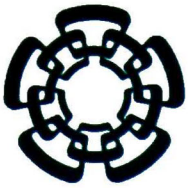
VT-T00066-CC1
Dom.-2014



CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS
DEL INSTITUTO POLITÉCNICO NACIONAL

Laboratorio de Tecnologías de Información,
CINVESTAV-Tamaulipas

# Servicio seguro de almacenamiento y búsqueda de documentos en un entorno de nube

Tesis que presenta:

**Sandra Daniela Hernández de la Rosa**

Para obtener el grado de:

**Maestro en Ciencias en Computación**

Directores de la Tesis:
Dr. Arturo Díaz Pérez, Director
Dr. Víctor Jesús Sosa Sosa, Co-Director



Cd. Victoria, Tamaulipas, México. Diciembre, 2013

La tesis presentada por Sandra Daniela Hernández de la Rosa fue aprobada por:

---

Dr. Hiram Galeana Zapién

Dr. José Luis González Compeán

Dr. Arturo Díaz Pérez, Director

Dr. Víctor Jesús Sosa Sosa, Co-Director

Cd. Victoria, Tamaulipas, México., 17 de Diciembre de 2013

A mis padres por el apoyo que siempre me han brindado.

# Agradecimientos


- A mis padres y hermano por el amor y apoyo incondicional que siempre me han brindado.
- A mis directores de tesis, el Dr. Arturo Díaz Pérez y el Dr. Víctor Jesús Sosa Sosa por el apoyo, consejos y asesoría brindada para el desarrollo de este trabajo.
- A mis revisores de tesis, el Dr. Hiram Galeana Zapién y el Dr. José Luis González Compeán por sus valiosas observaciones y recomendaciones para la mejora de esta tesis.
- Al Centro de Investigación y Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV) por la enseñanza académica de alta calidad brindada durante mis estudios de maestría.
- Al Consejo Nacional de Ciencia y Tecnología (CONACYT) por la beca de manutención otorgada durante mis estudios.
- A todos los investigadores del CINVESTAV por el conocimiento compartido durante mi estancia en la maestría.
- A mis compañeros y amigos de generación en el CINVESTAV por todos los buenos momentos y desveladas compartidas.

# Índice General

# Índice de Figuras

# Índice de Tablas

# Índice de Algoritmos

# Resumen

## Servicio seguro de almacenamiento y búsqueda de documentos en un entorno de nube

por

**Sandra Daniela Hernández de la Rosa**

Maestro en Ciencias del Laboratorio de Tecnologías de Información, CINVESTAV-Tamaulipas
Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, 2013
Dr. Arturo Díaz Pérez, Director
Dr. Víctor Jesús Sosa Sosa, Co-Director

El cómputo en la nube es un paradigma que está basado en un modelo de prestación de servicios a gran escala disponibles a través de la Internet, donde, de manera flexible los usuarios pueden disponer de una gran cantidad de recursos de cómputo bajo un modelo de pago por consumo y en función de la demanda. Estas características de costo-eficiencia han motivado a que cada día mayor número de organizaciones y personas opten por almacenar sus datos en una infraestructura de nube. Sin embargo, con la adopción del cómputo en la nube se introducen diversos desafíos de seguridad, siendo la principal preocupación de los propietarios de datos la seguridad y confidencialidad de la información.

La práctica común aplicada por los propietarios de datos para proteger la confidencialidad de sus datos es el cifrado de datos. Sin embargo, esta solución limita el uso de los mismos, particularmente la funcionalidad de búsquedas sobre los datos cifrados de tal manera que la privacidad sea preservada y que no se revele nada acerca del contenido de los datos originales. Los esquemas existentes de búsqueda sobre datos cifrados típicamente sólo soportan búsquedas simples (una sola palabra) o búsquedas booleanas. Además, por razones de eficiencia dentro del contexto de almacenamiento masivo como lo es en el cómputo en la nube, es altamente deseable que a través de la búsqueda sean recuperados únicamente los documentos más relevantes dada una consulta.

En el contexto descrito, el objetivo de este trabajo de tesis es desarrollar un esquema de servicios básicos de seguridad (confidencialidad, autenticación e integridad) para ser integrado como parte de un servicio de almacenamiento en la nube mediante el uso de técnicas y/o protocolos criptográficos. La solución que se plantea incluye la propuesta de un esquema seguro de búsqueda por relevancia y múltiples palabras clave sobre datos cifrados, el cual integra técnicas tradicionales de Recuperación de la Información (IR) para documentos de texto no estructurado y primitivas criptográficas como el Cifrado Homomórfico (HE). Se considera además, un entorno en donde el propietario de datos comparte sus documentos con múltiples usuarios con la facilidad de establecer distintos tipos de permisos de manera individual para un usuario. El mecanismo de control de acceso de grano fino que aquí se propone tiene como fundamento un Cifrado Basado en Atributos (ABE).

# Abstract

## Secure storage and search of documents in a cloud environment

by

**Sandra Daniela Hernández de la Rosa**

Master of Science from the Information Technology Laboratory, CINVESTAV-Tamaulipas
Center for Research and Advanced Studies from the National Polytechnic Institute, 2013
Dr. Arturo Díaz Pérez, Advisor
Dr. Víctor Jesús Sosa Sosa, Co-advisor

Cloud Computing is a paradigm based on a scalable service model through the Internet, which, on-demand provides flexibility to dispose a large amount of computing resources under pay-as-you-go model. Nowadays, many organizations and individuals are motivated to storing their data in a cloud infrastructure due to cost-efficiency offered. However, the adoption of the cloud computing introduces several security challenges, where, the main concern of the data owners is data security and privacy.

A commonly applied practice by data owners to protect data confidentiality is to encrypt the data before uploading them to the cloud. However, this solution restricts the use of them, particularly to maintain the search functionality over encrypted data, in such a way that privacy is preserved and without disclosing anything about the content of the original data. The existing searchable encryption schemes tipically only support simple search (one word) or boolean search. In a mass storage context such as cloud storage, for efficiency reasons; it is highly desired retrieving only the more relevant documents.

The objective of this thesis is to develope a basic security services scheme (confidentiality, authentication and integrity) to its integration into cloud storage service through the use of techniques and/or cryptographic protocols. Also, this document states the solution for secure ranked multi-

keyword search over encrypted data, which integrates traditional techniques of Information Retrieval (IR) for unstructured text documents and cryptographic primitives as Homomorphic Encryption (HE). Besides, in cloud computing the data owners may share their documents with multiples users with need to specify different access rights of individual users. The proposed scheme in this thesis uses as main cryptographic primitive the Attribute Based Encryption (ABE) to provide a fine-grained data access control mechanism.

# Nomenclatura

## Acrónimos principales

| | |
|---|---|
| **ABE** | Cifrado Basado en Atributos |
| **AE** | Cifrado Asimétrico/Llave pública |
| **BD** | Base de Datos |
| **CA** | Autoridad Certificadora |
| **CAPTCHA** | Completely Automated Public Turing Tests to Tell Computers and Humans Apart |
| **CRL** | Lista de Revocación de Certificados |
| **CSP** | Proveedor de Servicios de Nube |
| **DO** | Propietario de Datos |
| **FH** | Completamente Homomórfico |
| **HE** | Cifrado Homomórfico |
| **HMAC** | Código de Autenticación de Mensaje basado en Hash |
| **IaaS** | Infraestructura como Servicio |
| **IR** | Recuperación de Información |
| **NIST** | Instituto Nacional de Estándares y Tecnología |
| **PaaS** | Plataforma como Servicio |
| **PH** | Parcialmente Homomórfico |
| **PKI** | Infraestructura de Llave Pública |
| **RA** | Autoridad de Registro |
| **SE** | Cifrado Simétrico |
| **SaaS** | Software como Servicio |
| **StaaS** | Almacenamiento como Servicio |
| **TF-IDF** | Frequencia de Término-Frequencia Inversa de Documento |
| **TPA** | Third Party Auditor |
| **U** | Usuario |

## Notaciones generales

- $SE = (KeyGen_{SE}, Enc_{SE}, Dec_{SE})$ corresponde a un esquema de SE, donde $KeyGen_{SE}$ es el algoritmo de generación de llaves, $Enc_{SE}$ el algoritmo de cifrado y $Dec_{SE}$ el algoritmo de descifrado.
- $AE = (KeyGen_{AE}, Enc_{AE}, Dec_{AE})$ corresponde a un esquema de AE, donde $KeyGen_{AE}$ es el algoritmo de generación de llaves, $Enc_{AE}$ el algoritmo de cifrado y $Dec_{AE}$ el algoritmo de descifrado.

- $privK$ es la llave privada del esquema AE.
- $pubK$ es la llave pública del esquema AE.
- $Sign$ es un algoritmo de generación de firma digital y $Verify$ el algoritmo de verificación de firma digital.

# 1

# Introducción

## 1.1 Antecedentes y motivación para el proyecto

La creciente generación y procesamiento de grandes cantidades de información dentro de las organizaciones ha traído consigo un aumento en la demanda tanto de recursos de cómputo como de capacidades de almacenamiento y de comunicaciones, existiendo con ello una necesidad de nuevas soluciones tecnológicas como es el caso del cómputo en la nube. El cómputo en la nube es un paradigma que en años recientes ha cobrado gran auge como medio coadyuvante para solventar dicha demanda debido esencialmente a que, basado en un modelo de prestación de servicios a gran escala a través de la Internet ofrece una infinidad de recursos de cómputo (e.g., redes de acceso veloces, computadoras y servidores potentes, virtualización de alto rendimiento) disponibles bajo demanda, permitiendo a las organizaciones escalar los recursos necesarios conforme lo requieran y bajo un modelo de pago por consumo (*pay-as-you-go*). Esto genera para las organizaciones ahorros en costos de hardware, software, operaciones de cómputo, energía eléctrica y comunicaciones sobre

la red, ya que no requieren realizar una gran inversión para poder disponer de tal infinidad de recursos de cómputo contraparte a si ellas desplegaran su propia infraestructura.

Entre los principales usos del cómputo en la nube se encuentra el servicio de almacenamiento de datos, comúnmente denominado como *Storage as a Service* (StaaS). Este servicio provee la característica de elasticidad y permite a los usuarios almacenar sus datos en los servidores de la infraestructura de nube y acceder a ellos en cualquier momento y desde cualquier lugar (*anytime-anywhere*). Sin embargo, los servicios de almacenamiento en la nube enfrentan retos apremiantes para satisfacer requerimientos de alta disponibilidad, confiabilidad, rendimiento, replicación y consistencia de los datos [1]. Además, dado que StaaS radica en una infraestructura en donde los recursos (discos duros, máquinas virtuales) son virtualmente compartidos por distintas organizaciones bajo un modelo multi-inquilino, la protección de datos personales y sensitivos almacenados en la nube es extremadamente importante durante cada una de las fases del ciclo de vida de los datos [2]: generación, transferencia, uso, compartición, almacenamiento, archivado y destrucción.

Bajo el contexto anterior este trabajo de tesis se enfoca: proveer mecanismos de seguridad para un servicio de almacenamiento en la nube.

## 1.2 Planteamiento del problema

El actual crecimiento del cómputo en la nube ha motivado a que cada día mayor número de organizaciones y usuarios opten por almacenar sus datos en una infraestructura de nube, aprovechando con ello las ventajas que el servicio ofrece: disponibilidad, escalabilidad, costo-eficiencia, entre otras, y olvidándose de restricciones en capacidades de almacenamiento. Sin embargo, existe una preocupación inminente en torno al nivel de seguridad que el cómputo en la nube es capaz de garantizar. El hecho de que los propietarios de datos almacenen datos sensitivos en la infraestructura de nube genera en ellos una gran preocupación con respecto a la pérdida de control sobre sus datos

[3], pues consideran al proveedor del servicio de almacenamiento en la nube como no confiable por ser una entidad externa a su organización.

Como forma de mitigar sus preocupaciones, los propietarios proceden a cifrar sus datos previo a la transferencia hacia el almacenamiento de nube, para con ello preservar la confidencialidad de los datos evitando accesos no autorizados. Sin embargo, un problema que prevalece con el cifrado de datos es la limitación en el uso de los mismos, particularmente mantener la funcionalidad de efectuar búsquedas sobre los datos tal como si estuvieran en claro. La solución más simple consistiría en descargar todos los datos y de forma local descifrarlos para ejecutar la búsqueda, pero esto sin duda alguna es visiblemente impráctico pues incurre en altos costos en términos de comunicación y cómputo.

Si bien existen algunas propuestas en la literatura para efectuar búsquedas sobre datos cifrados su funcionalidad es limitada. Por ejemplo, *Searchable Encryption* [4, 5, 6] es una primitiva criptográfica que permite efectuar búsquedas de forma segura sobre datos cifrados. Utiliza un índice seguro [6] como estructura de datos y sólo a través de una trampilla (*trapdoor*) válida generada mediante una llave secreta a partir de un palabra, permite verificar la existencia de ésta en un documento sin necesidad de descifrar. No obstante, sólo soporta una búsqueda booleana convencional, es decir, identifica la presencia o ausencia de una palabra sin ser posible capturar un valor de relevancia para los documentos en relación al resultado de una consulta.

Dentro del contexto de almacenamiento a gran escala como lo es en el cómputo en la nube, un usuario debería tener la facilidad de efectuar búsquedas y recuperar únicamente los documentos de mayor interés y a los cuales tiene acceso, y no todos aquellos que contienen la palabra consultada. Realizar la descarga completa de documentos que contienen la palabra clave consultada puede traducirse a una carga de tráfico innecesaria en la red cuando algunos de esos documentos no

son relevantes para la búsqueda. En el área de la Recuperación de la Información (IR por sus siglas en inglés) existen diversas técnicas [7] (TF-IDF, modelo de espacio vectorial, modelos probabilísticos como Okapi BM25) ya bien establecidas para medir la relevancia de los documentos conforme a una consulta. Sin embargo, esta área aún no ha sido ampliamente aplicada a búsquedas sobre datos cifrados. Además, para dar flexibilidad al proceso de búsqueda es preciso que el método soporte búsquedas por múltiples palabras, lo cual también coadyuva a una mejor precisión del resultado de búsqueda por relevancia, ya que en una búsqueda simple de una única palabra es posible que se presenten demasiados resultados secundarios.

Aunado a lo anterior, en un ambiente de compartición de archivos los propietarios de los datos pueden autorizar a los usuarios el acceso total o parcial de sus datos cifrados. Es por ello, que se tiene como requerimiento contar con un mecanismo de control de acceso de grano fino, el cual se puede entender como el cumplimiento del acceso basado en políticas específicas, de manera tal que se facilite y sea flexible la asignación de diferentes tipos de permisos de manera individual para un usuario. Esta situación se ve reflejada en lo siguiente: cuando múltiples usuarios tienen distintos permisos y comparten un mismo índice se presenta la dificultad de garantizar que la búsqueda no se extienda más allá de lo que tienen como permitido, ya que una consulta implica acceder al índice completo.

El principal requerimiento que se tiene al momento de realizar búsquedas sobre datos cifrados es que la seguridad sea preservada, particularmente sobre los siguientes aspectos [4]:

- Secreto demostrable para el cifrado. Un servidor no confiable no debe ser capaz de aprender nada acerca del texto plano dado un texto cifrado.

- Búsqueda controlada. Un servidor no confiable no debe realizar búsquedas sin la autorización del usuario.

- Consulta oculta. La petición de búsqueda debe realizarse de manera tal de no revelar la palabra secreta al servidor.

- Aislamiento de la consulta. El servidor no debe aprender más allá del resultado de búsqueda.

Dada la problemática planteada en esta sección, es entonces que surge la pregunta de investigación motivo de este trabajo de tesis: *¿Es posible diseñar e implementar un esquema de seguridad que cumpla con aspectos de confidencialidad, autenticación, e integridad en un servicio de almacenamiento en un entorno de nube, y que ofrezca flexibilidad en el control de acceso y sea eficaz en la búsqueda de información?*.

## 1.3 Objetivos

El presente trabajo de tesis tiene como objetivo general: diseñar e implementar un esquema de servicios de seguridad para ser integrado como parte de un servicio de almacenamiento de documentos en la nube y que permita la indización de archivos cifrados y la aplicación de políticas de seguridad para controlar el acceso a los archivos.

A fin de alcanzar el objetivo general, se han planteado los siguientes objetivos específicos:

- Diseñar y construir un esquema de servicios de seguridad que incluya confidencialidad e integridad de archivos y autenticación de usuarios.

- Diseñar y construir un mecanismo de indización de archivos cifrados que permita realizar consultas a través de palabras clave de forma segura.

- Diseñar y construir un mecanismo de control de acceso que permita compartir archivos de forma flexible.

## 1.4 Metodología

La metodología a seguir para cumplir con los objetivos estipulados del presente trabajo de investigación consta de las siguientes fases:

- La revisión del estado del arte de los trabajos relacionados con la problemática en particular de cada servicio de seguridad planteado como objetivo en esta tesis.
- El diseño arquitectural de los servicios de seguridad y modelado funcional del sistema.
- La realización de un estudio comparativo y/o la implementación y evaluación de las distintas técnicas criptográficas que serán fundamento de los esquemas de servicios de seguridad a proponer.
- El diseño de los distintos esquemas de servicios de seguridad de la arquitectura y su implementación.
- El diseño experimental y evaluación del desempeño de los esquemas de servicios de seguridad propuestos.

## 1.5 Organización del trabajo de tesis

El contenido de esta tesis se encuentra conformado por siete capítulos organizados de la siguiente manera: el Capítulo 2 presenta los conceptos teóricos básicos para la comprensión del resto del documento. En el Capítulo 3 se describe la arquitectura general de servicios de seguridad propuesta, las entidades que la integran y sus funcionalidades. En el Capítulo 4 se presentan los esquemas propuestos de los servicios de seguridad a nivel de almacenamiento y acceso a datos (autenticación, control de acceso y confidencialidad), y en el Capítulo 5 los esquemas de los servicios de seguridad a nivel de funcionalidades sobre los datos (indización, búsqueda e integridad). Posteriormente, en el

Capítulo 6 se describe la experimentación realizada y los resultados obtenidos con la implementación de la propuesta. Finalmente, en el Capítulo 7, conforme a los resultados obtenidos se presentan las conclusiones y el trabajo futuro a realizar.

# 2

# Marco teórico

## 2.1 Cómputo en la nube

El paradigma del cómputo en la nube emergente en años recientes se basa en un modelo de prestación de servicios a gran escala disponibles a través de la Internet, en donde los usuarios pueden acceder de forma flexible a gran cantidad de recursos de cómputo conforme demanda y bajo un modelo de pago por consumo. Diversas definiciones han sido proporcionadas para el cómputo en la nube, sin embargo, en esta tesis se considera la definición formal especificada por el Instituto Nacional de Estándares y Tecnología (NIST por sus siglas en inglés) la cual es descrita en la Definición 1.

**Definición 1.** ***Cómputo en la nube*:**

*Modelo para habilitar de forma ubicua, conveniente y bajo demanda de acceso a red el compartir una fuente de recursos de cómputo configurables (e.g., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser rápidamente provisionados y liberados con el mínimo esfuerzo de gestión o de interacción por parte del proveedor del servicio.*

Además de la definición anterior y tal como se muestra en la Figura 2.1, el modelo del cómputo en la nube definido por NIST está compuesto por:

- Características esenciales: autoservicio por demanda, amplio acceso a red, agrupación de recursos, rápida elasticidad y servicio medido.
- Modelos de servicios: Software como Servicio (SaaS por sus siglas en inglés), Plataforma como Servicio (PaaS por sus siglas en inglés) e Infraestructura como Servicio (IaaS por sus siglas en inglés).
- Modelos de despliegue: nube privada, nube pública, nube comunitaria y nube híbrida.

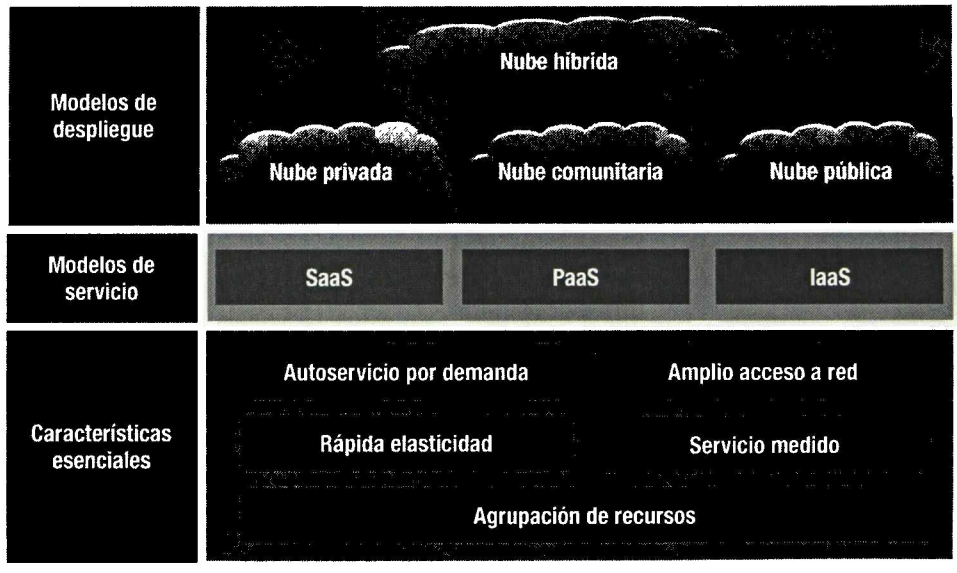



**Figura 2.1**: Modelo de Cómputo en la Nube (Definición provista por NIST)

## 2.1.1 Características esenciales

Acorde a la definición de cómputo en la nube presentada por NIST, existen cinco características esenciales que deben ser habilitadas para toda infraestructura de nube [8]:

- Autoservicio por demanda. Un consumidor[1] puede disponer de las capacidades de cómputo, tales como tiempo del servidor y almacenamiento en red, automáticamente en función de la demanda.

- Amplio acceso a red. Las capacidades están disponibles en la red para ser accedidas a través de mecanismos estándares y el uso de diferentes tipos de dispositivos (e.g., teléfonos móviles, tabletas, computadoras portátiles, y estaciones de trabajo).

- Agrupación de recursos. Los recursos de cómputo del proveedor se encuentran agrupados para servir a múltiples consumidores utilizando un modelo multi-inquilino, diferentes recursos físicos y virtuales son dinámicamente asignados y reasignados de acuerdo a la demanda del consumidor. Existe un sentido de independencia de la ubicación en que el consumidor generalmente no tiene control o conocimiento sobre la ubicación exacta de los recursos proporcionados, pero puede especificar la ubicación a alto nivel de abstración (e.g., país, estado, o centro de datos). Entre estos recursos se incluye el almacenamiento, procesamiento, memoria y ancho de banda de la red.

- Rápida elasticidad. Las capacidades deben ser rápida y elásticamente provisionadas y liberadas de tal forma que se alcance escabilidad. Para el consumidor, estas capacidades a menudo parecen ser ilimitadas y pueden ser adquiridas en cualquier cantidad y en cualquier momento.

- Servicio medido. Los modelos de servicio de cómputo en la nube automáticamente controlan y optimizan el uso de recursos en un cierto nivel de abstracción, realizan una medición apropiada según el tipo de servicio (e.g., almacenamiento, procesamiento, ancho de banda y cuentas de usuario activas) bajo un modelo de pago por consumo. El uso de recursos puede ser monitoreado, controlado y reportado, proveyendo transparencia para ambos, el proveedor y el consumidor del servicio utilizado.

