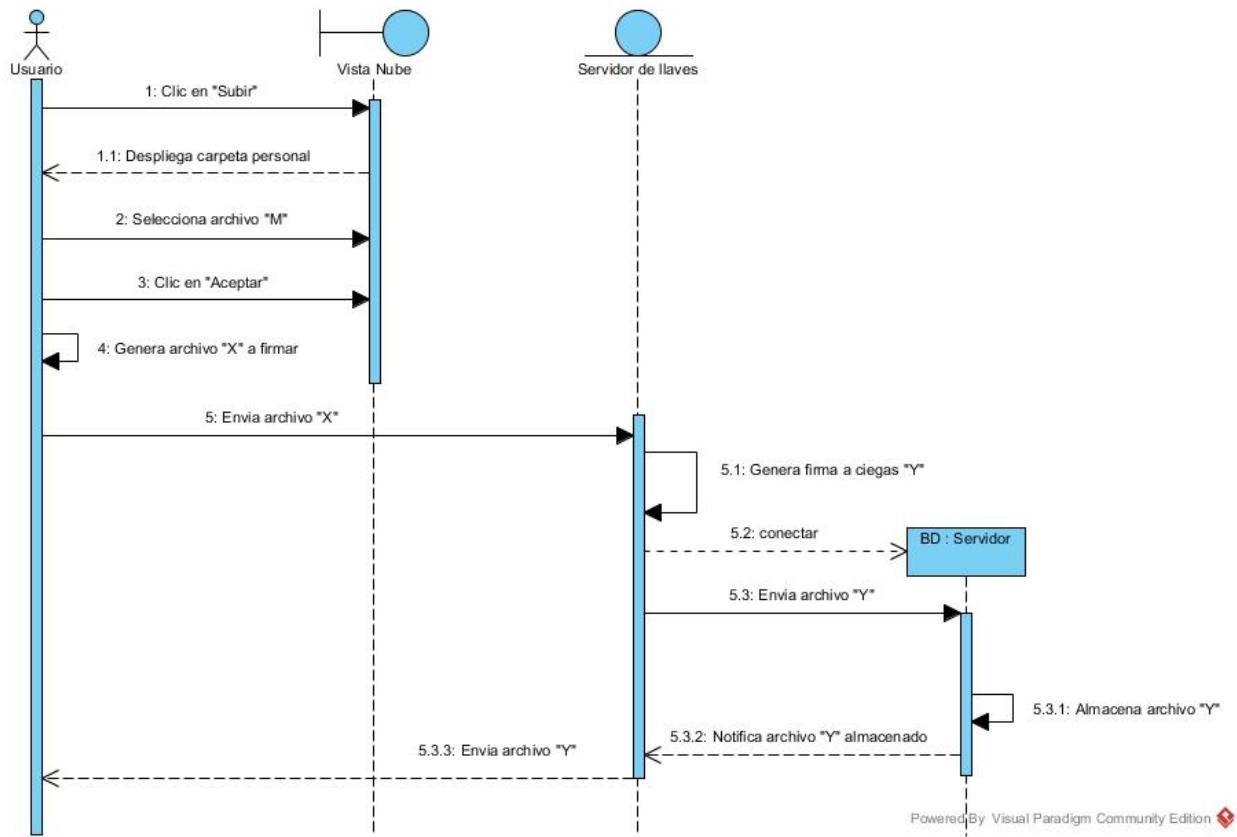


Figura 5.14: Diagrama de secuencias de la firma a ciegas del servidor de llaves.

5.3.5. Descargar Archivo

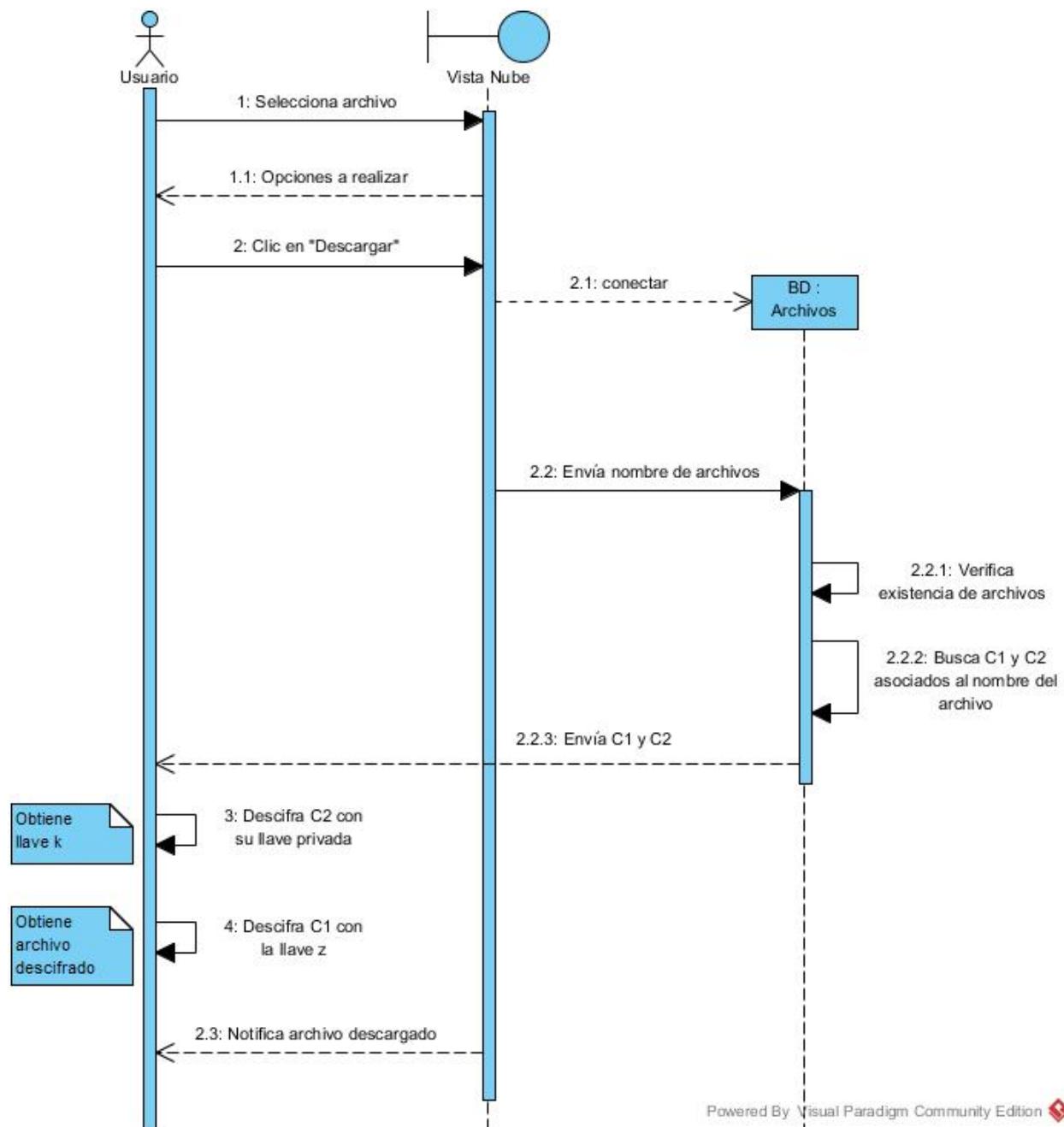


Figura 5.15: Diagrama de secuencias de descargar un archivo de la nube.

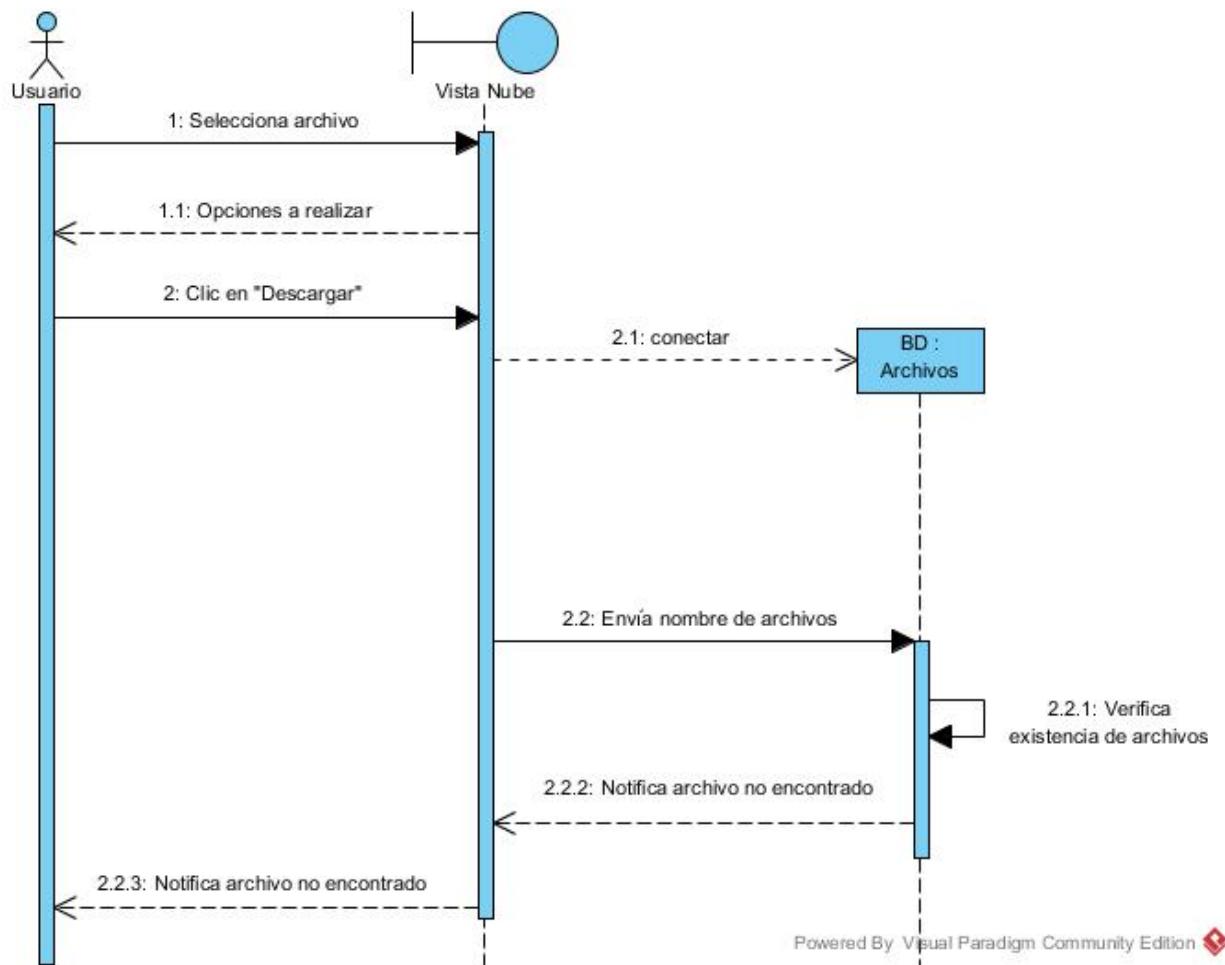


Figura 5.16: Diagrama de secuencias de descargar un archivo de la nube no encontrado.

5.3.6. Eliminar Archivo

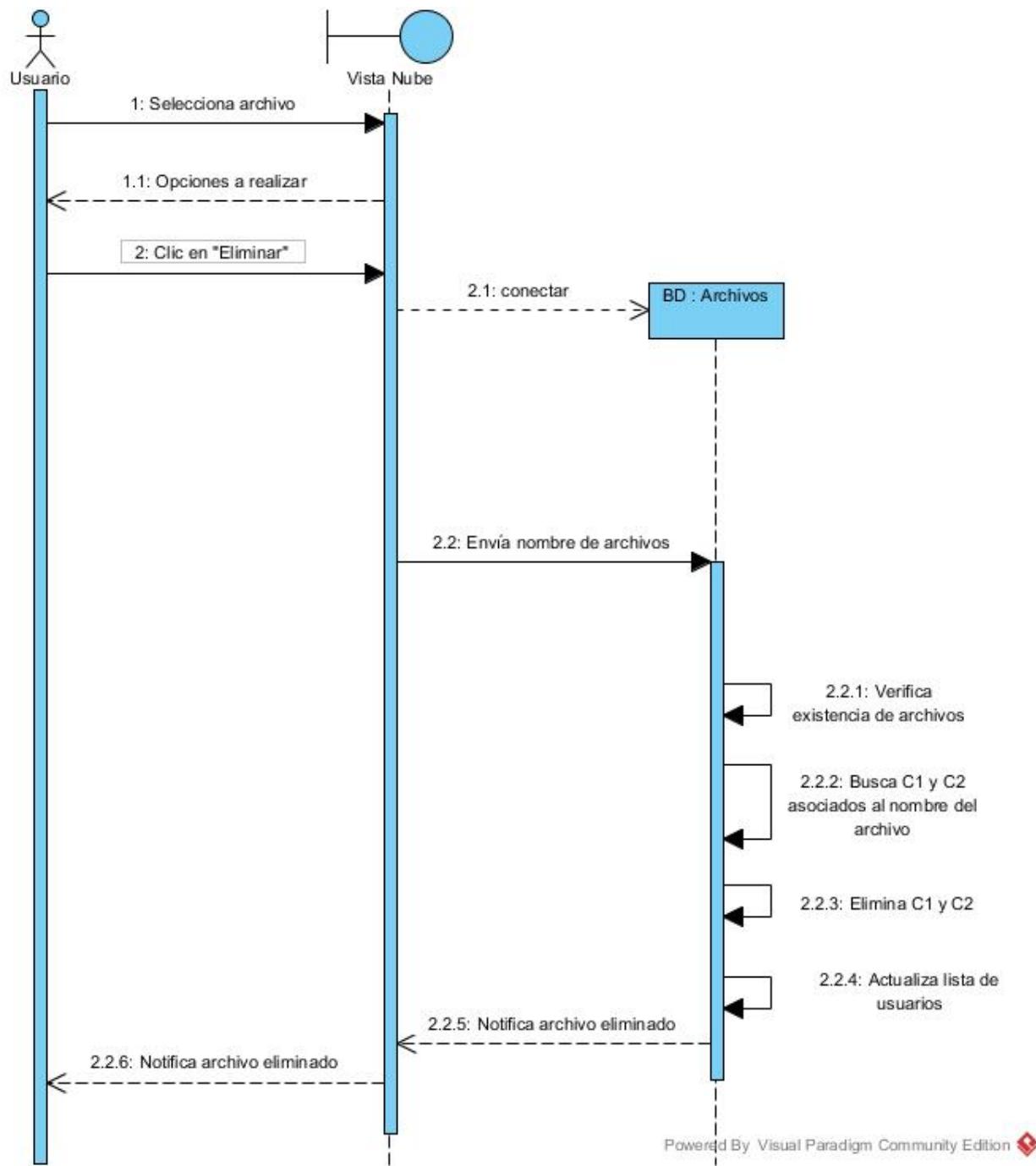


Figura 5.17: Diagrama de secuencias de eliminar un archivo de la nube.

5.4. Mensajes del sistema

Mensajes usados en el sistema, que se usan para informar al usuario mediante la interfaz de ciertas situaciones o eventos que ocurren en el prototipo y pueden ser de los siguientes tipos:

- **Notificación** : Estos mensajes se utilizan para indicar que la operación solicitada por el usuario se ejecutó correctamente.
- **Alerta** : Estos mensajes se utilizan para indicar alguna advertencia sobre la operación.
- **Error** : Estos mensajes se utilizan para indicar que ha ocurrido un error en la operación solicitada.

Varios mensajes se encuentran parametrizados. Es decir cuando algún mensaje es recurrente, hay palabras que pueden ser sustituidas por otras para transformar el mensaje a la situación.

