UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

ASEGURAMIENTO DEL INTERCAMBIO DE DATOS ENTRE UNA APLICACIÓN MÓVIL ANDROID Y UNA APLICACIÓN WEB JAVA MEDIANTE CÓDIGOS QR, EN BASE AL ESTÁNDAR ISO/IEC 27002.

Trabajo de grado previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autora:

Dayana Elizabeth Guerra Guzmán

Director:

Msc. Rea Peñafiel Xavier Mauricio

Ibarra - Ecuador

Julio 2019

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información.

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003992375-9		
APELLIDOS Y NOMBRES:	GUERRA GUZMÁN DAYANA ELIZABETH		
DIRECCIÓN:	IBARRA (ARGENTINA Y BOLIVIA)		
EMAIL:	deguerrag@utn.edu.ec		
TELÉFONO FIJO:	(062) 540-081	**TELÉFONO MÓVIL:**	0985119754

DATOS DE LA OBRA	
TÍTULO:	ASEGURAMIENTO DEL INTERCAMBIO DE DATOS ENTRE UNA APLICACIÓN MÓVIL ANDROID Y UNA APLICACIÓN WEB JAVA MEDIANTE CÓDIGOS QR, EN BASE AL ESTÁNDAR ISO/IEC 27002.
AUTOR:	GUERRA GUZMÁN DAYANA ELIZABETH
FECHA:	29/07/2019
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERA EN SISTEMAS COMPUTACIONALES
DIRECTOR:	MSc. REA PEÑAFIEL XAVIER MAURICIO
ASESOR:	PHD. IVÁN GARCÍA
ASESOR:	MSc. CARPIO PINEDA

2. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de esta y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 29 días del mes de Julio del 2019

Nombre: Dayana Elizabeth Guerra Guzmán
Cédula: 1003992375

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE GRADO A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, GUERRA GUZMÁN DAYANA ELIZABETH, con cédula de identidad Nro. 1003992375 manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la ley de propiedad intelectual del Ecuador, artículo 4, 5 y 6, en calidad de autora del proyecto de grado denominado: "ASEGURAMIENTO DEL INTERCAMBIO DE DATOS ENTRE UNA APLICACIÓN MÓVIL ANDROID Y UNA APLICACIÓN WEB JAVA MEDIANTE CÓDIGOS QR, EN BASE AL ESTÁNDAR ISO/IEC 27002.", que ha sido desarrollado para optar por el título de Ingeniera en Sistemas Computacionales, en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autora me reservo los derechos morales de la obra antes citada. En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Nombre: Dayana Elizabeth Guerra Guzmán
Cédula: 1003992375
Ibarra, 29 de julio del 2019

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

DECLARACIÓN

Yo, GUERRA GUZMÁN DAYANA ELIZABETH, declaro bajo juramento que el trabajo aquí descrito es de mi autoría y que este no ha sido previamente presentado para ningún grado o calificación profesional.

A través de la presente declaración cedo los derechos de propiedad intelectual correspondientes a este trabajo, a la Universidad Técnica del Norte, según lo establecido por las Leyes de la Propiedad Intelectual, Reglamentos y Normatividad vigente de la Universidad Técnica del Norte.

Nombre: Dayana Elizabeth Guerra Guzmán
Cédula: 1003992375
Ibarra, 29 de julio del 2019

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Certifico que el trabajo de grado “ASEGURAMIENTO DEL INTERCAMBIO DE DATOS ENTRE UNA APLICACIÓN MÓVIL ANDROID Y UNA APLICACIÓN WEB JAVA MEDIANTE CÓDIGOS QR, EN BASE AL ESTÁNDAR ISO/IEC 27002.”, ha sido desarrollado en su totalidad por la señorita: Dayana Elizabeth Guerra Guzmán, portadora de la cédula de identidad número 1003992375.

.....

ING. MGS. MAURICIO REA

DIRECTOR DE TRABAJO DE GRADO

Ibarra, 22 de julio de 2019

Ingeniero
Pedro Granda
COORDINADOR DE LA CARRERA CISIC/CSOFT

Por medio de la presente me permito informar que el software producto de la elaboración de la tesis "Aseguramiento del intercambio de datos entre una aplicación móvil Android y una aplicación web java mediante códigos QR, en base al estándar ISO/IEC 27002" realizado por la señorita Dayana Elizabeth Guerra Guzmán con cédula de identidad número 100399237-5, se encuentra instalado en el servidor de la carrera, ya que es de utilidad tanto como caso de estudio, así como de apoyo para temas de clase relacionados a seguridades y encriptación.

El software se encuentra funcional y el código fuente se ha registrado en el repositorio de versionamiento de proyectos de software de la carrera.

Atentamente,

Mgs. Mauricio Rea
DOCENTE FICA UTN

Ing. Mauricio Rea
DIRECTOR TESIS

Dedicatoria

Este trabajo está dedicado a mi mamá Marianita Guzmán, a mi papá Iván Guerra, quienes me apoyaron incondicionalmente en el transcurso de la carrera y a mis hermanos y amigos por representar un impulso en el camino, gracias a ello logre alcanzar el objetivo de culminar mis estudios obteniendo el título de Ingeniera en Sistemas Computacionales.

Agradecimiento

Agradezco a todos aquellos quienes me apoyaron para alcanzar este objetivo, en especial a mis papás por estar siempre con una voz de aliento y por ser el pilar más importante de mi vida, de ellos aprendí que la constancia es una de las claves del éxito y que las cosas se deben ganar con esfuerzo y perseverancia, siempre les estaré muy agradecida, y a mi tutor por su apoyo incondicional, sus enseñanzas y por guiarme en este proceso.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

RESUMEN

Considerando el impacto que tienen las aplicaciones móviles y los sistemas web actualmente, surge el cuestionamiento sobre la seguridad del flujo de información que manejan los usuarios cada día con el uso de las mismas, consecuentemente nace la necesidad de buscar la forma de mejorar la seguridad del intercambio de información de dichas tecnologías utilizando métodos y herramientas que se mantengan vigentes en el plano tecnológico, de lo contrario se estaría perdiendo oportunidades en el mercado y minimizando la calidad de los sistemas, con lo que se convertirían en tecnologías obsoletas.

La innovación tecnológica hace posible la construcción de nuevas herramientas y métodos que ayudan en la solución de diversos problemas que aquejan a la población mundial.

La finalidad de este proyecto de grado es realizar un estudio sobre métodos de cifrado, los cuales permitirán asegurar el intercambio de información entre aplicaciones móviles Android y aplicaciones web java utilizando códigos QR, lo que ayudará a los programadores a darle un plus de calidad a los sistemas que se desarrollen a futuro.

El estudio se enfocó directamente en el aseguramiento de la información con la utilización de métodos de cifrado, inclinándose en el uso de códigos QR y el hecho de que no todas las aplicaciones móviles y sistemas web cuentan con métodos que aseguren el intercambio de los datos que manejan. El análisis realizado entre los métodos propuestos establece que, en base a los parámetros evaluados, AES es más eficiente que RSA.

Este trabajo se realizó en base a la ISO/IEC 27002, apoyado específicamente en la directriz 10, que se refiere al Cifrado de datos.

Abstract

Considering the impact that mobile applications and web systems have today, the question arises about the security of the flow of information that users use every day with the use of them, consequently the need arises to find a way to improve the security of the exchange of information on these technologies using methods and tools that remain in force at the technological level, otherwise it would be losing opportunities in the market and minimizing the quality of the systems, with which they would become obsolete technologies.

Technological innovation makes possible the construction of new tools and methods that help in the solution of various problems that afflict the world population.

The purpose of this degree project is to carry out a study on encryption methods, which will allow the exchange of information between Android mobile applications and java web applications using QR codes, which will help programmers to give an extra quality to the systems that are developed in the future.

The study focused directly on securing information with the use of encryption methods, based on the use of QR codes and on the fact that not all mobile applications and web systems have methods that ensure the exchange of data they handle. The analysis carried out among the proposed methods establish that, based on the parameters evaluated, AES is more efficient than RSA.

This work was carried out based on ISO / IEC 27002, specifically supported in guideline 10, which refers to data encryption.

INTRODUCCIÓN

PROBLEMA

- **ANTECEDENTES**

La creación de códigos QR nace por la necesidad de aligerar los procesos de diferentes industrias. En 1994, Denso Wave anunció el lanzamiento de su código QR (Quick Response[1]). Es una matriz en dos dimensiones formada por una serie de cuadrados negros sobre fondo blanco, esta matriz es leída por un lector específico para QR en los dispositivos móviles con gran rapidez (Tomás, 2012). A partir de allí han sido implementados en múltiples tareas, sin embargo, una de las debilidades de estos códigos es la seguridad de la información que contienen, pueden ser textos, imágenes o audios.

Existen proyectos dentro del país en donde en los últimos diez años se ha hecho uso de los códigos QR, con aplicaciones móviles y aplicaciones web, en ciudades como Quito y Guayaquil en donde se ha implementado en el área del deporte y la industria respectivamente (Jami, 2015).

A nivel tecnológico existen proyectos conocidos sobre el tema a tratar, sin embargo, estos proyectos o aplicaciones forman parte de grandes empresas por lo que el código es privado. Pese a que los códigos QR están teniendo un impacto notable, no son lo suficientemente conocidos para que la gente haga uso frecuente de ellos o se implementen para proyectos locales o regionales. El proyecto planteado está enfocado al área de seguridad de la información, con lo que pretende cumplir con el objetivo 16, Paz, Justicia en Instituciones Sólidas de los objetivos de desarrollo sostenible (Censos).

- **SITUACION ACTUAL**

Actualmente existen muchas aplicaciones móviles con la capacidad de escanear y descifrar estos códigos y descubrir la información que contienen (Caneda, 2011), sin embargo, a pesar de estar cifrados, los lectores de código pueden descifrarlos y obtener la información sin ningún tipo de privacidad, lo que representa un aspecto negativo si la información es confidencial y se requiere que sólo algunos usuarios puedan acceder a ella (Pliego García, 2013). Ante esta situación nace la necesidad de investigar la forma en la que

[1] quick response: respuesta rápida, hace referencia a la rápida reposición a un cliente por parte de un proveedor con acceso directo a datos.

se generen códigos QR que sólo pueda ser descifrado por los usuarios a quienes va dirigida la información que contiene y a la vez asegurar la información que la aplicación móvil envía al servidor de la aplicación web en la que se muestra y genera el código QR.

- **PROSPECTIVA**

Para asegurar el intercambio de datos entre aplicaciones web y aplicaciones móviles mediante códigos QR, es necesario un estudio de métodos de aseguramiento de datos, actualmente se vive un mundo tecnológico en el que cada vez existen más formas de alterar o hurtar la información de los usuarios de distintas tecnologías, por este y otros motivos es necesario el estudio de un método de aseguramiento de datos (J, 2012).

Este proyecto espera asegurar el intercambio de datos entre aplicaciones web y aplicaciones móviles. El uso de estas tecnologías evitará que exista alteración de la información de los usuarios, (CUC, 2013) con la implementación de esta investigación se logrará fortalecer la seguridad en la transmisión de datos, mejorando la funcionalidad de los sistemas, tratar de que la experiencia de los usuarios sea satisfactoria sabiendo que su información está protegida, lo que conlleva a mejorar calidad de vida de la sociedad en general, ya que se vive en un mundo tecnológico.

- **PLANTEAMIENTO DEL PROBLEMA**

Los códigos QR tienen mayor capacidad para guardar información que otros códigos, incluso se puede mejorar la calidad de los archivos que almacena, las imágenes por ejemplo (Kuen-Tsair Lay, 2018), por lo que son mundialmente utilizados para agilizar múltiples procesos en diferentes áreas, no obstante, la información que contienen no tiene restricción o privacidad para los usuarios, así como la información que envía la aplicación móvil a un determinado servidor, lo cual se convierte en una debilidad de esta tecnología (Casanova Pastor, 2013). Así, nace la idea de asegurar la información de manera bidireccional, es decir, asegurar la información que contiene el código QR y la información que se envía al servidor web.

OBJETIVOS

- **OBJETIVO GENERAL**

Asegurar el intercambio de datos entre una aplicación móvil Android[2] y una aplicación web java[3] mediante códigos QR, en base al estándar ISO[4]/IEC[5] 27002.

- **OBJETIVOS ESPECÍFICOS**

  - Realizar un estudio comparativo sobre los métodos de cifrado Rivest, Shamir, Adleman (RSA[6]) y Advanced Encryption Standard (AES[7]), los cuales permitirán asegurar la información que se transmite entre una aplicación Android y una aplicación web java mediante códigos QR, en base al estándar ISO/IEC 27002. La directriz 10, Cifrado y la directriz 13, Seguridad de las Comunicaciones.
  - Utilizar los métodos de cifrado AES y RSA, para asegurar el intercambio de datos entre una aplicación web y una aplicación móvil mediante códigos QR.
  - Validar los resultados del proyecto.

ALCANCE Y LIMITACIONES

- **ALCANCE**

El presente proyecto tiene como finalidad implementar un método de cifrado de datos para fortalecer el intercambio de información entre aplicaciones móviles y aplicaciones web, el flujo de datos es bidireccional, por tanto, el aseguramiento también será bidireccional, de tal forma que la información viaje cifrada y sea fuerte ante ataques de hackers o peligros en la red. Así, los datos que sean leídos por la aplicación móvil viajan cifrados y la información que devuelve la aplicación móvil a la aplicación web y por tanto al servidor mediante servlets también este cifrada.

[2] android: nombre de un sistema operativo que se emplea en dispositivos móviles.
[3] java: lenguaje de programación y una plataforma informática.
[4] ISO: organización internacional de estandarización.
[5] IEC: comisión electrotécnica internacional.
[6] RSA: iniciales de los apellidos de los desarrolladores: Rivest, Shamir y Adleman.
[7] AES: estándar de cifrado avanzado.

Este proyecto estará conformado por una aplicación web java y una aplicación móvil Android. Es bastante común el desarrollo de este tipo de aplicaciones en los proyectos de tesis desarrollados en la carrera de sistemas computacionales de la Universidad Técnica del Norte, por lo que, este estudio es de aplicación directa por los estudiantes de la carrera mencionada. Luego de un estudio se determinará el método estadístico que permitirá validar los resultados.

Figura 1: Diagrama de la Implementación del método de aseguramiento
Fuente: Propia

JUSTIFICACIÓN

El presente proyecto tiene un enfoque hacia los objetivos de desarrollo sostenible:

Nº16: Paz, Justicia e Instituciones Sólidas:

16.10 Garantizar el acceso público a la información y proteger las libertades fundamentales, de conformidad con las leyes nacionales y los acuerdos internacionales (Moran, Desarrollo Sostenible, Paz y Justicia, 2016).

Nº9: Industria, Innovación e Infraestructura:

9.b Apoyar el desarrollo de tecnologías, la investigación y la innovación nacionales en los países en desarrollo, incluso garantizando un entorno normativo propicio a la diversificación industrial y la adición de valor a los productos básicos, entre otras cosas.

9.c Aumentar significativamente el acceso a la tecnología de la información y las comunicaciones y esforzarse por proporcionar acceso universal y asequible a Internet en los países menos adelantados de aquí a 2020 (Moran, Infraestructura, 2016).

Este proyecto se justifica por la falta de estudios sobre el aseguramiento del intercambio de datos entre aplicaciones móviles Android y aplicaciones Web Java mediante Códigos QR.

En la localidad no se han hecho estudios o implementaciones del tema, no obstante existen empresas como WhatsApp que ofrece sus servicio de “WhatsApp Web” en donde utiliza estas tecnologías, el problema es que no son de código abierto, es decir no están disponibles para los usuarios (Pastor, 2016), por lo que este proyecto será de gran utilidad para los desarrolladores de software que desarrollen aplicaciones móviles y necesiten transmitir datos a una aplicación web, porque contarán con un método de aseguramiento de la información, sin privaciones de ningún tipo que les ayudará a tener mayor seguridad al momento transmitir información.

Económico

El impacto a nivel económico implica evitar gastos que se pueden producir en la corrección de errores de seguridad en los sistemas. Asegurar la transmisión de datos evita fallos y alteraciones de la información que consecuentemente podría representar grandes pérdidas económicas a las organizaciones y usuarios individuales.

Ambiental

Los sistemas de automatización de procesos ayudan a reducir el consumo de recursos, principalmente de papel con lo que se aporta directamente a la preservación del medio ambiente.

Social

Tiene un alto impacto en la parte social, teniendo seguridad de datos los usuarios podrán tener una buena experiencia y mayor tranquilidad utilizando las tecnologías mencionadas anteriormente, lo cual genera un ambiente de bienestar colectivo.

Los beneficiarios directos de este proyecto serán los propietarios de celulares con sistema operativo Android en los cuales se implemente la aplicación y hagan uso diario de este tipo de tecnologías.

1. CAPÍTULO I: MARCO TEÓRICO

1.1. Criptografía

1.1.1. ¿Qué es Criptografía?

La criptografía es la ciencia que estudia la escritura oculta, proviene de un sentido etimológico del griego *Kriptos*=ocultar, *Graphos*=escritura, es la habilidad de escribir en un lenguaje convenido utilizando claves o cifras numéricas, esta ciencia enseña a diseñar cifrarios[8], también conocidos como códigos secretos construidos por los criptógrafos,(Díaz, 1995) de tal manera que el mensaje que se envía sea legible únicamente para el destinatario deseado, esto garantiza la privacidad de la información enviada ya que, si algún intruso o destinatario no deseado intercepta el mensaje, se encontrará con un mensaje sin sentido.

Así mismo, la labor de transformar texto cifrado en el mensaje original cuando se conoce la clave se denomina “descifrar” o “decodificar”, por otro lado, cuando se desconoce la clave, el término adecuado para esto es “perlustrar” o más conocido como “descriptar”. Sin duda, conocer esta terminología es importante al hablar del tema, por lo que debe quedar claro la distinción entre “descifrar” y “descriptar”: si el destinatario legítimo conoce la clave secreta, entonces “descifra” el criptograma, de no ser así, el destinatario no deseado lo intentará “descriptar” (Wallis, 2013).