[1] El término consumidor es equivalente al término usuario.

### 2.1.2 Modelos de servicio

Dentro del cómputo en la nube distintos tipos de servicios pueden ser provistos a un consumidor por parte de un proveedor de nube. La clasificación de estos servicios recae en los siguientes modelos:

- Software como Servicio (SaaS por sus siglas en inglés).
- Plataforma como Servicio (PaaS por sus siglas en inglés).
- Infraestructura como Servicio (IaaS por sus siglas en inglés).

#### *2.1.2.1. Software como Servicio (SaaS)*

En este modelo el software es ofrecido al consumidor por el proveedor de servicios mediante aplicaciones que se ejecutan en una infraestructura de nube. Dichas aplicaciones pueden ser accedidas a través de un navegador web y desde cualquier dispositivo cliente. El consumidor no administra ni posee control alguno sobre los recursos de la infraestructura de nube incluyendo la red, servidores, sistemas operativos o almacenamiento. Sin embargo, limitadamente puede configurar ciertas características específicas de la aplicación. Algunos ejemplos de este modelo de servicio son: Gmail[2] quien es un proveedor de correo electrónico basado en la web, y Salesforce.com[3] quien brinda el uso de aplicaciones *Customer Relationship Management* (CRM).

#### *2.1.2.2. Plataforma como Servicio (PaaS)*

Este modelo ofrece al consumidor la capacidad de desplegar aplicaciones sobre una infraestructura de nube mediante el uso de lenguajes, bibliotecas, servicios y herramientas soportadas por el proveedor de servicios. De la misma manera que en el modelo SaaS el consumidor no administra ni controla los recursos de red, servidores, sistemas operativos o almacenamiento de la infraestructura de nube, pero tiene control sobre las aplicaciones implementadas y posiblemente sobre los ajustes de configuración

[2] gmail.com
[3] www.salesforce.com

para el entorno de hospedaje de la aplicación. Servicios como el de Google App Engine[4] y Windows Azure[5] recaen bajo este modelo pues permiten el desarrollo de aplicaciones para una infraestructura de nube en lenguajes como .NET, PHP, Java y Python.

#### *2.1.2.3. Infraestructura como Servicio (IaaS)*

Dentro de los modelos de servicios del cómputo en la nube la capa física corresponde al modelo IaaS, el cual permite al consumidor disponer de procesamiento, almacenamiento, redes y otros recursos de cómputo. El consumidor es capaz de instalar y ejecutar software que puede incluir sistemas operativos y aplicaciones. Mediante virtualización el consumidor puede tener control sobre los sistemas operativos, almacenamiento y aplicaciones implementadas, y limitadamente sobre los componentes de red (e.g., cortafuegos). Los servicios de Google Compute Engine[6] y Amazon Elastic Compute Cloud[7] son ejemplos de servicios que ofrecen al cliente la posibilidad de instanciar y administrar máquinas virtuales para un cómputo escalable.

### 2.1.3 Modelos de despliegue

Los modelos de servicio explicados en la sección anterior pueden ser desplegados ya sea de forma privada, alojados en las instalaciones del propio consumidor de nube, compartidos entre un número limitado de socios de confianza, representados por una tercera entidad o ser un servicio de acceso público. Estas opciones presentan una serie de prestaciones que también varían con respecto a la forma en la que los consumidores pueden disponer, controlar y escalar los recursos de cómputo. Por tanto, según sean implementados dichos recursos se distinguen cuatro modelos de despliegue para una infraestructura de nube:

- Nube privada. La infraestructura de nube está dedicada para uso exclusivo de una organización

[4] developers.google.com/appengine
[5] www.windowsazure.com
[6] cloud.google.com/products/compute-engine
[7] aws.amazon.com/es/ec2

con múltiples consumidores o unidades de negocio. Puede ser administrada y operada por la organización o por una tercera parte, o una combinación de ambos, y puede existir dentro o fuera de las instalaciones.

- Nube comunitaria. La infraestructura de nube es proveída para uso exclusivo de un grupo de organizaciones con asuntos compartidos como por ejemplo misión, requerimientos de seguridad, políticas y consideraciones de cumplimiento. La nube comunitaria puede ser administrada y operada por una o más de las organizaciones que forman parte de la comunidad, o también por una tercera parte o una combinación de ambas, y puede existir dentro o fuera de las instalaciones de las organizaciones.

- Nube pública. La infraestructura de nube es proveída para uso abierto del público en general. Puede ser administrada y operada por una empresa, por una organización académica o de gobierno, o por una combinación de ambas. Radica en las instalaciones del proveedor de nube.

- Nube híbrida. La infraestructura de nube es una composición de dos o más distintas infraestructuras de nube (privada, comunitaria, pública) que siguen siendo únicas, pero están unidas por una tecnología estandarizada o propietaria que habilita los datos y la portabilidad de la aplicación (balanceo de carga entre las nubes).

### 2.1.4 Seguridad en el cómputo en la nube

Si bien el cómputo en la nube trae consigo un conjunto de ventajas tales como: un modelo de pago por consumo, la disponibilidad de una infinidad de recursos, elasticidad, simplicidad en el uso y gestión de los servicios, entre otras, el principal desafío de este paradigma es la seguridad de la información almacenada y/o procesada en la nube. En particular, los desafíos y problemas de seguridad varían de acuerdo al modelo de servicio del cómputo en la nube que se utilice [9], lo cual se describe a continuación.

### *Seguridad en SaaS*

En SaaS, a fin de garantizar la seguridad de los datos de las empresas diversos aspectos deben ser cubiertos a través de las capas de este modelo (véase Figura 2.2). Los siguientes elementos son clave para el despliegue seguro de una aplicación SaaS:

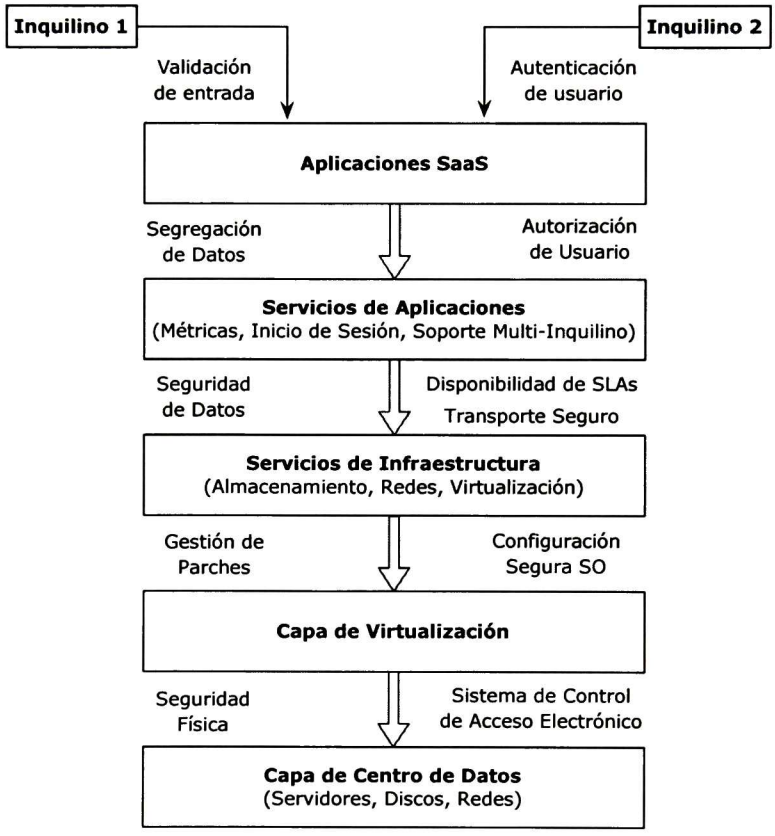



**Figura 2.2:** Seguridad en SaaS [9]

- Seguridad de datos. El proveedor de servicios debe adoptar controles de seguridad adicionales como técnicas de cifrado para prevenir determinadas vulnerabilidades: secuencias de comandos en sitios cruzados (XSS por sus siglas en inglés), debilidades de control de acceso, inyecciones de flujo OS o SQL, falsificaciones de peticiones en sitios cruzados (CSRF por sus siglas en inglés), manipulación de cookies, almacenamiento inseguro, entre otras.

- Seguridad de red. El flujo de datos a través de la red debe ser asegurado con el fin de prevenir pérdidas de información sensitiva. Usuarios maliciosos incluyendo terceros, podrían aprovechar debilidades en la gestión de sesiones y configuraciones inseguras de red para penetrar en ésta y analizar paquetes.

- Localización de datos. Debido a las leyes de conformidad y privacidad de varios países, la ubicación geográfica de los servidores de almacenamiento es de suma importancia para las organizaciones. En países como EUA ciertos tipos datos no deben ser almacenados fuera de los límites geográficos del país. Por tanto, el proveedor de servicios debe ser capaz de proporcionar fiabilidad ante el propietario de la información de la correcta ubicación de sus datos.

- Integridad de datos. Es uno de los elementos más críticos de cualquier sistema. En un sistema distribuido como el cómputo en la nube, las transacciones a través de múltiples fuentes de datos deben ser manejadas correctamente en modo a prueba de fallos. Las APIs de los servicios web de la mayoría de los proveedores de SaaS no ofrecen soporte de transacciones para garantizar la integridad de los datos.

- Segregación de datos. En el modelo multi-inquilino los datos de múltiples usuarios residen en una misma ubicación. Por tanto, la intrusión de datos se hace posible en este entorno. El modelo SaaS debe disponer de mecanismos que de manera inteligente puedan segregar los datos de diferentes usuarios, no sólo a nivel físico sino también a nivel de aplicación.

- Acceso a datos. Con el fin de evitar intrusiones no deseadas a los datos almacenados en la nube, es necesario el uso de políticas de acceso adicionales. Por tal motivo, el modelo SaaS debe ser suficientemente flexible para permitir que el propietario de datos pueda incorporar políticas específicas definidas por una organización.

- Autenticación y autorización. El uso de múltiples aplicaciones de SaaS incrementa la carga en la gestión adecuada de cuentas de usuario.

- Seguridad de aplicaciones web. Las principales amenazas de seguridad que enfrentan las aplicaciones web en SaaS son: fallas de inyección como SQL, OS y LDAP, secuencias de comandos en sitios cruzados (XSS), autenticación y administración de sesiones rotas, mala configuración de seguridad, falsificaciones de peticiones en sitios cruzados (CSRF), almacenamiento criptográfico inseguro, fallos de restricción de accesos URL y redireccionamientos inválidos.

- Violaciones de datos. En un entorno de nube, existe el riesgo de que se presenten incumplimientos internos por parte de los empleados de los proveedores de SaaS para atacar los datos de los usuarios.

- Vulnerabilidades en virtualización. Garantizar que las diferentes instancias virtuales que se ejecutan en una misma máquina física estén aisladas unas de otras es un importante desafío de seguridad.

- Disponiblidad. La aplicación de SaaS debe ser resistente ante fallos de hardware/software y a ataques de denegación de servicios para con ello asegurar en todo momento la disponibilidad de los datos.

- Respaldo. El proveedor de SaaS debe garantizar que todos los datos sensibles de las empresas sean respaldados regularmente para facilitar su rápida recuperación en caso de desastres.

- Gestión de identidad (IdM). Identifica a los individuos en un sistema y controla el acceso a los recursos de acuerdo a las restricciones establecidas para las identidades. IdM debería ser altamente configurable para facilitar la conformidad de las políticas de una empresa, así también, el proveedor de SaaS debe garantizar la seguridad de las credenciales durante el tránsito y almacenamiento.

***Seguridad en PaaS***

El modelo PaaS permite a los desarrolladores construir sus propias aplicaciones otorgándoles también mayor flexibilidad para añadir una capa de seguridad adicional. El proveedor debe ofrecer garantías sólidas de que los datos permanecen inaccesibles entre aplicaciones. En este modelo, debe evaluarse la eficacia de los programas de seguridad de aplicaciones mediante indicadores y cobertura de parches de seguridad. Entre los riesgos aquí presentes se encuentran los ataques a los códigos de las aplicaciones ejecutados por hackers.

***Seguridad en IaaS***

En el modelo IaaS, las responsabilidades de seguridad se encuentran compartidas entre proveedor y consumidor. El proveedor es responsable de la seguridad del hypervisor [8], lo que significa que debe abordar controles de seguridad para la seguridad física y la seguridad de la virtualización. Por otro lado, el consumidor es responsable de los controles de seguridad relacionados con el sistema IT (sistema operativo, aplicaciones y datos).

## 2.2 Criptografía

Proveniente de las palabras griegas *krypto* "ocultar" y *graphien* "escribir", la criptografía es el elemento fundamental para el desarrollo de aplicaciones en donde la seguridad de la información desea ser preservada. Esto en el sentido de poder habilitar a dos entidades el comunicarse sobre un canal inseguro, de tal forma que una tercera entidad (adversario) no sea capaz de entender o alterar el mensaje transmitido. Presente desde tiempos muy antiguos, la criptografía en sus inicios fue vista de forma homóloga al concepto de cifrado de datos, uno de los primeros cifradores utilizados es el cifrador César que data a principios de esta era. Estos inicios de la criptografía son conocidos como "criptografía clásica"

[8] Monitor de Máquina Virtual (VMM por sus siglas en inglés).

Con el avance de las computadoras, las tecnologías de información y las comunicaciones digitales, la criptografía ha evolucionado a una forma moderna en donde sus bases radican no sólo en la teoría matemática sino también en las ciencias de la computación y en la ingeniería. El objetivo de la criptografía moderna es el análisis y diseño de algoritmos, protocolos y criptosistemas que permitan preservar la seguridad de la información, comunicaciones y entidades.

En las siguientes subsecciones se introducen los conceptos básicos que son campo de estudio de la criptografía moderna para el desarrollo de protocolos y aplicaciones, y que involucran algoritmos criptográficos, funciones criptográficas y servicios de seguridad (véase Figura 2.3).

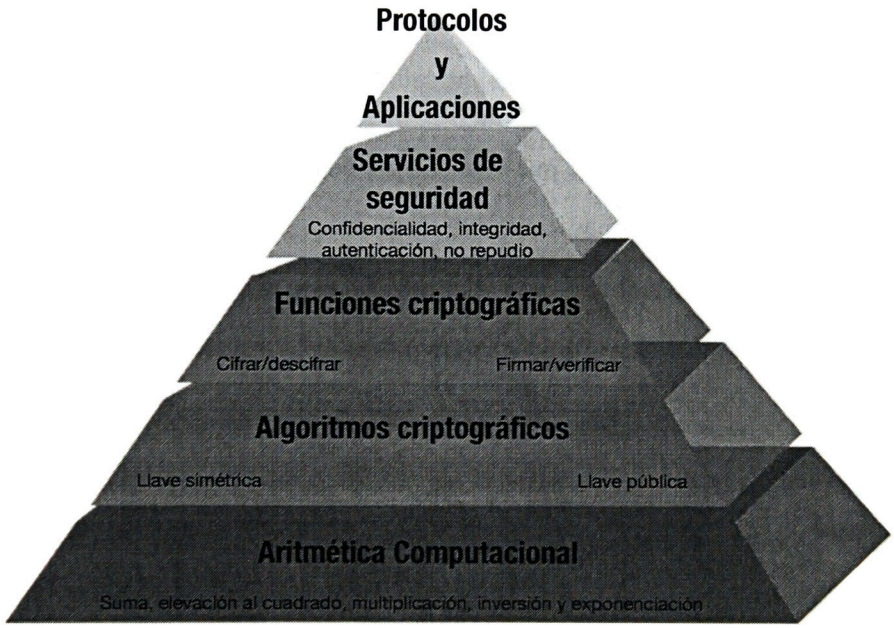



**Figura 2.3:** Modelo de Capas para Sistemas de Seguridad

### 2.2.1 Servicios de seguridad

Cuando dos entidades se comunican a través de un canal de comunicación inseguro es necesario hacer frente a eventuales amenazas y problemas relacionados con la seguridad de la información, para ello dentro de la criptografía han sido definidos un conjunto de servicios de seguridad:

- Confidencialidad. Este servicio garantiza que la información privada sólo pueda ser accedida por usuarios o entidades autorizadas.
- Integridad. Se enfoca en la protección contra la modificación o destrucción no autorizada de la información. Esto incluye cualquier inserción, eliminación o substitución de datos. El usuario debe tener la facilidad de verificar si la información ha sido alterada.
- Autenticación. Este servicio es utilizado para determinar el origen de la información, es decir, consiste en confirmar la identidad de una entidad o de un mensaje.
- No repudio. Impide que una entidad que se encuentra bajo acuerdo entre otras entidades niegue cualquier acción de la cual fue participe.

### 2.2.2 Funciones hash

Una función hash, $H$, es una función matemática que toma una entrada $x$ de tamaño arbitrario y produce como salida un valor de tamaño fijo denominado resumen o digesto del mensaje $h = H(x)$. El número de bits de salida se determina por el diseño de la función hash y es independiente del tamaño del mensaje de entrada. Estas funciones tienen amplia aplicación dentro de la criptografía, por ejemplo, son utilizadas como un componente de algoritmos y esquemas de seguridad para determinar la integridad de un mensaje, en donde cualquier cambio mínimo en el mensaje resulta en un resumen de mensaje diferente. Esta propiedad es útil para que las funciones hash sean utilizadas como generadores de números aleatorios y como bloque básico de las firmas digitales y de los Códigos

de Autenticación de Mensaje basados en Hash (HMAC, por sus siglas en inglés).

#### 2.2.2.1. *Propiedades*

Toda función hash debe cumplir con las siguientes propiedades:

- La entrada puede ser de cualquier longitud.
- La salida tiene una longitud fija.
- $H(x)$ es relativamente fácil de calcular para cualquier $x$ dado.
- Función de sólo ida (*One-way*). Dado un valor hash $h$, es computacionalmente imposible reconstruir $x$ tal que $H(x) = h$.
- Función débilmente libre de colisión (*weakly collision-free*). Dada una entrada $x$, es computacionalmente imposible encontrar una segunda entrada $x'$ que sea diferente de $x$ tal que $H(x) = H(x')$.
- Función fuertemente libre de colisión (*strongly collission-free*). Si $H$ es una funcion hash es computacionalmente imposible encontrar dos entradas $x$ y $x'$ tal que $H(x) = H(x')$.

#### 2.2.2.2. *Algoritmos*

Actualmente, acorde a la versión más reciente de los estándares de NIST siete algoritmos hash son referidos como seguros [10], véase Tabla 2.1.

#### 2.2.2.3. *Código de Autenticación de Mensaje basado en Hash (HMAC)*

Un Código de Autenticación de Mensaje basado en Hash tiene como propósito autenticar el origen de un mensaje y verificar su integridad, es una de las aplicaciones de las funciones hash. Un

| Algoritmo | Long. del msj. (bits) | Long. del bloque (bits) | Long. de palabra (bits) | Long. del resumen del msj. (bits) |
|---|---|---|---|---|
| SHA-1 | $< 2^{64}$ | 512 | 32 | 160 |
| SHA-224 | $< 2^{64}$ | 512 | 32 | 224 |
| SHA-256 | $< 2^{64}$ | 512 | 32 | 256 |
| SHA-384 | $< 2^{128}$ | 1024 | 64 | 384 |
| SHA-512 | $< 2^{128}$ | 1024 | 64 | 512 |
| SHA-512/224 | $< 2^{128}$ | 1024 | 64 | 224 |
| SHA-512/256 | $< 2^{128}$ | 1024 | 64 | 256 |

**Tabla 2.1:** Propiedades de algoritmos hash seguros

HMAC utiliza dos parámetros distintos, un mensaje entrante y una llave secreta conocida sólo por el originador del mensaje y los receptor(es) destinados. El emisor hace uso de una función HMAC para producir el valor MAC que será enviado al receptor en conjunto con el mensaje. Cuando el receptor recibe el mensaje procede a calcular el MAC del mensaje utilizando la misma llave secreta y la función HMAC que empleó el emisor, compara ambos resultados MAC. Si los resultados coinciden, entonces el mensaje es íntegro y el emisor auténtico. La especificación formal de un HMAC puede ser vista en la Definición 3 proporcionada por NIST [11].

**Definición 2.** ***Especificación HMAC*:**

*Sea $H$ una función hash aprobada, $K$ la llave secreta compartida entre originador y receptor, $K_0$ la llave $K$ resultante del preprocesamiento para formar una llave de $B$ bytes, $t$ el número de bytes MAC, $ipad$ un relleno interno de bytes, $opad$ un relleno externo de bytes, la siguiente operación es aplicada para calcular el MAC sobre el mensaje $text$:*

$$MAC(text)_t = HMAC(K, text)_t = H((K_0 \oplus opad) \,||\, H(K_0 \oplus ipad) \,||\, text))_t$$

### 2.2.3 Criptografía simétrica

La criptografía simétrica es el bloque básico para construir un sistema seguro que requiere confidencialidad, el cual consta de dos entidades que con el fin de intercambiar mensajes de forma confidencial comparten una misma llave secreta (simétrica). Dicha llave se usa entonces tanto para cifrar como para descifrar un mensaje. La Figura 2.4 ilustra el proceso básico de este esquema criptográfico, una entidad $A$ transforma un mensaje en claro en mensaje cifrado a través de un algoritmo de cifrado en conjunto con una llave secreta, cuando la entidad $B$ recibe el mensaje cifrado procede a transformarlo en mensaje en claro utilizando el algoritmo de descifrado y la misma llave secreta que fue compartida con $A$.

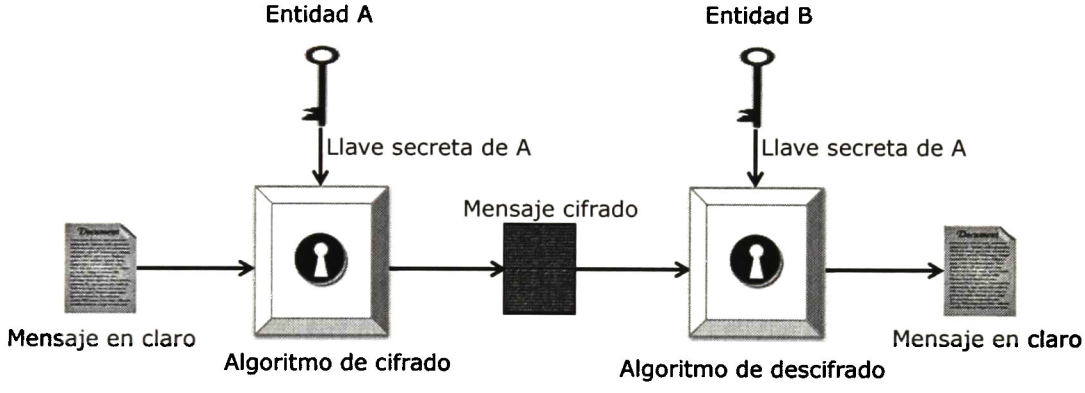



**Figura 2.4:** Esquema básico de la criptografía simétrica

La seguridad del esquema depende del hecho que la llave secreta sea conocida únicamente por las dos entidades que se comunican los mensajes, para ello es requerido que el intercambio de llave se realice a través de un canal de comunicación seguro o bien, haciendo uso de la criptografía asimétrica, la cual será descrita en la siguiente sección. La desventaja de la criptografía simétrica es que en un ambiente de red y dado que entre cada par de entidades se comparte una única llave, para $n$ entidades tenemos que $\frac{n*(n-1)}{2}$ pares de diferentes llaves serán generados. Este hecho sin duda trae consigo problemas de gestión y de almacenamiento seguro de dicha cantidad de llaves cuando $n$ es grande.