Parámetros más comunes:

ARTÍCULO: Se refiere a un artículo el cual puede ser DETERMINADO (El | La | Lo | Los | Las) o INDETERMINADO (Un | Una | Uno | Unos | Unas) se aplica generalmente sobre una ENTIDAD, ATRIBUTO o VALOR.

CAMPO: Se refiere a un campo del formulario. Por lo regular es el nombre de un atributo en una entidad.

CAUSA: Un razón por lo que la operación aconteció de cierta manera.

ENTIDAD: Es un sustantivo y generalmente se refiere a una entidad del modelo estructural del negocio.

OPERACIÓN: Se refiere a una acción que se debe realizar sobre los datos de una o varias entidades. Por ejemplo: registrar, eliminar, modificar, etc.

RESTRICCIÓN: Se refiere a alguna restricción para un tipo de dato. Por ejemplo: máximo, mínimo, etc.

TAMAÑO: Es el tamaño del atributo de una entidad, el cual se encuentra definido en el modelo conceptual.

VALOR: Es un sustantivo concreto y generalmente se refiere a un valor en específico.

5.4.1. Mensajes

Mensaje: MSG-SLL1 Generación de llaves

Tipo: Notificación

Objetivo: Notificar al actor que la operación se ha realizado de forma exitosa.

Redacción: La generación exitosa de llaves del servidor.

Ejemplo:

Mensaje: MSG-SLL2 Números iguales

Tipo: Error

Objetivo: Notificar al actor que los números aleatorios elegidos se encuentran repetidos.

Redacción: Los números aleatorios se encuentran repetidos o no son primos.

Ejemplo:

Mensaje: MSG-SLL3 Número incorrecto

Tipo: Error

Objetivo: Notificar al actor que el número elegido no cumple con las características específicas.

Redacción: El número aleatorio no cumple con las especificaciones.

Ejemplo:

Mensaje: MSG-N1 Archivo no encontrado

Tipo: Notificación

Objetivo: Notificar al usuario que el archivo enviado no existe almacenado en la nube.

Redacción: El archivo solicitado no existe.

Ejemplo:

Mensaje: MSG-CL1 Carpeta vacía

Tipo: Notificación

Objetivo: Notificar al usuario que su carpeta personal no tiene archivos

Redacción: La carpeta personal se encuentra sin archivos.

Ejemplo:

Mensaje: MSG-CL2 Archivo incompatible

Tipo: Error

Objetivo: Notificar al usuario que el archivo que intenta subir no es válido

Redacción: El archivo no es compatible con el almacenamiento.

Ejemplo:

Mensaje: MSG-CL3 Número incorrecto

Tipo: Notificación

Objetivo: Notificar al actor que el número aleatorio no esta dentro del rango de tamaño.

Redacción: El número aleatorio no se encuentra dentro del rango establecido.

Ejemplo:

Mensaje: MSG-CL4 Error al generar la llave

Tipo: Notificación

Objetivo: Notificar al actor que la operación se ha realizado de forma exitosa.

Redacción:

Ejemplo:

Mensaje: MSG-CL5 Archivo almacenado

Tipo: Notificación

Objetivo: Notificar al actor que la operación se ha realizado de forma exitosa.

Redacción:

Ejemplo:

Mensaje: MSG-CL6 Archivo inexistente

Tipo: Notificación

Objetivo: Notificar al actor que la operación se ha realizado de forma exitosa.

Redacción:

Ejemplo:

Mensaje: MSG1 Operación exitosa

Tipo: Notificación

Objetivo: Notificar al actor que la operación se ha realizado de forma exitosa.

Redacción: DETERMINADO ENTIDAD ha sido OPERACIÓN exitosamente.

Parámetros: El mensaje se muestra con base en los siguientes parámetros:

- DETERMINADO ENTIDAD: Artículo determinado más el nombre de la entidad sobre la que se realiza la operación.
- OPERACIÓN: Es la acción que el actor solicitó realizar. Puede ser registro, eliminación,modificación o revisión.

Ejemplo: El Cliente ha sido registrado exitosamente.

Mensaje: MSG4 Registro repetido

Tipo: Error

Objetivo: Notificar al actor que la entidad que desea registrar ya existe en el sistema.

Redacción: DETERMINADO ENTIDAD que intentas registrar ya existe.

Parámetros: El mensaje se muestra con base en los siguientes parámetros:

- DETERMINADO ENTIDAD: Artículo determinado más el nombre de la entidad sobre la que se realiza la operación.

Ejemplo: El Cliente que intentas registrar ya existe.

Mensaje: MSG5 Dato incorrecto

Tipo: Error

Objetivo: Notificar al actor que el dato no tiene el tipo solicitado.

Redacción: DETERMINADO ENTIDAD debe ser INDETERMINADO TIPODATO.

Parámetros: El mensaje se muestra con base en los siguientes parámetros:

- DETERMINADO ENTIDAD: Artículo determinado más el nombre de la entidad sobre la que se realiza la operación.
- INDETERMINADO: Artículo indeterminado.
- TIPODATO: Indica el tipo de dato, por ejemplo cadena o número.

Ejemplo: El dato debe ser un número.

Mensaje: MSG6 Longitud inválida

Tipo: Error

Objetivo: Notificar al actor que el dato no tiene la longitud correcta.

Redacción: DETERMINADO ENTIDAD debe tener RESTRICCIÓN TAMAÑO TIPO-DATO.

Parámetros: El mensaje se muestra con base en los siguientes parámetros:

- DETERMINADO ENTIDAD: Artículo determinado más el nombre de la entidad sobre la que se realiza la operación.
- RESTRICCIÓN: Puede ser máximo, al menos,, mínimo, etc.
- TAMAÑO: Tamaño del dato.
- TIPODATO: Indica el tipo de dato con el que se mide el campo.

Ejemplo: La contraseña debe tener mínimo 6 caracteres.

Mensaje: MSG9 Dato requerido

Tipo: Error

Objetivo: Notificar al actor que el dato es requerido y se ha omitido.

Redacción: Este dato es requerido.

Mensaje: MSG10 No existe información

Tipo: Error

Objetivo: Notificar al actor que aún no existe información registrada en el prototipo.

Redacción: DETERMINADO ENTIDAD no se encuentra en el sistema.

Mensaje: MSG11 Contraseña incorrecta

Tipo: Error

Objetivo: Notificar al actor que la contraseña que introdujo no fue correcta.

Redacción: No es correcta su contraseña.

Capítulo 6

Desarrollo

6.1. Descripción General del Desarrollo del Protocolo

El protocolo criptográfico para evitar duplicados almacenados en la nube, es un proyecto que involucra la tecnología de software para ofrecer un funcionamiento eficiente y útil para las necesidades de los usuarios que se encuentran inmersos en el cómputo nube. Nuestro protocolo, se compone a su vez de diferentes módulos. Cada uno de ellos busca satisfacer:

- La seguridad de los archivos de los usuarios en la nube
- Almacenamiento seguro en la sube
- Conexión de diversos usuarios
- Ahorro en el consumo de espacio ofrecido en la nube
- Fácil acceso al almacenamiento de los archivos de los usuarios

Para lograrlo, el protocolo criptográfico se compone de diversos módulos que a su vez integran diferentes aplicaciones criptográficas. Dichos módulos serán detallados en las próximas secciones.

6.2. Servidor de Llaves

El módulo Servidor de llaves tiene una función muy importante dentro del funcionamiento del protocolo criptográfico, ya que sin dicho servidor los usuarios no podrían conectarse para poder utilizar el servicio de almacenamiento seguro.

Objetivo

- Generar llaves certificadas para cifrar los archivos.
- Brindar seguridad ante ataques por fuerza bruta.

Entradas

- El factor de ocultamiento (Proporcionado por el cliente)
- La clave privada del servidor d , generada mediante el algoritmo RSA.

El desarrollo de éste módulo, fué realizado mediante el lenguaje de programación Python en su versión 2.7.

Las librerías utilizadas para llevar acabo la implementación de éste módulo son:

```
import SocketServer
import threading
import time
from rsagen import *
```

Para crear el socket utilizado en el servidor, ocupamos el modulo **socket** de Python, el cuál importamos desdse la librería **SocketServer**, dicho módulo simplifica la tarea de escribir servidores de red montados en el lenguaje de programación python.

La librería **threading** construye interfaces de subprocesamiento de nivel superior en la parte superior del thread.

La librería **time** disponible en Python proporciona funciones para trabajar con los tiempos y para convertir representaciones.

Finalmente, **rsagen** es una implementación de Python RSA pura. Es compatible con el cifrado, el descifrado, la firma, verificación de firmas y la generación de claves de acuerdo con PKCS # 1 versión 1.5. Se puede utilizar como una biblioteca de Python, así como en la línea de comandos.

Una vez que importamos las bibliotecas pertinentes, se generan las llaves del servidor, tanto la pública (e), como la privada (d), para esto usamos el método *gen_rsa(usuario)*, es decir utilizamos RSA para la generación de llaves y para ello hacemos uso de *rsagen*, donde ya viene la implementación de dicho algoritmo.

Posteriormente leemos el archivo n y la llave privada d del servidor.

```
usuario = 'server'
gen_rsa(usuario)
n = int(open("key_n_server.PEM", "r").read())
d = int((open("key_d_server.PEM", "r").read()))
```

Para continuar, creamos un TCP Handler el cual utiliza el protocolo TCP de Internet, que proporciona transmisiones continuas de datos entre el cliente y el servidor.

```
#creo mi TCP Handler
class MiTcpHandler(SocketServer.BaseRequestHandler):
```

Ya que tenemos la conexión entre las dos partes ahora estamos listos para recibir los datos enviados por el cliente, lo que vamos a recibir se le llama factor de ocultamiento x esto nos ayuda a que el servidor quien va a realizar una firma a ciegas no sepa que es lo que este firmando, y no se entere de la información manejada, ya que tenemos este factor ahora si, el servidor se dispone a realizar la firma a ciegas y para que pueda continuar con la creación de la llave para poder cifrar los archivos

```
def handle(self):
    data= ""
    while data != "salir":
        #intento recibir informacion
        try:
            data= self.request.recv(65536)
            print data
            x = int(data)
            print "x:", x
            y = pow(x,d,n)
                            #Devolvemos el mensaje al cliente
            print "y:", y
            self.request.send(str(y))
            time.sleep(0.1)

        #si hubo un error lo digo y termino el handle
        except:
            print "El cliente D/C o hubo un error"
            data="salir"
```

Obteniendo la firma a ciegas y se la enviamos a nuestro cliente para que pueda continuar en la creación de la llave z , esto se ve explicado mas adelante cuando se explique la interfaz.

6.3. Aplicación Criptográfica (Cifrado/Descifrado)

6.3.1. Cifrado

Éste algoritmo que forma parte de la aplicación criptográfica, se lleva a cabo del lado del cliente, dicho algoritmo de cifrado se encargará de brindar la seguridad a los archivos que los usuarios deseen almacenar en la nube.

El cifrado de archivos se lleva a cabo bajo la utilización del algoritmo de cifrado **AES** que proveé la librería criptográfica **PyCrypto 2.3** propia del lenguaje **Python**. Ésta librería poseé la seguridad necesaria para satisfacer a los requerimientos del protocolo criptográfico.

Objetivo

Proteger la información que se almacenará en la nube.

Entradas

- El archivo que se almacenará en la nube.
- La clave para poder cifrarlo, que en este caso es la llave (*z*) generada por el cliente en el módulo anterior.

La implementación del algoritmo, se llevó a cabo de la siguiente manera:

```
from random import randint
import socket
import hashlib
import hmac
import Crypto.Cipher.AES, Crypto.Util.Counter
import hmac
from rsagen import *
from Duplicidad import *
from django.core.files import File
from models import Hashes, Cipher
```

Siendo **haslib**, **Crypto.Cipher.AES**, **Crypto.Util.Counter**, **hmac** librerías criptográficas, es decir, utilizadas para llevar a cabo operaciones relacionadas con la implementación del algoritmo de cifrado *AES* en sus 3 tipos de tamaños de claves (*128, 192, 256 bits*).

- Para poder comenzar el proceso de cifrado, es necesario obtener la clave que se utilizará para llevar a cabo el proceso.

Dicha clave se obtiene de la siguiente manera:

```
contentK2 = open("llaves_clientes/key_z_" + filename + "_"
                  + nom_user + ".PEM", "rb").read()
```

Abrimos el archivo **key_z** y almacenamos su contenido en la variable **contentK**. Éste archivo contiene la clave que se necesita para poder cifrar el archivo que el usuario desea, dicha *clave (z)* fue generada y escrita en este archivo en el módulo anterior.

- Se crea un objeto de tipo *AES* que almacenamos en la variable **cipher**, el cual contiene como parámetros la clave que obtuvo del archivo **key_z**, el modo de operación que se utilizará (**CTR**), etc.

```
cipher = Crypto.Cipher.AES.new(contentK2, Crypto.Cipher.AES.
                                MODE_CTR, counter=ctr)
```

- Para cifrar el archivo, mandamos llamar al método *encrypt(m)* (*m* almacena el contenido del archivo que se desea cifrar) y almacenamos el resultado de dicho método en la variable **ctext**. y esta variable contiene el archivo cifrado por completo

```
ctext = cipher.encrypt(m)
```

6.3.2. Descifrado

Éste algoritmo que forma parte de la aplicación criptográfica, al igual que el cifrado, se lleva a cabo del lado del cliente, éste algoritmo será el encargado de que los usuarios puedan recuperar sus archivos originales, es decir, tomar de la nube aquel archivo que se encuentre cifrado y posteriormente descifrarlo para poder acceder a este archivo en su forma original. El descifrado de archivos se lleva a cabo bajo la utilización del algoritmo de descifrado **AES** que, al igual que el cifrado lo proveé la librería criptográfica **PyCrypto 2.3** propia del lenguaje **Python**.

Objetivo

Descifrar los archivos almacenados de forma íntegra.

Entradas

- El archivo que se almacenó en la nube.
- La clave para poder descifrarlo, que en este caso es la llave (*z*) generada por el cliente.

La implementación del algoritmo, se llevó a cabo de la siguiente manera:

- Las librerías utilizadas para llevar a cabo el descifrado de archivos son las siguientes:

```
from random import randint
import socket
import hashlib
import hmac
import Crypto.Cipher.AES, Crypto.Util.Counter
from rsagen import *
from Duplicidad import *
from django.core.files import File
from models import Hashes, Cipher
```

Siendo **Crypto.Cipher.AES**, **Crypto.Util.Counter**, librerías criptográficas, es decir, utilizadas para llevar a cabo operaciones relacionadas con la implementación del algoritmo de descifrado *AES* en sus 3 tipos de tamaños de claves (*128, 192, 256 bits*).

- Al igual que en el proceso de cifrado, para comenzar dicho proceso es necesario obtener la clave que se utilizará para el descifrado. Dicha clave se obtiene de la siguiente manera:

```
contentK2 = open("llaves_clientes/key_z_"+filename+"_"+  
nom_user+".PEM", "rb").read()
```

Abrimos el archivo **key_z** y almacenamos su contenido en la variable **contentK**.

- El siguiente paso, es buscar en la base de datos el archivo que se va a descifrar y se guarda en una variable.

```
buscar = Cipher.objects.filter(hash_c=doc).all()[0]  
file_c1 = File(buscar.docfile).read()
```

Una vez dentro del archivo, almacenamos el cifrado en la variable **textcifrado** para utilizarlo posteriormente.