1.1.2. Historia y Evolución de la Criptografía

El nacimiento de la criptografía se remonta a varios miles de años, aparece por la necesidad de proteger la información de diferentes grupos sociales, religiosos y étnicos en donde cada uno buscaba mantener la confidencialidad de los datos, ya sea en reposo o en tránsito, debido a que en épocas de guerras y conquistas era crucial lograr que el enemigo no conozca la información que se compartía. Un ejemplo claro de ello es la manera en la que se comunicaban los espartanos durante sus guerras, utilizaron un dispositivo especial para transportar mensajes secretos, un bastón llamado scytalee en el cual envolvían una tira de pergamino en el que se escribía el mensaje con una técnica criptográfica, con esto evitaban que la información sea interceptada (Wallis, 2013b).

[8] cifrario: sistema de reglas o normas de transcripción, gracias al cual un mensaje original que contiene información oculta se transforma en un mensaje cifrado, incomprensible para personas ajenas.

Las civilizaciones de Egipto, Mesopotamia, China, Grecia y Roma hacían uso de la criptografía como oficio práctico, lo que más adelante permitiría crear formas más robustas de asegurar la información. Actualmente, existen varias formas y técnicas que se utilizan para mantener la confidencialidad de los datos, paralelo a ello la complejidad y robustez ha aumentado conjuntamente con los avances en la tecnología informática, es decir, a medida que avanza el mundo tecnológico se fortalecerá e incluso se crearán nuevos métodos para asegurar la información (Delfs & Knebl, 2015).

Dentro de las ciencias, la criptografía proviene de una rama de las matemáticas que fue iniciada por el matemático Claude Elwood Shannon en 1948, denominada como “Teoría de la Información”, la cual se divide en 2 ramas: “Teoría de Códigos” y “Criptología”, y la última a su vez se divide en Criptoanálisis y Criptografía. Así, la Criptografía se encarga de transformar mensajes claros en mensajes cifrados de tal manera que solo se pueda descifrarlos si se conoce una o más llaves, por otro parte el Criptoanálisis se encarga de estudiar los métodos que se utilizan para recuperar los mensajes originales cuando no se conoce la llave o en su defecto conocer con que llave fueron cifrados los mensajes (Paredes, 2006).

1.1.3. Clasificación de la Criptografía

La Criptografía se clasifica en dos: Clásica y Moderna. La criptografía clásica o no digitalizada se utilizó desde mediados del siglo XX hasta antes de la época actual, los métodos utilizados eran simples y algunos muy complicados de criptoanalizar para la época. Por otro lado, la criptografía moderna se consideró iniciada después de la publicación de Shannon, la aparición del estándar del sistema de cifrado DES (Data Encryption Standard) en 1974 y la aparición del estudio realizado por Whitfield Diffie y Martin Hellman en el año de 1976 sobre el cifrado de llave pública (Paredes, 2006b).

Lo que define la clasificación de la criptografía son las técnicas o métodos que se utilizan para cifrar lo mensajes, como se muestra en la siguiente figura:

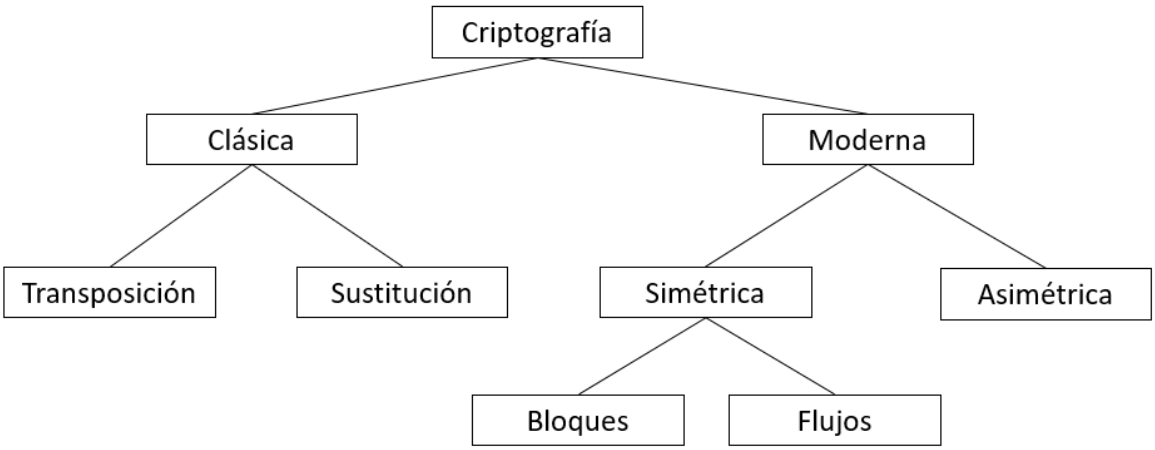

Figura 2: Diagrama de la clasificación de la Criptografía
Fuente: Propia

1.1.4. Definiciones Claves

1.1.4.1. Criptología

Es la ciencia que trata los problemas relacionados con la seguridad en el intercambio de mensajes entre emisor y receptor a través de un canal de comunicaciones. La Criptología se considera como una de las ciencias más antiguas y se divide en dos grandes ramas o ciencias: la criptografía y el criptoanálisis (Iris-Cert, 2011) .

1.1.4.2. Criptografía

La criptografía es la ciencia que estudia la escritura oculta, proviene de un sentido etimológico del griego Kriptos=ocultar, Graphos=escritura, es la habilidad de escribir en un lenguaje convenido utilizando claves o cifras numéricas, esta ciencia enseña a diseñar cifrarios, también conocidos como códigos secretos construidos por los criptógrafos, de tal manera que el mensaje que se envía sea legible únicamente para el destinatario deseado (Díaz, 1995).

1.1.4.3. Criptoanálisis

El criptoanálisis es la ciencia complementaria a la criptografía, se encarga de analizar y descifrar los criptogramas creados por la criptografía sin la necesidad de tener un código o clave (Iris-Cert, 2011).

A medida que el criptoanálisis mejore, la criptografía también necesitará mejorar y aumentar su complejidad y viceversa, es así como estas dos ciencias se complementan, aunque algunos autores opinan que son ciencias opuestas por el objetivo que tiene cada una.

1.1.4.4. Criptosistema

Un criptosistema es un sistema completo que se interpreta como una quíntupla conformada por: (M) textos claros o plaintext[9], (C) criptogramas que son los que representan el conjunto de los mensajes cifrados, (K) las claves de cifrado que representan al conjunto de claves que se emplean en el criptosistema, (E) el conjunto de transformaciones que se aplican al mensaje original para obtener el mensaje cifrado C y el quinto elemento (D), que representa el conjunto de transformaciones o funciones para descifrar el mensaje C y obtener el mensaje original (Security Manager, 2011).

1.2. Códigos QR

1.2.1. Características de los Códigos QR

Los códigos QR (Quick response) o códigos de respuesta rápida, son sistemas utilizados para almacenar información en una matriz bidimensional, este tipo de código a diferencia del código convencional de barras puede almacenar hasta 7.089 caracteres y diferente tipo de información, puede ser desde mensajes cortos, hasta publicidad o información sobre páginas web. Se pueden presentar en pantalla o forma impresa y son interpretables por cualquier dispositivo que cuente con el software adecuado para leer imágenes (M. Kumar & Singhal, 2012a).

1.1.1. Capacidad de almacenamiento del código QR

TABLA 1: Diferencias

Capacidad de los Códigos QR	
Tipo	**Capacidad**
Sólo numérico	7.089 caracteres
Alfanumérico	4.296 caracteres
Binario (8 bits)	2.953 bytes
Kanji/Kana	1.817 caracteres
Microcódigo QR	35 caracteres

Fuente: Propia

[9] plaintext: Texto plano.

1.1.2. Limitaciones del código QR

La capacidad de almacenamiento puede considerarse una limitación si se toma en cuenta que cada vez se requiere mayor capacidad en toda herramienta tecnológica que pueda almacenar algún tipo de información.

Por ahora, el código QR es poco práctico debido a que se necesita de un celular con cámara y tener una aplicación diseñada para leerlo, es decir no es funcional por sí solo. Otro aspecto negativo con respecto esta tecnología es que no es tan conocida y aunque su uso va en aumento, aún está muy lejos de ser una costumbre generalizada (Kevin Turcios, 2015).

1.1.3. Origen de los Códigos QR

El código fue inventado por Denso-Wave, la subsidiaria de Toyota en 1994 diseñando principalmente para la industria automotriz, a partir de allí se los utilizó para diferentes industrias sin ser extensamente conocidos. Recientemente, el Código QR ha sido popular en Internet móvil e Internet de las cosas debido a su fácil lectura y gran capacidad de almacenamiento (Chen, Du, Lin, & Tian, 2012).

1.1.4. Aplicaciones de los Códigos QR

Si se lo compara con el código de barras convencional, éste tiene mayor capacidad de almacenamiento, exploración rápida, mayor tolerancia a fallas en caso de que el código se dañe pueda ser leído exitosamente y la capacidad de corregir errores. En la actualidad, los códigos QR han sido cada vez más amplios en cuanto a su utilidad, se utiliza en flujos de aplicaciones relacionadas con marketing[10], temas académicos, seguridad, tarjeta electrónica, servicios de comercio electrónico, redes sociales, etc. La popularidad de los códigos QR aumenta a un ritmo alto, crece rápida y paralelamente con los usuarios de teléfonos inteligentes, por lo que está llegando a altos niveles de aceptación a nivel mundial (Tiwari, 2016).

1.2. Aplicaciones Web

1.2.1. Introducción

Una aplicación web es una aplicación del tipo cliente/servidor, donde el cliente, servidor y protocolo de comunicación están estandarizados y no son creados por el programador. Se debe entender como cliente al navegador o explorador, el servidor hace referencia al servidor

[10] marketing: en español: márquetin, es el conjunto de técnicas y estudios que tienen como objeto mejorar la comercialización de un producto.

web y el protocolo de comunicación a HTTP[11], este último forma parte de la familia de protocolos TC/IP que son los que se utiliza en Internet, los cuales permiten la conexión de diferentes sistemas facilitando así, el intercambio de información entre ordenadores (By-Nc-Nd, s. f.).

1.2.2. Cliente

Es un programa con el que el usuario puede interactuar para solicitar a un servidor web los recursos que desea obtener mediante el protocolo de comunicación HTTP. La misión del cliente es interpretar las páginas HTML[12] y los recurso que contienen que pueden ser imágenes, sonidos, etc. (By-Nc-Nd, s. f.).

Las tecnologías comúnmente empleadas para programar el cliente web son: HTML. CSS. DHTML. Lenguajes de script: JavaScript, VBScript, etc. ActiveX. Applets programados en Java. Distintas tecnologías que necesitan la existencia de un plug-in en el navegador: Adobe Acrobat Reader, Autodesk MapGuide, Live Picture PhotoVista, Macromedia Flash, Macromedia Shockwave, Virtual Reality Modeling Language (VRML), etc.

1.2.3. Servidor

Al igual que el cliente, es un programa que permanece en constante espera de solicitudes de conexión por parte de los clientes web. En Windows se les denomina servicios. La parte servidor de las aplicaciones web se conforma de: páginas estáticas, recursos adicionales y los programas o scripts[13] que ejecuta el servidor web cuando el navegador/cliente solicita alguna página (Luján-Mora, 2002).

1.2.4. Servidor de Aplicaciones

El servidor de aplicaciones se considera como el servidor de un gran sistema distribuido, provee servicios de ejecución y disponibilidad a las aplicaciones que se desplieguen, sin embargo, no todas las aplicaciones empresariales necesitan de un servidor de aplicaciones. Para el presente proyecto se utilizará como servidor de aplicaciones Wildlfy, anteriormente conocido como JBoss.

[11] HTTP: De las siglas en inglés: Hypertext Transfer Protocol, en español: protocolo de transferencia de hipertextos, se utiliza en algunas direcciones de internet.

[12] HTML: De las siglas en inglés: HyperText Markup Language, en español se traduce como Lenguaje de Formato de Documentos para Hipertexto, es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet

[13] Scripts: Es un documento que contiene instrucciones, escritas en códigos de programación. El script es un lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computador.

Wildfly es un servidor de aplicaciones Java EE de código abierto, que al estar basado en Java puede ser utilizado en cualquier sistema operativo que soporte la máquina virtual de Java. Está orientado a arquitectura de servicios, cuenta con gran capacidad de soporte, tiene flexibilidad consistente y es confiable a nivel de empresa lo que lo hace uno de lo más usados (Ortiz Ramírez & Balbuca Ramones, 2017).

1.2.5. Arquitectura de las Aplicaciones Web

Las aplicaciones web están basadas en la arquitectura cliente/servidor, de modo que, por un lado, está el cliente o navegador y por otro lado se encuentra el servidor web. En base a esto, existen diversas variantes dependiendo de cómo se implementen las funcionalidades del servidor. Las más comunes son: todo en un servidor, servidor de datos separado y todo en un servidor con servicio de aplicaciones (Luján-Mora, 2002). El presente proyecto se basa en la arquitectura básica.

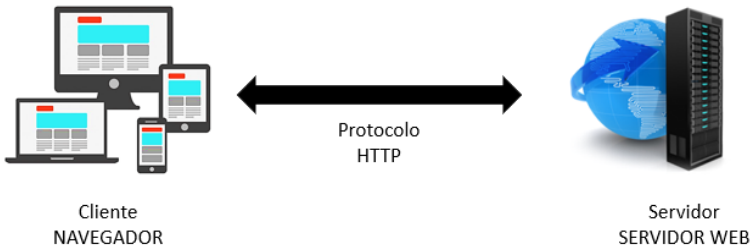

Figura 3: Esquema básico de una aplicación web
Fuente: *(Leonardo Favio Paillacho Vinueza, 2014)*

1.3. Aplicaciones Móviles

1.3.1. Características

Una aplicación móvil es un software desarrollado para dispositivos móviles como: teléfono móvil, cámara digital, cámara de video, navegador GPS, Tablet, reproductor de audio y asistente personal digital, con los cuales se puede acceder en cualquier lugar y momento a los datos, son lo suficientemente livianos para ser transportados por personas y cuentan con batería adecuada para funcionar de forma autónoma (Enriquez & Casas, 2014a).

Figura 4: Dispositivos móviles.
Fuente:*(Enriquez & Casas, 2014b)*

El manejo por parte del usuario tiene sus particularidades dependiendo del tipo de dispositivo, así también se toma en cuenta ciertos aspectos al momento de desarrollar una aplicación. Los sistemas operativos para móviles son más simples que los de una computadora y se enfocan más en la conectividad inalámbrica.

1.3.2. Incidencia de las Aplicaciones Móviles

Actualmente existen innumerables teléfonos inteligentes y otros dispositivos móviles ampliamente utilizados con los que los usuarios interactúan en comunidad, por lo que, el servicio de red móvil se vuelve cada vez más esencial para el diario vivir de las personas modernas (Lin & Lee, 2015). Por lo tanto, la seguridad de las aplicaciones móviles se ha vuelto importante y representa uno de los mayores desafíos para los diseñadores de aplicaciones, puesto que deben garantizar la seguridad de sus aplicaciones en dispositivos móviles para prolongar su usabilidad.

1.4. Métodos de Cifrado de Datos

1.4.1. Método de Rivest, Shamir y Adleman (RSA)

1.4.1.1. Introducción

RSA es un algoritmo de cifrado asimétrico[14] de clave pública desarrollado en 1977 que permite conservar la confidencialidad de la información compartida por los usuarios. Es el algoritmo más utilizado de este tipo, se utiliza tanto para cifrado de datos como para firmas digitales (Mónica Robles, 2016). Al ser un cifrador asimétrico, trabaja con dos claves, una

[14] Asimétrico: En el contexto de la información, hace referencia a que utiliza diferente clave para cifrar y descifrar la información.

privada y una pública, la información que sea cifrada con clave pública deberá ser descifrada con clave privada (Córdoba, 2016).

1.4.1.2. Funcionalidad

El problema del algoritmo RSA radica en la factorización de números enteros, el mensaje o información que se envía se representan con números, en donde se debe conocer el producto de dos números primos grandes elegidos al azar y mantenidos en secreto, así el atacante se enfrentará a dicho problema de factorización. La operación inversa a este problema es multiplicar, una operación con cierto costo computacional debido a que la complejidad crece a medida que aumenta el tamaño del número, sin considerar el tiempo que se demora (Camilo Gutiérrez Amaya, 2013a).

1.4.1.3. Seguridad

La seguridad de RSA está basada en el problema matemático de factorización, los valores de los factores primos deben ser mínimo de 155 dígitos, lo que representa un aproximado de 512 bits, el producto de estos factores representa un aproximado de 1024 bits, con esto se puede dar una idea de lo complejo que puede ser factorizar a nivel de recursos tecnológico (Wing H. Wong, 2013). El descifrado completo de un texto cifrado con RSA es computacionalmente intratable, no se ha encontrado un algoritmo que logre solucionar dicho problema (Mónica Robles, 2016).

1.4.1.4. Generación de Claves

Para enviar un mensaje cifrado a otra persona, es necesario generar las claves, este proceso es relativamente fácil y a la vez eficiente debido a que, las operaciones que se ejecutan son rápidas (Hernández Encinas, 2016). Este proceso lo puede hacer cada usuario. El usuario “A” deberá ejecutar los siguientes pasos:

Para ejemplificar utilizaremos número pequeños, aunque en la práctica real se utilizan números grandes que van de entre 1024 a 2048 bits, sin embargo, en el ejemplo se utilizarán valores pequeños para una mejor comprensión (Córdoba, 2016).

1. Generar dos números primos grandes p y q, se recomienda que el número de bits de cada uno este entre 1024 y 2048. Para propósito de demostración se harán los cálculos con números pequeños. *Ejemplo* p=3 y q=11.

2. Calcular *n*, el valor que le corresponde es el resultado de la operación *p*q*, el número de bits del resultado es la suma de p y q, el valor está entre 2048 y 4096, así determina

el valor indicador de Euler representado por z. *Ejemplo:* n=3*11=33, y z= (3 – 1) * (11 – 1) = 20.

3. Luego elige un número primo *e*, tal que *e* sea co-primo a z, por ejemplo, z no es divisible por *e*. Tenemos varias opciones aquí, valores de *e* como pueden ser 7, 11, 13, 17 o 19 son válidos. 5 es primo, pero no es co-primo de *e* puesto que 20 (z) es divisible por 5.

4. Elegimos e=7 para simplificar los cálculos con un número pequeño. La clave pública de "A" va a ser el conjunto de los números (n,e), es decir, clave publica = (33,7). Por cuestiones de seguridad, para proteger la clave privada, los números p y q se mantienen en secreto.

5. Ahora se calcula la clave privada. Para ello, se elige un número *d,* el cual tendrá el valor del primer número entero que dé como resultado de aplicar el Algoritmo extendido de Euclides denotado por la fórmula: d=[(y*z) + 1] / e, la variable e irá tomando valores consecutivos empezando desde 1.