### *Tipos de cifradores*

Los algoritmos criptográficos simétricos son utilizados para cifrar grandes volúmenes de datos debido a su eficiencia. Son clasificados en dos categorías:

- Cifradores por bloque. Los mensajes son cifrados a través de transformaciones sobre bloques de datos de una longitud fija de bits. Entre los algoritmos actuales mayormente utilizados se encuentran: Triple-DES, IDEA y AES.

- Cifradores en flujo. Denominados también cifradores de estado, se basan en el método de cifrado libreta de un solo uso (OTP por sus siglas en inglés), éste consiste en cifrar un mensaje bit a bit mediante operaciones XOR. Estos algoritmos son especialmente útiles en situaciones en la que los errores de transmisión son altamente probables, principalmente aplicados dentro de la telefonía móvil. Los algoritmos A5 y RC4 son ejemplos de este tipo de cifradores.

## 2.2.4 Criptografía asimétrica

En 1976 Diffie y Hellman introducen el concepto de criptografía de llave pública [12], este tipo de criptografía asigna a cada entidad un par de llaves, una privada y la otra pública, la llave pública será distribuida para ser conocida por todas las entidades del sistema. Por ejemplo, cuando la entidad $A$ desea emitir un mensaje cifrado a la entidad $B$ cifrará el mensaje en claro mediante un algoritmo de cifrado y la llave pública de $B$, posteriormente $B$ transformará el mensaje cifrado en mensaje en claro a través del algoritmo de descifrado y su propia llave privada, véase Figura 2.5.

### *Algoritmos Asimétricos*

La seguridad de los algoritmos criptográficos asimétricos está basada en la dificultad de resolver ciertos problemas matemáticos, en relación a esto se presentan las siguientes tres categorías:

- Algoritmos basados en el problema de factorización de enteros. Dado un entero $n$ es requerido descomponerlo en sus factores primos $p$ y $q$. Un algoritmo que recae en esta categoría es el

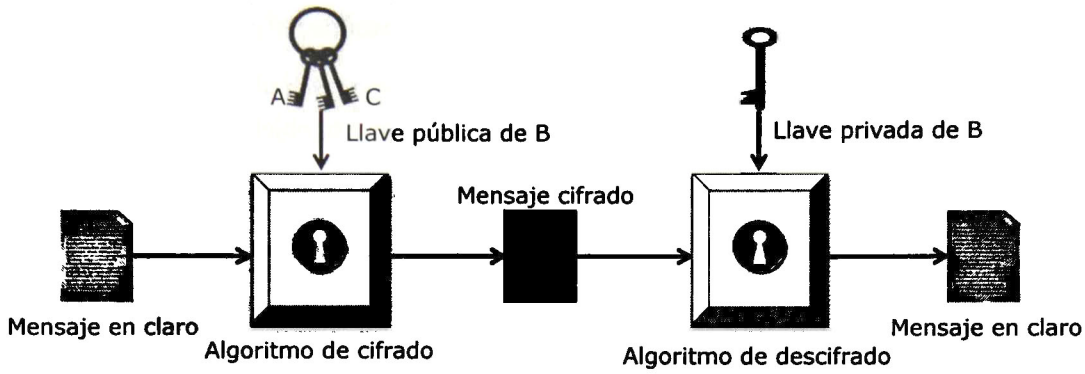



**Figura 2.5:** Esquema básico de la criptografía asimétrica

algoritmo RSA.

- Algoritmos basados en el problema de logaritmo discreto. Dado $\alpha$ y $\beta$ se requiere encontrar $x$ tal que $\beta = \alpha^x \bmod p$. Ejemplos de esta categoría de algoritmos son DSA y ElGamal.

- Algoritmos basados en curvas elípticas. Propuestos en 1985 de forma independiente por Miller [13] y Koblitz [14], la relevancia de estos algoritmos radica en el hecho de ofrecer un mismo nivel de seguridad utilizando una llave de menor longitud a la de otros algoritmos asimétricos, lo cual reduce el tiempo de procesamiento. La seguridad de estos algoritmos está basada en resolver el problema de logaritmo discreto sobre el conjunto de puntos de la curva, una curva elíptica está definida sobre la ecuación $y^2 = x^3 + ax + b$.

En general, estos algoritmos no son recomendados para el cifrado de grandes volúmenes de datos debido a la aritmética intensiva que realizan, caso contrario a los algoritmos criptográficos simétricos que son más eficientes para ello. Sin embargo, los algoritmos asimétricos son utilizados para proveer un intercambio seguro de llaves secretas o para la generación de firmas digitales de documentos.

### 2.2.4.1. *Firma digital*

Las firmas digitales son parte de la criptografía de llave pública, y su objetivo consiste en añadir un bloque de datos al mensaje a través de una operación sobre el hash del mensaje y el uso de una llave privada. La Figura 2.6 ilustra el proceso de generación y verificación de una firma digital. En primer instancia, la entidad $A$ aplica una función hash al mensaje original $H(m)$, el resumen es firmado digitalmente mediante una operación privada y el uso de la llave privada de $A$. Posteriormente, la firma digital se añade al mensaje original y éste se envía a la entidad $B$ que procede a aplicar la función hash al mensaje recibido para también obtener $H(m)$. Por otra parte, utilizando una operación pública y la llave pública de $A$ se calcula $H'(m)$, que finalmente se compara con el resumen $H(m)$. Si los dos resúmenes son iguales la firma digital se considera válida, y en caso contrario es considerada como una firma inválida.

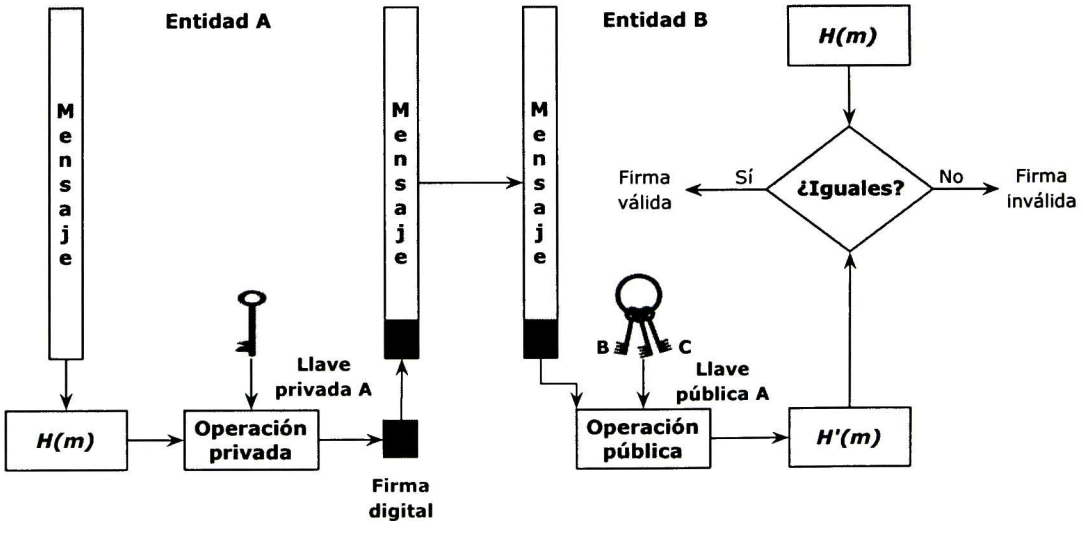



**Figura 2.6:** Esquema de firma digital

La importancia del uso de firmas digitales radica en otorgar diversos servicios de seguridad: el receptor puede asegurarse de la identidad de quien firmó el mensaje si verifica la firma digital utilizando la llave pública del emisor (autenticación). Asimismo, si un mensaje ha sido alterado es

posible detectarlo durante la verificación de la firma digital (integridad). Además, no es posible que el emisor niegue la propiedad de un mensaje firmado con su llave secreta, pues sólo él es quien posee dicha llave (no repudio).

### 2.2.5 Infraestructura de Llave Pública (PKI)

Una Infraestructura de Llave Pública (PKI, por sus siglas en inglés) consiste en un conjunto de hardware, software, entidades, técnicas criptográficas, políticas y procedimientos necesarios para proteger la seguridad de las comunicaciones y las transacciones sobre la red. PKI integra certificados digitales, autoridades certificadoras y criptografía de llave pública, lleva a cabo el registro de usuarios y la emisión, gestión, renovación y revocación de certificados digitales.

Una PKI se encuentra conformada por diversos componentes funcionales e interrelacionados [15] (véase Figura 2.7). La descripción de dichos componentes se presenta a continuación:

- Entidad final. Es el usuario final del sistema y corresponde al sujeto de un certificado digital. Realiza transacciones de solicitud y consulta de certificados digitales y Listas de Revocación de Certificados (CRLs, por sus siglas en inglés).

- Autoridad Certificadora (CA, por sus siglas en inglés). Es la entidad central de una PKI, entre sus responsabilidades se encuentran el emitir y publicar certificados digitales, mantener el estado de expiración o de revocación de los certificados y la emisión de CRLs.

- Autoridad de Registro (RA, por sus siglas en inglés). Es una entidad opcional a la cual una CA puede delegarle ciertas funciones de gestión, entre ellas el verificar los datos que el usuario proporciona durante la solicitud de obtención de un certificado digital.

- Emisor de CRL. Es un sistema que genera y firma CRLs, generalmente la CA es propiamente también el emisor de CRLs.

- Repositorio. Sistema o colección de sistemas distribuidos que almacenan certificados y CRLs, sirve como medio de distribución de certificados y CRLs para las entidades finales.

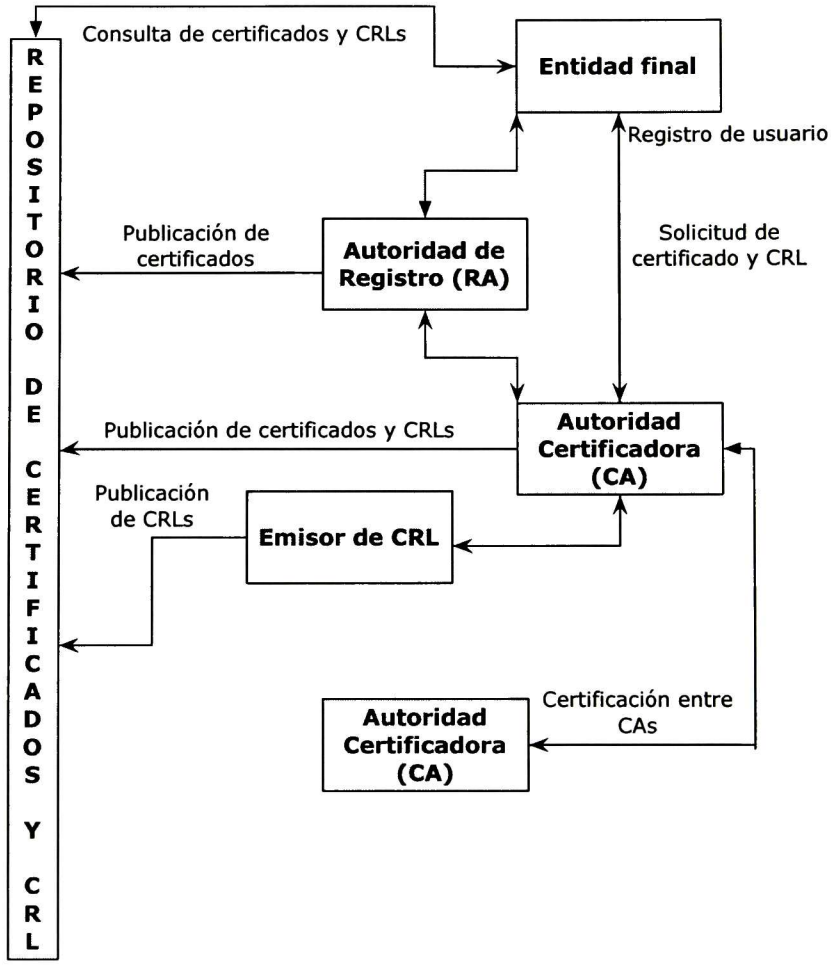



**Figura 2.7:** Relación entre componentes de una PKI

### *2.2.5.1. Certificado digital*

Un certificado digital es un archivo digital que contiene la llave pública e información de un sujeto y opera como mecanismo de autenticación, está firmado por una CA quien certifica la validez de la

información, se hace disponible a través de un repositorio público. En la Figura 2.8 se puede observar la estructura de un certificado digital bajo el estándar X.509 definido por la organización *Internet Engineering Task Force* (IETF) en el documento *Request for Comments* (RFC) 5280 [15].

| | | |
|---|---|---|
| Campos del certificado a ser firmados | **Versión** | Versión del certificado (v1(0), v2(1) o v3(2)). |
| | **Número de serie** | Entero positivo largo y no negativo, único por cada certificado que una CA emite. |
| | **Firma** | Identificador del algoritmo que es utilizado para firmar el certificado. |
| | **Emisor** | Entidad que firma y emite el certificado (nombre distinguido (DN) en formato X.500). |
| | **Validez** | Periodo de validez del certificado representado como secuencia de dos fechas: No antes y No después. |
| | **Sujeto** | Entidad asociada con la llave pública a ser contenida en el certificado (DN en formato X.500). |
| | **Información de la llave pública del sujeto** | Contiene la llave pública del sujeto y el identificador del algoritmo que es utilizado por la llave. |
| | Identificador único del emisor (*) | Identificador único del emisor del certificado. |
| | Identificador único del sujeto (*) | Identificador único del sujeto del certificado. |
| | Extensiones (**) | Secuencia de una o más extensiones del certificado. |
| | **Algoritmo de firma** | Identificador del algoritmo que es utilizado para firmar el certificado. |
| | **Valor de la firma** | Firma digital del certificado. |

**Figura 2.8:** Estructura de un certificado X.509 (* Aplica a versión v2 y v3, ** a versión v3)

### *2.2.5.2. Lista de Revocación de Certificados (CRL)*

Aunque es deseable que un certificado digital sea utilizado durante su periodo de validez que se defina al momento de expedirlo, existen circunstancias que causan que el certificado se convierta en inválido. Tal es el caso de un cambio de nombre, un cambio de la asociación entre el sujeto y la CA o que la llave privada se encuentre comprometida, por ello se hace necesario que la CA esté habilitada para revocarlo. El estándar X.509 [15] define un método para revocar certificados en donde periódicamente según sea la política (cada hora, diariamente, semanalmente, etc.) la CA emitará y firmará una estructura llamada Lista de Revocación de Certificados. Una CRL es una

lista que identifica certificados revocados y que está firmada por una CA o un emisor de CRLs, se hace disponible a través de un repositorio público. Cuando un sistema utiliza un certificado no sólo se verifica la firma y validez de éste, sino también que el número de serie no se encuentre como revocado. En la Figura 2.9 se puede observar la estructura de una CRL estándar X.509.

| Campos de la CRL a ser firmados | | | |
|---|---|---|---|
| Versión | | | Versión de la CRL. |
| **Firma** | | | Identificador del algoritmo que es utilizado para firmar la CRL. |
| **Emisor** | | | Entidad que firma y emite la CRL (nombre distinguido (DN) en formato X.500). |
| **Esta actualización** | | | Fecha de emisión de la CRL. |
| **Siguiente actualización** | | | Fecha de la próxima emisión de la CRL. |
| **Certificados revocados** | **Número de serie** | | Certificados revocados listados por su número de serie y fecha de revocación. Puede contener extensiones de entrada por cada certificado. |
| | **Fecha de revocación** | | |
| | Extensiones de entrada de la CRL (*) | | |
| Extensiones (*) | | | Secuencia de una o más extensiones de la CRL. |
| **Algoritmo de firma** | | | Identificador del algoritmo que es utilizado para firmar la lista de Certificados revocados. |
| **Valor de la firma** | | | Firma digital de la CRL. |

**Figura 2.9:** Estructura de una CRL X.509 (* Aplica a versión v2)

## 2.3 Resumen

En este capítulo, se han descrito las características esenciales de los modelos de servicio y de despliegue que componen al paradigma del cómputo en la nube. Asimismo, se mostró un panaroma general de la problemática de seguridad existente en cada uno de dichos modelos de servicio. Posteriormente, se abordaron los conceptos elementales del área de la criptografía, describiendo algunas primitivas criptográficas (funciones hash, firmas y certificados digitales, entre otras) que son fundamento para proveer servicios de seguridad en protocolos y aplicaciones. De aquí se realza la importancia de la criptografía para proveer seguridad en el cómputo en la nube.

# 3

# Arquitectura de seguridad y control de acceso para un servicio de almacenamiento en la nube

## 3.1 Modelo general de seguridad para el cómputo en la nube

Como se ha descrito hasta el momento, los problemas y aspectos de seguridad en el cómputo en la nube comprenden diferentes retos en cada uno de los modelos de servicio (véase Sección 2.1.4). En la Figura 3.1 se ilustra la propuesta realizada en [2] de un modelo general de seguridad para el cómputo en la nube que considera vulnerabilidades diversas dentro de los distintos modelos de servicios (SaaS, PaaS, IaaS) del cómputo en la nube. El modelo SaaS por ejemplo, se propone que esté integrado por módulos de seguridad de acceso multiusuario (autenticación, control de acceso, gestión de identidades) y de seguridad de aplicaciones en Internet (antivirus, antispam, etc.). En el caso del modelo PaaS se contempla la seguridad del framework, de los componentes y la interfaz. Por

otra lado, IaaS se basa en dos aspectos, uno de ellos se enfoca en el manejo seguro de la virtualización como mecanismo de prevención de posibles ataques hacia las máquinas virtuales, y el otro se enfoca en proteger la seguridad de los datos durante cada una de las fases de su ciclo de vida hasta llegar a la fase final de destrucción y/o en mantener la confidencialidad de los datos mediante algún tipo de cifrado dentro de un almacenamiento compartido. Se plantea también como requerimiento la necesidad de contar con una entidad externa que lleve a cabo la auditoría y verificación del cumplimiento de las políticas de seguridad que hayan sido establecidas dentro de los servicios del cómputo en la nube. La implementación del modelo general de seguridad completo puede ser complicada y tal vez no estrictamente necesaria, está en dependencia del servicio que se requiera.


**Figura 3.1:** Modelo general de seguridad para el cómputo en la nube

## 3.2 Modelo funcional del sistema

El modelo funcional habitual de un servicio multiusuario de almacenamiento en la nube considera en su estructura básica tres tipos de entidades: proveedor de servicios de nube, propietario de datos y usuarios. Con base en esto, en la Figura 3.2 se ilustra la interacción entre cada una de estas entidades

dentro de una arquitectura que contempla servicios de seguridad. De forma muy general, el rol que desempeña cada una de estas entidades es el siguiente:

- **Proveedor de Servicios de Nube (CSP).** Es la entidad que administra los recursos de cómputo de una infraestructura de nube con el fin de proveer distintos tipos de servicios. Su principal responsabilidad es almacenar y recuperar los datos acorde a peticiones autorizadas de usuarios.

- **Propietario de Datos (DO).** Es la entidad (individual/organizacional) poseedora de una colección de datos que desea almacenar en una infraestructura de nube de tal forma que las tareas de cómputo y mantenimiento de sus datos son delegadas a un CSP. Es la parte fundamental para la inicialización de los distintos servicios de seguridad. Asimismo, el DO es quien gestiona el acceso a la información almacenada en la nube a un conjunto de usuarios, los cuales pueden tener diferentes capacidades de acceso a la información.

- **Usuario (U).** Un usuario es una entidad que actúa como consumidor de los servicios del cómputo en la nube. Un usuario puede ser el propietario de datos.

Dentro del modelado funcional del sistema que se propone denotamos $C = \{D_1, ..., D_n\}$ como la colección de $n$ documentos de texto (no estructurado) que un propietario de datos posee y que desea almacenar en la infraestructura de nube. En una fase fuera de línea, el propietario de datos procede a cifrar cada uno de sus documentos con el fin de proteger la privacidad de sus datos sensitivos y confidenciales contra accesos no autorizados. Para habilitar la funcionalidad de efectuar búsquedas sobre los datos cifrados, a partir del conjunto $W = \{w_1, ..., w_p\}$ compuesto por $p$ distintas palabras extraídas de la colección de documentos $C$ y la aplicación de técnicas criptográficas el propietario debe construir un índice de búsqueda seguro $\widetilde{I}$. En relación a la problemática planteada en la Sección 1.2 el método de búsqueda deberá soportar búsqueda por relevancia. Posteriormente, el DO procede a transferir el índice de búsqueda seguro $\widetilde{I}$ y la colección de documentos cifrados $\widetilde{C}$

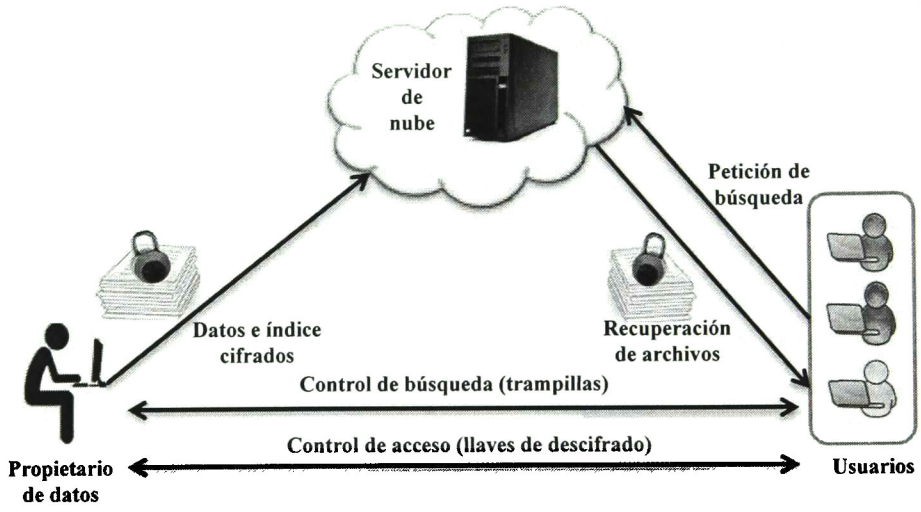



**Figura 3.2:** Modelo funcional en un servicio seguro de almacenamiento en la nube

hacia el almacenamiento en la nube.

En un entorno multiusuario el propietario de datos comparte sus datos con múltiples usuarios otorgándoles permiso para efectuar algún tipo de funcionalidad sobre los datos. Particularmente, para el modelo de sistema consideramos que el DO es la única entidad que tiene acceso total, es decir, puede escribir, modificar y eliminar datos. Los usuarios únicamente tendrán permiso de lectura y de efectuar consultas mediante múltiples palabras clave. Por consiguiente, es necesario contar con un mecanismo de control de búsqueda a través del cual el DO pueda distribuir a los usuarios autorizados los parámetros de seguridad $sk$ (por ejemplo, llaves secretas) necesarios para realizar consultas seguras. De manera particular, para efectuar búsquedas sobre la colección de documentos a partir de una consulta con múltiples palabras de interés $Q = \{w_1, ..., w_x\}$, un usuario autorizado deberá generar la trampilla de cada palabra $\widetilde{Q} = \{\widetilde{w}_1, ..., \widetilde{w}_x\}$ haciendo uso de los parámetros $sk$ que el DO haya distribuido. La trampilla representa la forma secreta de la palabra sin revelar ningún tipo de información de la palabra en claro, no podrá ser generada de ninguna otra forma más que a través de los parámetros de seguridad correspondientes. Una vez que el servidor de nube recibe la consulta

$\widetilde{Q}$ ejecuta la búsqueda sobre el índice $\widetilde{I}$ y retorna los resultados de relevancia de todos aquellos documentos que contienen las palabras especificadas. De esta manera, con base a los resultados de búsqueda un usuario puede obtener los top-$k$ documentos cifrados más relevantes.