- Creamos un objeto *AES* que almacenamos en la variable **cipher**, el cual contiene como parámetros la llave que obtuvo del archivo **key_z** , el modo de operación que se utilizará para el descifrado(**CTR**), etc.

```
cipher = Crypto.Cipher.AES.new(contentK, Crypto.Cipher.AES.  
MODE_CTR, counter=ctr)
```

- Desciframos el archivo, mandamos llamar al método *decrypt(textcifrado)* y almacenamos el resultado de dicho método en la variable **plaintext**

```
return cipher.decrypt(textcifrado)
```

- Para finalizar, mandamos escribir a un archivo **fname** (*Es el nombre del archivo original del usuario*) el archivo tal y como estaba antes de cifrarlo.

```
plaintext = descifrado(contentK2, textcifrado, iv)  
outf = open("Descifrados/" + nom_user + "/" + filename  
, "wb")  
outf.write(plaintext)  
outf.close()  
print "Mensaje Descifrado c1: "
```

6.4. Interfaz Web

En ésta sección, se encuentra el detalle de la implementación de los módulos anteriores dentro de una interfaz web.

Dicha interfaz se desarrolló bajo el sistema operativo Linux. Esto debido a que presenta varias ventajas, es un sistema operativo de software libre, por ende no es necesario comprar

una licencia para instalarlo y utilizarlo en un equipo informático. Es un sistema multitarea, multiusuario, compatible con *UNIX*, y proporciona una interfaz de comandos y una interfaz gráfica.

También elegimos como manejador de base de datos de la interfaz *SQLite*.

SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware. Además agrega extensiones que facilitan su uso en cualquier ambiente de desarrollo. Esto permite que *SQLite* soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL, y lo más importante es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente.

Sin embargo la mayor parte de la codificación fue desarrollada e implementada en *Django*, éste es un framework para aplicaciones web gratuito y open source, escrito en *Python*.

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Un lenguaje ideal para desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

Lo cuál hizo posible la compatibilidad entre los módulos desarrollados y una interfaz web que fuera intuitiva, segura y fácil de manejar para los usuarios.

6.5. Mapa de Navegación.

En la figura 6.1 se muestra el mapa de navegación de la aplicación web.

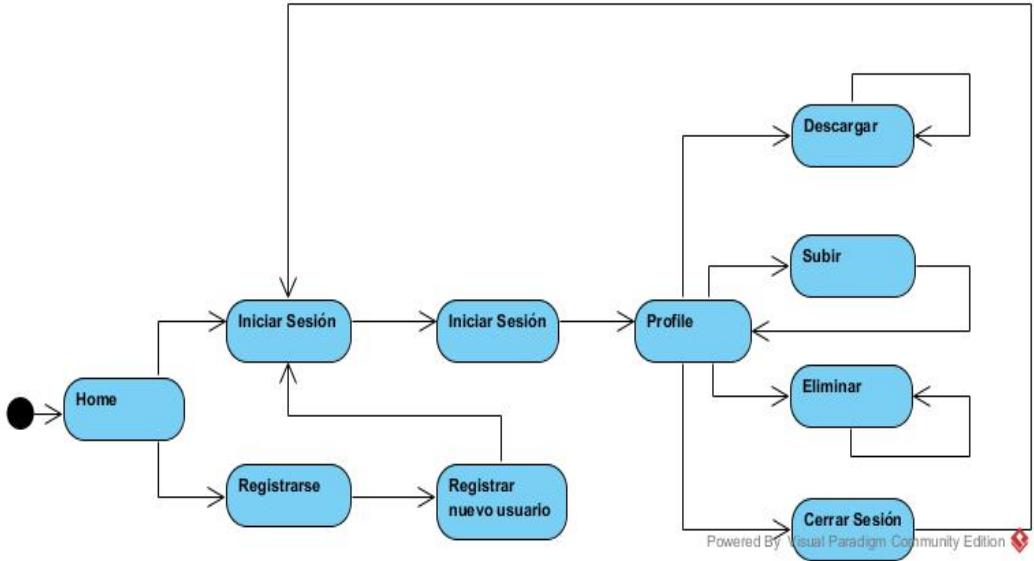


Figura 6.1: Mapa de navegación

Dentro de la interfaz web, existen diferentes módulos que la conforman. Dichos módulos son:

6.5.1. Registrar un Nuevo Usuario.

Éste módulo que compone a la interfaz, se encargará de poder llevar a cabo el registro en la plataforma de nuevos usuarios de nuestro protocolo criptográfico. En dicho módulo participará el usuario activamente con la interfaz, la cuál le solicitará datos personales como **Nombre, Apellidos, Correo electrónico, Contraseña**. Con ésta información, la plataforma generará un par de claves de usuario (Con el algoritmo de generación de claves RSA) para que estas sean utilizadas en el proceso de carga y descarga de archivos.

A continuación, se muestra a detalle como es que se lleva a cabo el registro de un usuario dentro de la interfaz web:

En este fragmento de código podemos observar la clase ***RegisterView*** que se encarga de crear un usuario, lo que hace es obtener los datos del formulario que ha llenado el usuario en la página y crea un objeto ***user*** para poder identificarlo como usuario, una vez que su registro haya sido exitoso se crean sus llaves certificadas.

```

class RegisterView(FormView):
    template_name = 'nube/register.html'
    form_class = RegistrationForm

    def form_valid(self, form):
        user= User.objects.create_user(

```

```

        username = form.cleaned_data['username'],
        password = form.cleaned_data['password1'],
        email = form.cleaned_data['email']
    )

    nom_user = user.username
    gen_rsa(nom_user)
    return super(RegisterView, self).
        form_valid(form)
def get_success_url(self):
    return reverse('register-success')

```

6.5.2. Iniciar Sesión de un usuario.

Para que un usuario pueda iniciar sesión es necesario que la información que ingresó en la plataforma sea validada por el sistema ya que debe estar registrada en él. Para esto tenemos la clase llamada **LoginView** donde se obtienen los datos del usuario y se valida con los datos que ya existen en la base de datos. Si los datos del usuario se encuentran almacenados, la plataforma redireccionará a la página del perfil del usuario. En caso contrario, la plataforma regresará al index.

```

class LoginView(FormView):
    template_name = 'nube/login.html'
    form_class = LoginForm

    @method_decorator(csrf_protect)
    def dispatch(self, *args, **kwargs):
        return super(LoginView, self).dispatch(*args, **kwargs)

    def form_valid(self, form):
        login(self.request, form.get_user())
        return super(LoginView, self).form_valid(form)

    def get_success_url(self):
        try:
            return config.LOGIN_REDIRECT_URL
        except:
            return "/profile/"

```

Para poder listar todos los archivos que tiene almacenados el usuario que ha iniciado sesión, realizamos un ciclo for para recorrer el arreglo que contiene los nombres de los archivos que se encuentran en la base de datos que le pertenecen al usuario.

```

{ % for element in lstFiles %}
    <h4> {{ element }} </h4>
{ % endfor %}

```

Dicho arreglo que se recorrió, fue creado en las vistas, ahí lo que hacemos cada que se entra a la página del perfil de usuario buscamos en la base de datos los archivos que coincidan con el nombre de usuario que ha iniciado sesión, esta consulta nos regresa todos los archivos como objetos, entonces lo que debemos hacer es crear un arreglo y recorrerlo junto con la lista de objetos de la base de datos e irlos asignando accediendo solo a la información del nombre del archivo y regresamos el arreglo para que se pueda imprimir en nuestro template.

```

def lista(request):
    nom_user = str(request.user)
    buscar = Cipher.objects.filter(user_name=nom_user).all()
    tamano = len(buscar)
    i = 0
    lstFiles = []
    for i in range(0, tamano):
        lstFiles.append(buscar[i].filename)
    print ('LISTADO FINALIZADO')
    return render(request, 'nube/profile.html', { 'lstFiles' :
        lstFiles })

```

6.5.3. Subir un archivo.

Para poder seleccionar el archivo que se va a subir, en el html mandamos a llamar el formulario creado para poder *cargar un archivo*.

```

<div class="row" id="content-container">
    <div class="page-header">
        <h2></h2>
    </div>
    <div class="jumbotron-content">
        <form action="" method="post" enctype="multipart/
            form-data">{{ csrf_token }}
            {{ form.as_p }}
            <input type="submit" value="Subir">
        </form>
    </div>
</div>

```

Este método fue implementado para crear el documento y guardarlo en la base de datos. Se crea un objeto de tipo documento donde le asignamos el archivo que ha seleccionado el usuario y se almacena en la base de datos con la línea `newdoc.save(form)`. En la siguiente línea (`subir_arch...`), se manda a llamar la función `subir_archivo` que es la que se encarga

de cifrar el archivo, aquí los parámetros que le pasamos son en primer lugar el nombre del archivo que obtenemos del archivo seleccionado, y el nombre de usuario correspondiente a la sesión.

```
def upload_file(request):
    if request.method == 'POST':
        form = UploadForm(request.POST, request.FILES)
        if form.is_valid():
            palabra2 = str(request.FILES['docfile'])
            newdoc = Document(docfile=request.FILES['docfile'])
            newdoc.save(form)
            nom_user = str(request.user)
            subir_arch(palabra2, str(nom_user))
            return redirect("profile")
    else:
        form = UploadForm()
    return render(request, 'nube/upload.html', {'form': form})
```

Cifrar un archivo.

A continuación, se explica con detalle la parte del cifrado de archivos.
Para empezar, importamos algunas bibliotecas criptográficas, como son:

```
from random import randint
import socket
import hashlib
import hmac
import Crypto.Cipher.AES, Crypto.Util.Counter
from rsagen import *
from Duplicidad import *
from django.core.files import File
```

- La biblioteca **Crypto.Cipher.AES** tiene implementado el algoritmo AES en todas sus versiones y lo utilizaremos para cifrar.
- La biblioteca **hmac** contiene la implementación de las funciones Hash que para el desarrollo de este protocolo criptográfico se utilizó **SHA256**.
- La biblioteca **rsagen**, en ésta implementamos el algoritmo de RSA para la generación de llaves.
- La biblioteca **Duplicidad**, se encuentra implementado nuestro algoritmo para detectar duplicados.

En ésta parte de la codificación, se tiene la implementación de la función *subir_arch*, donde lo primero que hacemos es abrir las llaves del servidor y crear una conexión con ayuda de sockets entre nuestro cliente y el servidor.

```
def subir_arch(filename , nom_user):
    n = int(open("llaves_servidor/key_n_server.PEM" , "r").read())
    e = int(open("llaves_servidor/key_e_server.PEM" , "r").read())
    host , port = "localhost" , 9999
    sock= socket.socket()
    sock.connect((host , port))
```

Una vez creada la conexión entre cliente - servidor, se obtiene el archivo que se va a subir, esto gracias a que cuando se manda a llamar la función, ésta pasa por parámetros el nombre del archivo y del usuario. Con esto se logra abrir el archivo y se le aplica una función hash de ***SHA256*** (*h(m)*) y se opera con la llave pública del servidor (*e*), a esto se le llama *factor de ocultamiento* (*x*), con este factor se puede llevar a cabo una firma a ciegas por parte de nuestro servidor, éste factor se le envía al servidor por medio de sockets.

```
while mensaje != "salir":
    r = randint(0 ,n)
    m = str(open(filename , "rb").read())
    h = int(hashlib.sha256(m).hexdigest() , 16)
    print "h(m): " , h
    x1 = pow(r , e , n)
    x = (h * x1) % n
    print "x: " , x
    f_x = open("x" , "w")
    f_x.write(str(x))
    f_x.close()
    mensaje = open("x" , "r").read()

try:
    sock.send(mensaje)

except:
    print "no se mando el mensaje"
    mensaje="salir"
```

Una vez realizada la firma a ciegas (*y*) la envía al cliente por medio de sockets, el cuál procede a calcular la llave *z* (que sirve para cifrar los archivos). Y se crea un archivo con dicha llave, ya que esta llave es demasiado grande para que funcione en *AES*, es por ello que se debe de sacar un hash con *SHA256* para que quede del tamaño permitido por *AES* (128 bits) y de igual manera creamos un archivo con la llave.

```
recibido = sock.recv(65536)
print ""
```

```

print "RECIBIDO: ", recibido
y = int(recibido)
print "y: ", y
rinv = modinv(r, n)
print "inverso: ", rinv
z_long = (y * rinv) % n

##### Calculo de llave Z en tamaño original
print "z: ", z_long
f_z = open("llaves_clientes/key_z_"+filename2+"_"+nom_user
           +".PEM", "w")
f_z.write(str(z_long))
f_z.close()
check = pow(z_long, e, n)
print "check: ", check
print "h: ", h % n

##### Hash a Llave Z ya que es muy grande Se manda
a escribir de nuevo al archivo
kz = str(open("llaves_clientes/key_z_"+filename2+"_"++
               nom_user+".PEM", "rb").read())
z = hashlib.sha256(kz).hexdigest()[:16]
wz = open("llaves_clientes/key_z_"+filename2+"_"+nom_user
           +".PEM", "w")
wz.write(str(z))
wz.close()

```

Para cifrar el archivo se guarda el contenido de la llave z y es utilizada para generar un vector de inicialización usando una función *SHA256*, dicho vector se almacena en un archivo para que pueda ser utilizado posteriormente a la hora de descifrar el archivo, posteriormente se cifra el archivo con ayuda de las funciones definidas en las bibliotecas criptográficas que se mencionan anteriormente, y se le indica el tamaño de la llave de *128 bits*, con un modo de operación *CTR* y con la función *encrypt()* se cifra el archivo, éste archivo se almacena en una carpeta y se le pone el nombre original pero agregandole una extensión *.aes*

```

contentK = open("llaves_clientes/key_z_"+filename2+"_"++
                 nom_user+".PEM", "rb").read()
iv = hmac.new(contentK, m, hashlib.sha256).hexdigest()[:32]
escribeIV = open("llaves_clientes/vector__"+filename2+"_"++
                  nom_user+".txt", "wb")
escribeIV.write(iv)
escribeIV.close()
ctr = Crypto.Util.Counter.new(128, initial_value=long(iv,

```

```

    16))
cipher = Crypto.Cipher.AES.new(contentK, Crypto.Cipher.AES
    .MODE_CTR, counter=ctr)
ctext = cipher.encrypt(m)
fc1 = str(ctext)
print ""
print "Mensaje Cifrado... C1 "

```

Para nosotros es importante guardar otro archivo cifrado $c2$ que va a contener la llave z cifrada con la llave publica e del usuario, esto es para que el usuario no tenga que estar guardando las llaves en su dispositivo donde subió el archivo, además si quiere abrirlo en otro dispositivo tendría que estar cargando con esa llave para poder descargarlo y así con todas las llaves de todos los archivos que se disponga a subir en el sistema, y si lo guardamos en la nube ya no pasaría esto y el usuario solo tendría que iniciar sesión con sus llaves personales que esas si tiene que ser responsable de tenerlas, de no ser así no puede hacer nada. Este proceso es similar al cifrado pasado solo que ahora vamos a cifrar la llave con que se cifró el archivo con su llave publica, y se almacena en una variable llamada $fc2$

```

contentK = open("llaves_clientes/key_e_"+nom_user+".PEM",
    "rb").read()
m = str(open("llaves_clientes/key_z_"+filename2+"_"+nom_user+".PEM",
    "rb").read())
iv = hmac.new(contentK, m, hashlib.sha256).hexdigest()
[ : 32] ## Generacion del IV
## Mandamos el vector IV a un archivo .txt para que tambien
## pueda ser utilizado por el descifrado
escribeIV = open("llaves_clientes/vector_c2_"+filename2+"_"+nom_user+".txt",
    "wb")
escribeIV.write(iv)
escribeIV.close()
contentK2 = hashlib.sha256(contentK).hexdigest()[ : 16]
ctr = Crypto.Util.Counter.new(128, initial_value=long(iv,
    16))
cipher = Crypto.Cipher.AES.new(contentK2, Crypto.Cipher.
    AES.MODE_CTR, counter=ctr)
ctext = cipher.encrypt(m)
fc2 = str(ctext)
print "Mensaje Cifrado... C2 "

```

Duplicación.