6. Se reemplaza los valores en la fórmula, d= [(1*20) + 1] / 7, en este caso al reemplazar la variable "y" por el valor "1" y resolver las operaciones da como resultado 3, entonces d=3. Con valores mucho más grandes la variable "y" deberá adoptar diferentes valores para encontrar el valor de "d".

7. De modo que la clave privada se compone por (n,d), que al reemplazar los valores obtenidos se traduce como clave privada= (33,3), esta clave no debe ser revelada nunca.

1.4.1.5. Cifrar un Mensaje

Como primera instancia se debe tener un mensaje en texto plano para cifrar, denotado por M, el algoritmo RSA transforma el mensaje de texto plano a caracteres hexadecimales para cifrarlos, para ello se utiliza la ecuación matemática: *C=M^e (Mod n).* Donde: M, es el mensaje de texto de plano, n y e son la clave pública y C es el mensaje cifrado.

Reemplazamos los valores en la fórmula: *C=14^7 (Mod 33),* se eleva 14 a la potencia 7, se multiplica por el módulo de 33. El resultado de la potencia 14^7 = 105413504, ahora se multiplica por el módulo de 33 y da como resultado el número 20, que representa el texto cifrado. Así si el mensaje original era el número 14, se envía al destinatario el número 20, que

en realidad es el número 14 cifrado con RSA utilizando la clave pública del destinatario (Córdoba, 2016).

1.4.1.6. Descifrar un Mensaje

Asumiendo que el destinatario posee su clave privada, recibe el mensaje cifrado C=20, debe seguir los siguientes pasos para descubrir el mensaje original. Continuando con el ejemplo el destinatario tiene la clave privada d=3. En RSA el proceso para cifrar y descifrar un mensaje es básicamente el mismo, por ello se realiza la siguiente operación matemática: M=C^d (mod n). Donde: C es el mensaje cifrado, d es la clave privada, M es el mensaje en texto plano y n es parte de la clave privada.

Se sustituye los valores: M=20^3 (mod 33), lo que dará como resultado 20^3 = 8000, el valor resultante se multiplica por el módulo de n (Mod 33), lo que dará como resultado 14, que es el mensaje original. Y es así como se descifra los mensajes en RSA (Córdoba, 2016).

Claramente los “sistemas reales”, funcionan exactamente igual al ejemplo propuesto, pero con la diferencia de que los valores que se toma son mucho más grandes, puesto que este algoritmo basa su seguridad en la complejidad computacional.

1.4.2. Método: Advanced Encryption Standard (AES)

1.4.2.1. Historia del algoritmo AES

AES es un algoritmo de cifrado simétrico[15] desarrollado por Vicent Rijmen y Joan Daemen, estudiantes belgas, quienes en un principio lo nombraron “Rijndael”, así, fue presentado en 1997 en un concurso realizado por el Instituto Nacional de Normas y Tecnologías, el cual tenía por objetivo elegir al mejor algoritmo de cifrado, años después y con algunos cambios se renombró como se lo conoce en la actualidad. Es un algoritmo con alto nivel de seguridad, en el año 2001 el gobierno de los Estados Unidos anunció que era lo suficientemente seguro para utilizarlo en la protección nacional de información. Actualmente los únicos ataques considerados como eficientes son los ataques de canal auxiliar (Pousa, 2011a).

1.4.2.2. Características de AES

Se trata de un algoritmo de clave simétrica que cifra por bloques. El tamaño de cada bloque es de 16 bytes (128 bits) siempre, pero el tamaño de la clave puede variar entre los 128, 192 y 256 bits. En el caso de que aumente el tamaño de la clave, también aumenta el número de

[15] simétrico: En el contexto de la información, hace referencia a que utiliza la misma clave para cifrar y descifrar la información.

rondas necesarias para cifrar, entiéndase por rondas un conjunto de iteraciones de cuatro funciones matemáticas diferentes e invertibles (Pliego García, 2013). Un cifrado de bloque generalmente consta de dos algoritmos pareados, uno para el cifrado en el lado del remitente y otro para el descifrado en el lado del receptor. Al ser AES un algoritmo de clave simétrica, tanto el emisor como el receptor comparten la misma clave, es decir la clave privada (L. P. Kumar & Gupta, 2016).

TABLA 2: Estructura de AES

Estructura AES	
Clasificación	**Ronda**
AES – 128	10
AES - 192	12
AES - 256	14

Fuente: Propia

La mayoría de los cálculos de AES se realizan en un campo finito especial. La estructura redonda AES consta de cuatro estructuras que son SubBytes, ShiftRows, MixColums y AddRoundKey. Entre los indicados, el componente de caja de sustitución (caja S) es el corazón del algoritmo del Estándar de Cifrado Avanzado (AES) (M. Kumar & Singhal, 2012b).

1.4.2.3. Funcionamiento de AES

AES es un algoritmo de cifrado por bloques que se basa en un conjunto de rondas, por tanto, el algoritmo aplica dichas rondas a un mensaje de texto plano para obtener un mensaje cifrado (Pousa, 2011b). Este estándar define un tamaño de bloque de 128 bits, por lo tanto, los datos a ser encriptados se dividen en segmentos de 16 bytes (128 bits) y cada segmento se lo puede ver como una matriz de 4x4 bytes al que se lo llama estado, este se organiza de la siguiente forma:

AE	03	1F	2A	1E	3F	01	7A	21	04	CF	7A	1C	33	11	27

Matriz de Estado

AE	1E	21	1C
03	3F	04	33
1F	01	CF	11
2A	7A	7A	27

Figura 5: Matriz ejemplo del funcionamiento de AES
Fuente: Propia

1.4.2.4. Rondas y Operaciones en AES

El algoritmo AES se basa en aplicar a cada estado un conjunto de operaciones agrupadas, a esto se le denomina rondas y en cada ronda se aplica una subclave diferente (Pousa, 2011a). Las rondas se clasifican en tres tipos:

- Inicial: Se aplica la subclave inicial.
- Estándar: Se aplican un conjunto de subclaves, de acuerdo con las operaciones.
- Final: Se aplica la última subclave.

Dentro de las rondas se realizan cuatro operaciones básicas: SubBytes, ShiftRows, MixColumns y AddRoundKey.

1.4.2.5. Diagrama de funcionamiento de AES

Se presenta una descripción gráfica de cómo se utiliza el algoritmo para cifrar información.

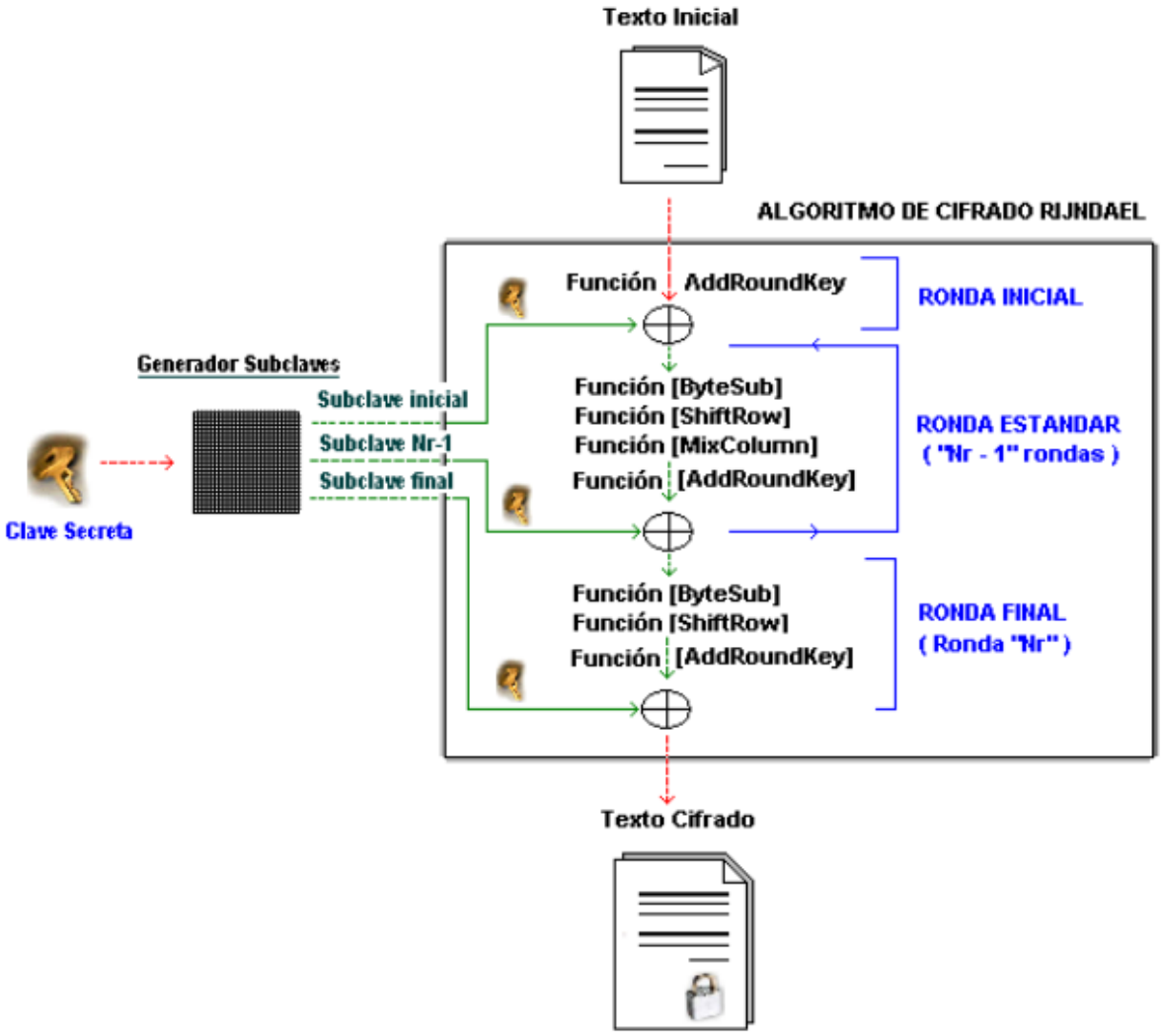

Figura 6: Descripción del proceso de Cifrado de AES
Fuente: *(Guamán, 2018)*

1.4.2.6. Transformaciones

El proceso de cifrado consiste en la aplicación de cuatro funciones matemáticas que se aplican a la información que se desea cifrar. Las transformaciones se realizan de forma reiterativa para cada ronda.

- *Función AddRoundKey*: Hace una operación exclusiva "O" que devuelve "verdadero" si uno u otro de sus operandos es verdadero, es la primera transformación que se ejecuta (David Dunning, 2016).

- *Función ByteSub:* Esta transformación opera independientemente en cada byte de la matriz, es decir que cada byte del bloque de entrada es invertido sobre GFH2 8 L y

luego pasa por una transformación afín. La operación completa puede realizarse mediante una matriz de sustitución, conocida con el nombre de S-Box.

- Función ShiftRow: En esta función los bytes de las tres últimas filas son desplazados cíclicamente en distintas cantidades o posiciones (offsets).

- *Caja S de AES:* Se utiliza como una tabla de búsqueda. Cuando se determina el inverso multiplicativo para un número dado en el campo finito de GF (28) Rijndael, se genera la caja S y viene dada por la ecuación: GF (28) = GF (2) [x] / (x8 + x4 + x3 + x + 1). La transformación afín se utiliza para transformar el inverso multiplicativo. Esta transformación es un vector en el que se toma la suma de múltiples rotaciones del byte, donde la suma es la operación XOR (operación exclusiva "O") (Pousa, 2011a).

1.4.2.7. Descifrado AES

Para el proceso para descifrar un mensaje aplica las mismas operaciones que en el proceso de cifrado, con la diferencia de que funcionan de forma inversa, pero utilizan las mismas subclaves generadas en orden inverso. Otra de las diferencias es que la matriz para la operación MixColumns es diferente, con esto se obtiene la transformación lineal aplicada en el proceso de cifrado (Pousa, 2011a).

1.5. Otros Métodos de Cifrado.

Los algoritmos como MD5, RC4 SHA-1 o SEAL son relativamente sencillos en cuanto a la implementación se refiere, su funcionamiento es parecido, los mensajes, palabras o firmas son cifrados considerando un número determinado de bits, posterior a esto se establece una clave para que el destinatario conozca el mensaje, evitando que los atacantes puedan conocer la información cifrada. Sin embargo, estos métodos tienen mucho tiempo desde su aparición y uso, paralelo a ello se han hecho mejoras para garantizar mayor seguridad, el problema radica en que a medida que aumenta su uso también ha aumentado el número de algoritmos desarrollados para descifrar la información que cifran dichos algoritmos (Lopez, 2002).

Esto denota que su nivel de seguridad no es altamente satisfactorio, pero un punto a considerar es que son algoritmos a los que se les puede añadir un pequeño complemento denominados como "salt/salts" o sal/sales en español, son dígitos aleatorios que se agregan al inicio o final del mensaje cifrado que se genera, esto hace que sea más difícil decodificarlos las contraseñas por parte del atacante y por ende más seguro (Fluid Attacks, 2016).

Para la realización del presente proyecto se ha seleccionado los algoritmos de cifrado RSA y AES, considerados los algoritmos más utilizados debido a su alto nivel de seguridad, son algoritmos accesibles en cuanto a la codificación y con gran capacidad de memoria, es decir, se puede cifrar gran cantidad de texto a la vez (Molina, 2017b). Por otra parte, algoritmos como RC4, por ejemplo, es propietario, lo que años atrás representó un gran negocio, hoy es una desventaja en su uso. El algoritmo SHA-1 se desarrolló con enfoque a firmas digitales y el MD5 ha mostrado ciertas debilidades, aunque no ha causado grandes problemas, por lo que se sigue considerando en la actualidad un algoritmo seguro, si bien su uso tiende a disminuir.

1.6. Protocolo HTTPS

HTTPS (Hyper Text Transfer Protocol Secure), en español: protocolo seguro de transferencia de hipertexto, es el protocolo dominante utilizado para proteger y garantizar la privacidad y la seguridad en las transacciones que se realizan online, esto puede ir desde datos públicos hasta datos confidenciales, como información de identidad, experiencia profesional o datos financieros.

Este protocolo consiste en capas de tráfico HTTP sobre los protocolos de transporte cifrados TLS[16] y SSL[17], aunque ha habido pequeñas fallas criptográficas en el TLS, el problema relevante ha sido la implementación inconsistente e incompleta de este protocolo. Los navegadores deben admitir una combinación de conexiones HTTP y HTTPS, para evitar los ataques conocidos como Man-In-TheMidle (Hombre en el Medio), son quienes intentan degradar la conexión de un usuario HTTP inseguro, no se puede descartar el peligro del atacante pese al soporte que tiene la implementación de HTTPS tanto para el servidor como para el cliente (Kranch & Bonneau, 2015).

Los protocolos SSL y TSL además de implementarse en HTTPS, son utilizados para la encapsulación de diferentes protocolos de nivel superior para poder crear conexiones más seguras (Bustamante, 2012).

En la actualidad sólo 116.675 webs de las más populares usan HTTPS por defecto, esto denota que la tendencia de la implantación del protocolo seguro está incrementando

[16] SSL: “Secure Sockets Layer” (en español «capa de conexión segura»), permite establecer conexiones seguras a través de Internet, de forma sencilla y transparente, consiste en interponer una fase de codificación de los mensajes antes de enviarlos por la red.

[17] TLS: “Transport Layer Security” (en español «seguridad de la capa de transporte»), protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

notablemente, con lo que se puede especular que en el futuro los sitios web más populares y sobre todo aquellos que tenga inicio de sesión obligatorio implementarán este sistema de comunicación cifrado (Agudo, 2017). Sin embargo, no todos los sitios web han adoptado este sistema por el costo de los certificados HTTPS, por lo que en su mayoría se ha implementado en sitios importantes, comerciales o empresas de seguridad.

1.7. X 509.

En criptografía X.509 es un estándar UIT-T (Unión Internacional de las Telecomunicaciones) para PKI (Public Key Infrastructure) o Infraestructura de Claves Públicas, este estándar especifica formatos estándar para certificados y un algoritmo de validación de la ruta de certificación. X.509 utiliza el lenguaje ASN1 (Abstract Syntax Notation One) para el estándar de certificados de clave pública. (Wazan, Laborde, Chadwick, Barrere, & Benzekri, 2016)

Una infraestructura de clave pública está basada en un modelo definido por el estándar X.509 original y se compone de tres entidades: AC (Autoridad de Certificación), el titular del certificado (o sujeto) y la parte que confía (RP), la AC cumple además el papel de un tercero de confianza entre el sujeto y el RP (Rrelying party) (BM Knowledge Center, 2014).

Actualmente se utilizan los certificados X.509 como estándar para verificación de una entidad con una clave pública, y se usan en aplicaciones de red: desde HTTPS en navegadores, conexiones SSH, correo electrónico, PDF y firma de código.