A través de un mecanismo de control de acceso los usuarios obtienen las llaves de descifrado de los documentos. Dicho mecanismo debe ser flexible en el sentido que el DO tenga la facilidad de otorgar a un usuario en particular el acceso total o parcial de sus datos con base a políticas específicas que éste defina. Por consiguiente, el método de búsqueda debe filtrar los documentos que un usuario puede descifrar para no devolver resultados que incluyan documentos a los cuales no tenga permitido acceder.

Otra cuestión importante de seguridad corresponde a la verificación de la integridad de los datos almacenados en el servidor de nube, por lo cual deberá contarse con un método que permita llevar a cabo la verificación. Es preciso además que la comunicación entre entidades sea realizada a través de un canal de comunicación seguro y que las entidades sean debidamente autenticadas antes de procesar cualquier petición.

Dentro de la literatura [16, 17, 18, 19], el servidor de nube es considerado comúnmente como una entidad *honesta-pero-curiosa*. Se asume que actúa de forma "honesta" al seguir la especificación del protocolo diseñado, pero que es "curioso" al inferir y analizar el flujo de mensajes y los datos almacenados con el fin de aprender información adicional. No obstante, el servidor de nube no tiene intención alguna de modificar o eliminar datos ni de interrumpir los servicios. Este mismo supuesto es retomado para este trabajo de tesis.

## 3.3 Diseño arquitectural

La arquitectura general de servicios de seguridad que se propone para un servicio de almacenamiento en la nube es ejemplificada en la Figura 3.3. Para el desarrollo de esta arquitectura, una organización fundamentalmente debe contar con los modelos de servicios PaaS e IaaS definidos para el cómputo en la nube. Como se puede observar, proveer los servicios de seguridad que se han planteado como objetivo requiere de varias entidades. Entre dichas entidades se encuentran: el Proveedor de Servicios de Nube (CSP), el Propietario de Datos (DO) y el Usuario (U), éstas son retomadas del modelo funcional descrito en la sección anterior y pueden ser categorizadas como las entidades básicas de la arquitectura.

Por otro lado, las entidades que se añaden a la arquitectura son: un *Third Party Auditor* (TPA) y una Infraestructura de Llave Pública (PKI). Para llevar a cabo una comunicación segura entre las entidades de la arquitectura se hace uso del protocolo seguro de Internet HTTPS, por ello es indispensable contar con una PKI a través de la cual pueda obtenerse un certificado digital que es el bloque básico del protocolo HTTPS. Finalmente, para la verificación de la integridad de sus datos el DO delega la tarea a un TPA debido al hecho de que éste cuenta con mayores capacidades y recursos de cómputo que el DO. Un TPA funge como entidad de confianza para llevar a cabo la auditoría pública de los datos de los DO's.

Las funcionalidades específicas de cada entidad son descritas en las siguientes subsecciones.

### 3.3.1 Propietario de Datos (DO)

El propietario de datos es la parte fundamental para la inicialización de los distintos servicios de seguridad. Sus funcionalidades son realizadas a través de cinco módulos funcionales básicos (véase Figura 3.4):

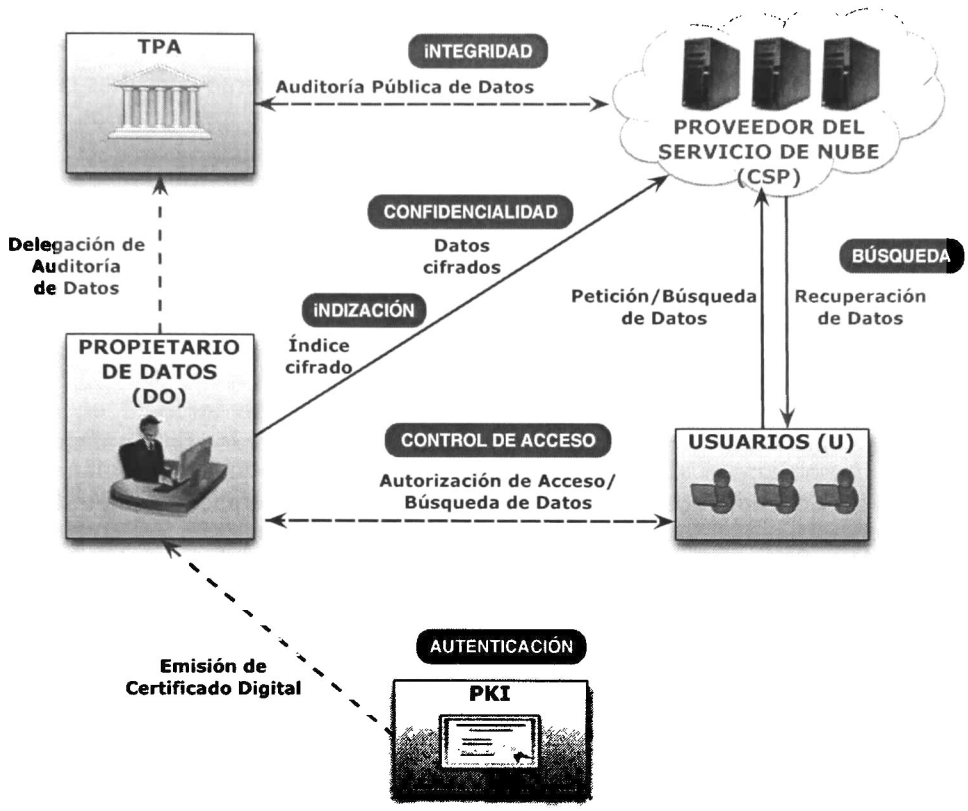



Figura 3.3: Arquitectura de servicios de seguridad para un almacenamiento en la nube

- **Módulo Generador de Llaves.** Es el módulo principal para inicializar los servicios de seguridad del sistema. Los parámetros que recibe como entrada definirán el nivel de seguridad de los algoritmos criptográficos que se utilizan en los servicios. A su salida se obtienen las llaves criptográficas correspondientes a cada uno de dichos algoritmos, éstas pueden ser resguardadas en el almacén de llaves.

- **Módulo de Indización.** Este módulo es utilizado para construir el índice de búsqueda a partir de la extracción de las palabras contenidas en la colección de documentos de texto.

- **Módulo de Control de Acceso.** En este módulo el DO autoriza el acceso sobre sus datos a los usuarios y define los atributos de éstos, así como políticas de acceso específicas para cada

documento. El módulo integra también un mecanismo para compartir con usuarios autorizados las llaves criptográficas que descifran los documentos y que generan consultas seguras.

- **Módulo de Firmas.** En este módulo se generan ciertos metadatos únicos por cada documento los cuales serán utilizados durante la verificación de integridad de los datos almacenados en el servicio de nube.

- **Módulo de Seguridad.** Este módulo contiene todas las funciones criptográficas del sistema. Por ejemplo, los documentos son cifrados bajo las políticas definidas dentro del control de acceso. Por otra parte, el índice de búsqueda pasa por este módulo para con ello obtener un índice de búsqueda seguro.

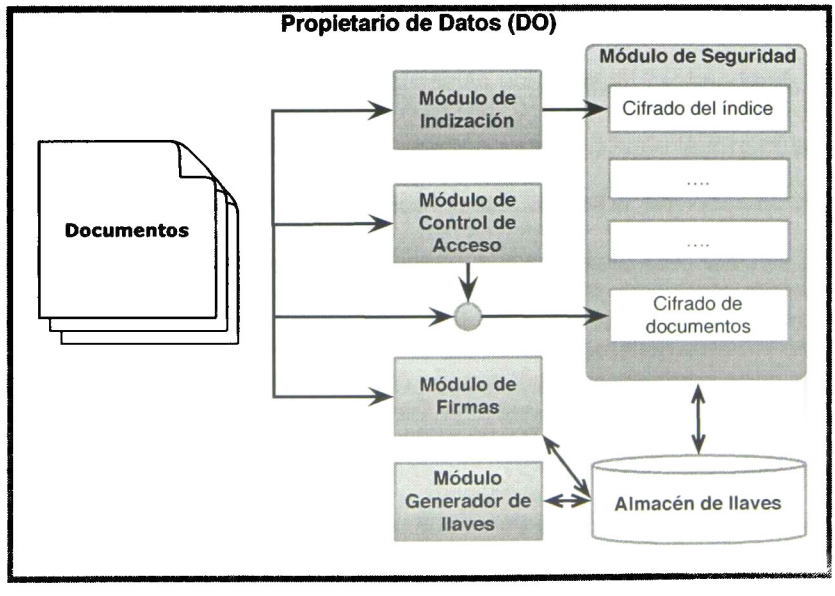



**Figura 3.4:** Detalle arquitectural-Propietario de Datos (DO)

### 3.3.2 Proveedor de Servicios de Nube (CSP)

El CSP es la entidad que provee los servicios del cómputo en la nube y se constituye esencialmente por los siguientes módulos funcionales (véase Figura 3.5):

- **Módulo de Autenticación.** Este módulo gestiona las identidades de los usuarios y los autentica previo al acceso a los servicios del almacenamiento en la nube.
- **Módulo de Control de Acceso.** Este módulo ejecuta el cumplimiento de las políticas de acceso durante las solicitudes de acceso a los documentos. Se integra por **BD-ACP**, que es la base de datos que resguarda las políticas de control de acceso definidas para los documentos, y por **BD-USER** que es la base de datos que gestiona los usuarios autorizados por el DO.
- **Módulo de Búsqueda.** En este módulo se atienden las peticiones de búsqueda de datos emitidas por los usuarios cumpliendo con el control de acceso que fue especificado por el DO.
- **Módulo de Integridad.** A través de este módulo el CSP da respuesta a las peticiones emitidas por el auditor para la verificación de integridad de los datos almacenados en la nube haciendo uso de los metadatos de verificación generados por el DO.

### 3.3.3 Infraestructura de Llave Pública (PKI)

A través del despliegue de una PKI descrita en la Sección 2.2.5, el resto de las entidades de la arquitectura podrán solicitar la obtención, renovación y revocación de un certificado digital. El certificado digital será utilizado como uno de los mecanismos de autenticación en el servicio de almacenamiento en la nube.

Dentro de la PKI, una Autoridad de Registro (RA) recibirá los datos de identidad en conjunto con la llave pública de una entidad. Una vez que los datos hayan sido validados por la RA, la Autoridad

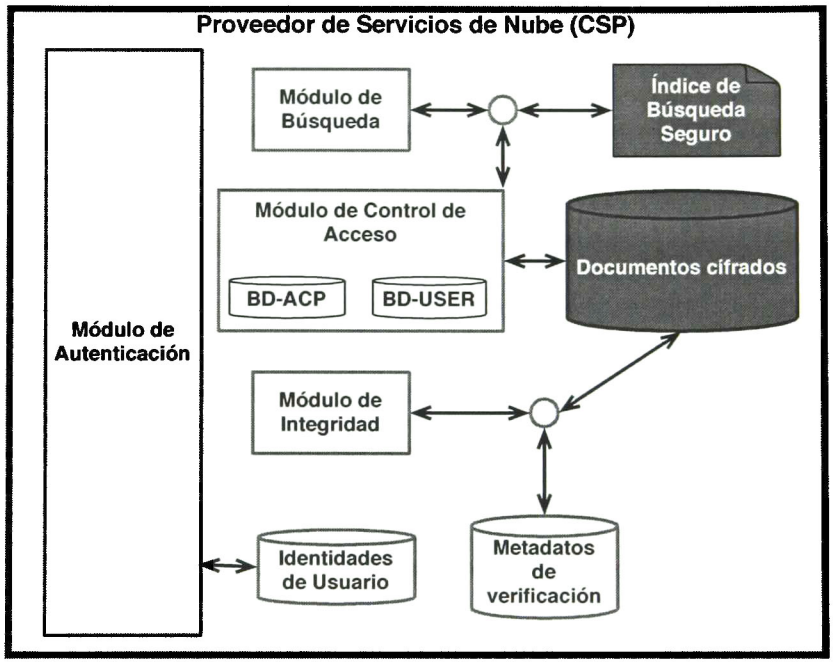



**Figura 3.5:** Detalle arquitectural-Proveedor de Servicios de Nube (CSP)

Certificadora (CA) procederá a crear el certificado digital de la entidad, mismo que hará llegar a la entidad y almacenará en un Repositorio. El Repositorio que será de ámbito público servirá como módulo de consulta para verificar la existencia de un certificado en particular, validar su vigencia y que no se encuentre como revocado.

### 3.3.4 Usuario (U)

En la arquitectura propuesta, de igual forma que los DO's los usuarios deberán cumplir con el registro de sus datos, la obtención de su certificado digital a través de la PKI y autenticarse ante el CSP cuando deseen acceder al servicio. Los módulos que conforman a un usuario son los siguientes (véase Figura 3.6):

- **Módulo Generador de Llaves.** A través de este módulo un usuario procederá a generar su

par de llaves pública y privada necesarias para la obtención de su certificado digital.

- **Módulo Generador de Consultas.** En este módulo, un usuario genera las consultas.
- **Módulo de Seguridad.** Este módulo contiene todas las funciones criptográficas del sistema. Aquí, un usuario autorizado podrá transformar una consulta en una consulta segura haciendo uso de la llave criptográfica que el DO le compartió. Asimismo, podrá descifrar únicamente los documentos sobre los cuales el DO le otorgó acceso.

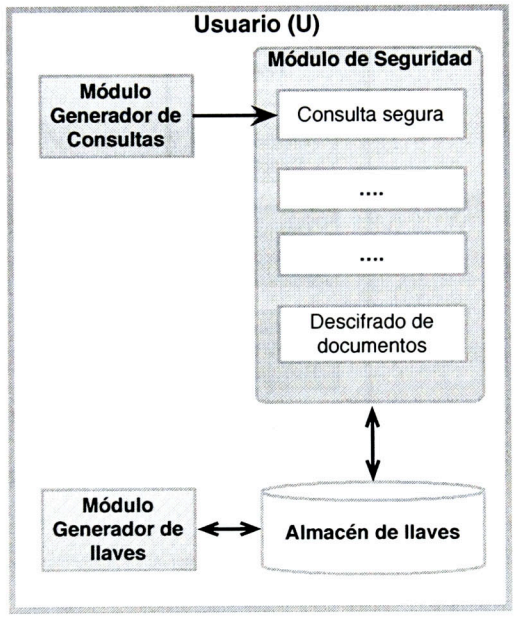



**Figura 3.6:** Detalle arquitectural-Usuario (U)

### 3.3.5 Third Party Auditor (TPA)

La auditoría de los datos de los DO's es llevada a cabo por un TPA que es una entidad de confianza. El TPA deberá permanecer siempre en línea para, a petición del DO, ejecutar la auditoría de los datos. Esencialmente, los módulos funcionales que lo integran son (véase Figura 3.7):

- **Módulo de Emisión de Desafíos.** Mediante este módulo el TPA generará los desafíos de auditoría de datos que serán emitidos ante el CSP.
- **Módulo de Verificación.** En este módulo el TPA verificará la prueba otorgada por el CSP para comprobar la integridad de los datos.
- **Módulo de Publicación de Resultados.** A través de este módulo el DO podrá comunicarse con el TPA para la obtención de un reporte de resultados de sus datos auditados en el servicio de nube.


**Figura 3.7:** Detalle arquitectural-Third Party Auditor (TPA)

## 3.4 Resumen

En este capítulo, se presentó un modelo general de seguridad para el cómputo en la nube propuesto dentro de la literatura, el cual contempla los distintos problemas de seguridad presentes en los modelos de servicios SaaS, IaaS y PaaS. Posteriormente, se describieron las funcionalidades de seguridad particulares definidas para nuestra propuesta. Finalmente, se presentó la arquitectura general de los servicios de seguridad que son objetivo en esta tesis. Por cada entidad de la arquitectura se ilustraron los módulos funcionales que la integran, el detalle de éstos se presenta en el Capítulo 4 y el Capítulo

5. En el primero de ellos, se abarcan los servicios de seguridad a nivel de almacenamiento y acceso a datos (autenticación, control de acceso y confidencialidad), y en el segundo, los servicios de seguridad a nivel de funcionalidades sobre los datos (indización, búsqueda e integridad).

# 4

# Servicios de seguridad para almacenamiento y acceso a datos en la nube

## 4.1 Registro inicial

El proceso inicial del servicio de almacenamiento en la nube consiste en el registro de datos por parte del usuario, la generación del conjunto de llaves criptográficas del esquema AE y la obtención de su certificado digital. Las entidades involucradas en este proceso son: Proveedor del Servicio de Nube (CSP), Usuario/Propietario de Datos (DO/U), y algunas de las entidades PKI (véase Sección 2.2.5): Autoridad de Registro (RA), Autoridad Certificadora (CA) y Repositorio.

El proceso de registro de usuarios que se diseñó se ilustra en la Figura 4.1, y consta de los siguientes pasos:

1. El DO/U procederá ante el CSP a registrar sus datos: identificador único de usuario (UID),

contraseña (pwd), entre otros.

2. Una vez registrado el DO/U estará habilitado para descargar la aplicación del sistema de gestión de llaves.

3. Mediante dicha aplicación y el algoritmo $KeyGen_{AE}$ el usuario generará de forma local su par de llaves $(privK_{UID}, pubK_{UID})$ del esquema AE. Dichas llaves podrán ser resguardadas en un almacén de llaves protegido por contraseña.

4. El usuario solicitará ante la RA la obtención de un certificado digital para ello deberá enviar todos los datos necesarios en conjunto con su llave pública $pubK_{UID}$.

5. La RA pre-registrará los datos recibidos del usuario.

6. Como mecanismo de autenticación la RA enviará al usuario vía correo electrónico un código de confirmación $code$ de forma cifrada utilizando la llave pública del usuario.

$$C = Enc_{AE}(pubK_{UID}, code)$$

7. El usuario descifrará el código utilizando su llave privada

$$code = Dec_{AE}(privK_{UID}, C)$$

De esta manera si el usuario es capaz de descifrarlo se autentica que en efecto él es el poseedor de la llave pública $pubK_{UID}$ que fue enviada durante la parte inicial del proceso de registro, dicho código de confirmación descifrado deberá ser retornado a la RA.

8. Una vez que el usuario ha ingresado el código de confirmación la RA aprobará el pre-registro.

9. La CA emitirá y firmará los certificados digitales correspondientes a las solicitudes aprobadas.

10. La CA almacenará en el Repositorio público los certificados digitales que han sido generados, asimismo enviará al usuario vía correo electrónico copia de su certificado digital.

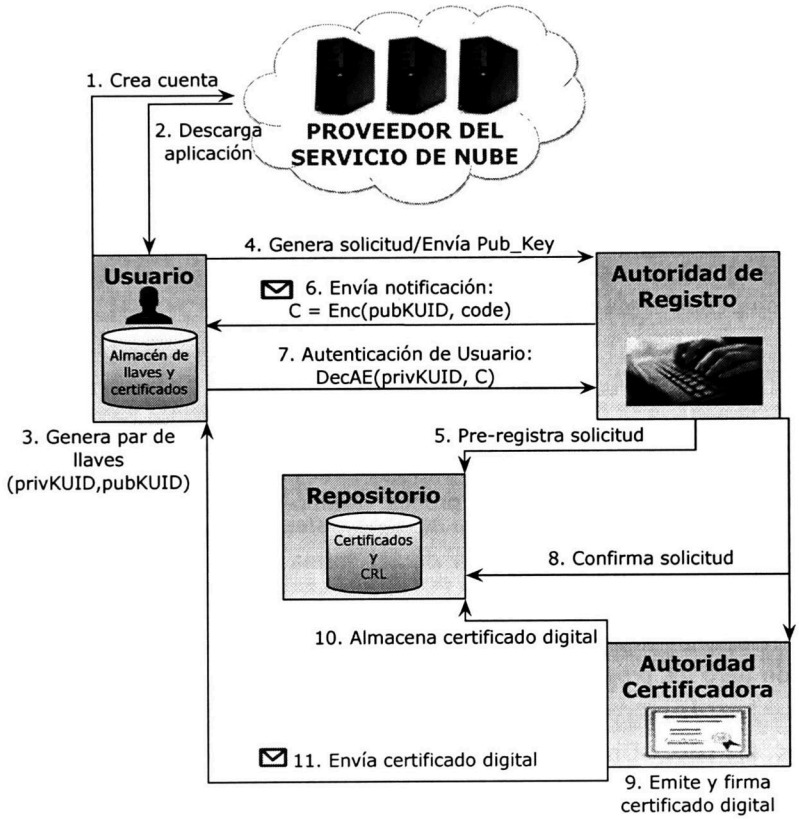



**Figura 4.1:** Esquema del proceso de registro

## 4.2 Autenticación

Uno de los requerimientos básicos de seguridad para el modelo de servicios SaaS del cómputo en la nube consiste en contar con un servicio de autenticación de identidades, en donde la identidad de

un usuario debe ser validada previo a que éste pueda utilizar alguno de los servicios de nube. Con el propósito de llevar a cabo una autenticación de identidades fuerte y efectiva es deseable desarrollar un esquema que combine dos o más tecnologías de seguridad para autenticación.

Para la propuesta de solución se planteó un esquema que tiene como bloque básico las siguientes tecnologías de seguridad:

- Identificador/contraseña. Es el método de autenticación más representativo y de uso simple. Se requiere que el valor de la contraseña tenga buen nivel de seguridad (combinación de mayúsculas, minúsculas, números, caracteres especiales, etc.) y que se cuente con un método de renovación periódica de contraseñas.

- CAPTCHA (*Completely Automated Public Turing Test to Tell Computers and Humans Apart*). Es ampliamente utilizado en sitios web para proteger los recursos de ataques iniciados por scripts automáticos. Consiste en una prueba desafío-respuesta para distinguir humanos de computadoras. Los desafíos se diseñan de tal forma en la que un usuario humano puede responder con facilidad a dichos desafíos. Por ejemplo, los humanos pueden leer texto distorsionado pero los programas de computadora no.

- Certificados digitales. Un certificado digital es un archivo que actúa como credencial de identidad de un sujeto y contiene su llave pública del esquema AE. Está firmado electrónicamente por una entidad de confianza que certifica la validez de la información que fue proporcionada por el sujeto.

El esquema de autenticación que se diseñó se ejemplifica en la Figura 4.2; en el primer paso del proceso se habilita la comunicación https para lo cual se requiere el certificado digital del usuario, posteriormente el servidor web de la nube enviará un desafío aleatorio CAPTCHA al cual el usuario deberá respondar incluyendo sus datos de identificador y contraseña, el servicio de autenticación validará la respuesta al desafío y que los datos sean válidos, si esto se cumple se autoriza el acceso.


**Figura 4.2:** Esquema del proceso de autenticación

## 4.3 Control de acceso y confidencialidad

La seguridad de datos y control de acceso es uno de los desafíos existentes en el cómputo en la nube. Control de acceso es un mecanismo de seguridad para la protección de recursos compartidos contra accesos no autorizados, fundamentalmente se constituye por: *objetos*, *sujetos*, y *permisos*. Los objetos son los recursos a proteger, los sujetos son las entidades que acceden a los objetos y los permisos son las acciones autorizadas que un sujeto puede realizar sobre un objeto.