Una vez que se cifró el archivo, del archivo $c1$ que fue almacenado en la llave $fc1$ se le aplica una función hash y el resultado de ésta se le envía a la función que va a checar la

duplicación, va a checar si el archivo ya ha sido almacenado una vez. Además se envía el nombre del archivo.

```
hc1 = hashlib.sha256(fc1).hexdigest()[:16]
fc2 = str(ctext)
deduplication(hc1, nom_user, filename2, fc1, fc2)
```

Ya que recibió el hash del mensaje busca en la base de datos en la tabla que tenemos donde guardamos los hash si existe alguno con el mismo hash, de ser así guardamos el hash del archivo en la variable *var*, de no haber encontrado un archivo se deja la variable var vacía.

```
def deduplication(hc1, nom_user, filename2, fc1, fc2):
    try:
        hc = Hashes.objects.get(hash_c=hc1)
        var = hc.hash_c
        doc = File(hc.docfile).read()
    except:
        var = ""
```

Ahora checamos si nuestra variable var es igual al hash que hemos recibido, si son iguales entonces tenemos el caso donde es un archivo duplicado, lo primero que hacemos es crear un archivo con la información de *c2* que es la llave cifrada y buscamos en la base de datos si ya se ha subido una llave con el mismo hash con el mismo usuario. Si se encuentra un archivo con el mismo nombre y la misma llave para el mismo usuario no se tiene que guardarnada en la base de datos, ya que dicho usuario ha subido éste archivo anteriormente, en caso contrario debemos guardar en la base de datos éste archivo que creamos.

```
if str(var) == hc1:
    print "Duplicado detectado"
    filec2 = open("Cifrados/" + nom_user + "/c2_" + filename2 + ".aes", "wb")
    filec2.write(str(fc2))
    filec2.close()
    f = open("Cifrados/" + nom_user + "/c2_" + filename2 + ".aes")
    f.read()
    try:
        Cipher.objects.get(filename="c2_" + filename2, user_name=nom_user)
        cont=1
        print "try : "+str(cont)
    except:
        cont=0
        print "except : "+str(cont)

    if cont==0:
```

```

newdoc = Cipher( filename="c2_"+filename2 , hash_c=hc1 ,
    user_name=nom_user , docfile=File(f))
newdoc . save ( File(f) )
f . close ()
else :
    print "nada"

```

Cuando el hash no ha sido encontrado en la base de datos se trata de un archivo nuevo y debemos guardar tres cosas, primero se crea un archivo con el hash que hemos recibido y se almacena en la base de datos en la tabla de *Hashes*, después creamos un archivo con la información del archivo cifrado y se almacena en la base de datos, el tercer archivo contiene la llave cifrada del archivo y se almacena en la tabla de cifrados.

```

else :
    print "Archivo nuevo"
    hash_c1 = open("hash/"+nom_user+"/"+filename2+".sha","wb")
    hash_c1 . write( str(hc1) )
    hash_c1 . close ()
    f = open("hash/"+nom_user+"/"+filename2+".sha" )
    f . read ()
    print str(f)
    newdoc = Hashes( filename=filename2 ,hash_c=hc1 , docfile=
        File(f) )
    newdoc . save ( File(f) )
    f . close ()
    filec1 = open("Cifrados/"+nom_user+"/"+filename2+".aes" ,"
        wb")
    filec1 . write( str(fc1) )
    filec1 . close ()
    f = open("Cifrados/"+nom_user+"/"+filename2+".aes" )
    f . read ()
    newdoc = Cipher( filename=filename2 , hash_c=hc1 , user_name=
        nom_user , docfile=File(f))
    newdoc . save ( File(f) )
    f . close ()
    filec2 = open("Cifrados/"+nom_user+"/c2_"+filename2+".aes" ,
        "wb")
    filec2 . write( str(fc2) )
    filec2 . close ()
    f = open("Cifrados/"+nom_user+"/c2_"+filename2+".aes" )
    f . read ()
    newdoc = Cipher( filename="c2_"+filename2 , hash_c=hc1 ,
        user_name=nom_user , docfile=File(f))
    newdoc . save ( File(f) )

```

```
f.close()
```

6.5.4. Descargar un archivo

Para descargar un archivo de un formulario donde vamos a tener un campo de entrada de texto y un botón donde se le va a solicitar al usuario que ponga el nombre del archivo a descargar y ésto se le envía a nuestro archivo de vistas.

```
<form action="" method="post" enctype="multipart/form-data">{ %  
  csrf_token %}  
  {{ form.as_p }}  
  <input type="submit" value="Descargar">  
</form>
```

Ya que obtenemos el nombre del archivo que se quiere descargar lo que hacemos es identificar al usuario correspondiente a la sesión y se envían éstos datos a una función llamada descargar archivo y de llevarse a cabo esta función de manera exitosa mostramos una alerta donde informamos al usuario que su archivo ha sido descargado con éxito. De lado contrario se le notifica que el archivo no existe.

```
def download(request):  
    if request.method == 'POST':  
        form = DownloadForm(request.POST, request.FILES)  
        if form.is_valid():  
            filename=request.POST['filename']  
            print filename  
            nom_user = str(request.user)  
            descargar_archivo(filename, nom_user)  
            messages.success(request, 'Archivo Descargado con  
            exito')  
            return redirect("download")  
        else:  
            messages.error(request, 'El archivo que deseas  
            descargar no existe')  
    else:  
        form = DownloadForm()  
    return render(request, 'nube/download.html', {'form': form})
```

Descifrar un archivo

Cuando recibimos el nombre del archivo y el usuario lo que hacemos es primero consultar en la base de datos en la tabla de *Cipher* el archivo *c2* correspondiente al usuario en sesión y se almacena en una variable *filec2*, después de éste archivo que hemos buscado obtenemos

el campo de la tabla que contiene el hash y lo guardamos en la variable *doc*, después buscamos en la tabla de cifrados el primer archivo que coincide con el hash que hemos sacado anteriormente, ésto nos garantiza que será el archivo original sin importar que tenga otro nombre o sea de otro usuario.

```
def descargar_archivo(filename, nom_user):
    buscar = Cipher.objects.get(filename="c2_" + filename, user_name=nom_user)
    file_c2 = File(buscar.docfile).read()
    hc = Cipher.objects.get(filename="c2_" + filename, user_name=nom_user)
    doc = hc.hash_c
    buscar = Cipher.objects.filter(hash_c=doc).all()[0]
    file_c1 = File(buscar.docfile).read()
    file_d = open("llaves_clientes/key_d_" + nom_user + ".PEM", "rb").read()
```

Una vez que obtenemos el archivo *c2* el siguiente paso es descifrarlo, para éste proceso lo primero que necesitamos es realizar un hash de la llave privada del usuario, buscar el vector de inicialización que se utilizó para cifrar este archivo y mandarle estos parámetros a nuestra función de descifrado, la cuál nos regresa el texto plano de nuestro archivo *c2*.

```
contentK = hashlib.sha256(file_d).hexdigest()[:16]
textcifrado = file_c2
iv = open("llaves_clientes/vector_c2_" + filename + "_" + nom_user + ".txt", "rb").read()
plaintext = descifrado(contentK, textcifrado, iv)
outf = open("Descifrados/" + nom_user + "/c2_" + filename + ".txt", "wb")
outf.write(plaintext)
outf.close()
print "Mensaje Descifrado c2: "
```

Para continuar con la descarga de nuestro archivo, ahora descifraremos nuestro archivo *c1*, de igual manera se busca el vector de inicialización utilizado para cifrar y se le envía a nuestra función de descifrar y el resultado de éste ya es nuestro archivo descargado el cual se guarda en una carpeta llamada *Descifrados del usuario*.

```
contentK = plaintext
textcifrado = file_c1
iv = open("llaves_clientes/vector_" + filename + "_" +
           nom_user + ".txt", "rb").read()
contentK2 = hashlib.sha256(contentK).hexdigest()[:16]
contentK2 = open("llaves_clientes/key_z_" + filename + "_" +
                 nom_user + ".PEM", "rb").read()
plaintext = descifrado(contentK2, textcifrado, iv)
```

```

outf = open("Descifrados/" + nom_user + "/" + filename , "wb")
outf.write(plaintext)
outf.close()
print "Mensaje Descifrado c1: "

```

Ésta es la implementación de nuestra función de descifrado la cual gracias a las librerías que ocupamos solo le tenemos que indicar que ocupamos un modo de operación *CTR* con una versión de *AES* de 128 bits y le pasamos el archivo que queremos descifrar.

```

def descifrado(contentK , textcifrado , iv):
    ctr = Crypto.Util.Counter.new(128 , initial_value=long(iv , 16))
    cipher = Crypto.Cipher.AES.new(contentK , Crypto.Cipher.AES.
        MODE_CTR, counter=ctr)
    return cipher.decrypt(textcifrado)

```

6.5.5. Eliiminar un archivo

Lo primero que tenemos que hacer para descargar un archivo es obtener los datos del formulario llenado por el usuario que contiene un campo de entrada de texto donde se solicita de manera muy atenta y explicitamente el nombre del archivo y un botón simple y humanamente sencillo y se le envía a nuestro archivo de vistas.

```

<form action="" method="post" enctype="multipart/form-
    data">{ % csrf_token %}
    {{ form.as_p }}
    <input type="submit" value="Eliminar">
</form>

```

Cuando el usuario haya llenado el formulario, recibimos el nombre del archivo que se desea eliminar y a su vez obtenemos el usuario de la sesión, después mandamos estos datos a nuestra función llamada *eliminararchivo* y de llevarse de manera exitosa le mostramos a nuestro usuario el mensaje de que su archivo ha sido correctamente eliminado.

```

def delete(request):
    if request.method == 'POST':
        form = DeleteForm(request.POST, request.FILES)
        if form.is_valid():
            filename=request.POST['filename']
            print filename
            nom_user = str(request.user)
            eliminar_archivo(filename ,nom_user)
            messages.success(request , 'Archivo Eliminado con exito')
        )
        return redirect("delete")
    else :

```

```

    messages.error(request , 'El archivo que deseas
                      eliminar no existe')
else :
    form = DeleteForm()
return render(request , 'nube/delete.html' , { 'form' : form})

```

Primero buscamos en nuestra base de datos en la tabla de *Cipher* el archivo *c2* que corresponda al usuario en cuestión, ya que obtenemos este archivo lo eliminamos con la sentencia *delete*, antes de eliminarlo obtenemos el valor del hash de este archivo y buscamos si existe mas de dos archivos con este mismo hash, de ser así solo debemos eliminar el archivo *c2* ya que es un duplicado y debemos mantener el archivo original, en el caso que solo exista un archivo más con este hash significa que era la única copia de este archivo en la base de datos, por lo tanto debemos eliminar el archivo original y también debemos eliminar el hash de la tabla de *Hashes* y de esta forma si se vuelve a subir el archivo se detecta como si fuera uno nuevo.

```

def eliminar_archivo(filename , nom_user):
    buscar = Cipher.objects.get(filename="c2_"+filename , user_name
                                 =nom_user)
    doc = buscar.hash_c
    buscar.delete()
    try :
        buscar2 = Cipher.objects.filter(hash_c=doc).all()[1]
    except :
        buscar2 = Cipher.objects.filter(hash_c=doc).all()[0]
        buscar2.delete()
        buscar3 = Hashes.objects.get(hash_c=doc)
        buscar3.delete()
        print "eliminado"

```

6.6. Funcionamiento de la Interfaz Web

6.6.1. Pantalla Principal



Figura 6.2: Pantalla Principal de la interfaz web.

6.6.2. Pantalla Registro de Usuario

A screenshot of a web browser window titled "Registro". The address bar shows "127.0.0.1:8000/register/". The page has a white background with a dark blue header bar containing the word "REGISTRARSE". Below the header is a welcome message: "Bienvenido! Elige un nombre de usuario, O si tienes una cuenta, [Ingresa aquí](#)". There is a section titled "Registrar Usuario" with four input fields: "Nombre de usuario", "Correo electrónico", "Contraseña", and "Confirma tu contraseña". Below these fields is a large blue button labeled "Registrar nuevo usuario". At the bottom of the page is a dark footer bar with two links: "Ubicación" and "Links". On the far right of the footer is a small red upward-pointing arrow icon.

Figura 6.3: Pantalla de Registro del usuario.

6.6.3. Registrandote por primera vez

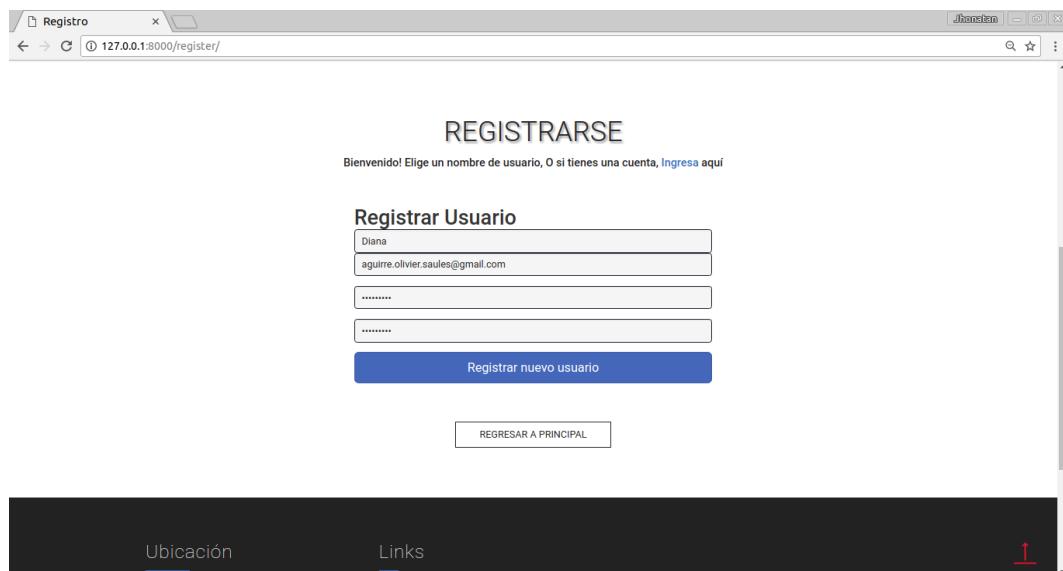


Figura 6.4: Pantalla con el formulario lleno con los datos de un usuario nuevo.

6.6.4. Pantalla de la Base de Datos

id	password	is_superuser	username	first_name	email	last_name	is_staff	is_active
1	pbkdf2_sha256\$2...	0	Leslie		les_olivier@hotmail.com		0	1
2	pbkdf2_sha256\$2...	0	Jhonatan		jsaulesc0476@gmail.com		0	1
3	pbkdf2_sha256\$2...	0	Diana		aguirre.olivier.saules@gmail.com		0	1

Figura 6.5: Tabla de Registro de Usuario en la Base de Datos.