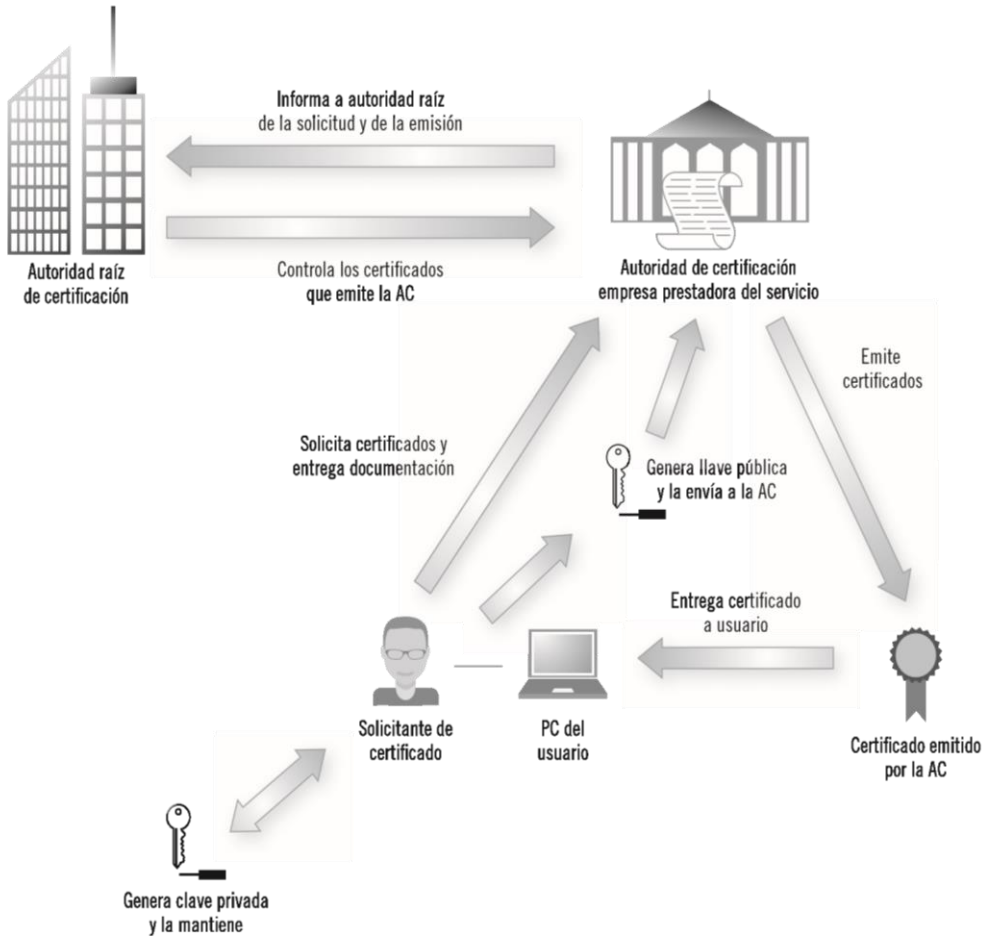

Figura 7: Funcionamiento General de una Infraestructura de Clave Pública
Fuente: *(Camilo Gutiérrez Amaya, 2013b)*

1.8. Modos de Cifrado

Los algoritmos de cifrado como AES o DES, cifran por bloques, es decir que separan el mensaje de texto plano en pedazos de tamaño fijo, 128 y 64 bits respectivamente. La manera en la que se cifra los bloques se denomina “Modo de Cifrado”, existen diferentes modos de cifrado que garantizan variados niveles de confidencialidad de los datos, los criterios que se evalúen para la elección de un modo de cifrado dependen de las necesidades del usuario.

1.8.1. ECB – Electronic Code Book Mode

Es el modo de cifrado más sencillo, ha sido estandarizado por el NIST (National Institute for Standars and Technology). ECB divide los mensajes en bloques y utiliza la misma clave K para cifrar cada uno de ellos. Una de las ventajas de este método es que puede cifrar en paralelo o el acceso aleatorio a diferentes bloques. Así mismo, ECB tiene desventajas, una

de ellas es que a los bloques de texto plano les corresponde bloques de cifrado iguales, de manera que a partir del texto cifrado se puede reconocer el texto original, por lo que este modo es útil para cifrar información reducida (Ibarra Quevedo, Serrano López, & Calixto Garera y Gonzalez, 2010).

1.8.2. CBC – Cipher Block Chaining Mode

Al igual que ECB, este modo de cifrado también ha sido estandarizado por el NIST (National Institute for Standars and Technology), pero CBC opera diferente, a cada bloque de texto plano (antes de ser cifrado) aplica la operación XOR con el bloque cifrado anterior antes, se utiliza un vector de inicialización con número aleatorio para que el primer bloque de texto plano haga la operación XOR. En este modo no es posible hacer operaciones de cifrado en paralelo debido a que cada bloque depende del anterior, lo cual representa una desventaja, pues si el atacante logra capturar un paquete dañara por completo la transmisión (Ibarra Quevedo et al., 2010).

1.9. Estándar Criptográfico de Claves Públicas (PKCS)

Laboratorios RSA, son los encargados de producir los estándares criptográficos de clave pública, esto lo hacen con un grupo de desarrolladores para promover soluciones PKI (Infraestructura de Clave Pública), estas combinan políticas, software, hardware y procedimientos de seguridad. La serie PKCS está referenciada en muchos estándares formales y de facto, incluidos ANSI X9, PKIX, SET, S/MIME Y SSL. Existen diferentes versiones de este estándar, y cada versión tiene su propia funcionalidad.

1.10. Seguridad de WhatsApp.

A medida que se desarrollan y aparecen mejoras en las tecnologías más usadas a nivel mundial, aumentar considerablemente la exigencia de los usuarios para que la información que comparte sea segura, por su puesto dar garantía de ello por parte de las empresas y organizaciones implica vivir en constante mejora y buscar respuestas que satisfagan necesidades. Es así como WhatsApp implementó nuevas medidas de seguridad.

1.10.1. Cifrado de extremo a extremo.

Se trata de un protocolo de seguridad en el que sólo el emisor y el receptor pueden leer los mensajes sin involucrar ni siquiera la misma compañía que presta el servicio. Es decir, los mensajes enviados no se almacenan en el servidor, tampoco la claves lo cual evita que los atacantes puedan interceptar esta información, este hecho también representa una ventaja para la compañía, ya que con esto se evitan de demandas por violación a la privacidad (Inés Matte Urrejola, 2014).

1.10.2. WhatsApp Web con Códigos QR.

Para entender la seguridad de WhatsApp Web es necesario conocer cómo funciona:

En el navegador se carga la página de WhatsApp Web en donde se muestra el código QR, internamente cada 20000 milisegundos se actualiza el código QR en codificación BASE64, esto se debe a la petición que hace el navegador al servidor mediante un WebSocket (WS) asociado a un único código QR.

Ahora bien, para que el servidor reconozca que el usuario es quien dice ser, se debe escanear el código QR mediante el lector de códigos de la app móvil de WhatsApp, así la información que recibe el servidor es el número de teléfono, credenciales de autenticación y se aprueba que el WebSocket asociado con el código QR leído puede recibir información (chats, imágenes, video., mp3) y mostrarlas en en el navegador mediante solicitudes de tipo GET (CodeDay, 2017).

1.11. Metodología para el desarrollo de software en Cascada.

La metodología en cascada es un modelo lineal de diseño de software. El desarrollo fluye secuencialmente, el proceso se compone de un conjunta de etapas que se ejecutan una tras otra. Es una metodología tradicional que se utiliza mayormente en pequeños equipos de trabajo donde la probabilidad de cambios en el proyecto, durante su ejecución es mínima (OBS Bissnes School, 2019).

Las etapas son: Requisitos, Diseño, Implementación, Verificación y Mantenimiento.

(Pablo Dominguez, 2017)

Requisitos: Se analiza las necesidades del cliente para determinar las características del software a desarrollar. Se debe comprender de forma clara el producto que quiere el cliente.

Diseño: Se describe la estructura interna del software, y las relaciones entre las entidades que lo componen, con esto se define la arquitectura y las herramientas para empezar con la implementación.

Implementación: En esta fase de la metodología, se programan los requisitos especificados con el cliente y se hace uso de todo lo definido en la fase de Diseño.

Verificación: Se verifica que el sistema y cada uno de sus componentes funcione correctamente y cumpla con los requisitos, de lo contrario se debe hacer los cambios necesarios para aumentar la calidad del software.

Mantenimiento: Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar, hecho esto, se pueden hacer

modificaciones ya sea para corregir errores o para mejorar el rendimiento o las características.

1.12. ISO/IEC 27002:2015

La ISO/IEC 27002 es una guía de buenas prácticas que establece directrices, controles y recomendaciones generales para iniciar, implementar, mantener y mejorar la gestión de la seguridad de la información en una organización. A continuación, se puede apreciar la directriz 10: Cifrado, que se utilizó para el presente trabajo, sin embargo, el Control de *Gestión de Llaves* no se implementó, debido a que no es aplicable a ninguno de los objetivos de este proyecto.

(INEN Servicio Ecuatoriano de Normalización, 2017).

1.12.1. Cláusula 10: Criptografía

1.12.1.1. Controles criptográficos

Objetivo: Asegurar un uso adecuado y eficaz de la criptografía para proteger la confidencialidad, autenticidad y/o integridad de la información.

- **Política de uso de controles criptográficos.**

Control

Se debería desarrollar e implementar una política sobre el uso de los controles criptográficos para proteger la información.

Guía de implementación

1. Al desarrollar una política criptográfica, debería tenerse en cuenta lo siguiente:

a) el enfoque de la dirección con respecto al uso de controles criptográficos en toda la organización, incluyendo los principios generales con base en los cuales debería protegerse la información de negocio;

b) tomando como base la evaluación de los riesgos, debería identificarse el nivel de protección necesario, teniendo en cuenta el tipo, la fortaleza y la calidad del algoritmo de cifrado requerido;

c) el uso del cifrado para proteger la información sensible transportada a través de medios extraíbles o dispositivos móviles o a través de líneas de comunicación;

d) el enfoque de la gestión de las claves, que incluye los métodos para ocuparse de la protección de las claves criptográficas y la recuperación de la información cifrada en caso de pérdida, vulneración o daño de las claves;

e) las funciones y responsabilidades; es decir, quién es responsable de:

   1) la implementación de la política,

   2) la gestión de las claves, incluyendo la generación de las mismas (ver 10.1.2),

f) las normas que deberían adoptarse para la implementación efectiva en toda la organización (qué solución se utilizará para cada proceso de negocio);

g) el impacto del uso de información cifrada en los controles que se basan en la inspección del contenido (por ejemplo, la detección de un malware).

Al implementar la política criptográfica de la organización, deberían tenerse en cuenta los reglamentos y restricciones nacionales que puedan resultar aplicables al uso de técnicas criptográficas en las distintas partes del mundo, así como a las cuestiones relativas al flujo transfronterizo de información cifrada (ver 18.1.5).

Los controles criptográficos pueden utilizarse para alcanzar distintos objetivos de seguridad de la información, por ejemplo:

a) confidencialidad: uso del cifrado de la información para proteger información sensible o crítica, tanto si esta se almacena como si se transmite;

b) integridad/autenticidad: uso de firmas electrónicas o códigos de autenticación de mensajes para verificar la autenticidad o la integridad de la información sensible o crítica que se almacene o se transmita;

c) no repudio: uso de técnicas criptográficas para obtener pruebas de la existencia o inexistencia de un evento o una acción;

d) autenticación: uso de técnicas criptográficas para autenticar usuarios y otras entidades del sistema que soliciten acceso a, o transacciones con, usuarios, entidades y recursos del sistema.

Otra información

Tomar una decisión en cuanto a si una solución criptográfica resulta adecuada debería considerarse como parte del proceso general de evaluación de riesgos y selección de controles. En ese caso, esta evaluación podría utilizarse para determinar si un control criptográfico es adecuado, qué tipo de control debería aplicarse, para qué fin y en qué procesos de negocio.

La política sobre el uso de controles criptográficos resulta necesaria para maximizar los beneficios y minimizar los riesgos de utilizar técnicas criptográficas, así como para evitar un uso inadecuado o incorrecto.

Debería consultarse con un especialista al seleccionar los controles criptográficos que sean apropiados para cumplir con los objetivos de la política de seguridad de la información.

1.12.1.2. Gestión de llaves

Control

Se debería desarrollar e implementar una política para el uso, la protección y la duración de las llaves criptográficas a lo largo de todo su ciclo de vida.

Guía de implementación

La política debería incluir los requisitos de gestión de las llaves criptográficas en todo su ciclo de vida incluyendo la generación, almacenamiento, archivo, recuperación, distribución, retirada y destrucción de estas.

Los algoritmos criptográficos, la extensión de las llaves y las prácticas de uso deberían seleccionarse de acuerdo con buenas prácticas. Una gestión adecuada de las llaves requiere procesos seguros de generación, almacenamiento, archivo, recuperación, distribución, retirada y destrucción de las llaves criptográficas.

Todas las llaves criptográficas deberían estar protegidas contra la modificación y la pérdida. Además, las llaves secretas y privadas necesitan protección contra una divulgación no autorizada de las mismas. Los equipos utilizados para generar, almacenar y archivar llaves deberían contar con protección física.

El sistema de gestión de llaves debería basarse en un conjunto consensuado de normas, procedimientos y métodos seguros para:

a) generar llaves para distintos sistemas criptográficos y diferentes aplicaciones;

b) generar y obtener certificados de llave pública;

c) distribuir las llaves a los usuarios previstos, incluyendo la forma en que dichas llaves deberían activarse cuando se reciban;

d) almacenar llaves, incluyendo la forma en que los usuarios autorizados pueden acceder a las mismas;

e) cambiar o actualizar las llaves, incluyendo las normas relativas a cuándo y cómo deberían cambiarse las llaves;

f) actuar ante las llaves comprometidas;

g) revocar llaves, incluyendo cómo deberían retirarse o desactivarse las llaves, por ejemplo, cuando estas han sido comprometidas o cuando un usuario deja una organización (en cuyo caso, las claves también deberían archivarse);

h) recuperar llaves pérdidas o corruptas;

i) realizar copias de respaldo o archivar llaves;

j) destruir llaves;

k) registrar y auditar las actividades relacionadas con la gestión de llaves.

Para reducir la posibilidad de un uso inadecuado, deberían definirse fechas de activación y desactivación de las llaves, de manera que estas solo puedan utilizarse durante un espacio limitado de tiempo definido en la correspondiente política de gestión de llaves.

Además de una gestión segura de las llaves secretas y privadas, también debería tenerse en cuenta la autenticidad de las llaves públicas. Este proceso de autenticación puede llevarse a cabo utilizando certificados de llave pública, que suelen ser expedidos por una autoridad de certificación, que debería ser una organización reconocida que cuente con controles y procedimientos adecuados para ofrecer el grado de confianza necesario.

El contenido de los acuerdos o contratos de nivel de servicio con proveedores externos de servicios criptográficos como, por ejemplo, una autoridad de certificación debería cubrir las cuestiones relativas a la responsabilidad, la fiabilidad de los servicios y los tiempos de respuesta para la prestación de servicios (ver 15.2).

Otra información

La gestión de llaves criptográficas resulta fundamental para un uso eficaz de las técnicas criptográficas. Las ISO/IEC 11770 [2] [3] [4] proporcionan más información sobre la gestión de llaves.

Las técnicas criptográficas pueden usarse asimismo para proteger las llaves criptográficas. Los procedimientos pueden necesitar ser considerados para el manejo de peticiones legales para el acceso a las llaves criptográficas, por ejemplo, puede que, en un juicio, deba presentarse información cifrada como prueba en un formato no cifrado.

1.13. Política del Uso de Controles Criptográficos

Una política es un documento que define reglas para el uso de algo, en este caso: controles criptográficos, con el fin de proteger la confidencialidad, integridad, autenticidad e

inviolabilidad de la información (Dejan Kosutic, 2019). A continuación, la política desarrollada para este proyecto:

Objetivo

Establecer las directrices generales para proteger la información mediante el uso de algoritmos criptográficos. Esta política está diseñada para mantener la confidencialidad de los datos y evitar su robo o alteración.

Alcance

Esta política aplica para todos aquellos que hagan uso de algoritmos criptográficos listados a continuación:

- Algoritmo de Cifrado: Advanced Encryption Standard (AES)
- Algoritmo de Cifrado: Rivest, Shamir y Adleman (RSA)

Definiciones

Dato: Representación simbólica numérica, alfabética, algorítmica de un atributo o variable cuantitativa.

Información: conjunto organizado de datos que poseen significado.

Control: medios para gestionar el riesgo, incluyendo políticas, procedimientos, directrices, prácticas o estructuras.

Política: Se refiere a una declaración general de principios que presenta la posición de la administración para un área o aspecto del negocio.

Cifrado: Que está escrito con letras, símbolos o números que solo pueden comprenderse si se dispone de la clave (llave criptográfica) necesaria para descifrarlos.

Cifrar: Es un procedimiento que utiliza un algoritmo de cifrado con cierta clave que transforma la información, sin atender a su estructura lingüística o significado, de tal forma que sea incomprensible o, al menos, difícil de comprender a toda persona que no tenga la clave secreta.

Llaves criptográficas: Son códigos (algoritmos) que se generan de forma automática y se guarda en un directorio especial durante la instalación. Habitualmente, esta información es una secuencia de números o letras mediante la cual, en criptografía, se especifica la transformación del texto plano en texto cifrado, o viceversa.

Política

- Con el fin de garantizar la confidencialidad de la información, se debe utilizar técnicas criptográficas como mecanismo de protección de datos.
- El desarrollador o encargado de implementar los algoritmos criptográficos debe procurar al menos un nivel básico para proteger las claves secretas y privadas evitando sean copiadas o modificadas sin autorización.
- Los responsables del software que contenga algoritmos de cifrado deberán velar porque la información de su custodia o propiedad, que se considere privada, se cifre al momento de almacenarse o transmitirse.
- La persona encargada de la configuración deberá configurar y administrar el sistema de cifrado, así como velar por el cumplimiento de la presente política y generar los reportes que se requieran.
- El administrador será el encargado de establecer el cambio de claves privadas, que se recomienda hacerlo cada 6 meses. En caso de alguna alteración en el sistema, el cambio será inmediato.

Incumplimiento

El incumplimiento de esta política traerá consigo las consecuencias legales de acuerdo con las normativas de quienes la utilicen, incluyendo lo establecido en las normas que competen al Gobierno Nacional en cuanto a seguridad y privacidad de la información se refiere.

Referencia a Otros Documentos

Política de Monitoreo de Niveles de Servicio.

Política de Seguridad de Laboratorios de Informática / Centros de cómputo.

Política de Protección de llaves/claves de cifrado para usuario final

Nota: Estos documentos no son parte de esta tesis.

Responsabilidad y Autoridad

Desarrollador: Implementa los algoritmos criptográficos en el sistema o software.

Administrador: Revisa que el funcionamiento sea correcto y administra las llaves de cifrado.

Oficial de Seguridad de la Información: Realiza el seguimiento y control del cumplimiento de esta política.

TABLA 3: Elaboración, Revisión y Aprobación de la Política del Uso de Controles Criptográficos

Autores de la Elaboración, Revisión y Aprobación de la Política del Uso de Controles Criptográficos					
ELABORÓ		**REVISÓ**		**APROBÓ**	
Nombre:	Dayana Guerra	Nombre:	Ing. Mauricio Rea	Nombre:	Ing. Mauricio Rea
Cargo:	Estudiante	Cargo:	Tutor de Proyecto	Cargo:	Tutor de Proyecto
Fecha:	05/07/2019	Fecha:	08/07/2019	Fecha:	11/07/2019

Fuente: Propia

1.14. Directriz 13: Seguridad en las Comunicaciones.

La directriz sobre la seguridad en las comunicaciones tiene por objetivo asegurar la protección de la información en redes y sus instalaciones de soporte en el procesamiento de información. La información que se considera confidencial y viaja por redes públicas requiere de una gestión segura en las redes, para ello las organizaciones se deberían basar en alguna política o estándar que regule los procedimientos para un manejo de información segura, donde se consideren controles de red, mecanismos de seguridad asociados a servicios de red, acuerdos de confidencialidad y mensajería electrónica.