Primitivas criptográficas como el cifrado de datos usualmente son aplicadas por el propietario de datos para garantizar la confidencialidad de sus datos. Por consiguiente, las llaves de descifrado deberán ser compartidas con los sujetos autorizados. La problemática de esta solución radica en encontrar un balance entre el cumplimiento de ambos objetivos de seguridad (confidencialidad/control de acceso) sin introducir gran sobrecarga de cómputo del lado del propietario de datos durante la gestión y cifrado de los datos y la distribución de llaves. Además, un requerimiento importante es disponer de un control de acceso de grano fino mediante el cual se facilite la asignación de permisos específicos por usuario.

En las siguientes subsecciones, se describen los modelos y mecanismos de control de acceso principales. Así también se introduce el concepto cifrado basado en atributos (ABE por sus siglas en inglés). Finalmente se describe el esquema de control de acceso y confidencialidad propuesto.

### 4.3.1 Modelos de control de acceso

En un mecanismo de control de acceso la distinción entre accesos autorizados y no autorizados se determina con base a *políticas de control de acceso*. Una política de control de acceso describe un acceso permitido o denegado dentro del sistema, y se configura a través de un *modelo de control de acceso*. Un modelo de control de acceso define cómo los objetos, sujetos y accesos pueden ser representados.

Existen tres modelos principales de control de acceso:

- Control de Acceso Obligatorio (MAC por sus siglas en inglés).
- Control de Acceso Discrecional (DAC por sus siglas en inglés).
- Control de Acceso Basado en Roles (RBAC por sus siglas en inglés).

En el modelo MAC las políticas de control de acceso son determinadas exclusivamente por el sistema. Es utilizado por instituciones militares y del gobierno en donde se procesa información altamente sensible. Por otra parte, en el modelo DAC el propietario del objeto es quien determina las políticas de control acceso. Finalmente, en el modelo no discrecional RBAC (véase Figura 4.3) la gestión de las políticas de control de acceso es simplificada en gran medida ya que únicamente se requiere la asignación de usuarios a roles, y permisos[1] a roles. Este modelo es muy adecuado para modelos organizacionales de instituciones y empresas. De los tres modelos, el modelo DAC es el menos restrictivo, ya que el propietario tiene total libertad para determinar quién tendrá permiso para acceder al objeto y qué privilegios tendrá, es decir, es discrecional.

[1]Un permiso es una relación objeto-operación.

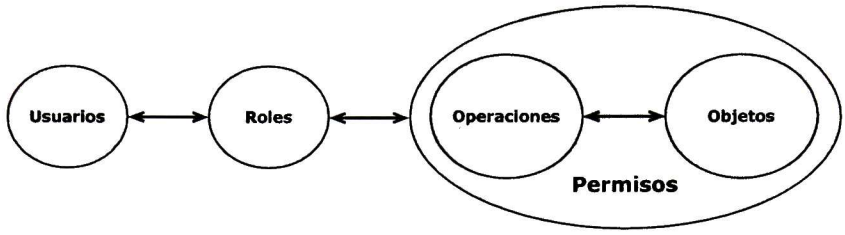



**Figura 4.3:** RBAC

### 4.3.2 Mecanismos de control de acceso

El modelo conceptual clásico para representar los permisos de los sujetos sobre los objetos es la *matriz de control de acceso* (ACM por sus siglas en ingés). Bajo este modelo los sujetos se representan en renglones, los objetos en columnas y los permisos a través de las entradas de la matriz. Este modelo es simple y ampliamente utilizado en la práctica, pero raramente se implementa como un todo, en su lugar, se descompone en renglones o columnas. Acorde a esto, se presenta la existencia de dos mecanismos de control de acceso:

- Lista de control de acceso (ACL por sus siglas en inglés).
- Lista de capacidades.

Una ACL es la descomposición en columnas de una ACM, esencialmente consiste en una lista por objeto en la cual se indican los permisos que posee cada sujeto sobre dicho objeto. Para asegurar control de acceso de grano fino y gestionar llaves de cifrado en el cómputo en la nube una ACL podría ser utilizada. Sin embargo, la representación de las políticas de acceso se dificulta cuando el número de usuarios en el sistema es muy grande.

Por otro lado, una lista de capacidades es la descomposición en renglones de una ACM, es decir, existe una lista por sujeto en la cual se indican los permisos que posee dicho sujeto sobre cada objeto.

Sanka *et al* [20] proponen un control de acceso criptográfico basado en capacidades para el cómputo en la nube en lugar de ACLs por ser más apropiado para un escenario de usuarios individuales. Dicho de otra forma, creando una ACL por objeto podría llegar a ser impráctico consultar los documentos que son accesibles para un usuario. Aquí, la lista de capacidades contiene el identificador del usuario (UID), el identificador del archivo (FID) y los correspondientes permisos de acceso (AR). Los documentos son cifrados bajo un esquema de cifrado simétrico, y para diversos propósitos de seguridad (intercambio de llaves, transferencias) y debido al gran tamaño de las llaves asimétricas eligen aplicar un doble cifrado.

### 4.3.3 Cifrado Basado en Atributos (ABE)

Uno de los modelos no tradicionales para resolver el problema de control de acceso con confidencialidad es el Control de Acceso Basado en Atributos (ABAC por sus siglas en inglés) el cual tiene como fundamento el esquema ABE. Recientemente ABE ha cobrado especial interés ya que ofrece un mecanismo de control de acceso y confidencialidad en donde las políticas de acceso se encuentran directamente incrustadas en los textos cifrados. Lo anterior hace posible que la confidencialidad de los textos cifrados sea preservada aún cuando el servidor de almacenamiento no sea de confianza.

Sahai y Waters [21] introdujeron las primeras nociones de ABE como nuevo mecanismo de control de acceso utilizando un esquema difuso de cifrado basado en identidad, donde una entidad se considera como un conjunto de atributos descriptivos. El realce de ABE radica en que los textos cifrados no necesariamente son cifrados para un usuario en particular tal como en la tradicional criptografía de llave pública. En su lugar, los textos cifrados y llaves privadas se asocian con un conjunto de atributos o con una política sobre los atributos. Esto habilita un control de acceso de grano fino, lo cual es muy deseable para el cómputo en la nube siendo esto uno de los objetivos planteados en este trabajo de tesis. El esquema ABE se define bajo dos enfoques: *Key Policy Attribute-*

*Based Encryption* (KP-ABE) [22] y *Ciphertext-Policy Attribute-Based Encryption* (CP-ABE) [23].

### *Políticas de acceso*

Las políticas de acceso en ABE son descritas a través de fórmulas booleanas, por ejemplo: $((a_1 \vee a_2) \wedge (a_3 \wedge a_4)) \vee (a_5 \wedge a_6 \wedge a_7)$, donde, $\vee$ representa una operación lógica OR, $\wedge$ una operación lógica AND y $a_1, a_2, .., a_7$ son atributos. También pueden ser vistas como un árbol de acceso, en cuya estructura la raíz y nodos internos son compuertas lógicas AND ($\wedge$) u OR ($\vee$), y las hojas son atributos. En [22, 23], las fórmulas booleanas son representadas a través de una estructura monótona de acceso (véase Definición 4). En este contexto, el rol de los objetos es tomado por los atributos, por tanto, la estructura de acceso $\mathbb{A}$ contiene el conjunto autorizado de atributos.

**Definición 3.** ***Estructura de acceso monótona*:**

*Sea* $\{P_1, P_2, ..., P_n\}$ *un conjunto de objetos. Una colección* $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}}$ *es monótona si para* $\forall B, C$*: si* $B \in \mathbb{A}$ *y* $B \subseteq C$ *entonces* $C \in \mathbb{A}$*. Una estructura de acceso monótona es una colección monótona* $\mathbb{A}$ *de subconjuntos no vacíos* $\{P_1, P_2, ..., P_n\}$*, por e.j.,* $\mathbb{A} \subseteq 2^{\{P_1, P_2, ..., P_n\}} \setminus \{\emptyset\}$*. Los conjuntos en* $\mathbb{A}$ *son denominados conjuntos autorizados, y los conjuntos que no están en* $\mathbb{A}$ *son llamados conjuntos no autorizados.*

### *Atributos*

Los atributos en ABE pueden ser de dos tipos: simples o compuestos. Los atributos simples son representados sencillamente por una etiqueta (cadena), mientras que los atributos compuestos tienen un valor asignado. Estos últimos pueden clasificarse en: 1) booleanos, adoptan uno de dos valores (si/no), 2) enumerados, pueden adoptar múltiples valores no numéricos y 3) numéricos, pueden adoptar múltiples valores numéricos. En la Tabla 4.1 se ilustran las reglas gramáticales para expresar atributos en ABE.

A manera de ejemplo, consideremos que una organización como por ejemplo una institución

| | | |
|---|---|---|
| ATRIBUTO | → | SIMPLE \| COMPUESTO |
| SIMPLE | → | ETIQUETA |
| ETIQUETA | → | cadena |
| COMPUESTO | → | ETIQUETA OPERADOR VALOR |
| OPERADOR | → | <\|>\|=\|<>\|≤\|≥ |
| VALOR | → | cadena \| número |

**Tabla 4.1:** Reglas gramáticales para atributos en ABE

educativa define el siguiente universo de atributos $\mathbb{U}$: {Departamento, Estudiante, Doctor, Cuatrimestre} y una política de acceso para un documento como la que se ilustra en la Figura 4.4 y se describe a continuación:

$$(\text{Departamento=LTI} \vee \text{Departamento=Computación}) \wedge ((\text{Estudiante} \wedge (\text{Cuatrimestre=1} \vee \text{Cuatrimestre=4})) \vee \text{Doctor})$$

De acuerdo a la política anterior, el documento sólo podrá ser visto por cada usuario(s) que pertenezca al Departamento LTI o Computación, y que además sea un Estudiante del Cuatrimestre 1 o 4 o bien un Doctor.

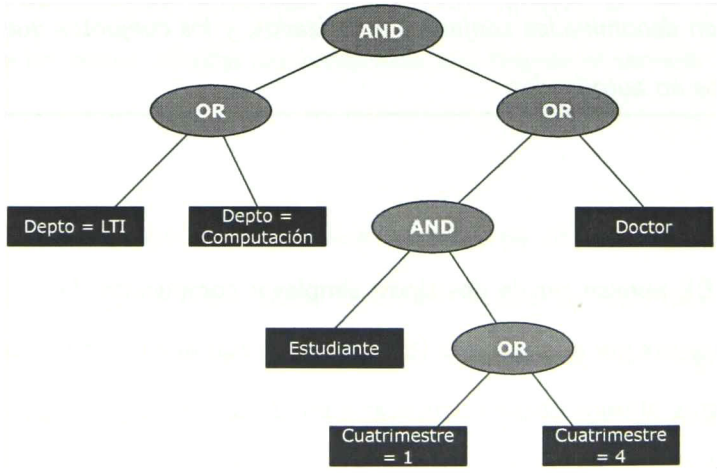



**Figura 4.4:** Ejemplo de política de acceso basada en atributos

La diferencia esencial entre KP-ABE y CP-ABE es la forma en la que se especifican las políticas

de acceso y los atributos. Es decir, en KP-ABE tal como se ejemplifica en la Figura 4.5 los textos cifrados son etiquetados con un conjunto de atributos, y las llaves privadas son asociadas a una estructura de acceso que determinará qué textos cifrados un usuario es capaz de descifrar. En CP-ABE, la relación se invierte (véase Figura 4.6) y una llave privada es asociada a un conjunto de atributos expresados como cadena. Cuando se cifra un mensaje se especifica y asocia a éste una estructura de acceso sobre los atributos. Un usuario sólo estará habilitado para descifrar un texto cifrado si sus atributos pasan a través de la estructura de acceso del texto cifrado.

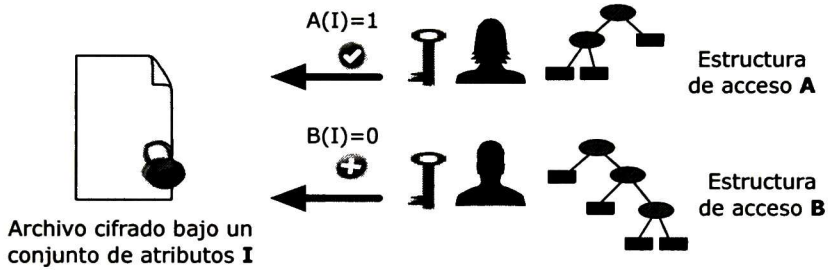



**Figura 4.5:** KP-ABE

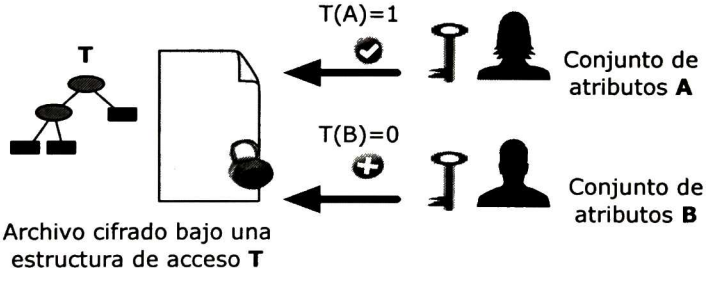



**Figura 4.6:** CP-ABE

### *CP-ABE*

El esquema CP-ABE se constituye por cuatro algoritmos fundamentales [23]:

1. **Setup**. Este algoritmo toma como entrada un parámetro de seguridad y genera los parámetros públicos $PK$ y una llave maestra $MK$.

2. **Encrypt($PK, M, \mathbb{A}$)**. El algoritmo de cifrado toma como entrada los parámetros públicos $PK$, un mensaje $M$, y una estructura de acceso $\mathbb{A}$ sobre el universo de atributos. El algoritmo cifra $M$ y produce el texto cifrado $CT$ de tal manera que sólo el usuario que posea el conjunto de atributos que satisfagan la estructura de acceso estará habilitado para descifrar el mensaje. El texto cifrado implícitamente contiene $\mathbb{A}$.

3. **KeyGeneration($MK, S$)**. El algoritmo de generación de llave toma como entrada la llave maestra $MK$ y el conjunto de atributos $S$ que describen la llave. Produce como salida una llave privada $SK$.

4. **Decrypt($PK, CT, SK$)**. El algoritmo de descifrado toma como entrada los parámetros públicos $PK$, un texto cifrado $CT$ que contiene una política de acceso $\mathbb{A}$, y una llave privada $SK$, que es la llave privada para un conjunto de atributos $S$. Si el conjunto de atributos $S$ satisfacen la estructura de acceso $\mathbb{A}$ el algoritmo descifra $CT$ y retorna un mensaje $M$.

### 4.3.4 Esquema de control de acceso y confidencialidad propuesto

En esta sección se describe el esquema de control de acceso y confidencialidad que se propone, para el cual retomamos el concepto de ABE como base principal de la propuesta. Particularmente se utiliza el esquema CP-ABE [23].

El esquema propuesto aplica un tipo de cifrado híbrido, más específicamente, se utiliza un esquema de cifrado simétrico [2] (SE por sus siglas en inglés) para los documentos, y el esquema CP-ABE para las llaves simétricas con las que se cifraron los documentos. Se determinó esta decisión ya que es más conveniente cifrar con el esquema de llave pública CP-ABE únicamente una llave simétrica de

[2]Los esquemas de cifrado simétrico son utilizados para cifrar grandes volúmenes de datos debido a su eficiencia.

128 bits, por ejemplo, que todo el contenido del documento que posee un mayor volúmen de bits. Se utiliza además un esquema de llave pública/cifrado asimétrico (AE por sus siglas en inglés) como mecanismo para compartición de llaves. En la Tabla 4.3 se describe el flujo de mensajes entre las entidades partícipes en cada una de las funcionalidades definidas dentro del esquema control de acceso propuesto, éstas se describen a mayor detalle en las siguientes subsecciones.

| **Almacenamiento de documentos** | | |
|---|---|---|
| **DO** | | **CSP** |
| Dado $C = \{D_1, ..., D_n\}$, para cada $D \in C$: | | |
| 1. $D \leftarrow DID, D \leftarrow T_{DID}$ | | |
| 2. $K_{DID} = KeyGen_{SE}(\lambda)$ | | |
| 3. $\widetilde{D} = Enc_{SE}(K_{DID}, D)$ | | |
| 4. $\widetilde{K}_{DID} = Enc_{CP\text{-}ABE}(PK, K_{DID}, T_{DID})$ | | |
| 5. $\sigma_{ACP} = Sign(privK_{DO}, ACP)$ donde $ACP = \{T_1, ..., T_n\}$ | | |
| 6. $\widetilde{C} = \{[\widetilde{K}_1, \widetilde{D}_1], ..., [\widetilde{K}_n, \widetilde{D}_n]\}, ACP$ y $\sigma_{ACP}$ | $\Longrightarrow$ | 7. Si $(Verify(pubK_{DO}, \sigma_{ACP}, ACP) \rightarrow \{verdadero, falso\}) == verdadero$, alta de $ACP$ en BD-ACP |
| **Autorización de acceso** | | |
| **DO** | | **CSP** |
| 1. $U \leftarrow UID$ | | |
| 2. $A_{UID} = \{Attr_1 \wedge Attr_2 \wedge Attr_3, ..\}$ | | |
| 3. $SK_{UID} = KeyGen_{CP\text{-}ABE}(MK, A_{UID})$ | | |
| 4. $\sigma_A = Sign(privK_{DO}, UID \parallel A)$ | | |
| 5. $\{UID, A_{UID}, \sigma_A\}$ | $\Longrightarrow$ | 6. Si $(Verify(pubK_{DO}, \sigma_A, UID \parallel A_{UID}) \rightarrow \{verdadero, falso\}) == verdadero$, alta de $UID$-$A_{UID}$ en BD-USER |
| **Compartición de llaves** | | |
| **DO** | | **U** |
| Dado $K = \{PK, A_{UID}, SK_{UID}, SearchKeys\}$ y $UID$ | | |
| 1. $\sigma_{KU} = Sign(privK_{DO}, K)$ | | |
| 2. $\widetilde{K} = Enc(pubK_{UID}, K)$ | | |
| 3. $\{\widetilde{K}, \sigma_K\}$ | $\Longrightarrow$ | 4. $K = Dec(privK_{UID}, \widetilde{K})$<br>5. Si $(Verify(pubK_{DO}, \sigma_{KU}, KU) \rightarrow \{verdadero, falso\}) == falso$, acepta $K$ |
| **Acceso a documento** | | |
| **U** | | **CSP** |
| 1. Solicita documento con el identificador DID | | 2. Verifica $UID \in$ BD-USER<br>3. Si $T_{DID}(A_{UID}) = 1$ |
| 4. $K_{DID} = Dec_{CP\text{-}ABE}(\widetilde{K}_{DID}, SK)$ | $\Longleftarrow$ | $[\widetilde{K}_{DID}, \widetilde{D}_{DID}]$ si paso 2 y paso 3 otorga verdadero |
| 5. $D = Dec_{SE}(K_{DID}, \widetilde{D})$ | | |
| **Revocación de acceso** | | |
| **DO** | | **CSP** |
| 1. $R = \{UID_1, ..., UID_u\}$ | | |
| 2. $\sigma_R = Sign(privK_{DO}, R)$ | | |
| 3. $\{R, \sigma_R\}$ | $\Longrightarrow$ | 4. Si $(Verify(pubK_{DO}, \sigma_R, R) \rightarrow \{verdadero, falso\}) == falso$, por cada $UID \in R$ dar de baja en BD-USER |

**Tabla 4.3:** Flujo de mensajes del esquema de control de acceso y confidencialidad

#### 4.3.4.1. *Inicialización*

En el **Módulo Generador de Llaves** y a través del algoritmo $Setup_{CP\text{-}ABE}$ el DO inicializa el servicio de control de acceso generando una llave maestra $MK$ de ámbito privado, y los parámetros públicos $PK$ que serán compartidos con aquellos usuarios a quienes el DO otorgue acceso. Asimismo, dentro del **Módulo de Control de Acceso** se establece el universo de atributos de tipo simple $\mathbb{U} = \{A_1, A_2, A_3, ..., A_x\}$ del sistema. Las políticas de acceso $T$ de los documentos serán descritas a través de fórmulas booleanas tal como se detalló en la sección anterior, mientras que los atributos que las compondrán serán tomados de $\mathbb{U}$ con posibilidad a ser extendidos a un tipo de atributo compuesto con base a las reglas gramaticales de la Tabla 4.1. Además, el conjunto de atributos de los usuarios denotado por $A_{UID} = \{Attr_1 \wedge Attr_2 \wedge Attr_3, ..\}$ son asignados a partir de $\mathbb{U}$ también con posibilidad a ser extendidos a un tipo de atributo compuesto.

#### 4.3.4.2. *Almacenamiento de documentos en la nube*

Dada una colección de documentos $C = \{D_1, ..., D_n\}$ que un DO posee y desea almacenar de forma cifrada en el servicio de nube, el DO procede de la siguiente manera para cada $D \in C$:

1. Asigna un identificador único de documento $DID$, y define las políticas de acceso específicas $T_{DID}$ del documento.

2. Bajo un parámetro de seguridad $\lambda$ que define el nivel de seguridad del esquema SE, genera una llave simétrica $K_{DID} = KeyGen_{SE}(\lambda)$ [3].

3. Haciendo uso de $K_{DID}$ procede a cifrar el documento $\widetilde{D} = Enc_{SE}(K_{DID}, D)$.

4. Posteriormente, mediante el algoritmo de cifrado de CP-ABE el DO cifra la llave $K_{DID}$ conforme a las políticas de acceso que fueron definidas para el documento $\widetilde{K}_{DID} = Enc_{CP\text{-}ABE}(PK, K_{DID}, T_{DID})$.

[3] La llave $K_{DID}$ es propia para el documento con el identificador único $DID$, pero puede ser reutilizada para cifrar todos aquellos documentos que posean exactamente la misma política de acceso.

5. El conjunto total de las políticas de acceso de los documentos se expresa como $ACP = \{T_1, ..., T_n\}$, y para garantizar su integridad durante el envío hacia el CSP el DO genera una firma digital del mensaje $\sigma_{ACP} = Sign(privK_{DO}, ACP)$.

6. Finalmente, el DO transfiere hacia el CSP la colección de documentos cifrados $\widetilde{C} = \{[\widetilde{K}_1, \widetilde{D}_1], ..., [\widetilde{K}_n, \widetilde{D}_n]\}$, las políticas de control de acceso $ACP$ de los documentos y la firma digital $\sigma_{ACP}$ para su almacenamiento.

7. Al recibir la tupla $\{C', ACP, \sigma_{ACP}\}$, el CSP valida la integridad y autenticidad del conjunto de políticas. Para ello, verifica $Verify(pubK_{DO}, \sigma_{ACP}, UID \parallel ACP) \rightarrow \{verdadero, falso\}$ que $\sigma_{ACP}$ sea una firma válida y generada por el DO. Si la salida de la verificación arroja $verdadero$ almacena el conjunto $ACP$ en **BD-ACP**.

#### *4.3.4.3. Autorización de acceso*

Cuando un nuevo usuario U desea tener acceso al sistema multiusuario del servicio de nube emite una solicitud ante el DO. Si el DO está de acuerdo con la solicitud procede de la siguiente manera:

1. Asigna a U un identificador único de usuario $UID$.

2. Asimismo, determina el conjunto de atributos $A_{UID} = \{Attr_1 \wedge Attr_2 \wedge Attr_3, ..\}$ para U.

3. Genera para U una llave secreta en correspondencia a los atributos asignados $SK_{UID} = KeyGen_{CP\text{-}ABE}(MK, A_{UID})$.