6.6.5. Pantalla Inicio de Sesión

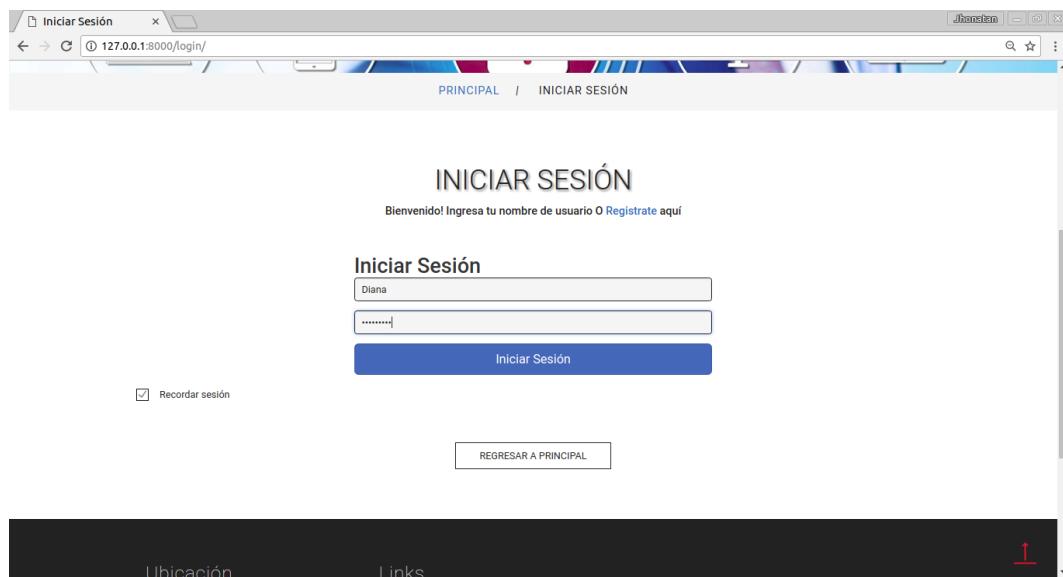


Figura 6.6: Pantalla de Inicio de Sesión.

6.6.6. Pantalla Tabla Cipher

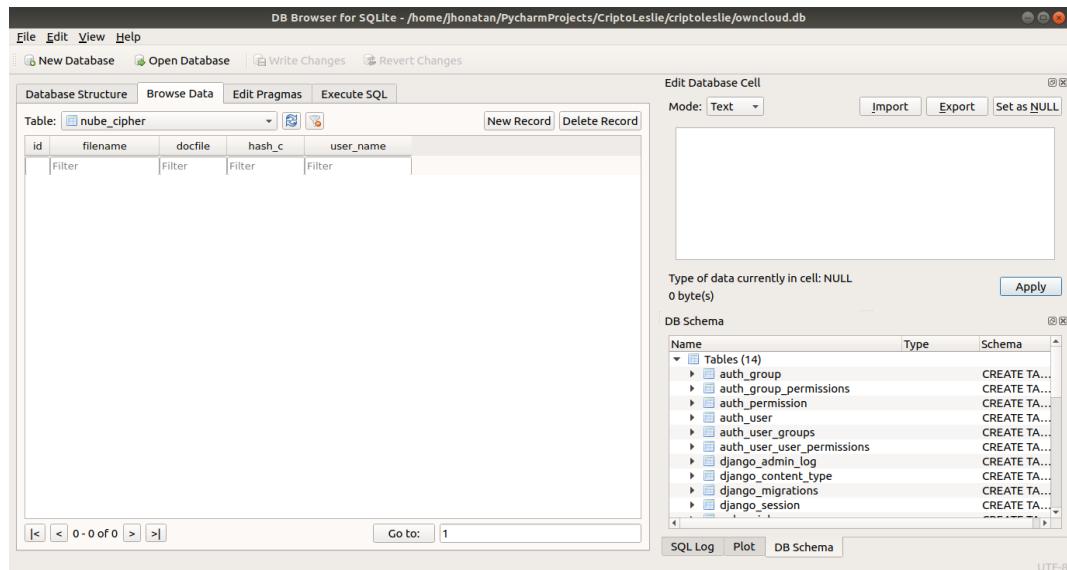


Figura 6.7: Tabla que nombramos Cipher donde se almacenan los archivos cifrados del usuario.

6.6.7. Pantalla Tabla Hashes

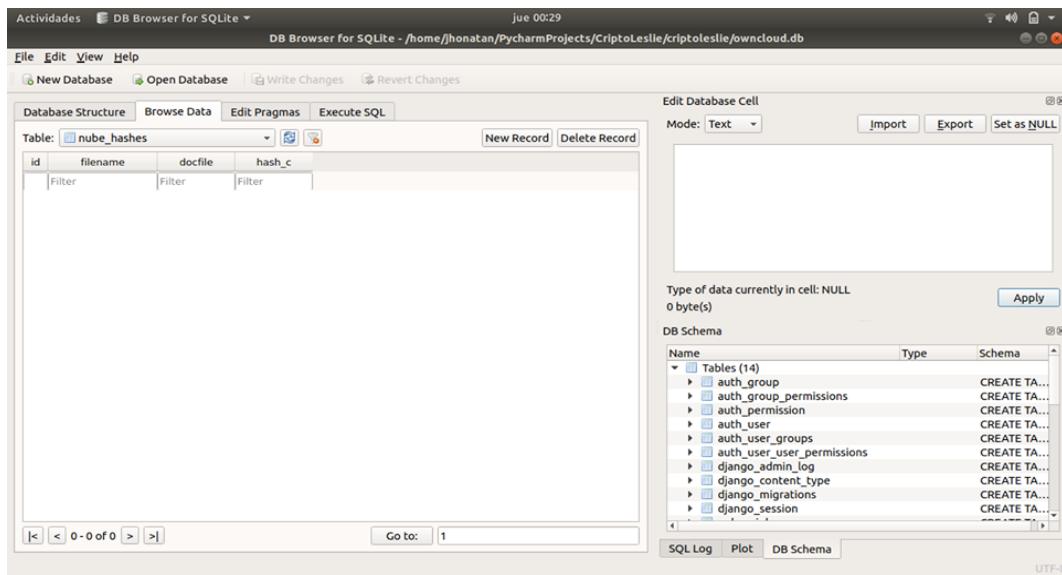


Figura 6.8: Tabla que nombramos Hashes donde se almacena el hash de los archivos almacenados en la nube.

6.6.8. Pantalla del Perfil del usuario

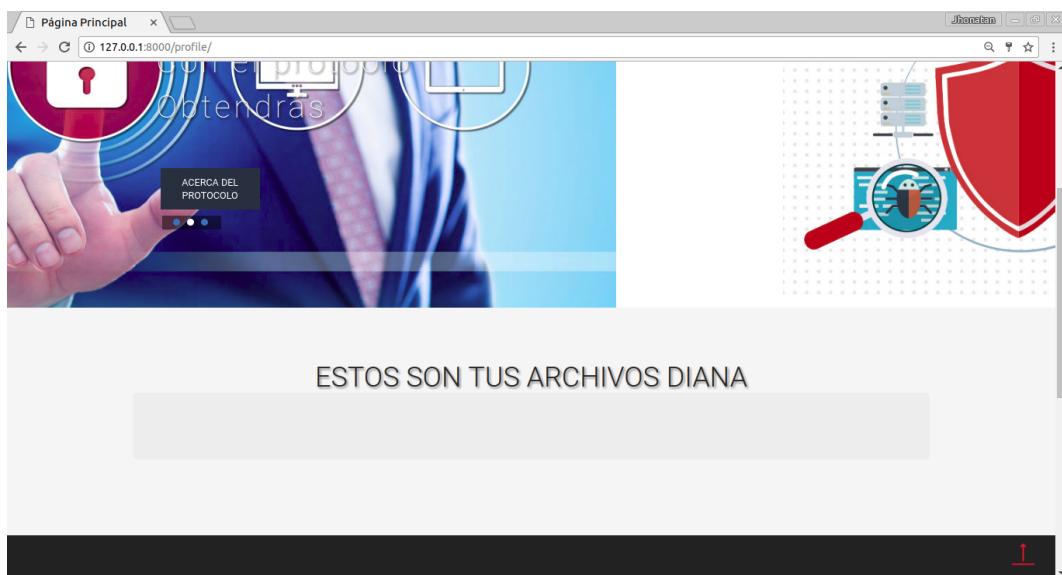


Figura 6.9: Pantalla del perfil de usuario, donde puede ver sus archivos y realizar otras acciones.

6.6.9. Pantalla Subir Archivo

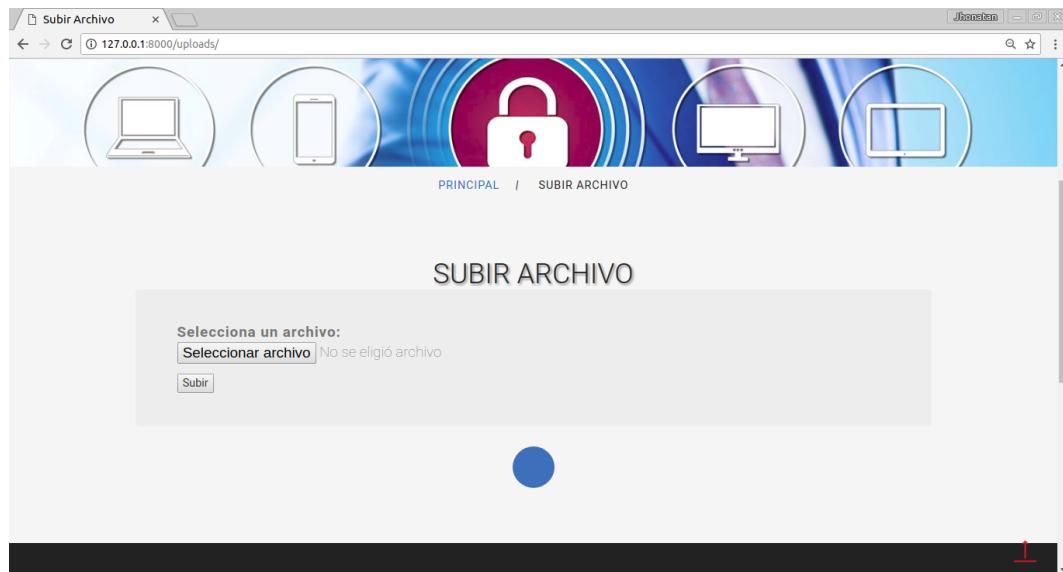


Figura 6.10: Pantalla al dar clic en la opción Subir Archivo.

6.6.10. Pantalla Elegir archivo a Subir

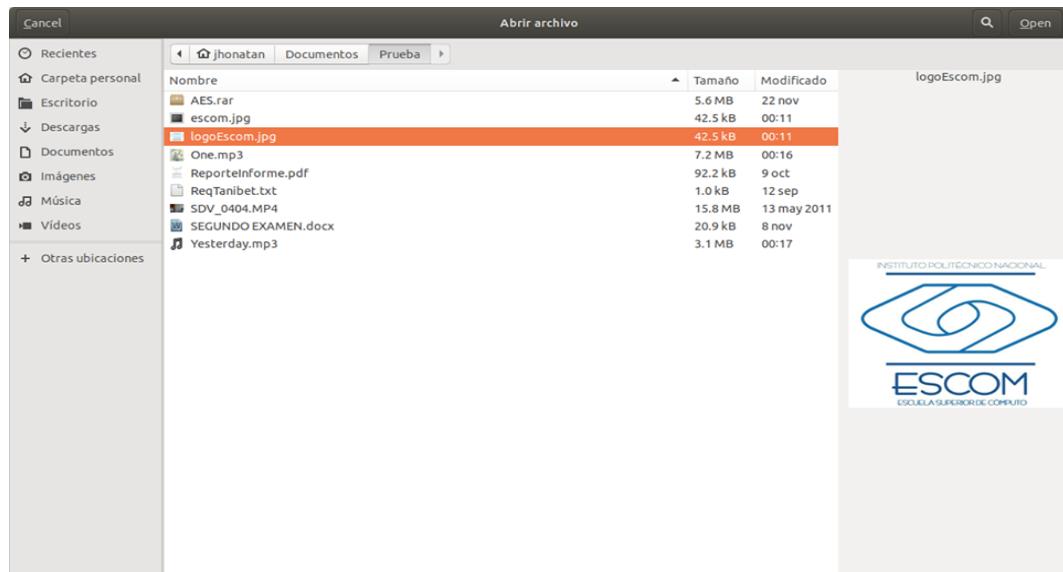


Figura 6.11: Al dar clic en seleccionar archivo, despliega esta pantalla donde podemos elegir un archivo para almacenarlo en la nube.

6.6.11. Pantalla del primer caso: Cuando Subimos un Archivo Nuevo

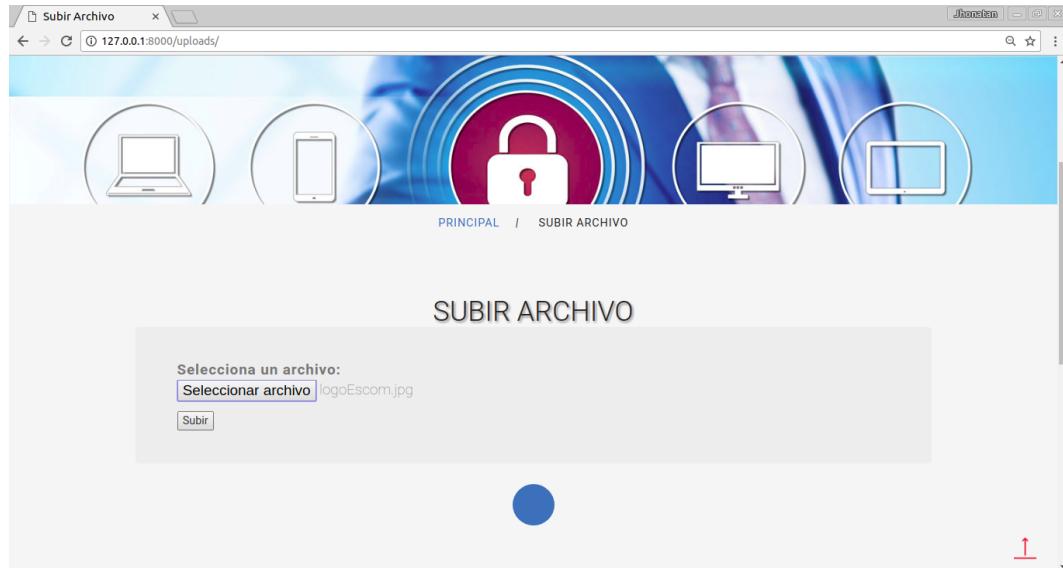


Figura 6.12: Una vez que elegimos el archivo aparecerá como en esta pantalla.

Pantalla de Perfil Actualizado



Figura 6.13: En el perfil de usuario aparecerá una lista de los archivos cifrados que tiene almacenados en la nube.

Pantalla Tabla Hashes Actualizada

The screenshot shows the DB Browser for SQLite interface with the database file '/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The main window displays the 'nube_hashes' table with the following data:

id	filename	docfile	hash_c
1	logoEscom.jpg	hash/logoEscom.jpg.sha	672d172cc5fc1072

The right side of the interface shows the 'Edit Database Cell' panel with the cell content set to NULL. The 'DB Schema' panel lists various tables and their schema definitions.

Figura 6.14: En esta pantalla mostramos la tabla Hashes actualizada.

Pantalla Tabla Cipher Actualizada

The screenshot shows the DB Browser for SQLite interface with the database file '/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The main window displays the 'nube_cipher' table with the following data:

id	filename	docfile	hash_c	user_name
1	logoEscom.jpg	cipher/logoEscom.jpg.aes	672d172cc5fc1072	Diana
2	c2_logoEscom.jpg	cipher/c2_logoEscom.jpg...	672d172cc5fc1072	Diana

The right side of the interface shows the 'Edit Database Cell' panel with the cell content set to NULL. The 'DB Schema' panel lists various tables and their schema definitions.

Figura 6.15: En esta pantalla mostramos que al subir el archivo ya se cifró y almacenó en la base de datos.