La seguridad en las Comunicaciones se divide en dos categorías, a continuación, se muestra los controles que tiene cada categoría:

GESTIÓN DE LA SEGURIDAD DE LAS REDES

- Controles de red
- Seguridad de los servicios de red
- Separación en las redes

TRANSFERENCIA DE INFORMACIÓN

- Políticas y procedimientos de transferencia de información
- Acuerdos de transferencia de información
- Mensajería Electrónica

Acuerdos de confidencialidad o no revelación

Como puede apreciarse, los puntos que se tratan en esta normativa tratan sobre todo aquello que incluya el aseguramiento de información mediante redes, por lo que, el alcance de esta tesis no se aplica a esta información, esto debido a que este trabajo investigativo se refiere a la parte de software.

2. CAPÍTULO II: DESARROLLO

1.1 Requerimientos

Se detallan los requerimientos funcionales y no funcionales en base a los cuales se desarrolló el aplicativo del presente proyecto.

TABLA 4: Requisitos Funcionales y no Funcionales

REQUISITOS FUNCIONALES	
R1	**El sistema web contará con una pantalla en donde se muestre el código QR**
R2	En la misma pantalla en la que se genera el código QR debe haber al menos un campo en donde se ingrese información a cifrar
R3	La aplicación móvil deberá tener implementado un lector de códigos QR.
R4	La aplicación móvil tendrá una pantalla en la cual se deberá ingresar al menos un campo con información.
R5	Se podrá ingresar cualquier tipo de texto para cifrar. Para este proyecto se utiliza únicamente datos alfanuméricos, mas no archivos multimedia.
R6	La aplicación web debe poder utilizarse con los navegadores web Chrome, Firefox e Internet Explorer
REQUISITOS NO FUNCIONALES	
R1	El sistema principalmente deberá asegurar el intercambio de información entre una aplicación web y aplicación móvil
R2	El sistema será desarrollado en Eclipse para la aplicación web y Android Studio para la aplicación móvil
R3	El proceso de cifrado se llevará a cabo implementando los algoritmos de cifrado RSA y AES.
R4	Un determinado usuario tendrá acceso al sistema para ingresar la información que se desea cifrar, sin embargo, esto no supone que exista control de acceso mediante login o inicio de sesión.
R5	El sistema debe poseer interfaces gráficas sencillas y de fácil manejo.
R6	La metodología para el desarrollo de software será la “Metodología en Cascada”

Fuente: Propia

1.2 Diseño

1.2.1 Herramientas

Las herramientas que se van a utilizar son:

Eclipse IDE Versión 2018-12: Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar

Aplicaciones Web. Este IDE es utilizado para el desarrollo en el lenguaje de programación Java.

JDK: El Java Development Kit (JDK) proporciona el conjunto de herramientas básico para el desarrollo de aplicaciones con Java estándar. Se puede obtener de manera gratuita en internet, descargándola desde el sitio de Oracle.

WildFly versión 14: El sistema Wildfly o JBoss, es un servidor de aplicaciones JEE, por tanto, facilita los diferentes estándares de esta tecnología: Java Persistence API

(JPA), Enterprise Java Beans (EJB), Java Server Faces (JSF), etc. (Suarez, Mielgo, Cabrera, Gonzalez, & Rocha, 2017).

Android Studio: Es el entorno de desarrollo específico de Android, está basado en Gradle que permite al desarrollador aplicar distintas configuraciones del mismo código para producir distintas versiones del mismo código de aplicación (Hohensee, 2014).

1.2.2 Librerías

Las librerías que se van a utilizar son:

- *barcode4j-light-2.1:* Es un generador flexible para códigos de barras escritos en Java, admite una variedad de simbologías de códigos de barras como Interleaved 2 of 5, Code 39, Code128, Codabar, EAN-8/13, UPC-E / A, POSTNET, Royal Mail Customer Barcode, PDF417 y DataMatrix. (Jeremias Maerki, 2011)
- *Biblioteca zxing de Google (core-3.3.0 y javase 3.3.0):* Es una biblioteca de procesamiento de imágenes de código de barras 1D / 2D multiformato de código abierto implementada en Java («QR Code in java», 2017).
- *primefaces 7.0:* Es una librería de componentes visuales de código abierto para el conjunto Java Server Faces 2.0 desarrollada y mantenida por Prime Technology. Su objetivo principal es ofrecer un conjunto de componentes para facilitar la creación y diseño de aplicaciones web (Pech-May, Gomez-Rodriguez, & Lara-Jeronimo, s. f.).
- *qrgen-1.4*: Es un pequeño envoltorio de ZXing para generar códigos QR en java. Para su utilización, simplemente se debe añadir la dependencia en el proyecto.

1.2.3 Arquitectura

Dentro del marco de desarrollo de software existe un gran conflicto por parte del programador o desarrollador en cuanto a las etapas del ciclo de vida de un sistema

informático al no predefinir el inicio y fin de cada etapa, lo que consecuentemente afecta al proyecto que se esté desarrollando.

La arquitectura de software es un conjunto de patrones que proporcionan un marco de referencia necesario que ayuda a guiar la construcción y desarrollo de un software, esto permite que los analistas de software, programadores y todo el conjunto de desarrollo se encaminen por una misma línea de trabajo con la finalidad de cumplir todos los objetivos del software (EcuaRed, 2016).

Este tema representa un problema frecuente en la rama de la informática debido a que cada desarrollador prefiere definir y aplicar su propia arquitectura para iniciar un proyecto, sin tomar en cuenta el concepto básico de Arquitectura de software y lo que esto implica en los factores internos y externos que son adyacentes al desarrollo de la programación.

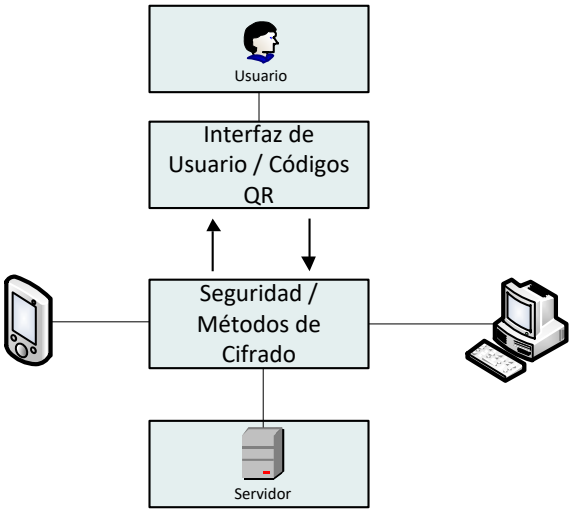

Figura 8: Diagrama de la arquitectura del Proyecto
Fuente: Propia

1.2.4 Funcionalidad

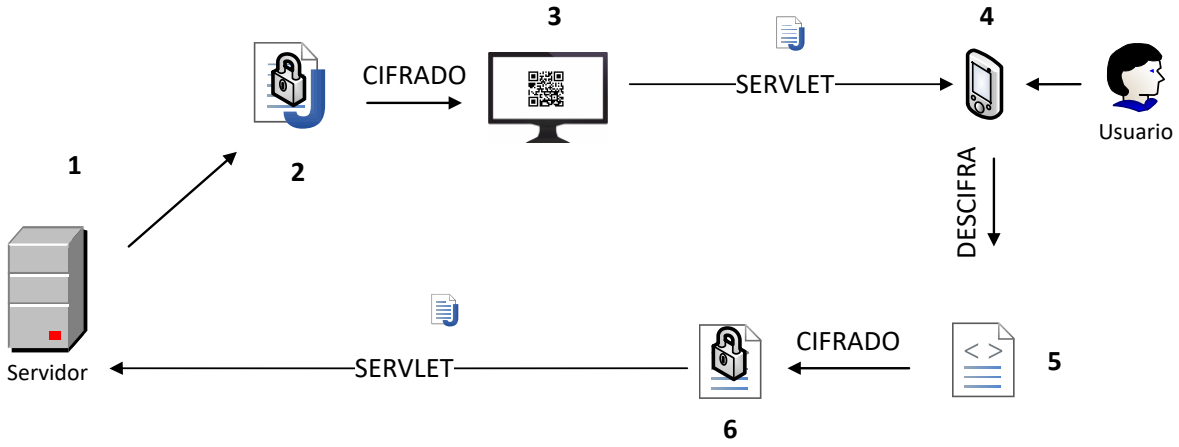

Figura 9: Diagrama de Funcionalidad del Proyecto
Fuente: Propia

A continuación, en la tabla Nº 5, se detalla cada paso del diagrama de funcionalidad del proyecto *(Figura 9).*

TABLA 5: Detalle del Diagrama de Funcionalidad

Detalle de cada etapa del diagrama de funcionalidad	
Paso Nº	Función
1	Se requiere de un servidor para la aplicación web, para ello se deben hacer las configuraciones respectivas.
2	En la aplicación web java estarán implementados los métodos de cifrado y un servlet, el cual permitirá el intercambio de información.
3	En la aplicación web estará implementado el método para generar el código QR, el cual se genera con la información cifrada.
4	El usuario debe hacer uso de su dispositivo móvil el cual deberá tener la aplicación móvil que permitirá la lectura del código QR.
5	La aplicación móvil descifra la información ingresada en la aplicación web.
6	Hecho esto, en la aplicación móvil se deberá ingresar un mensaje el cual será cifrado y añadido a la información anterior contenida en el Servlet.

Fuente: Propia

1.3 Implementación

1.3.1 Servidor de Aplicaciones

Para la implementación del proyecto QREncryption, como paso primordial se debe configurar el servidor de aplicaciones web. En este caso se eligió WildFly, el cual servirá para la implementación del algoritmo AES como para el algoritmo RSA.

1.3.2 Algoritmo de Cifrado RSA

Para el algoritmo RSA es necesario generar 2 claves, una pública y otra privada, para ello existen generadores de claves RSA online de los cuales se puede hacer uso, o bien, se puede implementar un generador de claves. Para este proyecto se optó por hacer la implementación del generador. En RSA se puede generar claves desde 512 bits, 1024 bits, 2048 bits y en la actualidad se puede generar claves de hasta 4096 bits, sin embargo, esta última, puede saturar la capacidad de almacenamiento del código QR.

```
public class RSAKeyPairGenerator {

	private PrivateKey privateKey;
	private PublicKey publicKey;

	public RSAKeyPairGenerator() throws NoSuchAlgorithmException {
		KeyPairGenerator keyGen = KeyPairGenerator.getInstance("RSA");
		keyGen.initialize(1024);
		KeyPair pair = keyGen.generateKeyPair();
		this.privateKey = pair.getPrivate();
		this.publicKey = pair.getPublic();
	}

	public void writeToFile(String path, byte[] key) throws IOException {
		File f = new File(path);
		f.getParentFile().mkdirs();
		FileOutputStream fos = new FileOutputStream(f);
		fos.write(key);
		fos.flush();
		fos.close();
	}

	public PrivateKey getPrivateKey() {
		return privateKey;
	}

	public PublicKey getPublicKey() {
		return publicKey;
	}
```

Figura 10: Método generador de claves RSA.
Fuente: Propia

Con generador implementado, de manera predeterminada, la clave privada se genera en formato PKCS # 8 y la clave pública se genera en formato X.509. Recuerde, la clave pública está escrita en el archivo de texto como formato X.509. Para el presente proyecto se codificó las claves públicas y privadas con Base64 para facilitar el intercambio con el cliente, debido a que es más legible. El tamaño de la clave variará en función de su tamaño, en este caso se generó una clave de 1024 bits. La clave pública se ve de la siguiente forma:

```
private static String publicKey
=MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCgFGVfrY4jQSoZQWWygZ83roKXWD4YeT2x2p41dGkPi
xe73rT2IW04glagN2vgoZoHuMJUTXFFFoK73D0vmCHu6D1auJhE2tXP+yLkpSiYMQucDKmCsW08nSzL5K7
OSL77TXXcfvTvyZcjObGtpqFzs6+FqpFbUO9SJEfh6wIDAQAB";
```

Figura 11: Claves pública RSA (clave de ejemplo)
Fuente: Propia

```
private static String privateKey =
"MIICdQIBADANBgkqhkiG9w0BAQEFAASCAl8wggJbAgEAAoGBAKAUZV+tjiNBKhlBZbKBnzeugp
dYPhh5PbHanjV0aQ+LF7vetPYhbTiCVqA3a+Chmge44+prlqd3qQCYra6OYIe7oPVq4mETa1c/7
IuSlKJgxC5wMqYKxYydb1eULkrs5IvvtNddx+9O/JlyM5sTPosgFHOzr4WqkVtQ71IkR+HrAgMB
AAECgYAkQLo8kteP0GAyXAcmCAkA2Tql/7kfdkjfdpoiUW/91kkypCDNF5oCsdxZSJgV8owViYW
ZPnbvEcNqLtqgs7nj1UHuX9S5yYIPGN/mHL6OJJ7sosOd6rqdpg6JRRkAKUV+tmN/7Gh0+GFXM+
ug6mgwQJBAO9/+CWpCAVoGxCA+YsTMb82fTOmGYMkZOAfQsvIV2v6DC8eJrSa+c0yCOTa3tirlC
khBfB08f8U2iEPS+Gu3bECQQCrG7O0gYmFL2RX1O+37ovyyHTbst4s4xbLW4jLzbSoimL235lCd
IC+fllEEP9UDJJuu87uinsVRbAkB0ME8AZjp/9Pt8TDXD5LHzo8mlruUdnCBcIo5TMoRG2+3hRe
1dHPonNCjgbdZCoyqjsWOiPfnQ2Brigvs7J4xhAkBGRiZUKC92x7QKbqXVgN9xYuq7oIanIM0nz
/wq190uq0dh5Qtow7hshC/dSK3kmIEHe8z++tpoLWvQVgM538apAkBoSNfaTkDZhFavuiVl6L8c
WCoDcJBItip8wKQhXwHp0O3HLg10OEd14M58ooNfpgt+8D8/8/2OOFaR0HzA+2Dm";
```

Figura 12: Clave privada RSA (Clave de ejemplo)
Fuente: Propia

Se crea una clase java, la cual tendrá métodos definidos para el cifrado y descifrado con RSA. Se empieza por el cifrado haciendo uso de la clave pública generada, la misma que se generó en formato X.509. Por lo tanto, se necesita la clase X509EncodedKeySpec para convertirla nuevamente a la clave pública RSA. Las claves públicas están codificadas en base64, por lo tanto, primero se decodifica Base64 para genera la clave pública.

```
	public static PublicKey getPublicKey(String base64PublicKey) {
		PublicKey publicKey = null;
		try {
			X509EncodedKeySpec keySpec = new
X509EncodedKeySpec(Base64.getDecoder().decode(base64PublicKey.getBytes()));
			KeyFactory keyFactory = KeyFactory.getInstance("RSA");
			publicKey = keyFactory.generatePublic(keySpec);
			return publicKey;
		} catch (NoSuchAlgorithmException e) {
			e.printStackTrace();
		} catch (InvalidKeySpecException e) {
			e.printStackTrace();
		}
		return publicKey;
	}
```

Figura 13: Método getPublicKey
Fuente: Propia

Se crea el método para encriptar el cual lleva la cadena o información para ser cifrada y la clave RSA codificada en Base64, se hace uso del método creado *getPublicKey* y por su puesto del parámetro que contiene a la clave pública.

```
public static byte[] encrypt(String data) throws BadPaddingException,
IllegalBlockSizeException,
                  InvalidKeyException, NoSuchPaddingException,
NoSuchAlgorithmException {
            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            cipher.init(Cipher.ENCRYPT_MODE, getPublicKey(publicKey));
            return cipher.doFinal(data.getBytes());
      }
```

Figura 14: Método de Encriptación RSA
Fuente: Propia

Para el descifrado se usa la clave privada generada en formato PKCS # 8. Por lo tanto, se debe generar la clave privada desde una cadena codificada en base64 utilizando PKCS8EncodedKeySpec.

```
public static PrivateKey getPrivateKey(String base64PrivateKey) {
            PrivateKey privateKey = null;
            PKCS8EncodedKeySpec keySpec = new
PKCS8EncodedKeySpec(Base64.getDecoder().decode(base64PrivateKey.getBytes()));
            KeyFactory keyFactory = null;
            try {
                  keyFactory = KeyFactory.getInstance("RSA");
            } catch (NoSuchAlgorithmException e) {
                  e.printStackTrace();
            }
            try {
                  privateKey = keyFactory.generatePrivate(keySpec);
            } catch (InvalidKeySpecException e) {
                  e.printStackTrace();
            }
            return privateKey;
                                          }
```

Figura 15: Método de Generación de Clave Privada RSA
Fuente: Propia

1.3.3 Algoritmo de Cifrado AES

Para el cifrado con el algoritmo AES se utiliza dos componentes de java en donde se específica el tipo de algoritmo que se va a utilizar y el modo de cifrado.

```
   private final static String alg = "AES";

   private final static String cI = "AES/CBC/PKCS5Padding";
```

Figura 16: Especificación de Componentes Java
Fuente: Propia

Se crea el método de tipo *String* que recibe la llave (key) y el texto que se va a cifrar.

```
   public static String encryptAES(String cleartext) throws Exception {
             String key = "M6D1auFqpFbUOxSJEfh6wI98Tfe+yu5z";
             Cipher cipher = Cipher.getInstance(cI);
             SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(), alg);
             byte[] bytes = new byte[16];
             IvParameterSpec ivParameterSpec = new IvParameterSpec(bytes);
             cipher.init(Cipher.ENCRYPT_MODE, skeySpec, ivParameterSpec);
             byte[] encrypted = cipher.doFinal(cleartext.getBytes());
             return new
String(javax.xml.bind.DatatypeConverter.printBase64Binary(encrypted));
       }
```