4. Como medida para garantizar la integridad del mensaje durante su transmisión hacia el CSP, el DO genera la firma digital del mensaje haciendo uso de su llave privada $\sigma_A = Sign(privK_{DO}, UID \parallel A)$.

5. Para concretar el alta de usuario en el servicio de almacenamiento en la nube el DO transfiere la tupla $\{UID, A_{UID}, \sigma_A\}$ hacia el CSP.

6. Cuando el CSP recibe la tupla $\{UID, A_{UID}, \sigma_A\}$ utiliza la llave pública del DO para verificar que $\sigma_A$ sea una firma digital válida, si la verificación $Verify(pubK_{DO}, \sigma_A, UID \parallel A_{UID}) \rightarrow \{verdadero, falso\}$ otorga $verdadero$, entonces añade la relación $UID$-$A_{UID}$ a **BD-USER**.

#### *4.3.4.4. Compartición de llaves*

El mecaniso de seguridad que se diseñó para que el DO comparta con los usuarios autorizados las llaves criptográficas de los servicios de control de acceso y de búsqueda tiene como fundamento el uso de dos técnicas criptográficas: cifrado de datos y firmas digitales. Con estas técnicas, respectivamente se garantiza tanto la confidencialidad como la integridad del mensaje durante su transmisión. Por tanto, dado un mensaje dirigido al usuario con el identificador $UID$ y denotado por $K = \{PK, A_{UID}, SK_{UID}, SearchKeys\}$, donde $PK$ son los parámetros públicos de CP-ABE, $A_{UID}$ los atributos de usuario, $SK_{UID}$ la llave secreta del usuario, y $SearchKeys$ es el conjunto de llaves del servicio de búsqueda (por definir en la Sección 5.1.5.2), el DO procede a obtener la firma digital del mensaje haciendo uso de su llave privada $\sigma_{KU} = Sign(privK_{DO}, K)$. Posteriormente, cifra el mensaje utilizando la llave pública del usuario a quien va dirigido el mensaje $\widetilde{K} = Enc(pubK_{UID}, K)$ y envía $\{\widetilde{K}, \sigma_K\}$ al usuario. A la recepción de $\{\widetilde{K}, \sigma_K\}$, el usuario primeramente descifra el mensaje recibido con su llave privada obteniendo el mensaje en claro $K = Dec(privK_{UID}, \widetilde{K})$. Después, verifica la firma digital del mensaje en claro utilizando la llave pública del DO $Verify(pubK_{DO}, \sigma_{KU}, KU) \rightarrow \{verdadero, falso\}$, si el valor de salida es $verdadero$, el usuario autentica que cada uno de los elementos de $K$ fueron emitidos por el DO y acepta como válidas las llaves.

#### *4.3.4.5. Acceso a un documento*

Durante una petición emitida por un usuario U para acceder a un documento de un DO, el CSP validará que el $UID$ de U esté registrado en **BD-USER** y que el conjunto de atributos $A_{UID} = \{Attr_1 \wedge Attr_2 \wedge Attr_3, ..\}$ del usuario satisfagan la política de acceso $T_{DID}$ definida para

el documento. Denotamos $T_{DID}(A_{UID}) = 1$ para decir que los atributos de usuario $A_{UID}$ satisfacen la estructura de acceso $T_{DID}$. En la práctica, las políticas de acceso fueron especificadas mediante lenguaje XML, y para verificar la satisfacción de la estructura de acceso se utilizó XQuery, que es un lenguaje de consulta para colecciones de datos XML. Por tanto, si la verificación se cumple U podrá descargar la estructura de archivo $[\widetilde{K}_{DID}, \widetilde{D}_{DID}]$. Para acceder al contenido del documento U primeramente obtiene la llave simétrica $K_{DID} = Dec_{CP\text{-}ABE}(\widetilde{K}_{DID}, SK)$ para después descifrar el documento $D = Dec_{SE}(K_{DID}, \widetilde{D})$.

#### *4.3.4.6. Revocación de acceso*

La revocación de acceso a un usuario es llevada a cabo por el CSP a petición exclusiva del DO, y se expresa mediante el conjunto $R = \{UID_1, ..., UID_u\}$ compuesto por el $UID$ de cada usuario a quien el DO desea revocar el acceso. La petición debe ser firmada digitalmente para garantizar su autenticidad $\sigma_R = Sign(privK_{DO}, R)$. Cuando el CSP recibe la petición de revocación $\{R, \sigma_R\}$ deberá validar la integridad y autenticidad de ésta antes de proceder con su ejecución. Para ello, verifica $Verify(pubK_{DO}, \sigma_R, R) \rightarrow \{verdadero, falso\}$ que en efecto $\sigma_R$ sea una firma válida y generada por el DO. Si la salida de la verificación arroja $verdadero$, entonces el CSP procede a eliminar la relación $UID\text{-}A_{UID}$ de **BD-USER** por cada $UID \in R$. De esta manera, un usuario que haya sido revocado ya no podrá acceder a ningún documento.

# 5

# Servicios de indización, búsqueda e integridad para un almacenamiento seguro en la nube

## 5.1 Indización y Búsqueda

Una las aplicaciones mayormente utilizadas del cómputo en la nube es el modelo Almacenamiento como Servicio (StaaS). StaaS permite a los usuarios almacenar sus datos en los discos remotos de los servidores de nube y acceder a ellos en cualquier momento y desde cualquier lugar, ofreciendo características de ventaja como disponibilidad y escalabilidad, entre otras. Sin embargo, con la adopción de este modelo la principal preocupación de los propietarios de datos es la seguridad y privacidad de sus datos sensitivos. La práctica comúnmente aplicada por los propietarios de datos para proteger la confidencialidad de sus datos es almacenarlos de forma cifrada. Sin embargo, esta solución limita el uso de los mismos, particularmente la funcionalidad de búsquedas sobre los datos cifrados de tal manera que la privacidad sea preservada y que no se revele nada acerca del contenido

de los datos originales.

En relación a esto, la problemática de desarrollar un esquema que permita realizar búsquedas sobre datos (documentos de texto no estructurado) cifrados almacenados en la nube y que preserve la privacidad es abordada en esta sección.

### 5.1.1 Cifrado de Búsqueda

El Cifrado de Búsqueda (SE por sus siglas en inglés) es una primitiva criptográfica que permite a una entidad efectuar búsquedas de forma segura sobre datos cifrados. Dentro de la literatura, la mayoría de los esquemas SE emplean la construcción de un índice seguro. Un índice seguro [6] es una estructura de datos que sólo a través de una trampilla (*trapdoor*) válida generada mediante una llave secreta a partir de un palabra, permite verificar la existencia de ésta en un documento sin necesidad de descifrar.

Song *et al* [4] fueron los primeros en proponer un esquema SE en un enfoque simétrico (SSE) y que tiene como base algoritmos de cifrado y de búsqueda que son simples y rápidos a través de operaciones de cifrado por flujo y bloques. En el mismo rubro de SSE, Goh [6] presenta una construcción eficiente de un índice seguro a través del uso de funciones pseudo-aleatorias y un índice por documento mediante *Bloom filter* [1].

En el enfoque de llave pública Boneh *et al* [5] presentan el primer esquema SE, el cual denominan como *Public Key Encryption with keyword Search* (PEKS). En este esquema el cifrado de búsqueda

[1]Bloom filter es un conjunto $S = \{s_1, s_2, ..., s_n\}$ de $n$ elementos representado por un arreglo de $m$ bits. Todos los bits del arreglo son inicializados a 0. El filtro utiliza $r$ funciones hash independientes $h_1, ..., h_r$, donde $h_i : \{0,1\}^* \rightarrow [1, m]$ para $i \in [1, r]$. Para cada elemento $s \in S$, el arreglo de bits en las posiciones $h_1(s), ..., h_r(s)$ es inicializado a 1. Para determinar si un elemento $a$ pertenece al conjunto $S$ se verifican los bits en las posiciones $h_1(a), ..., h_r(a)$. Si todos los bits están establecidos a 1, entonces $a$ es considerado miembro del conjunto. Existe sin embargo, una mínima probabilidad de un falso positivo.

de una palabra es creado utilizando la llave pública de una entidad. A manera de ejemplo, considere un servidor de correos que almacena varios mensajes públicamente cifrados para Alicia por diversos emisores. Bajo este esquema, sólo la entidad propietaria (Alicia) de la llave privada podrá generar la trampilla que habilite al servidor de correos a identificar todos los mensajes que contienen la palabra especificada, pero sin aprender nada más.

### 5.1.2 Búsqueda difusa sobre datos cifrados

El problema de búsqueda difusa sobre datos cifrados fue resuelto por Li *et al* [24], que desarrollaron la técnica *Wildcard-based Fuzzy Set Construction* (WFSC) la cual consiste en utilizar una distancia de edición como medida de similitud entre cadenas para la construcción de un conjunto de palabras difusas. La distancia de edición ed $(w_1, w_2)$ entre dos palabras $w_1$ y $w_2$ es el número de operaciones [2] requeridas para transformar una de ellas en otra. Se denota $S_{w,d}$ como el conjunto de palabras difusas $w'$ que satisfacen ed($w$,$w'$)$\leq d$ para cierto número entero $d$ y una palabra $w$. Por ejemplo, para la palabra CASTLE y una distancia de edición igual a 1, el conjunto de palabras difusas que se construye es: $S_{CASTLE,1}$={CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, ..., CASTL*E, CASTL*, CASTLE*}. Con el uso del comodín (*) el total de palabras generadas es solamente 13+1 en lugar de 13x26(letras en el abecedario)+1. Por tanto, a cada palabra le corresponden $O(l^d)$ palabras difusas, en donde $l$ es la longitud de la palabra y $d$ la distancia de edición. El índice de una palabra $w_i$ es construido por el propietario de los datos mediante el cálculo de una trampilla $T_{w_i'} = f(sk, w')$ para cada $w' \in S_{w,d}$, donde $sk$ es una llave secreta que comparte con usuarios autorizados y $f$ es una función pseudo-aleatoria o aleatoria. Para emitir la búsqueda de $w$ el usuario calcula la trampilla $Tw$ y la envía al servidor, éste busca dentro del índice y retorna el resultado.

Por su parte, Liu *et al* [25] argumentan que la técnica WFSC descrita previamente en [24] brinda

[2] Las primitivas de operación que se aplican son: substitución, eliminación e inserción de un carácter en una palabra.

flexibilidad para una búsqueda no exacta, pero que no todas las $O(l^d)$ combinaciones generadas son palabras válidas dentro del lenguaje. La aportación de este trabajo radica en añadir la construcción y uso de un diccionario, esto con propósito de optimizar el conjunto de palabras difusas generado. Dicho de otra forma, todas las palabras difusas a generar deberán pertenecer al diccionario. Esta integración reduce los tamaños de los índices así como costos de comunicación y almacenamiento, siendo la ventaja más notoria conforme la distancia $l$ aumenta.

Chuah y Hu [17] proponen una solución de búsqueda por múltiples palabras y difusas que utiliza un índice bed-tree como estructura. Primeramente, para cada palabra $w_i$ con distancia de edición $d$, un índice bloom filter $b_{wi,d}$ es generado para contener el conjunto de palabras difusas $S_{wi,d}$. Después, se genera un valor hash $h_{wi}$ que representa la palabra y se produce un vector de datos basado en n-gram [3]. Un nodo hoja de un índice bed-tree (véase Figura 5.1) consiste de la siguiente información: $[h_{wi}, b_{wi,1}, b_{wi,2}, b_{wi,d}, \{FID_{wi}\}]$, aquí $FID_{wi}$ denota el identificador de archivo en forma cifrada $Enc(sk, FID_{wi} \parallel w_i)$ que contiene la palabra.

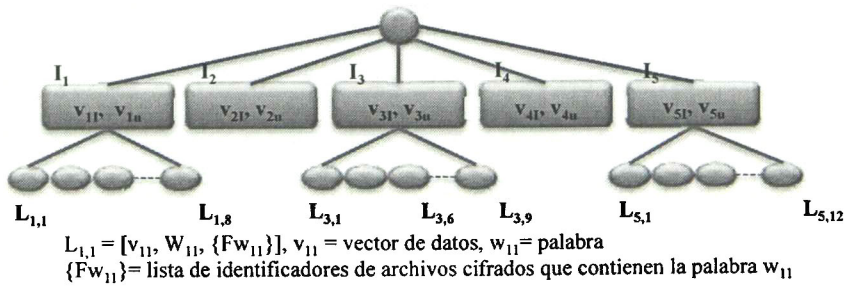



**Figura 5.1:** Construcción del índice bed-tree

A diferencia de los dos trabajos anteriores [24, 25] este trabajo sí soporta búsqueda difusa de múltiples palabras, la diferencia consiste en que se aplicó una probabilidad de co-ocurrencia para identificar e incluir palabras (dobles) que asociadas fueran de utilidad durante las consultas. Presentan

[3]Un n-gram es una secuencia de $n$ caracteres correspondientes a una cadena $s$, en donde, dado $s$ existen $s+n-1$ n-grams sobrepuestos, por ejemplo, si tenemos que $s_1$="network" y $n$=2 el conjunto de n-grams resultantes es: $Q(s_1)=\{\#n, ne, et, tw, wo, rk, k\#\}$.

como resultado una mejora en tiempos de construcción y espacio de almacenamiento del índice en comparativa con un enfoque que no soporta consultas por múltiples palabras y otro que utiliza una estructura de índice trie-traverse.

### 5.1.3 Búsqueda por relevancia sobre datos cifrados

Wang *et al* [16, 19] proponen un framework para realizar búsquedas seguras por relevancia sobre archivos cifrados en un almacenamiento de nube a partir de una función que utiliza la frecuencia de las palabras para calcular la relevancia. En su propuesta hacen uso de índices invertidos para contener la frequencia de cada palabra, e integra el cifrado *Order Preserving Symmetric Encryption* (OPSE) para cifrar los valores de frecuencia. OPSE es un esquema de cifrado determinista en donde el orden numérico de los textos planos es preservado por la función de cifrado. Aunque este método soporta búsqueda por relevancia sólo es de una sola palabra.

Cao *et al* [18] proponen un esquema que preserva la privacidad y que soporta búsquedas por relevancia de múltiples palabras sobre datos cifrados. En este caso para capturar la relevancia de los documentos arrojada en cada consulta se realiza mediante la medida de similitud "coordinate matching", que se evalúa con el uso de similitud de producto interior, lo cual es adaptado de [26] en donde desarrollan una técnica segura del método k-vecinos más cercanos (KNN por sus siglas en inglés). Durante la construcción del índice cada documento $F_i$ es asociado a un vector binario $D_i$, cada bit $D_i[j] \in \{0, 1\}$ representa si la palabra correspondiente $W_j$ está contenida en el documento. Asimismo, la consulta de búsqueda es descrita como un vector binario $Q$ indicando las palabras de interés donde cada bit $Q[j] \in \{0, 1\}$ representa la existencia correspondiente de la palabra $W_j$ en la consulta $\widetilde{W}$. La puntuación de similitud para un documento $F_i$ se mide con el producto interior entre el vector de consulta y el vector de datos $D_i \cdot Q_i$. La desventaja de este esquema es que el usuario requiere conocer a priori todas las palabras de la colección de documentos y su orden dentro del índice para poder construir el vector de consulta. Además, el método de búsqueda requiere recorrer

linealmente el índice de todos los documentos con cálculos de multiplicación de vectores.

### 5.1.4 Cifrado Homomórfico

La seguridad de la información es sin duda alguna uno de los principales requerimientos de cualquier sistema de información digital, para cumplir con ello diversas técnicas criptográficas pueden ser aplicadas (véase Sección 2.2). Por otro lado, con base en el planteamiento expuesto en la Sección 1.2 se hace notar que es muy difícil proteger la privacidad y confidencialidad de los datos de los usuarios contra posibles ataques de un proveedor de servicios (no confiable) a menos que los datos sean almacenados de forma cifrada. Sin embargo, esta solución hace que el problema se vuelva bastante complejo cuando se tiene la necesidad de efectuar cálculos de forma pública con los datos cifrados, y se garantice la privacidad de los mismos. La idea discutida en esta sección es precisamente eso, mantener la funcionalidad de poder realizar operaciones directamente sobre los datos cifrados sin necesidad de descifrarlos. Como primera impresión esto pareciera ser algo imposible, pero es en este contexto en donde el Cifrado Homomórfico (HE por sus siglas en inglés) puede ser utilizado.

Un esquema HE ofrece la gran ventaja de que permite realizar operaciones aritméticas sobre datos cifrados sin revelar nada acerca del contenido de los datos originales. Cuando el resultado es descifrado el valor coincide tal como si las operaciones hubieran sido realizadas sobre texto en claro. La noción de HE fue introducida por primera vez en 1978 por Ronald Rivest, Len Adleman y Michael Dertouzos [27]. En el artículo se interroga la existencia de funciones de cifrado que permitan operar datos cifrados sin requerir un descifrado preliminar de los operandos, funciones a las cuales denominaron como *homomorfismo de privacidad* (véase Definición 5).

**Definición 4.** ***Homomorfismo de privacidad*:**

*Un sistema algebraico consiste de un conjunto $S$, algunas operaciones $f_1, f_2, \ldots$, algunos predicados $p_1, p_2, \ldots$, y algunas constantes distinguidas $s_1, s_2, \ldots$. Dada la definición anterior, codificar y decodificar significa mapear elementos del sistema algebraico $U$ al sistema algebraico $C$, o viceversa. Dicho de otra forma, si $U = \{S; f_1, \ldots, f_k; p_1, \ldots, p_l; s_1, \ldots, s_m\}$ entonces $C = \{S'; f'_1, \ldots, f'_k; p'_1, \ldots, p'_l; s'_1, \ldots, s'_m\}$, $\phi : S' \to S$ es una función de decodificación y su inversa $\phi^{-1} : S \to S'$ la función de codificación. Por tanto, se dice que $\phi$ es un homomorfismo de $C$ a $U$, más explícitamente:*

1. $(\forall i)(a, b, c, \ldots)[f'_i(a, b, \ldots) = c \Rightarrow f_i(\phi(a), \phi(b), \ldots) = \phi(c)]$,
2. $(\forall i)(a, b, \ldots)p'(a, b, ...) \equiv p(\phi(a), \phi(b), ...)$,
3. $(\forall i)\phi(s'_i) = s_i$.

A manera de ejemplo y con base en la definición anterior, supongamos que un usuario desea conocer el valor $f_1(d_1, d_2)$, emitirá por ello una petición al sistema para calcular $f'_1(\phi^{-1}(d_1), \phi^{-1}(d_2))$. Por tanto, ya que $\phi$ es un homomorfismo, el sistema podrá tomar los datos cifrados de la respuesta sin necesidad de descifrar resultados intermedios $\phi(f'_1(\phi^{-1}, \phi^{-1}(d_2))) = f_1(d_1, d_2)$. Esta terminología está estrechamente relacionada con el concepto de homomorfismo de grupos. En efecto, la función de cifrado de un esquema HE con las propiedades mencionadas anteriormente es un *homomorfismo* (véase Definición 6).

**Definición 5.** ***Homomorfismo*:**

*La función $f : G \to H$ que va de un grupo $G$ a otro grupo $H$, es un homomorfismo de grupos si la operación de grupo $(*)$ es preservada en el sentido que:*

$$f(g_1 *_G g_2) = f(g_1) *_H f(g_2) \qquad \forall g_1, g_2 \in G.$$

#### 5.1.4.1. *Criptosistemas homomórficos*

Los criptosistemas homomórficos recaen en la categoría de criptosistemas de llave pública y son una instancia del homomorfismo de privacidad. Tienen la propiedad especial de realizar cálculos sobre datos cifrados sin requerir conocer la llave privada, dicho de otra forma, no requieren descifrar los datos para poder ejecutar los cálculos. De manera formal un criptosistema homomórfico se especifica bajo la Definición 7.

**Definición 6. *Criptosistema homomórfico*:**

*Un criptosistema de llave pública es homomórfico bajo adición o multiplicación si:*

$$D(E(a) \oplus E(b)) = a + b \quad o \quad D(E(a) \otimes E(b)) = a * b \qquad \forall a, b \in P,$$

*donde $P$ es el espacio de texto planos, $C$ el espacio de texto cifrados, $\oplus$ un operador de adición, $\otimes$ un operador de multiplicación, $E : P \to C$ una función de cifrado y $D : C \to P$ la correspondiente función de descifrado.*

El esquema de cifrado de un criptosistema puede ser determinista o probabilístico. Cuando el esquema es probabilístico, se introduce aleatoriedad en su algoritmo de cifrado. Esto significa que para un mensaje en claro cifrado más de una vez pueden obtenerse distintos textos cifrados, por lo que es difícil detectar dos cifrados diferentes del mismo mensaje. Además de lo anterior, los criptosistemas homomórficos se clasifican en dos categorías:

- **Criptosistemas Parcialmente Homomórficos (PH por sus siglas en inglés).** Evalúan una única operación aritmética, ya sea adición o multiplicación. Dependiendo de la operación que soporte, el criptosistema homomórfico se denomina aditivo o multiplicativo, respectivamente.

- **Criptosistemas Completamente Homomórficos (FH por sus siglas en inglés).** Evalúan un número arbitrario de adiciones y un número arbitrario de multiplicaciones. Por tanto,

a partir de la combinación de adiciones y multiplicaciones se hace posible derivar cualquier función u operación compleja.

Entre los criptosistemas homomórficos multiplicativos se encuentran los criptosistemas RSA [28] y ElGamal [29] que basa su seguridad en el problema del logaritmo discreto. El criptosistema RSA es el criptosistema de llave pública más ampliamente utilizado, es determinista y su seguridad está basada en el problema de factorización de enteros. Sin embargo, su propiedad homomórfica no es utilizada debido a que ésta no satisface buenas nociones de seguridad. El primer criptosistema probabilístico desarrollado fue el criptosistema Goldwasser-Micalli [30] el cual basa su seguridad en el problema del residuo cuadrático y provee la operación homomórfica XOR. Tiene como limitante el tamaño (un bit) permitido de los mensajes a cifrar. Los criptosistemas Benaloh [31], Okamoto-Uchiyama [32], Paillier [33] y Damgard-Jurik [34] son algunos ejemplos de criptosistemas homomórficos aditivos. Un criptosistema que toma especial interés es el criptosistema Boneh-Goh-Nissim [35] que utiliza emparejamientos bilineales y permite una operación homomórfica de multiplicación además de un número arbitrario de adiciones.

Desde 1978, la idea de homomorfismo de privacidad presentada por Riviest *et al* [27] para la construcción de un criptosistema completamente homomórfico (FH) había sido un problema abierto. No fue hasta el 2009 cuando Craig Gentry investigador de IBM a través de su tesis doctoral publica el primer criptosistema FH utilizando criptografía basada en *lattice* [36]. En este criptosistema, bajo distintos niveles de seguridad el tamaño de la llave pública puede ir de 70 MB-2.3 GB, y el tiempo de una operación *bootstrapping* está en el rango de 0.5-30 minutos [37]. Dada la complejidad de las operaciones de cifrado y descifrado así como los tamaños de llaves y texto cifrado este criptosistema aún no sea viable integrarlo en alguna aplicación real.

### 5.1.5 Esquema de Indización propuesto

De acuerdo a la revisión de literatura presentada en las secciones 5.1.1, 5.1.2 y 5.1.3 se recapitulan los siguientes enfoques de búsqueda sobre datos cifrados:

- Cifrado de Búsqueda. Los trabajos bajo este enfoque [4, 5, 6] sólo soportan una búsqueda booleana convencional.