6.6.12. Pantalla Subir Otro Archivo Nuevo

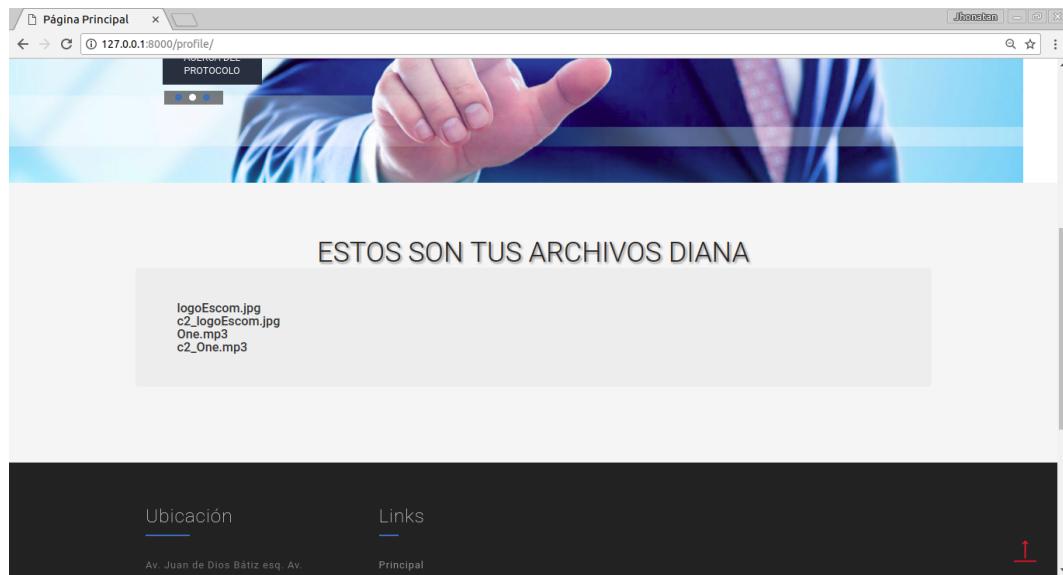


Figura 6.16: En esta pantalla mostramos que el usuario subió otro archivo y se actualizó su perfil.

Pantalla Tabla Cipher Actualizada

A screenshot of DB Browser for SQLite. The title bar says "DB Browser for SQLite - /home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db". The main interface shows a table named "nube_cipher" with four rows of data. The columns are "id", "filename", "docfile", "hash_c", and "user_name". The data is as follows:

id	filename	docfile	hash_c	user_name
1	logoEscom.jpg	cipher/logoEscom.jpg.aes	672d172cc5fc1072	Diana
2	c2_logoEscom.jpg	cipher/c2_logoEscom.jpg...	672d172cc5fc1072	Diana
3	One.mp3	cipher/One.mp3.aes	a2970d43aa576d0d	Diana
4	c2_One.mp3	cipher/c2_One.mp3.aes	a2970d43aa576d0d	Diana

On the right side, there is an "Edit Database Cell" panel with a text input field and an "Apply" button. Below it is a "DB Schema" panel showing a tree view of tables and their creation scripts. The tables listed are: auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_migrations, and django_session.

Figura 6.17: En esta pantalla mostramos la tabla Cipher actualizada.

Pantalla Tabla Hashes Actualizada

The screenshot shows the DB Browser for SQLite interface. The main window displays a table named 'nube_hashes' with three columns: 'id', 'filename', and 'hash_c'. There are two rows of data:

id	filename	hash_c
1	logoscom.jpg	672d172cc5fc1072
2	One.mp3	a2970d43aa576d0d

On the right side of the interface, there is a 'DB Schema' panel showing the database schema with various tables like auth_group, auth_permission, auth_user, etc.

Figura 6.18: En esta pantalla mostramos la tabla Hashes actualizada.

6.6.13. Pantalla segundo caso: Cuando otro usuario sube el mismo archivo con el mismo nombre.

Inicio de Sesión con otro usuario

The screenshot shows a Firefox browser window titled 'Iniciar Sesión - Mozilla Firefox'. The address bar indicates the URL is '127.0.0.1:8000/login/'. The page itself is a login form titled 'INICIAR SESIÓN' with the sub-instruction 'Bienvenido! Ingresa tu nombre de usuario O [Registerate aquí](#)'. It features fields for 'Nombre de usuario' (containing 'Jhonatan') and 'Contraseña' (containing '.....'). A blue 'Iniciar Sesión' button is at the bottom. A red arrow points upwards from the bottom right towards the top right corner of the browser window.

Figura 6.19: Pantalla de Inicio de Sesión de un usuario diferente.

Perfil de Usuario

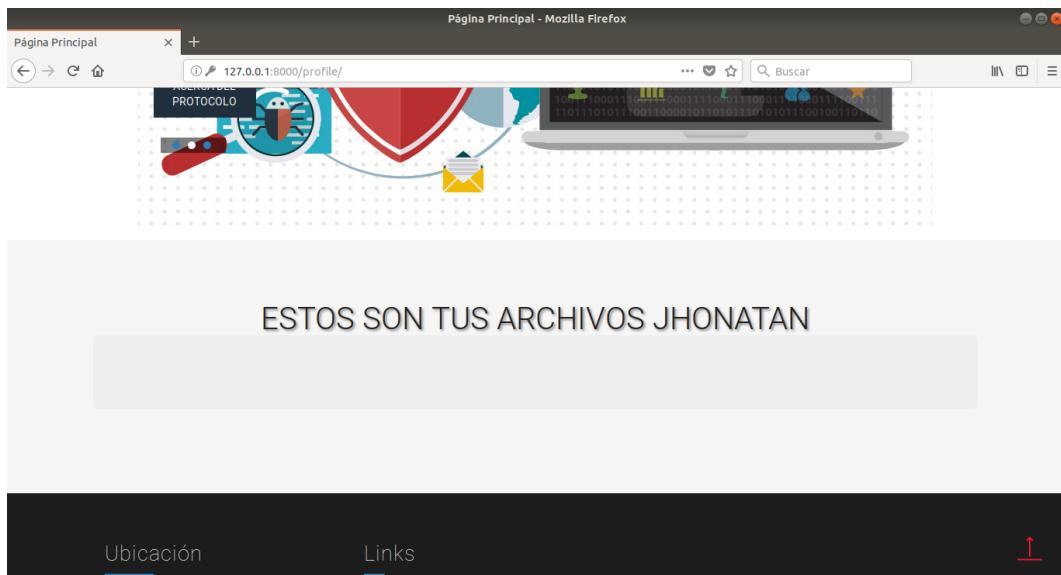


Figura 6.20: Pantalla de Perfil.

Subir Archivo

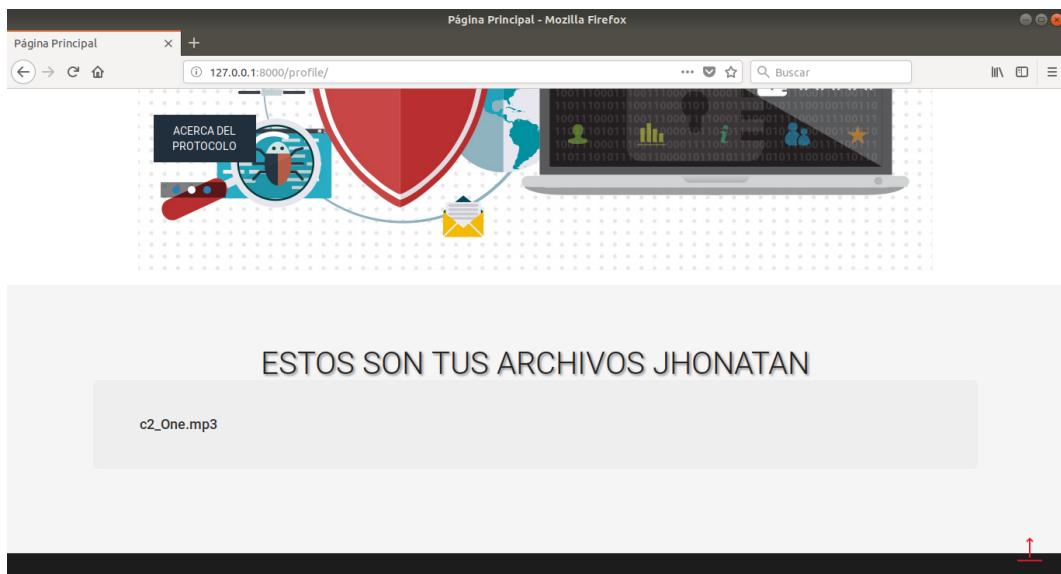


Figura 6.21: Pantalla de Perfil actualizada ya que el usuario subió un archivo, como es un duplicado sólo se guarda el c2 de dicho archivo.

Tabla Cipher actualizada

The screenshot shows the DB Browser for SQLite interface with the database file '/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The 'nube_cipher' table is selected, displaying five rows of data:

id	filename	docfile	hash_c	user_name
1	logoEscom.jpg	cipher/logoEscom.jpg.aes	672d172cc5fc1072	Diana
2	c2_logoEscom.jpg	cipher/c2_logoEscom.jpg...	672d172cc5fc1072	Diana
3	One.mp3	cipher/One.mp3.aes	a2970d43aa576d0d	Diana
4	c2_One.mp3	cipher/c2_One.mp3.aes	a2970d43aa576d0d	Diana
5	c2_One.mp3	cipher/c2_One.mp3.blU8n...	a2970d43aa576d0d	Jhonatan

Figura 6.22: Tabla Cipher actualizada.

Tabla Hashes actualizada

The screenshot shows the DB Browser for SQLite interface with the database file '/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The 'nube_hashes' table is selected, displaying two rows of data:

id	filename	docfile	hash_c
1	logoEscom.jpg	hash/logoEscom.jpg.sha	672d172cc5fc1072
2	One.mp3	hash/One.mp3.sha	a2970d43aa576d0d

Figura 6.23: Tabla Hashes actualizada.

Perfil Actualizado

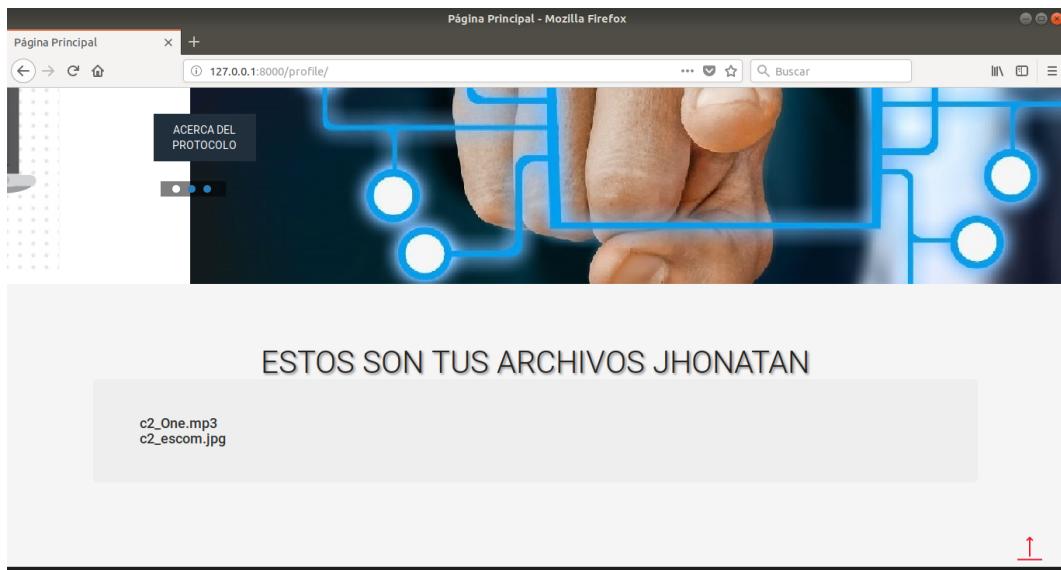


Figura 6.24: Perfil actualizado ya que el nuevo usuario subió más archivos.

Tabla Cipher actualizada

A screenshot of DB Browser for SQLite. The title bar says "DB Browser for SQLite - /home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db". The main window shows a table named "nube_cipher" with the following data:

	id	filename	docfile	hash_c	user_name
1	1	logoscom.jpg	cipher/logo...	672d172cc5fc1072	Diana
2	2	c2_logoEscom.jpg	cipher/c2_lo...	672d172cc5fc1072	Diana
3	3	One.mp3	cipher/One....	a2970d43aa576d0d	Diana
4	4	c2_One.mp3	cipher/c2_O...	a2970d43aa576d0d	Diana
5	5	c2_One.mp3	cipher/c2_O...	a2970d43aa576d0d	Jhonatan
6	6	c2_escom.jpg	cipher/c2_e...	672d172cc5fc1072	Jhonatan

The right side of the interface shows the "DB Schema" pane, which lists 14 tables and their corresponding CREATE TABLE statements. The tables listed are: auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_migrations, and django_session.

Figura 6.25: Tabla Cipher actualizada.

Tabla Hashes actualizada

The screenshot shows the DB Browser for SQLite interface. The main window displays a table named 'nube_hashes' with three columns: 'id', 'filename', and 'hash_c'. There are two rows of data:

id	filename	hash_c
1	logoEscom.jpg	hash/logoEscom.jpg.sha 672d172cc5fc1072
2	One.mp3	hash/One.mp3.sha a2970d43aa576d0d

The right side of the interface shows the database schema with 14 tables listed under 'Tables (14)'. The bottom right corner indicates the encoding is 'UTF-8'.

Figura 6.26: Tabla Hashes actualizada.

6.6.14. Perfil de usuario1 actualizado con distintos formatos de archivos

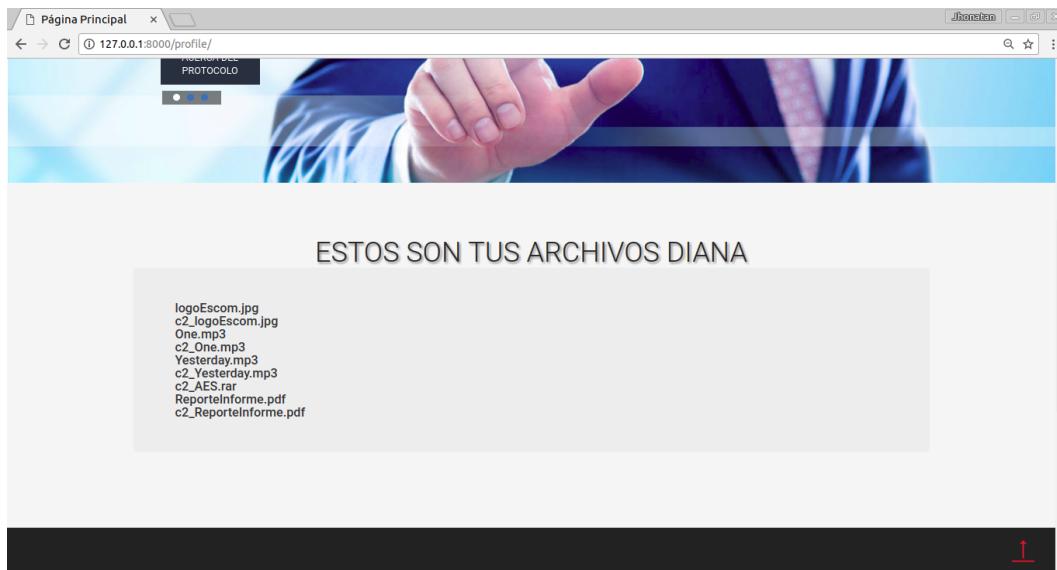


Figura 6.27: Pantalla donde se muestran distintos tipos de archivos cifrados almacenados en la nube.