Figura 17: Método de Encriptación AES
Fuente: Propia

1.3.4 Conexión mediante Servlet

Se crean dos Servlets que permitirán el intercambio de información entre la aplicación móvil y web, debido a que se crean dos sesiones, una en la aplicación móvil y otra en la aplicación web para cada algoritmo por lo que es necesario crear un servlet para cada algoritmo de cifrado.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
             String    informacion    =    request.getParameter("informacionweb"),
mensajemovil = request.getParameter("mensaje");
             String mensaje = mensajemovil;
             //guardarBloc("C:/Mensaje/", "MensajeEncryptAES", mensaje);

             desencriptar des=new desencriptar();
             try {
                    String dese= des.decryptAES(request.getParameter("mensaje"));
                    String mensaje1 = informacion ;
                    String mensaje2 = dese;
```

```
                    //guardarBloc("C:/Mensaje/", "MensajeDesAES", mensaje1);
                    beanInformacion.setInformacionMovilAES(mensaje2);
                    beanInformacion.setInformacionWebAES(mensaje1);
                    beanInformacion.setInformacionMovilEncrAES(mensaje);
                    System.out.println(beanInformacion.getInformacionMovilAES());

             } catch (Exception e) {
                    e.printStackTrace();
             }
}
```

Figura 18: Método doGet del Servlet para AES
Fuente: propia

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
             String    informacion    =    request.getParameter("informacionweb"),
mensajemovil = request.getParameter("mensaje");
             String mensaje = mensajemovil;
             try {
                    String                                                    dese=
RSAUtil.decrypt(request.getParameter("mensaje"),privateKey);
                    String mensaje1 =  informacion ;
                    String mensaje2= dese;
                    beanInformacion.setInformacionMovilRSA(mensaje2);
                    beanInformacion.setInformacionWebRSA(mensaje1);
                    beanInformacion.setInformacionMovilEncrRSA(mensaje);

                    System.out.println(dese);
             } catch (Exception e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
             }
      }
```

Figura 19: Método doGet del Servlet para RSA
Fuente: propia

Dentro del Controller se crean métodos para intercambiar la información cifrada tanto para el algoritmo AES como para RSA.

```
   public void generarInformacionAES() throws Exception {
             HttpServletRequest request = (HttpServletRequest)
FacesContext.getCurrentInstance().getExternalContext()
                          .getRequest();
             url = request.getRequestURL().toString();
             int posicion = url.indexOf("/EncriptacionAES.xhtml");
             url = url.substring(0, posicion);
             url = url + "/servletEnvioInformacionAES";
             url = url + "?informacionweb=" + informacionweb;
             url = encryptAES(url);
```

```
	}
```

Figura 20: Método de Conexión para intercambio de información entre aplicaciones para AES.
Fuente: Propia

```
	public void generarInformacionRSA() throws Exception {
		HttpServletRequest request = (HttpServletRequest)
FacesContext.getCurrentInstance().
				getExternalContext().getRequest();
		urlRSA = request.getRequestURL().toString();
		int posicion = urlRSA.indexOf("/EncriptacionRSA.xhtml");
		urlRSA = urlRSA.substring(0, posicion);
		urlRSA = urlRSA + "/servletEnvioInformacionRSA";
		urlRSA = urlRSA + "?informacionweb=" + informacionwebRSA;
		;
		urlRSA = Base64.getEncoder().encodeToString(RSA.encrypt(urlRSA));//
	}
```

Figura 21: Método de Conexión para intercambio de información entre aplicaciones para RSA.
Fuente: Propia

1.3.5 Aplicación Móvil Android

Para la aplicación móvil, al igual que en la aplicación web se debe desarrollar funciones que encripten y desencripten la información que se intercambia. Las funcionen deben ser tanto para RSA como para AES. A continuación, se presenta las funciones con las cuales se desencripta los mensajes que se transmiten desde la aplicación web.

```
public String decryptAES(String encrypted) throws Exception {
    Cipher cipher = Cipher.getInstance(cI);
    SecretKeySpec skeySpec = new
SecretKeySpec("M6D1auFqpFbUOxSJH5dQwI98Tfe+gulz".getBytes(), alg);
    byte[] bytes = new byte[16];
    IvParameterSpec ivParameterSpec = new IvParameterSpec(bytes);
    byte[] enc = Base64.decode(encrypted,Base64.DEFAULT);
    cipher.init(Cipher.DECRYPT_MODE, skeySpec, ivParameterSpec);
    byte[] decrypted = cipher.doFinal(enc);
    return new String(decrypted);
}
```

Figura 22: Método de Conexión para intercambio de información entre aplicaciones para RSA.
Fuente: Propia

```
public static String decrypt(String data) throws IllegalBlockSizeException,
InvalidKeyException, BadPaddingException, NoSuchAlgorithmException,
```

```
NoSuchPaddingException {
    byte[] datos = android.util.Base64.decode(data,
android.util.Base64.DEFAULT);
 return decrypt(datos, getPrivateKey(privateKey));
```

Figura 23: Método de Conexión para intercambio de información entre aplicaciones para RSA.
Fuente: Propia

Para desencriptar los mensajes ya sea en AES o RSA, se utilizan las claves correspondientes para cada algoritmo mostradas anteriormente en la Figura 11 y 12, una vez desencriptado se debe transmitir un mensaje desde la aplicación móvil para propósito de cumplir con el proceso de intercambio seguro de información. Entonces, después se deberá cifrar dicha información, para ello se utilizan las mismas funciones de encriptación que se utilizan en la aplicación web.

1.3.6 Resultado Final de las Aplicaciones

2.3.6.1. Aplicación Web – Inicio

Figura 24: Pantalla de Inicio del Demo QREncryption.
Fuente: Propia

2.3.6.2. Aplicación Web - Cifrado AES

Figura 25: Pantalla de encriptación para el algoritmo AES.
Fuente: Propia

2.3.6.3. Aplicación Web - Cifrado RSA

Figura 26: Pantalla de encriptación para el algoritmo RSA.
Fuente: Propia

2.3.6.4. Aplicación Móvil – Inicio

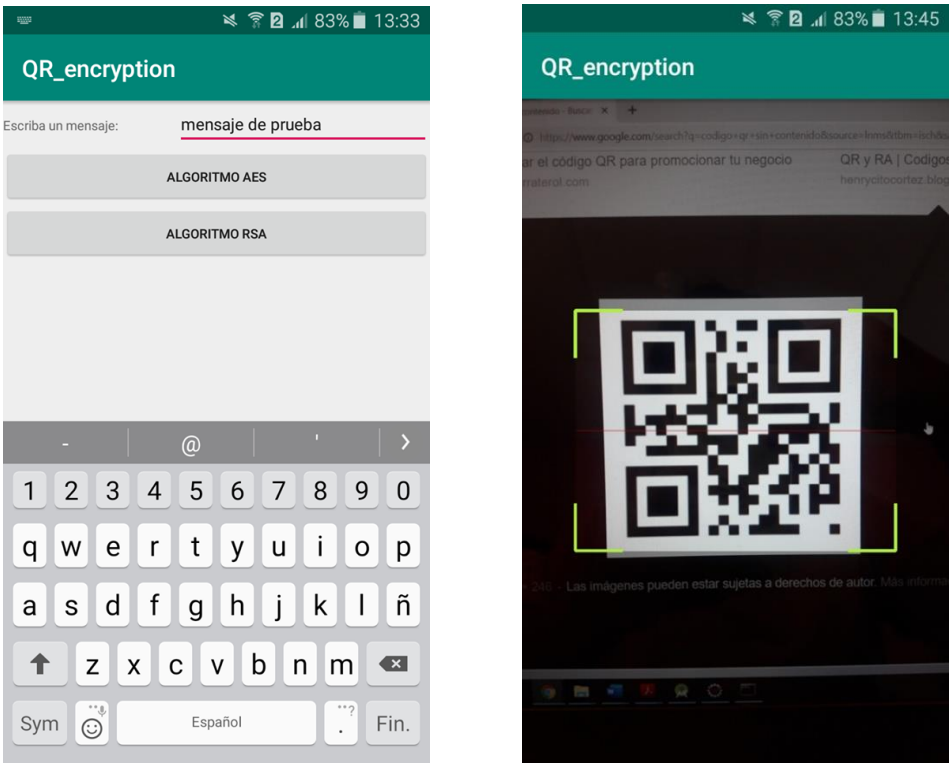

Figura 27: Pantalla de funcionamiento de la aplicación móvil, de AES y RSA.
Fuente: Propia

2.3.6.5. Otros lectores de Códigos QR

En la siguiente figura se muestra el resultado de leer un mensaje encriptado con el Algoritmo AES, utilizando un lector de código QR que no contiene las funciones para interpretar el mensaje cifrado.

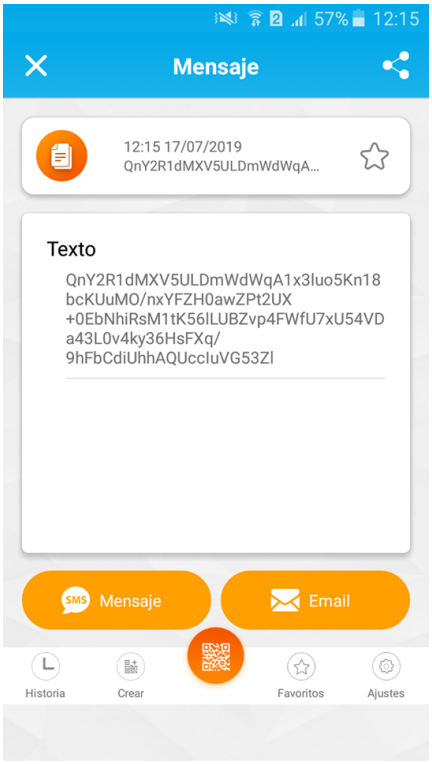

Figura 28: Interfaz del lector de códigos QR *"Escáner QR"*
Fuente: Propia

2. CAPÍTULO III: VALIDACIÓN DE RESULTADOS

3.1. Capacidad de caracteres AES y RSA

Al utilizar códigos QR para almacenar la información encriptada por los algoritmos AES y RSA, la capacidad de información que se puede encriptar varía significativamente. Si bien, un código QR admite 4.296 caracteres alfanuméricos y 7.089 caracteres numéricos, sin embargo, al cifrar un mensaje de texto plano con 15 caracteres, por ejemplo, da como resultado un mensaje cifrado con un mayor número de caracteres, la cantidad varía en función del algoritmo que se utilice para cifrar.

Así, en el caso del algoritmo RSA, es relativamente fácil conocer el número de caracteres que se podrá cifrar, puesto que involucra a las claves, estas pueden ser de 512 bits, 1024 bits, 2048 bits y actualmente admite hasta 4092 bits. Para este proyecto se utilizó una clave de 1024 bits. Ahora bien, es recomendable combinar este algoritmo con algún esquema de relleno o *Padding Scheme*, esto para evitar que el atacante pueda obtener el mensaje original cuando se utilicen valores numéricos bajos en la encriptación. Con valores bajos no tomaría demasiado tiempo en descubrir el mensaje original, bastaría con hacer las funciones matemáticas necesarias para obtener los números primos, estos casos suceden, aunque no sean comunes, debido a esto se considera como una vulnerabilidad de RSA.

Utilizar un esquema de relleno asegura que el mensaje original sea un mensaje sin cifrar seguro, para este proyecto se implementó Padding del tipo PKCS5, el cual ocupa 11 bytes en la capacidad que permiten el algoritmo para cifrar y se utiliza antes de que el mensaje sea cifrado. El RSA-Padding Scheme ayudará principalmente a prevenir ataques sofisticados, los cuales podrían ser facilitados por la predictibilidad de la estructura del mensaje.

Entonces, la clave de 1204 bits se divide por 8, la cantidad de bytes que da como resultado es de 128 bytes y es el número de caracteres que admitirá el algoritmo, a este valor se resta 11 bytes del relleno, y ese es el valor real que representa la cantidad de caracteres que se podrá cifrar, en este caso 117.

Para el algoritmo AES, la manera de definir la capacidad que admite es ingresar el mensaje con un número *n* de caracteres que permita generar el código QR, es decir que, al ingresar el mensaje de texto plano, el número de caracteres del mensaje cifrado debe ser menor o igual a 4.296, que es la capacidad que admite el código QR. En la siguiente tabla se muestra

la capacidad máxima de caracteres en mensaje de texto plano que admite tanto AES como RSA.

TABLA 6: Capacidad Máxima de Almacenamiento de Caracteres en el Código QR

Capacidad Máxima de Almacenamiento de Caracteres en el Código QR	
Algoritmo	**Número de Caracteres**
AES	2100
RSA	117

Fuente: Propia

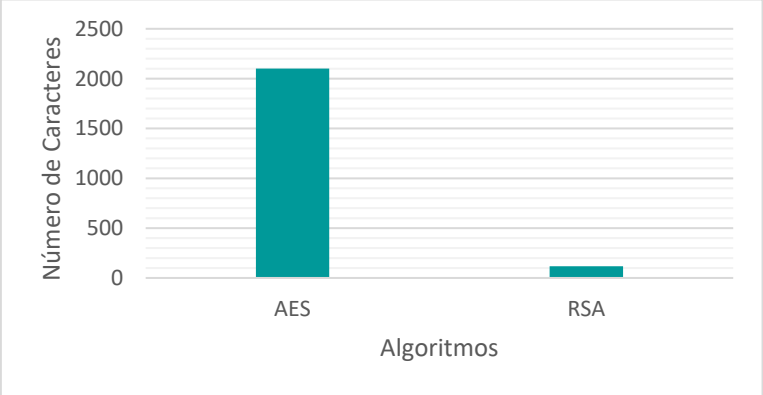

Figura 29: Capacidad Máxima de Almacenamiento de Caracteres en el Código QR

Fuente: Propia

3.2. Resultados de la Comparativa en base al número de caracteres

La cantidad de número de caracteres para la comparativa se escogió en base a la mayor capacidad de los algoritmos, en este caso AES tiene mayor capacidad, por tanto, la comparativa de tiempos se realizó en base a ello. En el caso de RSA, solo admite 117 caracteres en mensaje de texto plano por lo que, no fue posible compararlo con cantidades mayores. Se elaboraron 10 pruebas para cada cantidad de caracteres tanto para AES como RSA. En la siguiente tabla se pueden apreciar los tiempos resultantes, es importante considerar que los datos obtenidos hacen referencia al tiempo que se demora en cifrar el mensaje y mostrar el código QR con la información cifrada.

TABLA 7: Capacidad de caracteres de los algoritmos AES y RSA

Capacidad de caracteres de los algoritmos AES y RSA		
Datos del tiempo de ejecución de los algoritmos		
Nro de Caracteres	**Algoritmo**	**Tiempo (s)**
100	AES	0,002
	RSA	0,007
500	AES	0,001
	RSA	No aplica

1000	AES	0,004
	RSA	No aplica
1500	AES	0,005
	RSA	No aplica
2100	AES	0,006
	RSA	No aplica

Fuente: Propia

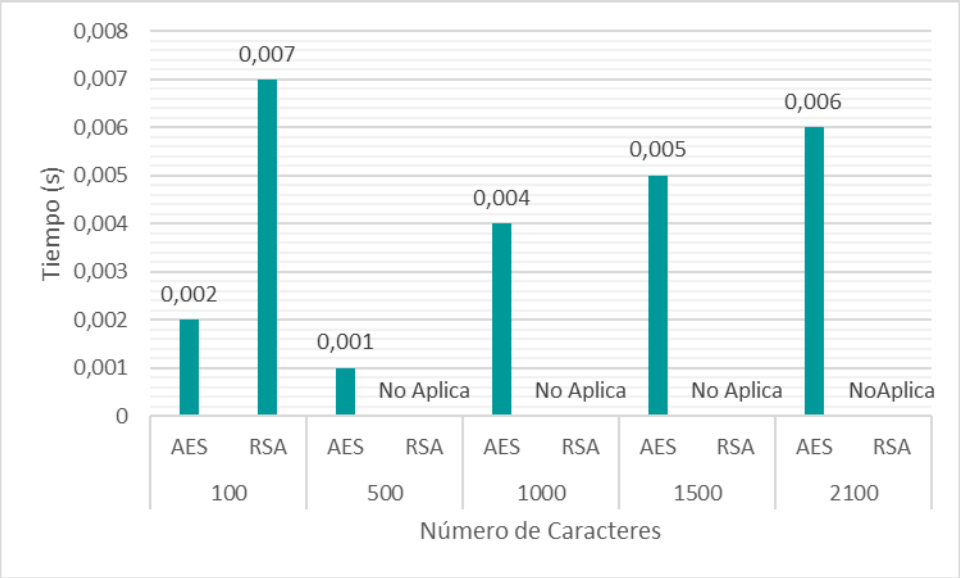

Figura 30: Datos del tiempo de ejecución de los algoritmos AES y RSA

Fuente: Propia

3.3. Evaluación del algoritmo AES

Para empezar, se evaluó el tiempo que se demora en encriptar los mensajes que se transmiten desde la aplicación Web, los tiempos exactos al ejecutar el código del algoritmo AES se muestran en la siguiente tabla:

TABLA 8: Tiempo de Encriptación de AES en la Aplicación Web

Tiempo de Encriptación en la Aplicación Web	
Nro Caracteres	**Tiempo (s)**
100	0,001
500	0,001
1000	0,001
1500	0,002
2100	0,002

Fuente: Propia

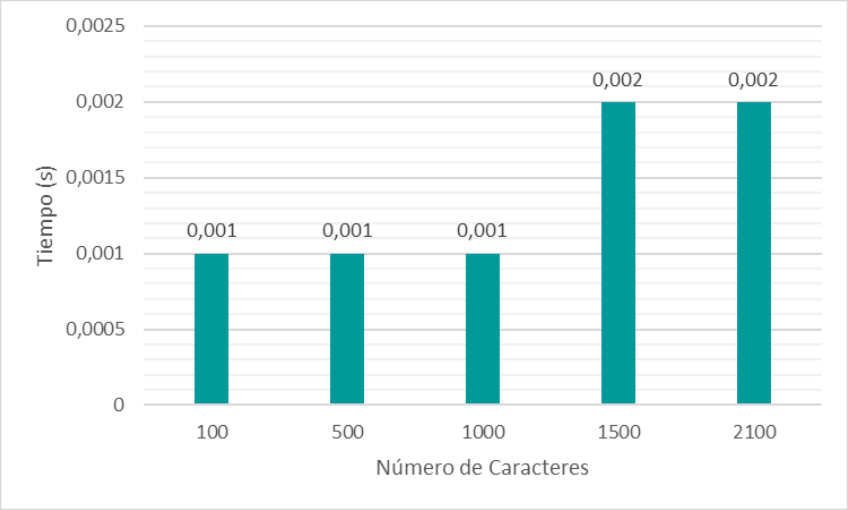

Figura 3132: Tiempo de Encriptación de AES en la Aplicación Web

Fuente: Propia

Otro de los aspectos que se analizó es el tiempo de ejecución en la Aplicación Móvil Android, esto se refiere al tiempo que toma descifrar el mensaje que se transmite desde la Aplicación Web con la cantidad de caracteres que se muestra en la *TABLA 7* y cifrar el mensaje que se transmite desde la Aplicación Móvil. Para estas pruebas se utilizó la cantidad de caracteres de la *TABLA 11*, y desde la aplicación se ingresó un mensaje de 100 caracteres para todas las pruebas. El tiempo se tomó en segundos (s). Las cantidades de 1000, 1500 y 2100 caracteres no tienen valor debido a que el lector de códigos QR no reconoce los puntos de alineación que se generan con estas cantidades, por tanto, no puede interpretar la información que contiene.