- Búsqueda difusa sobre datos cifrados. Diversos trabajos [17, 24, 25] han sido propuestos bajo este enfoque, su objetivo es soportar búsquedas con criterio de similitud mediante palabras difusas.

- Búsqueda por relevancia sobre datos cifrados. Existen algunas propuestas [16, 18, 19] pero su funcionalidad en general sigue siendo algo limitada. Por ejemplo, a excepción de una de las propuestas [18] el resto no soporta búsquedas por múltiples palabras.

Para el contexto de almacenamiento a gran escala como lo es en el cómputo en la nube, por razones de eficiencia destacamos la necesidad de habilitar un esquema de búsqueda por relevancia sobre datos cifrados que permita a los usuarios recuperar los top-$k$ documentos más relevantes. Asimismo, el método de búsqueda debe ser flexible en el sentido de soportar consultas por múltiples palabras. Por tanto, enfocamos este trabajo al desarrollo de una solución que permita realizar búsqueda por relevancia y por múltiples palabras sobre datos cifrados.

El esquema que se propone para un servicio de indización seguro se divide en dos fases: construcción del índice de búsqueda y seguridad en el índice de búsqueda, las cuales se describen en las secciones subsecuentes.

#### *5.1.5.1. Construcción del índice de búsqueda*

La primera fase del servicio de indización consiste en la construcción de un índice de búsqueda a partir de una colección de documentos $C = \{D_1, ..., D_n\}$ que un propietario de datos (DO) posee y mediante la aplicación de técnicas tradicionales de Recuperación de la Información (IR por sus siglas en inglés).

El DO ejecuta los siguientes cinco pasos a través del **Módulo de Indización**:

1. **Extracción de texto.** Dada una colección de documentos de texto a indizar, el primer paso para construir un índice de búsqueda corresponde a la extracción del contenido de cada documento, en donde los bytes de un archivo son convertidos a una secuencia de caracteres. Para que dicha conversión se efectúe de forma correcta es requerido considerar el esquema de codificado de caracteres (ASCII, Unicode, etc.) así como el formato del archivo, esto último en relación a que ciertos formatos (e.g., XML, HTML) representan su contenido a través de un lenguaje de marcado, además de utilizar secuencias especiales de caracteres para representar a otro carácter, por ejemplo la secuencia "&" equivale al carácter "&".

2. **Tokenización.** Este proceso consiste en dividir la secuencia de caracteres correspondiente a un documento y formar piezas denominadas tokens, términos o palabras. Como divisor de términos se utilizó al carácter espacio en blanco.

3. **Preprocesamiento de texto.** En orden de contribuir a la eficacia de IR y a la optimización del tamaño del índice que será construido, una tarea que cobra especial importancia es el preprocesamiento de texto. Los tipos de preprocesamiento que fueron aplicados son:

    - Lista de palabras vacías. En la mayoría de los documentos existen palabras que tienen gran frecuencia de aparición y que no aportan gran relevancia a la representación del contenido de un documento, es por ello que dentro del preprocesamiento de texto

generalmente se hace uso de una lista de palabras vacías con el objetivo de omitir para su indización a aquellas palabras del documento que coincidan con las de la lista. La lista de palabras vacías debe incluir a todas aquellas que son muy usuales dentro de las expresiones gramaticales: determinantes (artículos, posesivos, demostrativos), etc.. Asimismo, es necesario remover caracteres no funcionales.

- Normalización. Para casos en donde tokens no coinciden exactamente en secuencia de caracteres pero tienen un mismo significado es deseable que todos ellos sean considerados dentro un mismo resultado de búsqueda. Debido a lo anterior se hace necesario aplicar un proceso de normalización para transformar los tokens, generando como resultado una clase de equivalencia. Distintas estrategias de normalización pueden ser aplicadas: mayúsculas a minúsculas, acentos, puntos intermedios, diacríticos como la diérisis o la tilde, etc..
- *Stemming*. En el lenguaje natural existen diversas formas de expresar una simple idea, esto puede causar desventaja a los resultados de búsqueda al tratar de hacer coincidir a un término exacto y no considerar las otras formas de la palabra, técnicas como *stemming* abordan este problema. *Stemming* es una técnica que de forma heurística y el uso de reglas elimina el sufijo en una palabra obteniendo el término raíz, por ejemplo:

$$connect \leftarrow \{connected,\ connecting,\ connection,\ connections\}.$$

Dado que la experimentación del esquema propuesto será realizada utilizando documentos de texto en idioma inglés, el preprocesamiento de texto fue enfocado a dicho idioma. Por ejemplo, la lista de palabras vacías se construyó con palabras en idioma inglés, asimismo, se hizo uso del algoritmo mayormente utilizado para *stemming* del idioma inglés: "The Porter Stemming Algorithm" [38]. Cabe mencionar que el resto del esquema de indización propuesto puede ejecutarse tal cual independientemente del idioma, lo único necesario a cambiar en dado caso de utilizar documentos con texto en otro idioma es adaptar el enfoque de estas técnicas de preprocesamiento de texto.

4. **Ponderación de términos**. Dentro del área IR, los esquemas de ponderación de términos expresan qué tan relevante es un término para un documento dentro de una colección, y son utilizados para tareas de IR y minería de texto. Uno de los esquemas de ponderación mayormente aplicados y que retomamos para la propuesta es TF-IDF. Este esquema corresponde al producto de dos medidas estadísticas: Frecuencia de Término (TF) y Frecuencia Inversa de Documento (IDF). La medida TF determina la ocurrencia de un término en un documento, e IDF mide la importancia de un término en una colección de documentos. La Ecuación 5.1 define el esquema TF-IDF.

$$tf\text{-}idf_{t,D} = tf_{t,D} \times idf_t \tag{5.1}$$

Aquí, $tf\text{-}idf_{t,D}$ corresponde al peso de un término $t$ en un documento $D$, $tf_{t,D}$ es el número de ocurrencias de un término $t$ en el documento $D$, e $idf_t$ es calculada a partir de la Ecuación 5.2, en donde $n$ representa el número total de documentos de la colección, y $DF_t$ es el número de documentos en la colección que contienen un término $t$ y se denomina como Frecuencia de Documento (DF).

$$idf_t = log\frac{n}{df_t} \tag{5.2}$$

5. **Índice de búsqueda.** En sistemas IR los índices son estructuras de datos que se utilizan para acelerar un proceso de búsqueda, y uno de los más populares para soportar búsquedas sobre texto completo es el índice invertido. Este tipo de índice es el utilizado dentro de la solución propuesta y normalmente se compone de: (i) un vocabulario que contiene todos los distintos términos encontrados en el texto y (ii), por cada término $t$ del vocabulario, una lista que contiene estadísticas acerca de las ocurrencias de $t$ en el texto, tal lista es conocida como lista invertida. Para mayor ilustración, la Figura 5.2 ejemplifica la estructura de datos de un

índice invertido, sobre ella se observa que cada término $t$ del vocabulario apunta hacia una lista invertida compuesta por la concatenación de los identificadores de documentos $DID$ que contienen a dicho término más un valor estadístico respectivo, $tf\text{-}idf_{t,d}$ para nuestro caso.

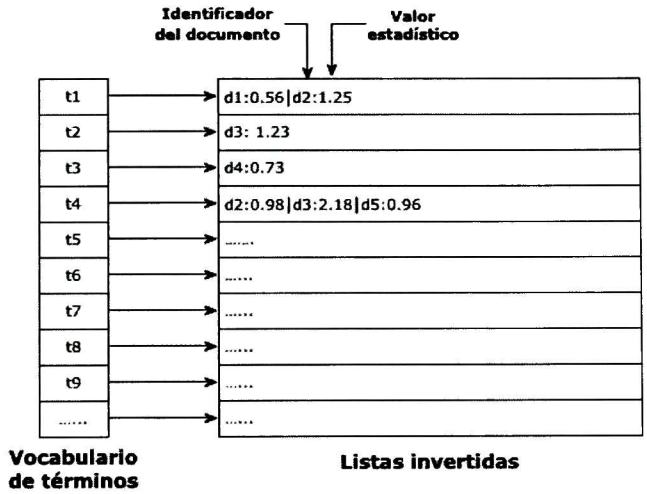



**Figura 5.2:** Índice invertido

### *5.1.5.2. Seguridad en el índice de búsqueda*

La segunda fase del servicio de indización consiste en aplicar primitivas criptográficas para proteger la privacidad del índice de búsqueda, particularmente sobre los siguientes requerimientos:

- **Privacidad del vocabulario.** Para proteger la privacidad de las palabras pertenecientes al vocabulario se propone el uso de una función pseudoaleatoria $f$ y una llave secreta $sk$ (véase Definición 8). Por tanto, sea $V$ el vocabulario compuesto de $p$ palabras únicas obtenidas de la fase de preprocesamiento de texto, para cada $w \in V$ el DO calcula $\widetilde{w} = f(sk, w)$, donde $\widetilde{w}$ representa la forma secreta de la palabra $w$. De esta manera sólo los usuarios que hayan sido autorizados por el DO, es decir, a quienes el DO les comparta la llave secreta $sk$ podrán emitir consultas válidas. De forma aplicada se utiliza HMAC como función pseudoaleatoria, la

cual toma una función hash segura de entrada y otorga un valor de salida con un tamaño fijo de bits (véase Sección 2.2.2.3).

- **Privacidad de las listas invertidas.** El mayor desafío aquí presente es proteger la privacidad de los valores de ponderación en las listas invertidas mientras que se mantiene la funcionalidad de búsqueda por relevancia. En este punto, retomando el concepto de cifrado homomórfico (HE) abordado en la Sección 5.1.4 realzamos la importancia de la aplicación de éste como bloque de seguridad para el cómputo en la nube ya que hace posible el siguiente escenario: un usuario podría cifrar sus datos mediante HE, almacenarlos en los servidores de nube, emitir peticiones de cálculos sobre ellos ante el CSP, éste ejecutar los cálculos directamente sobre los datos cifrados sin afectar su confidencialidad, y retornar los resultados. Finalmente, el usuario haciendo uso de su llave privada descifraría los resultados.

En relación al fundamento anterior, se propone aplicar HE para cifrar cada una de los valores de ponderación de las listas invertidas y hacer uso de la propiedad homomórfica para efectuar los cálculos de relevancia durante una búsqueda en el servicio de nube, lo cual se describe en la siguiente sección. Más formalmente, denotando un criptosistema homomórfico como $HE = (KeyGen_{HE}, Enc_{HE}, Dec_{HE}, OpeHom_{HE})$, donde $KeyGen_{HE}$ es el algoritmo de generación de llaves, $Enc_{HE}$ el algoritmo de cifrado, $Dec_{HE}$ el algoritmo de descifrado, y $OpeHom_{HE}$ el algoritmo que realiza el cálculo de la propiedad homomórfica, el DO procede de la siguiente manera:

1. A través del algoritmo $KeyGen_{HE}(\lambda)$ del criptosistema homomórfico y $\lambda$ que es el parámetro que define el nivel de seguridad del criptosistema, procede a generar una llave privada $sk_{HE}$ y una llave pública $pk_{HE}$.
2. Para cada $w \in V$ recupera la lista invertida $L$ asociada y procede a cifrar cada una de las entradas en la lista, lo cual puede ser visto de la siguiente forma: $L = \{DID_1 \parallel$

$Enc_{HE}(sk_{HE}, tf\text{-}idf_{w,DID_1}), .....\}$.

El criptosistema homomórfico que se elija debe cumplir con la característica de ofrecer un tipo de cifrado probabilístico y no determinista, esto en orden de volver indistinguible el cifrado de dos mismos valores de ponderación para prevenir con ello ataques de análisis estadísticos sobre el índice. Aunque los valores de ponderación al estar cifrados se preservan confidenciales, comparando dichos valores sí se podría detectar los documentos que son igual de relevantes, de aquí el planteamiento de cifrado probabilístico.

Una vez construido el índice de búsqueda seguro, el DO transferirá la tupla $\{\widetilde{V}, I, pk_{HE}\}$ hacia el CSP, donde, $\widetilde{V} = \{\widetilde{w}_1, ..., \widetilde{w}_p\}$ $(1 \leq i \leq p)$ es el vocabulario de palabras, $I$ es el conjunto de listas invertidas $I = \{L_1, ..., L_p\}$, donde, $L_i = \{DID_1 \parallel Enc_{HE}(sk_{HE}, tf\text{-}idf_{w_i,DID_1}), ....\}$, y $pk_{HE}$ es la llave pública. Asimismo, el DO deberá distribuir a todos aquellos usuarios autorizados el conjunto de llaves $SearchKeys = \{sk, sk_{HE}\}$ necesarios para el servicio de búsqueda, esto último se realiza de acuerdo al mecanismo de compartición de llaves definido previamente en la Sección 4.3.4.4.

**Definición 7. *Función pseudoaleatoria (PRF por sus siglas en inglés)*:**

*Una función* $f\ \{0,1\}^n \times \{0,1\}^s \rightarrow \{0,1\}^m$ *es una función pseudoaleatoria* $(t, \epsilon, q)$ *si*

1. *Dada una llave* $K \in \{0,1\}^s$ *y una entrada* $X \in \{0,1\}^n$ *hay un algoritmo "eficiente" para calcular* $F_K(X) = F(X, K)$.

2. *Para cualquier algoritmo oráculo* $A$ *de tiempo* $t$, *tenemos que*

$$\mid Pr_{k \leftarrow \{0,1\}^s}[A^{f_K}] - Pr_{f \in F}[A^F] \mid < \epsilon$$

*donde* $F = \{f : \{0,1\}^n \leftarrow \{0,1\}^m\}$ *y* $A$ *hace la mayoría de las* $q$ *consultas al oráculo.*

$k \in \{0,1\}^s$ *y una entrada* $X \in \{0,1\}^n$ *es un algoritmo eficiente para calcular* $F_K(X) = F(X, K)$.

### 5.1.6 Esquema de Búsqueda propuesto

Para efectuar búsquedas seguras sobre la colección de documentos cifrados almacenados en la nube a partir de una consulta con múltiples palabras de interés, el esquema de búsqueda propuesto considera las tres siguientes fases:

1. **Generación de consulta.** Dada una consulta $Q = \{w_1, ..., w_x\}$ compuesta por múltiples palabras de interés, para cada $w \in Q$ el usuario aplica las mismas técnicas de preprocesamiento de texto que fueron utilizadas por el DO durante la indización, esto con el objetivo de mantener la precisión en el resultado de búsqueda. Posteriormente, haciendo uso de la llave secreta $sk$ que le fue compartida por el DO procede a obtener la trampilla de cada palabra $T_w = f(sk, w)$ (function GENERACIONCONSULTA en Algoritmo 1), de esta manera se cumple con el requerimiento de que la consulta sea oculta, es decir, la petición de búsqueda no debe revelar las palabras secretas. Finalmente, envía $\widetilde{Q} = \{T_{w_1}, ..., T_{w_x}\}$ hacia el CSP y espera por el resultado.

2. **Búsqueda por relevancia.** Definido el esquema de ponderación de términos se hace necesario contar con un método con el cual sea posible medir la relevancia de un documento durante una búsqueda. Es en este punto donde se introducen las funciones ranking, estas funciones calculan una puntuación numérica de relevancia para un documento dada una consulta. Para la propuesta de solución se eligió una de las funciones ranking más sencillas pero efectiva, y que es ampliamente utilizada en IR, ver Ecuación 5.3. En esta función la puntuación de un documento $D$ es calculada mediante la sumatoria de los valores TF-IDF de todos los términos $t$ coincidentes con los de una consulta $Q$.

$$Score(Q, d) \sum_{t \in Q} tf\text{-}idf_{t,d} \tag{5.3}$$

De forma aplicada al esquema propuesto, se utilizó la propiedad aditiva del criptosistema

homomórfico para realizar los cálculos sobre los valores de ponderación cifrados en la sumatoria, para ello sólo se requiere que el CSP conozca la llave pública $pk_{HE}$. Por tanto, cuando el CSP recibe una consulta de usuario $\widetilde{Q} = \{T_{w_1}, ..., T_{w_x}\}$, primeramente buscará las trampillas dentro del vocabulario $\widetilde{V}$ para identificar cada lista invertida $L = \{DID_1 \parallel Enc_{HE}(sk_{HE}, tf\text{-}idf_{w,DID_1}), .....\}$ asociada, y procederá a realizar los cálculos de relevancia mediante el algoritmo $OpeHom_{HE}()$ utilizando la llave pública $pk_{HE}$. Finalmente, retorna $R = \{DID \parallel Enc_{HE}(sk_{HE}, Score_1), .....\}$ como resultado, éste se conforma por el DID y la puntuación de relevancia cifrada de cada documento en relación a $\widetilde{Q}$ (function BUSQUEDARELEVANCIA en Algoritmo 2).

Es fundamental que los usuarios sólo obtengan las puntuaciones de relevancia de los documentos sobre los cuales tienen acceso, para ello es necesario que se ejecute de forma semejante como se describió en la Sección 4.3.4.5 el control de acceso establecido. Esto es, verificar que los atributos de usuario $A_{UID} = \{Attr_1 \wedge Attr_2 \wedge Attr_3, ..\}$ satisfagan las políticas de acceso $T_{DID}$ definidas para los documentos (function CONTROLACCESO en Algoritmo 2). Para no afectar los tiempos de respuesta durante una búsqueda, esta función podría ser ejecutada sólo una vez durante el inicio de sesión de un usuario y no durante cada petición de consulta, manteniéndose la lista $AC$ en sesión para ser reutilizada.

3. **Ranking.** Al recibir el resultado de búsqueda $R = \{DID \parallel Enc_{HE}(sk_{HE}, Score_1), .....\}$ el usuario descifra las puntuaciones de relevancia de los documentos utilizando la llave privada $sk_{HE}$ que el DO le compartió. Una vez descifradas todas las puntuaciones, realiza el ordenamiento de éstas y con base a sus valores podrá solicitar al servidor de nube los top-$k$ documentos más relevantes (function RANKING en Algoritmo 1).

**Algorithm 1** Búsqueda segura por relevancia y múltiples palabras (Funciones Usuario)

**Require:** Consulta de múltiples palabras $Q = \{w_1, ..., w_x\}$, llave secreta $sk$
**Ensure:** Consulta segura de múltiples palabras $\widetilde{Q} = \{T_{w_1}, ..., T_{w_x}\}$
1: **function** GENERACIONCONSULTA($Q$)
2: **for all** $w \in Q$ **do**
3: $T_w \leftarrow f(sk, w)$
4: **end for**
5: **return** $\widetilde{Q}$
6: **end function**

**Require:** Resultado de búsqueda $R = \{DID \parallel Enc_{HE}(sk_{HE}, Score_1), .....\}$
**Ensure:**
7: **function** RANKING
8: **for all** $(DID, Score) \in R$ **do**
9: $Score \leftarrow Dec_{HE}(sk_{HE}, Score)$
10: **end for**
11: **end function**

### 5.1.7 Evaluación de criptosistemas homomórficos aditivos

A lo largo de esta sección se presenta la descripción de distintos criptosistemas homomórficos aditivos probabilísticos, para finalmente realizar una evaluación de ellos y optar por el más adecuado para el esquema de indización y búsqueda propuesto.

#### *5.1.7.1. Criptosistema Goldwasser-Micalli (GM)*

En 1982, Goldwasser y Micalli introducen el primer criptosistema probabilístico [30] cuya seguridad está basada en el problema del residuo cuadrático. Dada una llave pública $(x, n)$, donde, $n = pq$, siendo $p$ y $q$ dos primos distintos, este criptosistema cifra mensajes $b$ de un solo bit $E(b) = r^2 x^b mod n$, y provee la operación homomórfica XOR o módulo 2. La mayor desventaja en este criptosistema es el tamaño de los mensajes cifrados, que crecen en factor de uno a $n$ bits, acorde al módulo $n$ utilizado.

**Algorithm 2** Búsqueda segura por relevancia y múltiples palabras (Funciones CSP)

**Require:** Identificador único de usuario $UID$, base de datos de políticas de acceso BD-ACP
**Ensure:** Lista de DID autorizados $AC = \{DID_1, .....\}$

**function** CONTROLACCESO($UID$)
  **for all** $T_{DID} \in$ BD-ACP **do**
    **if** $T_{DID}(A_{UID}) \equiv 1$ **then**
      $AC$.Insertar(DID)
    **end if**
  **end for**
**end function**

**Require:** Consulta segura de múltiples palabras $\widetilde{Q} = \{T_{w_1}, ..., T_{w_x}\}$, llave pública $pk_{HE}$, vocabulario $\widetilde{V} = \{\widetilde{w}_1, ..., \widetilde{w}_p\}$ $(1 \leq i \leq p)$, listas invertidas $I = \{L_1, ..., L_p\}$, donde $L_i = \{DID_1 \parallel Enc_{HE}(sk_{HE}, tf\text{-}idf_{w_i,DID_1}), ....\}$, documentos autorizados $AC = \{DID_1, .....\}$
**Ensure:** Resultado de búsqueda $R = \{DID \parallel Enc_{HE}(sk_{HE}, Score_1), .....\}$

**function** BUSQUEDARELEVANCIA($\widetilde{Q}$) ▷ Ejecución por CSP
  $R \leftarrow InicializarTablaHash()$
  **for all** $Tw \in \widetilde{Q}$ **do**
    $i \leftarrow BusquedaVocabulario(\widetilde{V}, Tw)$
    **if** $i \neq null$ **then**
      **for all** $(DID, Weight) \in L_i$ **do**
        **if** $DID \in AC$ **then** ▷ Control de acceso
          $Score \leftarrow R.get(DID)$
          **if** $Score \neq null$ **then**
            $Score \leftarrow OpeHom_{HE}(pk_{HE}, Score, Weight)$
          **end if**
          $R.put(DID, Score)$
        **end if**
      **end for**
    **end if**
  **end for**
  **return** $R$
**end function**

#### *5.1.7.2. Criptosistema Benaloh*

El criptosistema Benaloh [31] (véase Tabla 5.1) a diferencia del criptosistema homomórfico GM [30] que cifra mensajes de un solo un bit, permite establecer una longitud límite $r$ para un mensaje $m$. El decremento de la expansión del mensaje viene al costo de incrementar la complejidad de descifrado

ya que $m$ se determina exhaustivamente. Este proceso puede ser acelerado por pre-procesamiento, para cada $m \in \mathbb{Z}_r$ un valor canónico puede ser pre-calculado como $T_m = y^{m(p-1)(q-1)/r} \bmod n$, ya que es cierto que $c^{(p-1)(q-1)/r} \bmod n = T_m$. Así, los $r$ distintos valores $T_0, T_1, ..., T_{r-1}$ pueden ser pre-calculados, y cualquier valor cifrado $c$ puede ser descifrado encontrando $c^{(p-1)(q-1)/r} \bmod n$ en una tabla de búsqueda. Si $r$ es de tamaño moderado, puede realizarse una combinación de los dos métodos anteriores para no comprometer almacenamiento ni tiempo de cómputo, teniendo una complejidad de descifrado de $O(\sqrt{r})$.