Perfil de usuario2 actualizado con distintos formatos de archivos

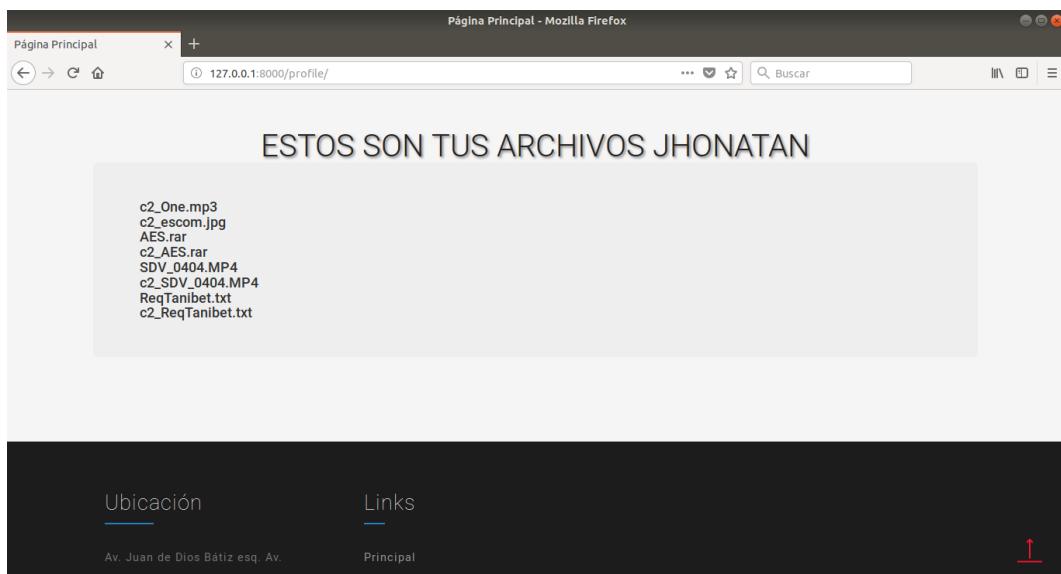


Figura 6.28: Pantalla donde se muestran distintos tipos de archivos cifrados almacenados en la nube.

Tabla Cipher actualizada

The screenshot shows the DB Browser for SQLite interface with the database file `/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db`. The "Database Structure" tab is selected, showing the "nube_cipher" table. The table has four columns: "id", "filename", "docfile", and "hash_c". The "user_name" column is also present but appears to be empty. The table contains 17 rows of data, each corresponding to one of the files listed in Figure 6.28. The "DB Schema" tab is open on the right, displaying the schema for the database, which includes various tables like auth_group, auth_permission, auth_user, etc., with their respective CREATE TABLE statements.

id	filename	docfile	hash_c	user_name
3	c2_One.mp3	cipher/c2_One.mp3.aes	a2970d43aa576d0d	Diana
4	c2_One.mp3	cipher/c2_One.mp3.aes	a2970d43aa576d0d	Diana
5	c2_One.mp3	cipher/c2_One.mp3_b1b8... a2970d43aa576d0d	jhonatan	
6	c2_escom.jpg	cipher/c2_escom.jpg.aes	672d172cc5fc1072	jhonatan
7	AES.rar	cipher/AES.rar.aes	905c60453a3f202c	jhonatan
8	c2_AES.rar	cipher/c2_AES.rar.aes	905c60453a3f202c	jhonatan
9	SDV_0404.MP4	cipher/SDV_0404.MP4.aes	02fbcd448336f779	jhonatan
10	c2_SDV_0404.MP4	cipher/c2_SDV_0404.MP4... 02fbcd448336f779	jhonatan	
11	ReqTanibet.txt	cipher/ReqTanibet.txt.aes	1f84ac44bb539da3	jhonatan
12	c2_ReqTanibet.txt	cipher/c2_ReqTanibet.txt... 1f84ac44bb539da3	jhonatan	
13	Yesterday.mp3	cipher/Yesterday.mp3.aes	eefb5dcde99826c49	Diana
14	c2_Yesterday.mp3	cipher/c2_Yesterday.mp3... eefb5dcde99826c49	Diana	
15	c2_AES.rar	cipher/c2_AES.rar_oXach... 905c60453a3f202c	Diana	
16	ReportelInforme.pdf	cipher/ReportelInforme.pdf... 3bebfcf9034d38ba	Diana	
17	c2_ReportelInforme...	cipher/c2_ReportelInforme... 3bebfcf9034d38ba	Diana	

Figura 6.29: Tabla Cipher actualizada.

Tabla Hashes actualizada

The screenshot shows the DB Browser for SQLite interface. The main window displays a table named 'nube_hashes' with the following data:

	id	filename	docfile	hash_c
1	1	logosEscom.jpg	hash/logoEscom.jpg.sha	672d172cc5fc1072
2	2	One.mp3	hash/One.mp3.sha	a2970d43aa576d0d
3	3	AES.rar	hash/AES.rar.sha	905c60453a3f202c
4	4	SDV_0404.MP4	hash/SDV_0404.MP4.sha	02fbcd448336f779
5	5	ReqTaniBet.txt	hash/ReqTaniBet.txt.sha	1f84ac44bb539da3
6	6	Yesterday.mp3	hash/Yesterday.mp3.sha	efb5dcde99826c49
7	7	ReporteInforme.pdf	hash/ReporteInforme.pdf...	3bebfcce9034d38ba

Figura 6.30: Tabla Hashes actualizada.

6.6.15. Descargar Archivo

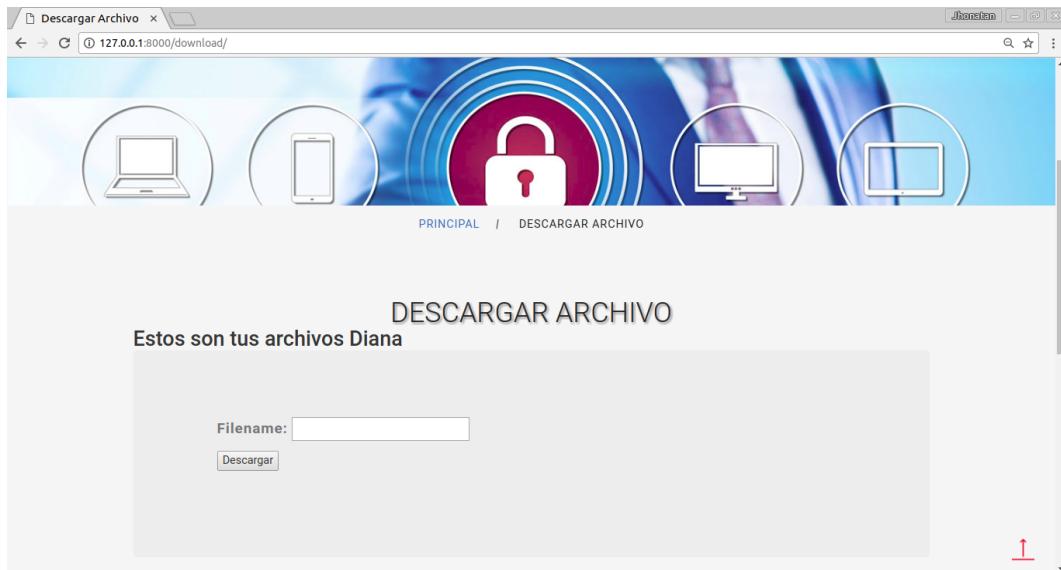


Figura 6.31: Pantalla donde se muestran el formulario donde se puede descargar los archivos.

Seleccionando Archivo



Figura 6.32: Pantalla donde se muestra el archivo seleccionado que se desea descargar.

Archivo Descargado

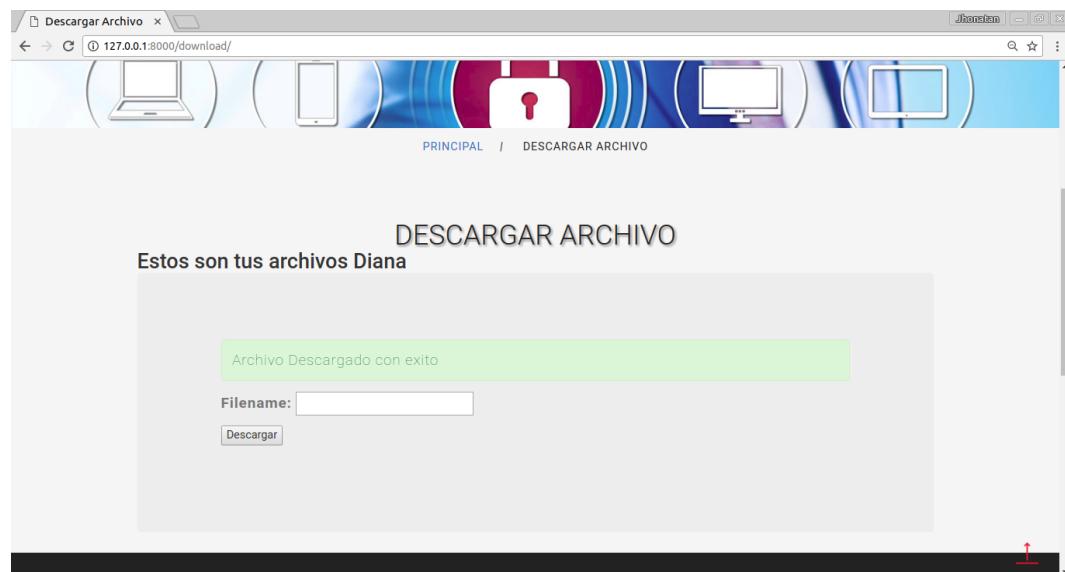


Figura 6.33: Pantalla donde se muestra que se descargo correctamente el archivo.

Archivo Descargado en la Carpeta

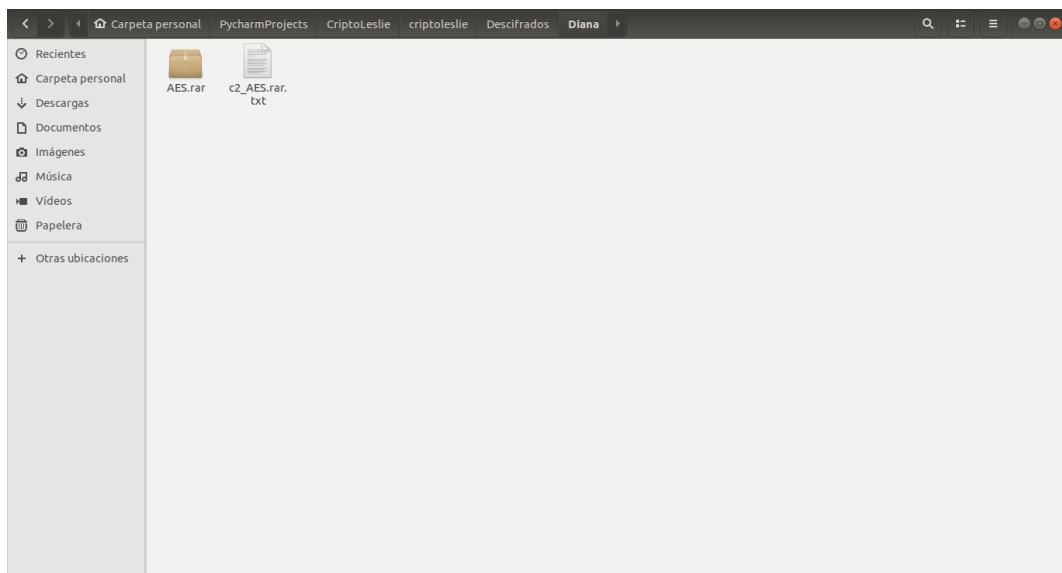


Figura 6.34: Pantalla donde se muestra el archivo en la carpeta donde se descargo.

Archivos Descargados del usuario 1 en la Carpeta

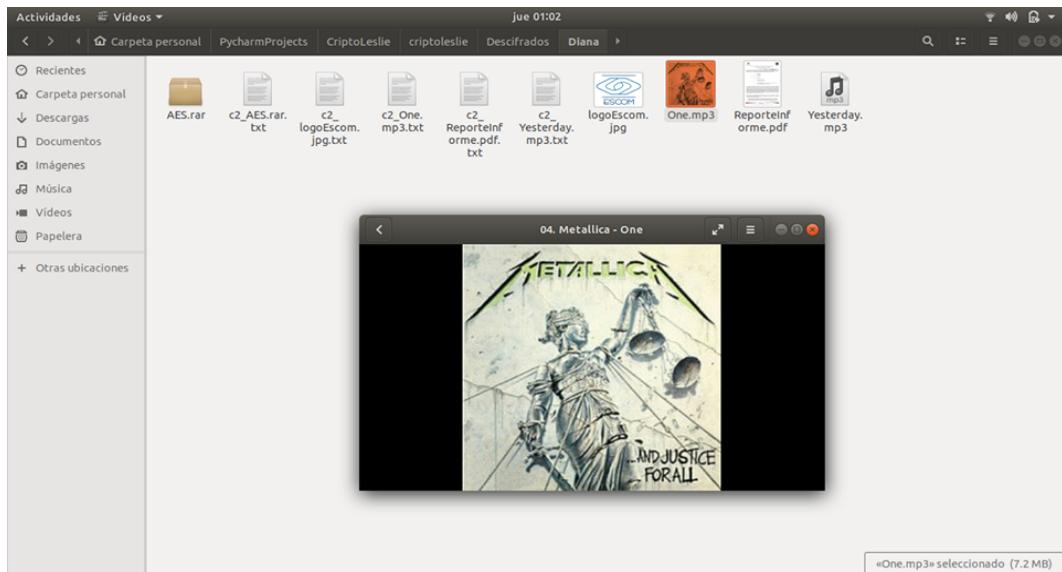


Figura 6.35: Pantalla donde se muestran los archivos en la carpeta donde se descargo del usuario 1.

Archivos Descargados del usuario 2 en la Carpeta



Figura 6.36: Pantalla donde se muestran los archivos en la carpeta donde se descargo del usuario 2.

6.6.16. Eliminar Archivo

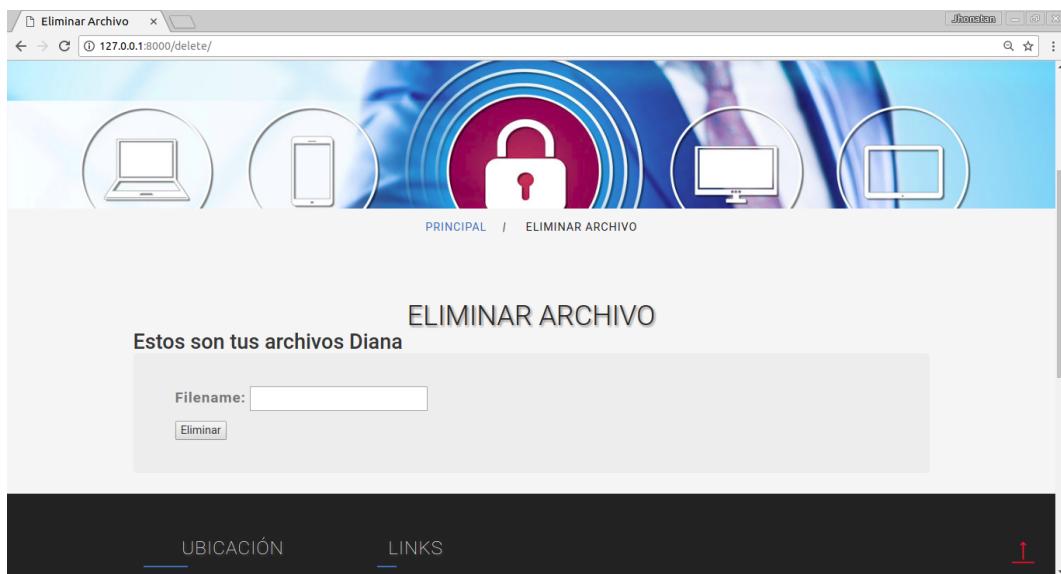


Figura 6.37: Pantalla donde se muestra el formulario donde se puede eliminar un archivo.