A continuación, se muestran los valores resultantes:

TABLA 9: Tiempos de Ejecución en la Aplicación Móvil con AES

Tiempos de Ejecución en la Aplicación Móvil con AES			
Nro. de Caracteres	**Tiempo Inicio (s)**	**Tiempo Fin (s)**	**Tiempo de Ejecución (s)**
100	10,719	10,766	0,047
500	35,512	35,57	0,058
1000	No aplica	No aplica	Sin valores
1500	No aplica	No aplica	Sin valores
2100	No aplica	No aplica	Sin valores

Fuente: Propia

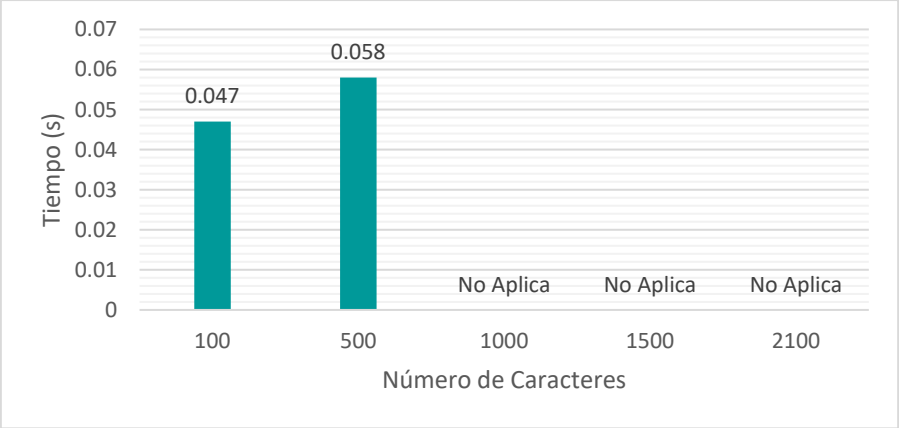

Figura 33: Tiempos de Ejecución en la Aplicación Móvil con AES

Fuente: Propia

Para las pruebas de la Aplicación Web con el algoritmo AES, el tiempo de ejecución hace referencia a las funciones que ejecuta la aplicación, esto se refiere al tiempo que toma descifrar el mensaje que se transmite desde la Aplicación Móvil y mostrarlo en la pantalla correspondiente en el sistema. El tiempo se tomó en segundos (s). A continuación, se muestran los valores resultantes:

TABLA 10: Tiempos de Ejecución en la Aplicación Web con AES

Tiempos de Ejecución en la Aplicación Web con AES			
Nro. de Caracteres	**Tiempo Inicio (s)**	**Tiempo Fin (s)**	**Tiempo de Ejecución (s)**
100	34,992	34,994	0,002
500	10,004	10,006	0,002
1000	No aplica	No aplica	Sin valores
1500	No aplica	No aplica	Sin valores
2100	No aplica	No aplica	Sin valores

Fuente: Propia

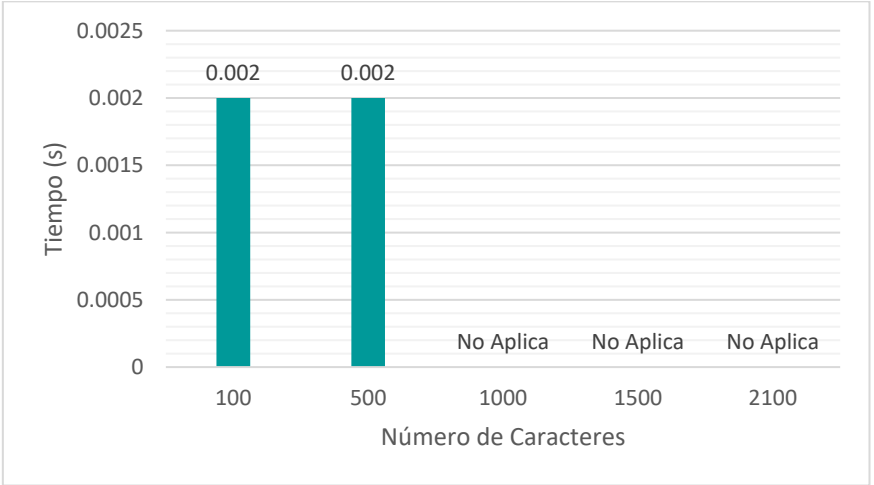

Figura 34: Tiempos de Ejecución en la Aplicación Web con AES
Fuente: Propia

Otro de los aspectos que se evaluó es el uso de memoria, desde que se ejecuta la Aplicación Web, se encripta el mensaje que se inserta y muestra el código QR con la información encriptada. A continuación, los datos obtenidos:

TABLA 11: Uso de memoria del Algoritmo AES

Uso de memoria del Algoritmo AES	
Nro. Caracteres	**Uso de Memoria (Kbytes)**
100	0,048
500	0,096
1000	0,080
1500	0,656
2100	0,624

Fuente: Propia

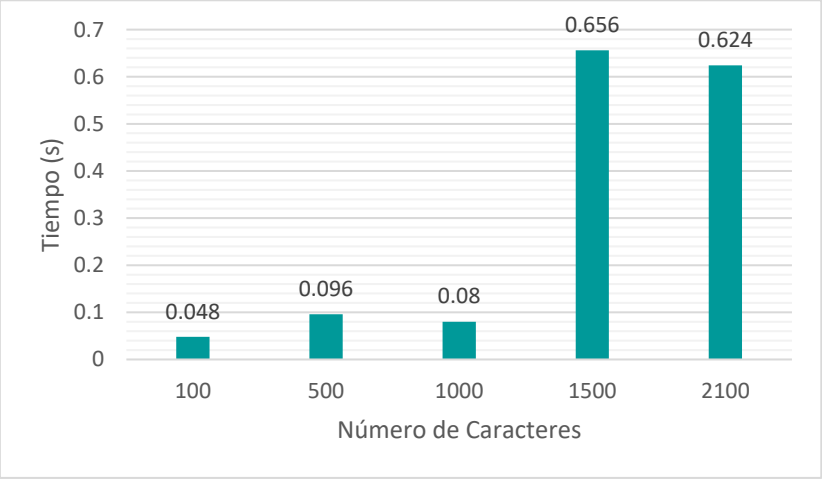

Figura 35: Uso de memoria del Algoritmo AES

Fuente: Propia

3.4. Evaluación del algoritmo RSA

Para este algoritmo también se evaluó el tiempo que se demora en encriptar los mensajes que se transmiten desde la aplicación Web, en este caso para RSA sólo se puede evaluar el tiempo de encriptación para 100 caracteres, debido a que el tamaño de la clave es de 1024 bits y se utiliza un esquema de relleno.

TABLA 12: Tiempo de Encriptación en la Aplicación Web con RSA

Tiempo de Encriptación en la Aplicación Web	
Nro. Caracteres	**Tiempo (s)**
100	0,004
500	No Aplica

1000	No Aplica
1500	No Aplica
2100	No Aplica

Fuente: Propia

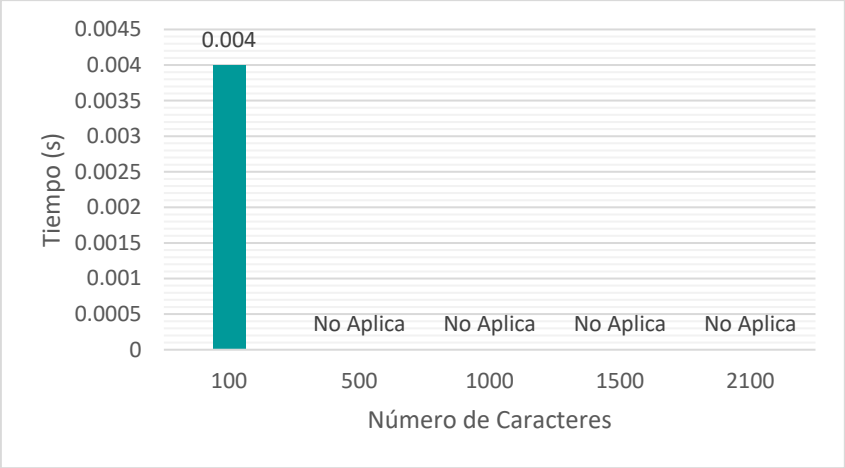

Figura 36: Tiempo de Encriptación en la Aplicación Web con RSA

Fuente: Propia

El tiempo de ejecución para el algoritmo RSA, tuvo el mismo procedimiento que el algoritmo AES (Revisar introducción de la *TABLA 9),* la diferencia es que, debido a la capacidad de este algoritmo solo se puede evaluar cuando el número de caracteres es 100. El tiempo se tomó en formato de segundos (s). A continuación, se muestran los valores resultantes:

TABLA 1314: Tiempos de Ejecución en la Aplicación Móvil con RSA

Tiempos de Ejecución en la Aplicación Móvil con RSA			
Nro. de Caracteres	**Tiempo Inicio (s)**	**Tiempo Fin (s)**	**Tiempo de Ejecución (s)**
100	13,846	14,006	0,160
500	No aplica	No aplica	Sin valores
1000	No aplica	No aplica	Sin valores
1500	No aplica	No aplica	Sin valores
2100	No aplica	No aplica	Sin valores

Fuente: Propia

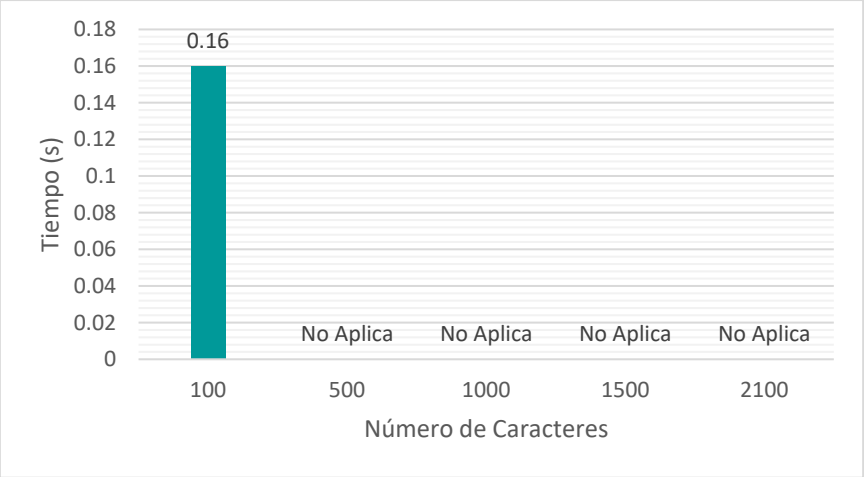

Figura 37: Tiempos de Ejecución en la Aplicación Móvil con RSA

Fuente: Propia

Al igual que AES, los datos que se muestran a continuación se refieren al tiempo que toma descifrar el mensaje que se transmite desde la Aplicación Móvil y mostrarlo en la pantalla correspondiente en el sistema.

TABLA 15:Tiempo de Ejecución en la Aplicación Web con RSA

Tiempo de Ejecución en la Aplicación Web con RSA			
Nro. de Caracteres	**Tiempo Inicio (s)**	**Tiempo Fin (s)**	**Tiempo de Ejecución (s)**
100	13,159	13,164	0,005
500	No aplica	No aplica	Sin valores
1000	No aplica	No aplica	Sin valores
1500	No aplica	No aplica	Sin valores
2100	No aplica	No aplica	Sin valores

Fuente: Propia

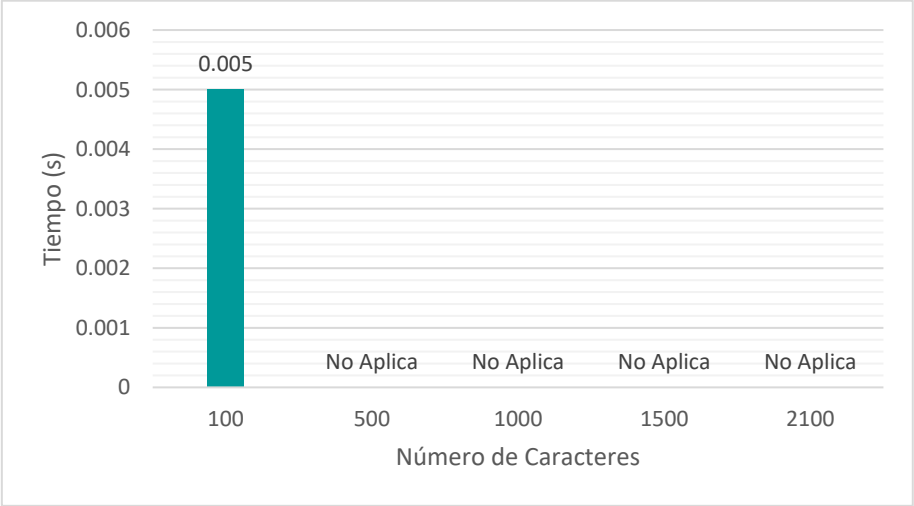

Figura 38: Tiempo de Ejecución en la Aplicación Web con RSA

Fuente: Propia

Para RSA también se evaluó el uso de memoria, la diferencia es que este algoritmo admite únicamente 100 caracteres. A continuación, los datos obtenidos:

TABLA 16: Uso de memoria del Algoritmo RSA

Uso de memoria del Algoritmo RSA	
N.º Caracteres	**Uso de Memoria (Kbytes)**
100	0,024
500	No Aplica
1000	No Aplica
1500	No Aplica
2100	No Aplica

Fuente: Propia

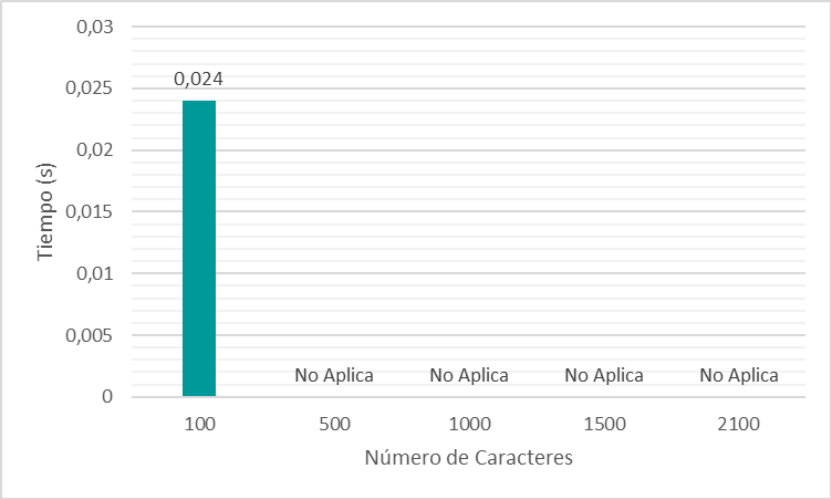

Figura 39: Uso de memoria del Algoritmo RSA

Fuente: Propia

3.5. Comparativa entre los algoritmos AES y RSA

Para la comparativa se consideraron los factores más importantes de cada algoritmo. Así, en base a la investigación teórica de este proyecto y las pruebas realizadas se obtiene las características resultantes de cada algoritmo. La siguiente tabla es un resumen de la investigación realizada tanto a nivel teórico como práctico.

TABLA 17: Comparativa entre AES y RSA

Características de AES y RSA		
Factores	**AES**	**RSA**
Desarrolladores	Vicente Rijmen y Joan Daemen	Rivest, Shamir y Adleman
Año de Publicación	2001	1977
Tamaño de la Clave	128, 192, 256 bits	Recomendado mayor a 1024 bits

Tamaño del Bloque	128	Mínimo 512 bits
Clave de cifrado y descifrado	La misma	Diferente
Tipo de Algoritmo	Algoritmo Simétrico	Algoritmo Asimétrico
Tiempo de Cifrado	Rápido	Lento
Tiempo de Descifrado	Rápido	Lento
Costo Computacional	Menor	Mayor
Uso de Memoria	Menor	Mayor
Posibles Vulnerabilidades	Ataque de fuerza bruta	Ataque de fuerza bruta
Implementación de Software	Eficiente	Eficiente
Nivel de Seguridad	Excelente Seguridad, Alto	Menos seguro, Medio
Rondas	10, 12, 14	1

Fuente: Propia

3.6. Análisis de Impactos

Económico

El impacto a nivel económico implica evitar gastos que se pueden producir en la corrección de errores de seguridad en los sistemas. Asegurar la transmisión de datos evita fallos y alteraciones de la información que consecuentemente podría representar grandes pérdidas económicas a las organizaciones y usuarios individuales.

Ambiental

Los sistemas de automatización de procesos ayudan a reducir el consumo de recursos, principalmente de papel con lo que se aporta directamente a la preservación del medio ambiente.

Social

Tiene un alto impacto en la parte social, teniendo seguridad de datos los usuarios podrán tener una buena experiencia y mayor tranquilidad utilizando las tecnologías seguras, lo cual genera un ambiente de bienestar colectivo.

Los beneficiarios directos de este proyecto serán los propietarios de celulares con sistema operativo Android en los cuales se implemente la aplicación y hagan uso diario de este tipo de tecnologías.

Tecnológico

El aseguramiento de la información tiene un impacto directo en el campo tecnológico, las organizaciones necesitan funcionar con datos seguros y con la menor posibilidad de vulneraciones a sus sistemas y áreas tecnológicas.