Seleccionando Archivo a eliminar

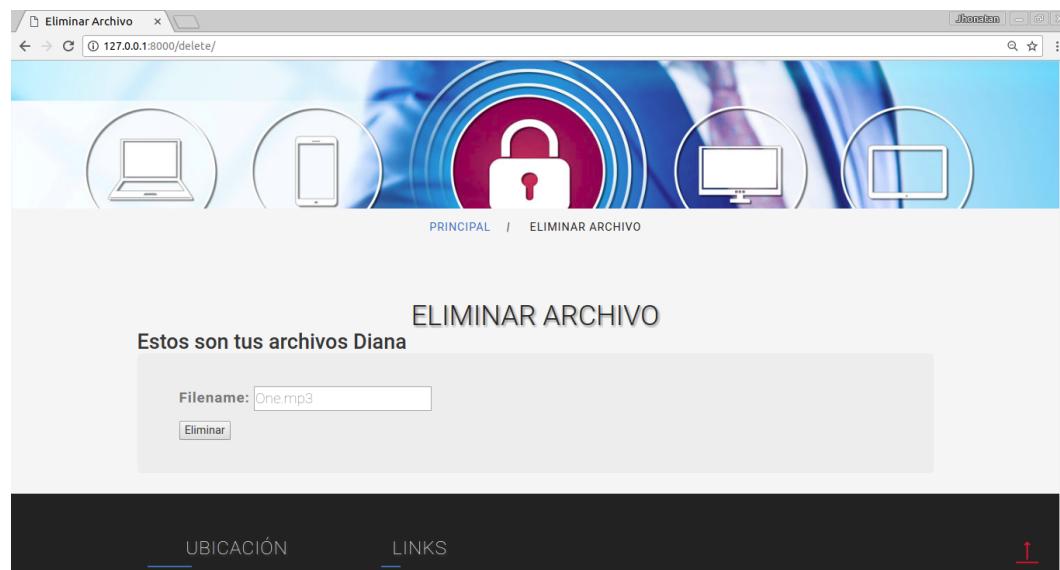


Figura 6.38: Pantalla donde se muestra el formulario donde se eligió un archivo a eliminar.

Tabla Cipher con Archivo eliminado de usuario 1

The screenshot shows the 'DB Browser for SQLite' interface connected to a database at '/home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The main window displays a table named 'nube_cipher' with the following data:

	id	filename	docfile	hash_c	user_name
1	1	logoEscom.jpg	cipher/logoEscom.jpg.aes	672d172cc5fc1072	Diana
2	2	c2_logoEscom.jpg	cipher/c2_logoEscom.jpg....	672d172cc5fc1072	Diana
3	3	One.mp3	cipher/One.mp3.aes	a2970d43aa576dd0d	Diana
4	5	c2_One.mp3	cipher/c2_One.mp3_b1U8n...	a2970d43aa576dd0d	Jhonatan
5	6	c2_escom.jpg	cipher/c2_escom.jpg.aes	672d172cc5fc1072	Jhonatan
6	7	AES.rar	cipher/AES.rar.aes	905c60453a3f202c	Jhonatan
7	8	c2_AES.rar	cipher/c2_AES.rar.aes	905c60453a3f202c	Jhonatan
8	9	SDV_0404.MP4	cipher/SDV_0404.MP4.aes	02fbcd448336f779	Jhonatan
9	10	c2_SDV_0404.MP4	cipher/c2_SDV_0404.MP4....	02fbcd448336f779	Jhonatan
10	11	ReqTaniBet.txt	cipher/ReqTaniBet.txt.aes	1f84ac44bb539da3	Jhonatan
11	12	c2_ReqTaniBet.txt	cipher/c2_ReqTaniBet.txt....	1f84ac44bb539da3	Jhonatan
12	13	Yesterday.mp3	cipher/Yesterday.mp3.aes	efb5dcde99826c49	Diana
13	14	c2_Yesterday.mp3	cipher/c2_Yesterday.mp3....	efb5dcde99826c49	Diana
14	15	c2_AES.rar	cipher/c2_AES.rar_oEXach...	905c60453a3f202c	Diana
15	16	ReporteInforme.pdf	cipher/ReporteInforme.pdf...	3bebfcf0934d38ba	Diana

To the right of the table, there is a 'Edit Database Cell' panel showing the current value as 'NULL' and a 'DB Schema' panel listing various tables and their creation scripts.

Figura 6.39: Pantalla donde se muestra la base de datos ya con el archivo eliminado.

Pantalla archivos actualizados

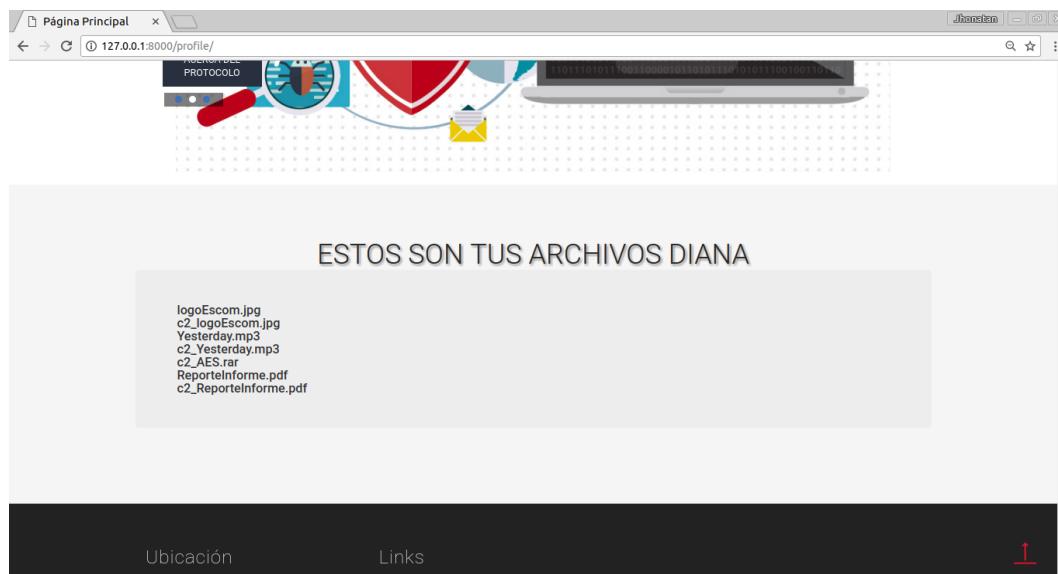


Figura 6.40: Pantalla donde se muestra la lista de archivos actualizada despues de eliminar el archivo.

Tabla Cipher con Archivo eliminado de usuario 2

The screenshot shows the 'DB Browser for SQLite' application interface. The title bar reads 'DB Browser for SQLite - /home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db'. The main window displays a table named 'nube_cipher' with the following data:

	id	filename	docfile	hash_c	user_name
1	1	logoEscom.jpg	cipher/logoEscom.jpg.aes	672d172cc5fc1072	Diana
2	2	c2_logoEscom.jpg	cipher/c2_logoEscom.jpg.aes	672d172cc5fc1072	Diana
3	6	c2_escom.jpg	cipher/c2_escom.jpg.aes	672d172cc5fc1072	Jhonatan
4	7	AES.rar	cipher/AES.rar.aes	905c60453a3f202c	Jhonatan
5	8	c2_AES.rar	cipher/c2_AES.rar.aes	905c60453a3f202c	Jhonatan
6	9	SDV_0404.MP4	cipher/SDV_0404.MP4.aes	02fbcd448336f779	Jhonatan
7	10	c2_SDV_0404.MP4	cipher/c2_SDV_0404.MP4.aes	02fbcd448336f779	Jhonatan
8	11	ReqTanibet.txt	cipher/ReqTanibet.txt.aes	1f84ac44bb539da3	Jhonatan
9	12	c2_ReqTanibet.txt	cipher/c2_ReqTanibet.txt.aes	1f84ac44bb539da3	Jhonatan
10	13	Yesterday.mp3	cipher/Yesterday.mp3.aes	efb5dcde99826c49	Diana
11	14	c2_Yesterday.mp3	cipher/c2_Yesterday.mp3.aes	efb5dcde99826c49	Diana
12	15	c2_AES.rar	cipher/c2_AES.rar_oEXachH...	905c60453a3f202c	Diana
13	16	ReporteInforme.pdf	cipher/ReporteInforme.pdf.aes	3bebfcf9034d38ba	Diana
14	17	c2_ReporteInforme.pdf	cipher/c2_ReporteInforme.pdf.aes	3bebfcf9034d38ba	Diana

The right side of the interface shows the 'Edit Database Cell' panel with 'Mode: Text' selected. The 'DB Schema' panel lists the database schema with various tables and their creation scripts.

Figura 6.41: Pantalla donde se muestra la base de datos ya con el archivo eliminado en los dos usuarios.

Pantalla archivos actualizados de usuario 2

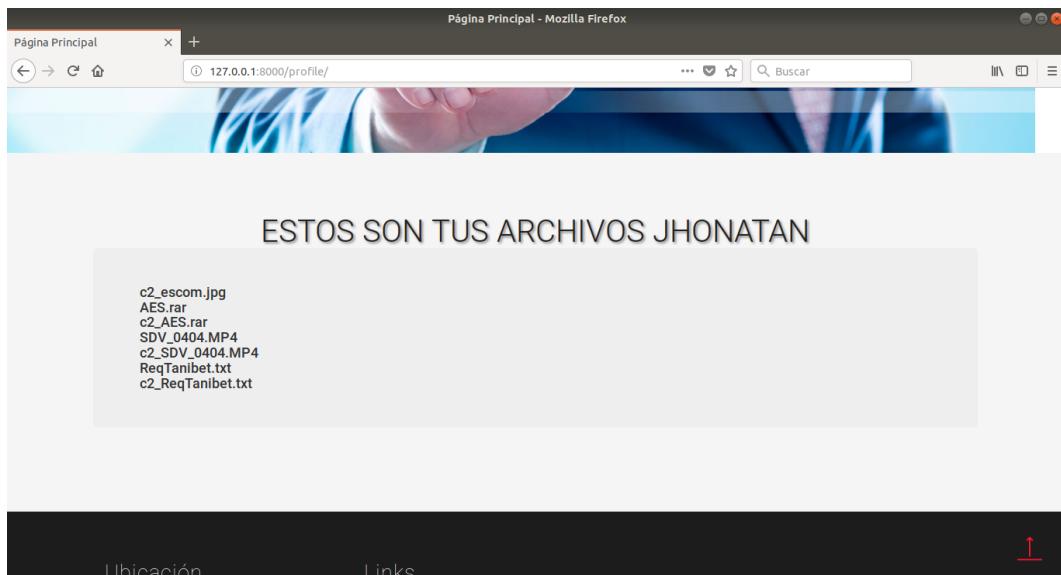


Figura 6.42: Pantalla donde se muestra la lista de archivos actualizada despues de eliminar el archivo del usuario 2.

Tabla Hashes con Archivo eliminado de los dos usuarios

The screenshot shows the DB Browser for SQLite interface with a database named /home/jhonatan/PycharmProjects/CriptoLeslie/criptoleslie/owncloud.db. The main window displays the "nube_hashes" table with the following data:

	id	filename	docfile	hash_c
1	1	logoEscom.jpg	hash/logoEscom.jpg.sha	672d172cc5fc1072
2	3	AES.rar	hash/AES.rar.sha	905c60453a3f202c
3	4	SDV_0404.MP4	hash/SDV_0404.MP4.sha	02fbcd448336f779
4	5	ReqTanibet.txt	hash/ReqTanibet.txt.sha	1f84ac44bb539da3
5	6	Yesterday.mp3	hash/Yesterday.mp3.sha	efb5dcde99826c49
6	7	ReporteInforme.pdf	hash/ReporteInforme.pdf...	3bebfcf9034d38ba

The right side of the interface shows the "DB Schema" pane, which lists the tables in the database, including auth_group, auth_group_permissions, auth_permission, auth_user, auth_user_groups, auth_user_user_permissions, django_admin_log, django_content_type, django_migrations, and django_session.

Figura 6.43: Pantalla donde se muestra la base de datos en la tabla de Hashes despues de haber eliminado un archivo por completo

Capítulo 7

Lista de acrónimos

7.1. Definiciones, acrónimos y abreviaturas

Acrónimos

- HP: Hewlett-Packard.
- DupLESS: Server-Aided Encryption for Deduplicated Storage (Cifrado Asistido por un Servidor para Almacenamiento Sin Duplicados).
- OPRF: Oblivious Pseudorandom Function (Función Pseudoaleatoria Inconsistente)
- MLE: Message-Locked Encryption (Cifrado por bloqueo de mensajes)
- ABS: The Apportioned Backup System (Sistema de Respaldo Asignado).
- SIGOPS: Special Interest Group on Operating Systems (Grupo de Interés Especial sobre Sistemas Operativos).
- TahoeFS: The Least-Authority Filesystem (Sistema de Archivos de Menor Autoridad).
- AES: Advanced Encryption Standard (Estándar de Cifrado Avanzado).
- DES: Data Encryption Standard (Estándar de Cifrado de Datos).
- RSA: Rivest Shamir Adleman.
- MD: Message Digest (Resumen del Mensaje).
- SHA: Secure Hash Algorithm (Algoritmo Seguro de Hash).
- NIST: National Institute of Standards and Technology (Instituto Nacional de Estándares y Tecnología).
- SaaS: Software as a Service (Software como Servicio).

- PaaS: Platform as a Service (Plataforma como Servicio).
- IaaS: Infrastructure as a Service (Infraestructura como Servicio).
- API: Application Programming Interface (Interfaz de Programación de Aplicaciones).
- BPMN: Business Process Model and Notation (Modelo de Proceso Empresarial y Notación).

Referencias

- [1] M. Bellare, S. Keelveedhi, and T. Ristenpart. Dupless: Server-aided encryption for deduplicated storage. *IACR Cryptology ePrint Archive*, 2013:429, 2013.
- [2] R. Bellare, Keelveedhi. *Message-locked encryption and secure deduplication.*, volume 7881. EUROCRYPT, 2013.
- [3] T. C. y. P. A. Cooley J. Abs: the apportioned backup system. MIT Laboratory for Computer, 2004. Disponible.
- [4] A. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
- [5] C. Paar and J. Pelzl. *Understanding Cryptography - A Textbook for Students and Practitioners*. Springer, 2010.
- [6] G. Z. C. Patricia. Diseño y desarrollo de un sistema para elecciones electrónicas seguras (seles). Master's thesis, Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2005.
- [7] D. J. R., A. A., B. W. J., S. D., and T. M. *Reclaiming space from duplicate files in a serverless distributed file system*. ICDCS, 2002.
- [8] S/A. Cifrado simetrico. *Guía de Gnu Privacy Guard*, 2015. Disponible en: <https://www.gnupg.org/gph/es/manual/c190.html#AEN201>.
- [9] T. O. Sergio. Introducción a la criptología. *InfoCentreUV*, 2003.
- [10] I. Sommerville. *Software Engineering, 6. Auflage*. Pearson Studium, 2001.
- [11] W. Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson Education, 5a edition, 2002.
- [12] D. R. Stinson. *Cryptography - theory and practice*. Discrete mathematics and its applications series. CRC Press, 1995.
- [13] H. D. y. W. N. Wilcox-O'Hearn Z. Tahoe: *The least-authority*. In Proceedings of the 4th ACM, 2008.

- [14] E. A. M. y M.C. Ma. Jaquelina López Barrientos. "*fundamentosdecriptografia*". Universidad Nacional Autónoma de México, 2012. Disponible en:<http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/index.php/1-panorama-general/11-concepto-de-criptografia>.
- [15] E. A. M. y M.C. Ma. Jaquelina López Barrientos. "*fundamentosdeseguridadinformtica*". Universidad Nacional Autónoma de México, 2012. Disponible en: <http://redyseguridad.fi-p.unam.mx/proyectos/criptografia/criptografia/index.php/1-panorama-general/14-ataques/142-ataques-a-los-metodos-de-cifrado>.