3.7. Discusión

Esta investigación tuvo como propósito asegurar el intercambio de datos entre aplicaciones web y aplicaciones móviles Android, con la utilización de códigos QR. Sobre todo, se pretendió analizar aspectos específicos; el uso de memoria, el tiempo de cifrado y descifrado para cada algoritmo. Además, se enfatizó en una tecnología creada en 1994, los códigos QR.

De los resultados obtenidos en este trabajo, se puede deducir que el método de encriptación Advanced Encryption Standard (AES) es más eficiente que Rivest, Shamir, Adleman (RSA) en términos de tiempo de ejecución (s) y uso de memoria (KB), por lo que se podría implementar en aplicaciones informáticas que requieran seguridad y protección en el flujo de información que manejan. Los hallazgos de esta investigación coinciden con los del trabajo de (Molina, 2017). Sin embargo, en dicho trabajo se evaluaron más parámetros como la escalabilidad, uso de energía, caballo de troya y repositorio de claves, debido a que era parte de sus objetivos, no obstante, los resultados son similares. La principal similitud es que en los dos casos se evalúa a los mismos métodos de cifrado bajo el argumento de que son los más utilizados en la actualidad (Andrea Pfundmeier, 2018). Por otro lado, la diferencia principal entre las investigaciones es que, en el trabajo realizado por Molina no se utilizan códigos QR, el estudio se enfoca en los métodos de cifrado como tal.

Finalmente, se tiene que la utilización de códigos QR en este proyecto, puede considerarse como una limitación por la capacidad de almacenamiento de estos, lo que no sucede si se encripta información ejecutando el algoritmo únicamente.

CONCLUSIONES

- De acuerdo con los resultados de las pruebas realizadas, se establece que el tiempo del algoritmo AES en cifrar y descifrar un mensaje de texto plano tanto en la Aplicación Móvil Android como en la Aplicación Web, es menor al tiempo que toma el algoritmo RSA en ejecutar el mismo proceso, lo que a su vez representa menor costo computacional.
- Los patrones de alineación de los códigos QR varían en función a la cantidad de caracteres que se ingrese. Para el caso del algoritmo AES, cuando se ingresa una sola letra genera un punto de alineación. Los puntos de alineación permiten la interpretación de la información contenida en el código QR, a mayor cantidad de información, más puntos de alineación se muestran. A diferencia de AES, el algoritmo RSA genera 6 puntos de alineación con una sola letra, esto debido a que la cantidad de caracteres que genera RSA en los mensajes cifrados es mayor, en el caso de AES es necesario ingresar 12 caracteres en la aplicación Web para que el código muestre 6 puntos de alineación. Para los dos casos es importante considerar que, el enlace (link) del Servlet contiene 86 caracteres, es decir que a la cantidad de caracteres que se ingresa en la aplicación Web se le debe sumar dicha cantidad. Así, para que AES muestre 6 puntos de alineación en realidad el código QR contiene el mensaje cifrado de 98 caracteres en mensaje de texto plano.
- El lector de código QR que se implementó en la aplicación móvil Android no admite el reconocimiento de los puntos de alineación de los códigos QR cuando la cantidad de información que se ingresa es demasiado grande.
- De acuerdo con los resultados de este estudio el algoritmo AES tiene un mejor rendimiento en comparación con el algoritmo RSA, considerando su velocidad de cifrado y descifrado, su facilidad de implementación en cuanto a código y menor costo computacional.

RECOMENDACIONES

- Implementar los algoritmos AES y RSA de tal manera que se utilicen las librerías hechas para estos algoritmos, así no dependerá del lenguaje de programación en el que se desarrolle o las herramientas que se utilice para la implementación. Para RSA, por ejemplo, al ser un algoritmo matemático no sería correcto implementarlo utilizando operaciones matemáticas, aunque dicho algoritmo se base en ello. Lo mejor es utilizar librerías y complementos diseñados para su desarrollo.
- Considerar la criptografía como una herramienta importante y eficiente para asegurar la información y establecer más estudios, investigaciones y análisis para elegir un algoritmo de acuerdo con el funcionamiento, estructura y características que cumplan con los requerimientos de las organizaciones.
- Hacer campañas de investigación para ampliar y mejorar el conocimiento en cuanto a algoritmos criptográficos para que se pueda aprovechar las ventajas que ofrecen y se haga un uso consciente de los mismos.
- El presente proyecto de titulación puede ser referente para trabajos futuros que se refieran al análisis e investigación de algoritmos de cifrado, en donde además se podría incluir el cifrado de documentos más grandes y archivos multimedia, lo cual supone un mayor alcance.
- Introducir el tema de aseguramiento de la información mediante criptografía / algoritmos de cifrado en alguna materia de la carrera.
- El demo QREncryption tiene la arquitectura y componentes compatibles para ser implementado en el sistema QRegistro Rápido, mismo que fue utilizado en las jornadas académicas CISIC/CSOFT para el registro de asistencia. Es altamente aplicable por lo que, se recomienda implementarlo para proteger la información.

GLOSARIO DE TÉRMINOS

AES: estándar de cifrado avanzado.

Android: nombre de un sistema operativo que se emplea en dispositivos móviles.

Asimétrico: En el contexto de la información, hace referencia a que utiliza diferente clave para cifrar y descifrar la información.

Cifrario: sistema de reglas o normas de transcripción, gracias al cual un mensaje original que contiene información oculta se transforma en un mensaje cifrado, incomprensible para personas ajenas.

HTML: De las siglas en inglés: HyperText Markup Language, en español se traduce como Lenguaje de Formato de Documentos para Hipertexto, es un lenguaje de marcado que se utiliza para el desarrollo de páginas de Internet

HTTP: De las siglas en inglés: Hypertext Transfer Protocol, en español: protocolo de transferencia de hipertextos, se utiliza en algunas direcciones de internet.

IEC: comisión electrotécnica internacional.

ISO: organización internacional de estandarización.

Java: lenguaje de programación y una plataforma informática.

Marketing: en español: márquetin, es el conjunto de técnicas y estudios que tienen como objeto mejorar la comercialización de un producto.

Plaintext: Texto plano.

Quick response: Conjunto de técnicas y métodos con el fin de comercialización de un producto.

RSA: iniciales de los apellidos de los desarrolladores: Rivest, Shamir y Adleman.

Scripts: Es un documento que contiene instrucciones, escritas en códigos de programación. El script es un lenguaje de programación que ejecuta diversas funciones en el interior de un programa de computador.

Simétrico: En el contexto de la información, hace referencia a que utiliza la misma clave para cifrar y descifrar la información.

SSL: "Secure Sockets Layer" (en español «capa de conexión segura»), permite establecer conexiones seguras a través de Internet, de forma sencilla y transparente, consiste en interponer una fase de codificación de los mensajes antes de enviarlos por la red.

TLS: "Transport Layer Security" (en español «seguridad de la capa de transporte»), protocolo criptográfico que proporciona comunicaciones seguras por una red, comúnmente Internet.

REFERENCIAS BIBLIOGRÁFICAS

Agudo, S. (2017, enero 12). Protocolo HTTPS | Seguridad en la Red. Recuperado 14 de enero de 2019, de Genbeta website: https://www.genbeta.com/a-fondo/por-que-es-importante-https-y-que-implica-no-usarlo

Andrea Pfundmeier. (2018). Cifrado AES y RSA. Recuperado 25 de julio de 2019, de https://www.boxcryptor.com/es/encryption/

Bustamante, E. R. T. (2012). *Comunicación Entre Ordenadores y la Cloud Computing*. 2.

By-Nc-Nd, C. (s. f.). *Creative Commons License Deed*. 354.

Camilo Gutiérrez Amaya. (2013a, enero 18). Funcionamiento del algoritmo RSA. Recuperado 21 de noviembre de 2018, de WeLiveSecurity website: https://www.welivesecurity.com/la-es/2013/01/18/funcionamiento-del-algoritmo-rsa/

Camilo Gutiérrez Amaya. (2013b, enero 18). Funcionamiento del algoritmo RSA. Recuperado 21 de noviembre de 2018, de WeLiveSecurity website: https://www.welivesecurity.com/la-es/2013/01/18/funcionamiento-del-algoritmo-rsa/

Chen, Q., Du, Y., Lin, R., & Tian, Y. (2012). Fast QR Code Image Process and Detection. En Y. Wang & X. Zhang (Eds.), *Internet of Things* (pp. 305-312). Springer Berlin Heidelberg.

CodeDay. (2017). Mecanismo detrás del escaneo de códigos QR de la aplicación web de WhatsApp. Recuperado 21 de mayo de 2019, de https://codeday.me/es/qa/20190219/203887.html

Córdoba, D. (2016, noviembre 8). RSA: ¿Cómo funciona este algoritmo de cifrado? Recuperado 21 de noviembre de 2018, de Junco TIC website: https://juncotic.com/rsa-como-funciona-este-algoritmo/

David Dunning. (2016). Advanced Encryption Standard. Recuperado 29 de noviembre de 2018, de Techlandia website: https://techlandia.com/funciona-aes-info_215975/

Dejan Kosutic. (2019, mayo 24). Política del uso de controles criptográficos. Recuperado 22 de julio de 2019, de 27001Academy website: https://advisera.com/27001academy/es/documentation/politica-del-uso-de-controles-criptograficos/

Delfs, H., & Knebl, H. (2015). Introduction. En H. Delfs & H. Knebl (Eds.), *Introduction to Cryptography: Principles and Applications* (pp. 1-10). https://doi.org/10.1007/978-3-662-47974-2_1

Díaz, J. C. G. (1995). *Criptografía: Historia de la escritura cifrada*. Editorial Complutense.

Enriquez, J. G., & Casas, S. I. (2014a). Usabilidad en aplicaciones móviles. *Informes Científicos - Técnicos UNPA*, *5*(2), 25-47. https://doi.org/10.22305/ict-unpa.v5i2.71

Enriquez, J. G., & Casas, S. I. (2014b). Usabilidad en aplicaciones móviles. *Informes Científicos - Técnicos UNPA*, *5*(2), 25-47. https://doi.org/10.22305/ict-unpa.v5i2.71

Fluid Attacks. (2016, octubre 22). Cifrar contraseñas con SALT. Recuperado 10 de enero de 2019, de Especialistas en Ethical Hacking | Fluid Attacks website: https://fluidattacks.com/web/es/defends/php/cifrar-contrasenas-salt/

Guamán, A. L. (2018). *DISEÑO E IMPLEMENTACIÓN DE UN NUEVO ALGORITMO CRIPTOGRÁFICO SIMÉTRICO PARA MENSAJERÍA INSTANTÁNEA EN UN ENTORNO WEB*. 163.

Hernández Encinas, L. (2016). *La criptografía*. Recuperado de http://ebookcentral.proquest.com/lib/utnortesp/detail.action?docID=4536091

Hohensee, B. (2014). *Introducción A Android Studio. Incluye Proyectos Reales Y El Código Fuente*. Babelcube Inc.

Ibarra Quevedo, R., Serrano López, M. A., & Calixto Garera y Gonzalez, C. (2010). *Teoría de la información y encriptamiento de datos*. Recuperado de http://ebookcentral.proquest.com/lib/utnortesp/detail.action?docID=3187419

IBM Knowledge Center. (2014, octubre 24). Recuperado 19 de mayo de 2019, de undefined

INEN Servicio Ecuatoriano de Normalización. (2017, abril). *INEN-ISO/IEC 27002*.

Inés Matte Urrejola. (2014). WhatsApp: ¿Qué significa el mensaje de cifrado de extremo a extremo? Recuperado 21 de mayo de 2019, de Https://www.facebook.com/teletrece website: https://www.t13.cl/noticia/tendencias/tecnologia/que-significa-nuevo-mensaje-esta-apareciendo-whatsapp

Iris-Cert. (2011, noviembre 12). Criptología. Recuperado 20 de mayo de 2019, de https://www.rediris.es/cert/doc/unixsec/node29.html

Jeremias Maerki. (2011). Barcode4j-light Java documentation Version 2.0. Recuperado 18 de enero de 2019, de https://jar-download.com/artifacts/net.sf.barcode4j/barcode4j-light/2.0/documentation

Kevin Turcios. (2015). Codigo QR: Ventajas/Desventajas. Recuperado 22 de julio de 2019, de Codigo QR website: http://kevqrcode.blogspot.com/p/ventajasdesventajas.html

Kranch, M., & Bonneau, J. (2015). Upgrading HTTPS in mid-air: An Empirical Study of Strict Transport Security and Key Pinning. *Proceedings 2015 Network and Distributed System Security Symposium*. Presentado en Network and Distributed System Security Symposium, San Diego, CA. https://doi.org/10.14722/ndss.2015.23162

Kumar, L. P., & Gupta, A. K. (2016). Implementation of speech encryption and decryption using advanced encryption standard. *2016 IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 1497-1501. https://doi.org/10.1109/RTEICT.2016.7808081

Kumar, M., & Singhal, A. (2012a). Efficient implementation of Advanced Encryption Standard (AES) for ARM based platforms. *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, 23-27. https://doi.org/10.1109/RAIT.2012.6194473

Kumar, M., & Singhal, A. (2012b). Efficient implementation of Advanced Encryption Standard (AES) for ARM based platforms. *2012 1st International Conference on Recent Advances in Information Technology (RAIT)*, 23-27. https://doi.org/10.1109/RAIT.2012.6194473

Leonardo Favio Paillacho Vinueza. (2014, mayo 19). Teoria de Autómatas y Computación: El nacimiento del HTML. Tim Berners Lee. Recuperado 23 de julio de 2019, de Teoria de Autómatas y

Computación website: http://taycalbacete.blogspot.com/2014/05/el-nacimiento-del-html-tim-berners-lee.html

Lin, H., & Lee, G. (2015). Building a Secure Cross Platform Enterprise Service Mobile Apps Using HTML5. *2015 18th International Conference on Network-Based Information Systems*, 162-166. https://doi.org/10.1109/NBiS.2015.28

Lopez, M. J. L. (2002). *Criptografía y Seguridad en Computadores*. 233.

Luján-Mora, S. (2002). *Programación de aplicaciones web: Historia, principios básicos y clientes web*. Recuperado de http://rua.ua.es/dspace/handle/10045/16995

Molina, S. N. L. (2017a). *ESTUDIO COMPARATIVO DE LA EFICIENCIA DE LOS ALGORITMOS CRIPTOGRÁFICOS AES Y RSA: CASO DE ESTUDIO DE UNA INSTITUCIÓN EN LA CIUDAD DE GUAYAQUIL*. 23.

Molina, S. N. L. (2017b). *Magíster en Auditoría de Tecnologías de la Información*. 23.

Mónica Robles, I. P. (2016). Encriptación RSA. Recuperado 21 de noviembre de 2018, de Calameo.com website: https://www.calameo.com/books/0016613242449cde12304

OBS Bissnes School. (2019, enero 10). Metodología en cascada | OBS Business School. Recuperado 10 de enero de 2019, de https://www.obs-edu.com/int/blog-project-management/metodologia-agile/pros-y-contras-de-la-metodologia-en-cascada

Ortiz Ramírez, J. M., & Balbuca Ramones, D. M. (2017). *Análisis comparativos de servidores de aplicaciones open SOURCE para la plataforma JAVA EE. caso práctico: Módulo de gestión de juntas administradoras de agua potable y riego para la dirección provincial de Chimborazo de la Senagua.* Recuperado de http://dspace.unach.edu.ec/handle/51000/3397

Pablo Dominguez. (2017, octubre 30). Modelo en Cascada. Recuperado 10 de enero de 2019, de OpenClassrooms website: https://openclassrooms.com/en/courses/4309151-gestiona-tu-proyecto-de-desarrollo/4538221-en-que-consiste-el-modelo-en-cascada

Paredes, G. G. (2006a). INTRODUCCIÓN A LA CRIPTOGRAFÍA. *Revista Digital Universitaria*, 17.

Paredes, G. G. (2006b). INTRODUCCIÓN A LA CRIPTOGRAFÍA. *Revista Digital Universitaria*, 17.

Pech-May, F., Gomez-Rodriguez, M. A., & Lara-Jeronimo, S. U. (s. f.). *Desarrollo de Aplicaciones web con JPA, EJB, JSF y PrimeFaces*. 9.

Pliego García, C. (2013). *Desarrollo de una aplicación generadora y lectora de códigos QR seguros en Android*. Recuperado de https://e-archivo.uc3m.es/handle/10016/19043

Pousa, L. A. (2011a). *ALGORITMO DE CIFRADO SIMÉTRICO AES. ACELERACIÓN DE TIEMPO DE CÓMPUTO SOBRE ARQUITECTURAS MULTICORE.* 41.

Pousa, L. A. (2011b). *ALGORITMO DE CIFRADO SIMÉTRICO AES. ACELERACIÓN DE TIEMPO DE CÓMPUTO SOBRE ARQUITECTURAS MULTICORE.* 41.

QR Code in java. (2017, junio 19). Recuperado 30 de enero de 2019, de CalliCoder website: https://www.callicoder.com/generate-qr-code-in-java-using-zxing/

Security Manager. (2011). Encriptación de la Información. Recuperado 20 de mayo de 2019, de http://www.juntadeandalucia.es/empleo/recursos/material_didactico/especialidades/materialdidactico_administrador_servidores/Content/4-seguridad/6-Encriptacion.pdf

Suarez, F. J. G., Mielgo, C. R., Cabrera, I. J., Gonzalez, J. J., & Rocha, J. (2017). *Herramienta para el auto-aprendizaje de las Matem´aticas en la Educaci´on Superior. TestAutoEval*. 8.

Tiwari, S. (2016). An Introduction to QR Code Technology. *2016 International Conference on Information Technology (ICIT)*, 39-44. https://doi.org/10.1109/ICIT.2016.021

Wallis, W. D. (2013a). Cryptography. En W. D. Wallis (Ed.), *Mathematics in the Real World* (pp. 157-167). https://doi.org/10.1007/978-1-4614-8529-2_11

Wallis, W. D. (2013b). Cryptography. En W. D. Wallis (Ed.), *Mathematics in the Real World* (pp. 157-167). https://doi.org/10.1007/978-1-4614-8529-2_11

Wazan, A. S., Laborde, R., Chadwick, D. W., Barrere, F., & Benzekri, A. (2016). How Can I Trust an X.509 Certificate? An Analysis of the Existing Trust Approaches. *2016 IEEE 41st Conference on Local Computer Networks (LCN)*, 531-534. https://doi.org/10.1109/LCN.2016.85

Wing H. Wong. (2013). Cifrado RSA. Recuperado 24 de julio de 2019, de https://www.ecured.cu/RSA#Padding_schemes_.28Esquema_de_relleno.29