**Propiedades homomórficas.** Para cualquier $c_1 \in E(m_1)$ y $c_2 \in E(m_2)$ se satisface que $(c_1 \otimes c_2) \in E((m_1 + m_2) \bmod r)$ y $(c_1 \oslash c_2) \in E((m_1 - m_2) \bmod r)$.

| Generación de llaves: KeyGen$(p, q)$ | | | |
|---|---|---|---|
| **Entrada:** | $p, q$ tal que $r$ divida a $(p-1)$, $r$ y $(p-1)/r$ sean primos relativos, $r$ y $(q-1)$ sean primos relativos | | |
| Calcular: | $n = pq$ | Elegir: | $y \in \mathbb{Z}_n^*$ tal que $y^{(p-1)(q-1)/r} \bmod n \neq 1$ |
| **Salida:** | $pk = (n, y)$ | | $sk = (p, q)$ |
| **Cifrado:** E$(m, pk)$ | | **Descifrado:** D$(c, sk)$ | |
| **Entrada:** | $m \in \mathbb{Z}_r$ | **Entrada:** | $c \in \mathbb{Z}_n$ |
| Elegir: | $u \in \mathbb{Z}_n^*$ | Determinar: | exhaustivamente el entero no negativo más pequeño tal que $(y^{-m} c \bmod n) \in Enc(0)$ |
| Calcular: | $c = y^m u^r \bmod n$ | | |
| **Salida:** | $c \in \mathbb{Z}_n$ | **Salida:** | $m \in \mathbb{Z}_r$ |

**Tabla 5.1:** Criptosistema Benaloh

### *5.1.7.3. Criptosistema Okamoto-Uchiyama (OU)*

El criptosistema Okamoto-Uchiyama [32] (véase Tabla 5.2) está basado en la dificultad del problema de factorización de números de la forma[4] $n = p^2 q$. Emplea algoritmos eficientes (tiempo polinomial) para resolver el logaritmo discreto en un subgrupo finito específico. Este esquema es semánticamente seguro bajo la asunción del problema del subgrupo-$p$. El tamaño de un texto cifrado

[4]El tamaño de $n = p^2 q$ puede ser el mismo que $n = pq$ si $n$ es suficientemente largo ($|n|$ es al menos 1024 bits).

| **Generación de llaves:** KeyGen$(p, q)$ | |
|---|---|
| **Entrada:** $p, q$ tal que $(\mid p \mid = \mid q \mid = k)$, $gcd(p, q-1) = 1$ y $gcd(q, p-1) = 1$ | |
| Calcular: $n = p^2 q$ | Elegir: $g \in \mathbb{Z}_n^*$ tal que el orden de $g_p = g^{p-1} \bmod p^2$ es $p$ |
| Calcular: $h = g^n \bmod n$ | |
| **Salida:** $pk = (n, g, h, k)$ | $sk = (p, q)$ |
| **Cifrado:** E$(m, pk)$ | **Descifrado:** D$(c, sk)$ |
| **Entrada:** $m \in \mathbb{Z}_p$ | **Entrada:** $c \in \mathbb{Z}_n$ |
| Elegir: $r \in \mathbb{Z}_n^*$<br>Calcular: $c = g^m h^r \bmod n$ | Calcular: $c_p = c^{p-1} \bmod p^2$<br>$m = \frac{L(c_p)}{L(g_p)} \bmod p$ |
| **Salida:** $c \in \mathbb{Z}_n$ | **Salida:** $m \in \mathbb{Z}_n$ |

**Tabla 5.2:** Criptosistema Okamoto-Uchiyama

es aproximadamente $3k$ bits más largo que el correspondiente texto plano de $k$ bits, no obstante el texto plano de $k$ bits es suficiente para el propósito de distribución de llaves secretas. En el mismo artículo, se presenta una modificación práctica en donde el algoritmo de cifrado puede ser acelerado si el tamaño $r$ en el proceso de cifrado es limitado al tamaño del mensaje, versión a la cual denominan versión aleatoria de tamaño limitado.

**Propiedades.** $E(m_0, r_0)E(m_1, r_1) \bmod n = E(m_0 + m_1, r_3)$ si $m_0 + m_1 < p$, donde, $E(m, r)$ significa un texto cifrado del texto plano $m$ aleatorizado por $r$ y $m_0 + m_1 < p$. Esta propiedad es utilizada para votación electrónica y otros protocolos criptográficos. Además, cualquiera puede cambiar un texto cifrado $c = E(m, r)$ en otro texto cifrado $c' = ch^{r'} \bmod n$ mientras se preserva el texto plano de $c$ ($c' = E(m, r'')$ por e.j.), y la relación entre $c$ y $c'$ es ocultada.

#### *5.1.7.4. Criptosistema Paillier*

El criptosistema de Paillier [33] (véase Tabla 5.3) es considerado el criptosistema homomórfico mayormente utilizado debido a su eficiencia y la sencillez de sus métodos de cifrado y descifrado. Su seguridad se basa en el problema de decisión del residuo compuesto. La particularidad de este criptosistema reside en utilizar un número al cuadrado $n^2$ como módulo, donde $n = pq$ y $p$ y $q$ son dos números primos grandes, por lo cual la expansión del mensaje será de $2n$ bits.

| Generación de llaves: KeyGen$(p,q)$ | | |
|---|---|---|
| **Entrada:** $p, q$ | | |
| Calcular: | $\phi(n) = pq$ por la función tociente de Euler | $\lambda(n) = lcm(p-1, q-1)$ por la función Carmichael |
| Elegir: | $g \in \mathbb{Z}^*_{n^2}$ tal que $gcd(L(g^\lambda \bmod n^2), n) = 1$, donde, $L(u) = \frac{u-1}{n}$ | |
| **Salida:** | $pk = (n, g)$ | $sk = (p, q)$ o el equivalente $\lambda(n)$ |
| **Cifrado:** E$(m, pk)$ | | **Descifrado:** D$(c, sk)$ |
| **Entrada:** | $m \in \mathbb{Z}_n$ | **Entrada:** $c \in \mathbb{Z}^2_n$ |
| Elegir: | $r \in \mathbb{Z}^*_n$ | Calcular: $m = \frac{L(c^\lambda \bmod n^2)}{L(g^\lambda \bmod n^2)} \bmod n$ |
| Calcular: | $c = g^m r^n \bmod n^2$ | |
| **Salida:** | $c \in \mathbb{Z}^2_n$ | **Salida:** $m \in \mathbb{Z}_n$ |

**Tabla 5.3:** Criptosistema Paillier

**Propiedades homomórficas.** Dos mensajes cifrados son homórficamente aditivos en $\mathbb{Z}_n$ bajo las identidades $\forall m_1, m_2 \in \mathbb{Z}_n$ y $k \in \mathbb{N}$. Las propiedades homomórficas de Paillier son las siguientes:

$$D(E(m_1)E(m_2) \bmod n^2) = m_1 + m_2 \bmod n$$

$$D(E(m)^k \bmod n^2) = km \bmod n$$

$$D(E(m_1)g^{m_2} \bmod n^2) = m_1 + m_2 \bmod n$$

$$\left.\begin{array}{l} D(E(m_1)^{m_2} \bmod n^2) \\ D(E(m_2)^{m_1} \bmod n^2) \end{array}\right\} = m_1 m_2 \bmod n$$

El criptosistema Paillier también soporta re-aleatorización de textos cifrados, cualquier texto cifrado puede ser públicamente cambiado en otro sin afectar el texto plano $\forall \in \mathbb{Z}_n$ y $r \in \mathbb{N}$: $D(E(m)r^n \bmod n^2) = m$ o $D(E(m)g^{nr} \bmod n^2) = m$.

#### *5.1.7.5. Criptosistema Damgard-Jurik (DJ)*

Damgard y Jurik proponen una generalización del criptosistema Paillier en donde el factor de expansión del mensaje es reducido mediante el ajuste de la longitud de bloque [34]. Es decir, el

mensaje de entrada es tomado de $Z_{n^s}$ y mapeado a un elemento $Z_{n^{s+1}}$. Esta generalización permite reducir $(\frac{s+1}{s})$ veces el factor de expansión del criptosistema Paillier. Esto viene con el aspecto negativo de incrementar el tamaño del módulo $n^{s+1}$, lo cual se ve reflejado en las operaciones aritméticas a realizar.

#### *5.1.7.6. Criptosistema Boneh-Goh-Nissim (BGN)*

Boneh-Goh-Nissim presentan un criptosistema [35] basado en grupos finitos de orden compuesto y mapas bilineales. Este criptosistema es el primero en soportar más de un tipo de operación homomórfica, además de un número arbitrario de adiciones soporta una única operación de multiplicación sobre datos cifrados. A pesar de dicha funcionalidad, este esquema limita el tamaño del mensaje a cifrar debido a la necesidad de calcular logaritmos discretos durante el descifrado. Dicho de otra forma, un mensaje $m$ limitado por $T$ $(0 \leq m \leq T)$, tomará un tiempo de $(O\sqrt{T})$ utilizando el método Pollard lambda para cálculo del logaritmo discreto.

#### *5.1.7.7. Selección de criptosistema homomórfico aditivo*

Los criptosistemas de llave pública dentro de los cuales recaen los homomórficos, basan su seguridad en algún tipo de problema matemático que es considerado como intratable bajo la selección apropiada de parámetros. Los distintos niveles de seguridad están en relación al tamaño de la llave pública que generalmente es de la forma $n = pq$, donde $p$ y $q$ son dos primos largos. Para un nivel de seguridad de $2^{80}$ se recomienda el uso de una llave pública de 1024 bits para la mayoría de los casos. En la Tabla 5.4 se sintetiza la revisión que se realizó acerca de distintos criptosistemas homomórficos aditivos probabilísticos, exceptuando al criptosistema BGN que además de un número arbitrario de adiciones soporta también una única operación homomórfica de multiplicación. En referencia a cada criptosistema se indica la operación modular a realizar con respecto a la llave pública $n$, el tamaño permitido para el mensaje de entrada así como el factor de expansión de éste.

De acuerdo a la revisión, destinar para nuestro propósito el criptosistema GM es una opción

| Criptosistema | Operación modular | Mensaje ($m$) | Expansión de $m$ |
|---|---|---|---|
| Goldwasser-Micali (GM) | [30] $n = pq$ | 1 | $n$ |
| Benaloh | [31] $n = pq$ | $m \in \mathbb{Z}_r$ | $\frac{n}{r}$ |
| Paillier | [33] $n^2$, donde $n = pq$ | $m \in \mathbb{Z}_n$ | 2 |
| Okamoto-Uchiyama (OU) | [32] $n = p^2q$ | $m \in \mathbb{Z}_p$ | 3 |
| Damgard-Jurik (DJ) | [34] $n^{s+1}$, donde $n = pq$ | $m \in \mathbb{Z}_{n^s}$ | $\frac{s+1}{s}$ |
| Boneh-Goh-Nissim (BGN) | [35] $n = pq$ | $0 \leq m \leq T$ y $T \leq q$ | $\frac{n}{T}$ |

**Tabla 5.4:** Criptosistemas homomórficos aditivos

inviable a aplicar debido principalmente al gran espacio de almacenamiento que ocuparía el índice, ya que en este criptosistema los mensajes son cifrados bit a bit expandiéndose en factor de $n$. Por otro lado, la complejidad de descifrado en los criptosistemas Benaloh y BGN debido a la necesidad del cálculo del logaritmo discreto hace que no sean adecuados para ser aplicados en el esquema. Además, aún cuando el descifrado pueda realizarse en un tiempo $O(\sqrt{r})$ mediante el algoritmo baby-step giant-step o el método Pollard lambda para el cálculo del logaritmo discreto el espacio de almacenamiento destinado para los valores precalculados sería muy demandante. Por otro lado, el hecho que en el criptosistema Paillier se utilice un número al cuadrado $n^2$ como módulo, la expansión del mensaje será de $2n$ bits que para términos de almacenamiento y operaciones sobre el índice no es muy adecuado. Por otra parte, el criptosistema DJ corresponde a una generalización del criptosistema Paillier que mediante el ajuste de una longitud de bloque $s$ permite reducir $(\frac{s+1}{s})$ veces el factor de expansión del mensaje, pero trayendo consigo un incremento en el tamaño del módulo $n^{s+1}$ En realidad el ajuste de bloque para nuestro caso no es necesario ya que los valores de ponderación están dentro del orden de los números enteros. Finalmente, aunque en el criptosistema OU los mensajes son limitados por $p$ y la aritmética es realizada módulo $n = p^2q$ lo cual corresponde a una expansión de tres veces el mensaje, consideramos que la limitante en el mensaje no afecta al esquema propuesto puesto que los valores de ponderación se encuentran dentro del rango de $p$. Por

todos estos fundamentos, se opta por utilizar el criptosistema Okamoto-Uchiyama para el esquema de indización y búsqueda.

## 5.2 Integridad

Dentro del paradigma del cómputo en la nube y el modelo Almacenamiento como Servicio (*Storage as a Service*) (StaaS) el abordar el desafío de garantizar ante los propietarios de datos que sus datos no han sido alterados inautorizadamente hace necesario contar con mecanismos que permitan verificar la integridad de los datos. El mecanismo más simple consiste en ejecutar una descarga para proceder a verificar los datos, pero esto sin duda alguna es algo impráctico pues además de incurrir en altos costos de comunicación a través de la red llega a convertirse en una tarea exhaustiva para el propietario de los datos. Por ello, el esquema de verificación que se implemente debe tratar de requerir recursos mínimos en términos de cómputo, comunicación y almacenamiento. Para dar solución al problema de verificación de integridad de datos, diversos esquemas y protocolos han sido propuestos bajo distintas técnicas criptográficas y fundamentos de seguridad. Los modelos *Provable Data Possesion* (PDP) y *Proofs of Retrievability* (POR) toman especial importancia pues han sido base para el desarrollo de esquemas de verificación de integridad de datos remotos, tal como se presentará en esta sección. Estos modelos permiten verificar la integridad de los datos remotos sin necesidad de recuperarlos del servidor, por tanto, pueden ser utilizados para habilitar un servicio de auditoría.

### *Provable Data Possesion (PDP)*

Un modelo PDP permite a un cliente que almacena datos en un servidor que no es de confianza verificar que el servidor posea los datos originales sin necesidad de recuperarlos. Tal como se ilustra en la Figura 5.3, un protocolo PDP verifica que un sitio externo de almacenamiento conserve un

archivo $F$ de $n$ bloques de datos. En la fase de inicio, el cliente $C$ preprocesa $F$, genera una pieza de metadatos $m$ que almacena localmente, transmite $F$ hacia el servidor $S$ y elimina la copia local. La correcta posesión de datos es validada a través de un protocolo de desafío-respuesta llevado a cabo entre cliente y servidor, bajo este protocolo el cliente solicitará al servidor calcular una función sobre el archivo almacenado, y utilizando los metadatos locales el cliente verifica la prueba otorgada por el servidor.

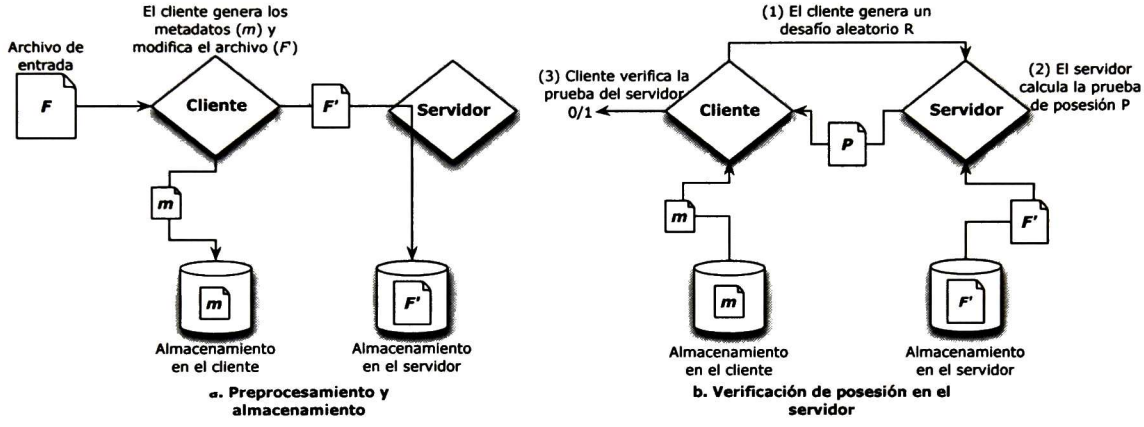



**Figura 5.3:** Protocolo PDP

### *Proofs of Retrievability (POR)*

El modelo POR habilita a un usuario no solamente el verificar que un demostrador posea un archivo $F$ o un objeto de datos sino también a recuperar el archivo en su totalidad. En este modelo, el verificador calcula y almacena un valor hash $r = h_k(F)$ junto con una llave aleatoria $k$. Para comprobar que el servidor posea $F$, el verificador libera $k$ y solicita al demostrador calcular y retornar $r$. La Figura 5.4 representa el funcionamiento de un esquema POR, en donde en primer instancia un algoritmo de codificado transforma un archivo $F$ en un archivo codificado $F'$ para ser almacenado por el demostrador. Un algoritmo de generación de llaves produce un llave $k$ almacenada por el verificador y utilizada en el codificado. El verificador ejecuta un protocolo de desafío-respuesta con

el demostrador para comprobar que se recupera $F$.

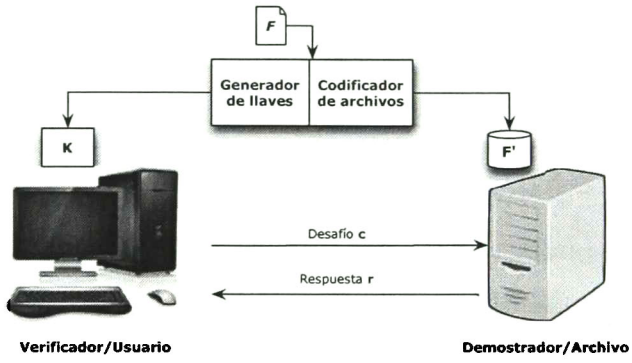



**Figura 5.4:** Protocolo POR

Por otra parte, acorde al tipo de entidad que esté habilitada para llevar a cabo la verificación de integridad de los datos se distinguen dos tipos de protocolos de auditoría:

- Auditoría privada. Permite solamente al propietario de los datos (DO) llevar a cabo la auditoría de los datos.
- Auditoría pública. Este tipo de protocolo es llevado a cabo por un auditor externo ( *Third Party Auditor*) (TPA) al proveedor del servicio de almacenamiento. Un TPA fungirá como entidad de confianza para llevar a cabo la verificación.

### 5.2.1 Análisis de esquemas de verificación de integridad de datos

Filho *et al* [39] proponen un protocolo para verificar la integridad de datos utilizando funciones hash homomórficas basadas en RSA. Dicho esquema es homomórfico bajo la siguiente propiedad: $H(d+d') \equiv b^{d+d'} \equiv b^d b^{d'} \equiv H(d)H(d')(mod\, n)$, donde, $n$ es un módulo RSA producto de dos números primos grandes $p$ y $q$, $\phi(n) = (p-1)(q-1)$ es de orden $(\mathbb{Z}/n\mathbb{Z})^*$, y $b$ es un entero aleatorio. El valor hash homomórfico de un dato $d$ es calculado a partir de $H(d) = b^d mod n$. Definido lo anterior, el protocolo propuesto se expresa de la siguiente manera, durante la fase de inicialización el DO calcula

el valor hash $h(d) = d \, mod \, \phi(n)$, envía los datos $d$ hacia el servidor y resguarda $h(d)$. Durante el desafío de posesión de datos, el DO eligirá un entero aleatorio $1 < b < n - 1$ el cual enviará al servidor, éste calcula y retorna el valor $r = b^d \, mod \, n$. Finalmente, el DO calcula $r' = b^{h(d)} \, mod \, n$ y verifica si $r = r'$, si esto se cumple se comprueba que el servidor preserva $d$ de forma correcta. La desventaja de este esquema consiste en la cantidad de cómputo que el servidor tiene que realizar durante la fase de respuesta al desafío, esto en relación a la necesidad de exponenciar $d$ que equivale a los datos enteros.

Con el propósito de reducir la complejidad de cómputo en el servidor, Sebe *et al.* [40] proponen dividir un archivo $m$ en bloques de $l$ bits. Por cada uno de los bloques $m_i$ el DO calculará y almacenará un valor hash homomórfico $M_i = m_i \, mod \, \phi(N)$ basado en RSA. Durante el protocolo de desafío-respuesta el DO genera una semilla aleatoria $S$ y un elemento aleatorio $a \in \mathbb{Z}_N$, enviando $(a, S)$ como desafío al servidor. Al recibir el desafío, el servidor genera $n$ valores pseudoaleatorios $c_i \in [1, 2^t]$, para $i = 1$ hasta $n$, utilizando un *(PseudoRandom Number Generator)* (PRNG) alimentado por $S$. Calculará $r = \sum_{i=1}^{n} c_i m_i$ y $R = a^r mod N$, retornando $R$ como prueba. De igual forma, el DO regenera los $n$ valores pseudoaleatorios $c_i \in [1, 2^t]$, calcula $r' = \sum_{i=1}^{n} c_i m_i$ y $R' = a^{r'} \, mod \, N$. La posesión de los datos remotos es válida si $R = R'$. Aunque esta propuesta no requiere exponenciar los datos enteros, se incurre en una sobrecarga de almacenamiento del lado del DO debido al almacenamiento de los valores hash, en total $N$ bits por bloque, donde $N$ es el tamaño del módulo RSA utilizado.

Por otra parte, Yamamoto *et al.* [41] desarrollan un algoritmo para verificar la integridad de datos a través de una verificación por lotes y el uso de una función hash homomórfica para etiquetar bloques de datos. Sean $(x_1, x_2, \ldots, x_d)$ un conjunto de bloques de datos, $g \in \mathbb{Z}/N\mathbb{Z}$, el DO calculará una etiqueta $y_i = g^{x_i} (mod \, N)$ para cada bloque. Dado $k$ el parámetro de seguridad del sistema, $s$ un número primo tal que $\mid s \mid > k$, el servidor elige aleatoriamente $e \in [0, s - 1]$ y calcula $e_i = e^{i-1} (mod \, s)$ para cada $i = 1, 2, \ldots, d$, y $\overline{x} = \sum_i e_i x_i$ en $\mathbb{Z}$. Para la verificación el DO calculará

$\overline{y} = \prod_i y_i^{e_i}$ y validará $g^{\overline{x}} = \overline{y}(mod\, N)$. Como se observa, si $h$ es una función homomórfica verificar un conjunto $y_1 = h(x_1), y_2 = h(x_2), \ldots, y_d = h(x_d)$ requiere solamente validar $y = h(x)$, donde $x$ e $y$ son combinaciones lineales de $x_i$ e $y_i$ con un valor aleatorio elegido. A pesar de que esta característica puede mejorar la eficiencia de la verificación, este esquema no puede ser aplicado a un tipo de verificación pública ya que el protocolo requiere almacenar las etiquetas del lado del DO.

Ateniese *et al* [42] fueron los primeros en definir formalmente un protocolo PDP y en considerar una verificación pública de datos remotos. Ellos introducen el concepto de etiquetas homomórficas verificables (HVTs por sus siglas en inglés) basadas en RSA como bloque de construcción de sus esquemas: *Sampling-PDP* (S-PDP) y *Efficient-PDP* (E-PDP). El primero de ellos provee una fuerte garantía de posesión de datos, mientras que el segundo ofrece eficiencia al costo de debilitar la garantía de posesión de los datos. El protocolo básico S-PDP opera de la siguiente manera:

- Generación de llaves. Sea $f$ una función pseudoaleatoria, $\pi$ una permutación pseudoaleatoria y $H$ una función hash, el DO genera una llave pública $pk = (N, g)$ y una llave secreta $sk = (e, d, v)$ de tal manera que $ed \equiv 1(mod\, p'q')$, $e$ es un número primo largo tal que $e > \lambda$ y $d > \lambda$, $g$ es el generador de $QR_N$ y $v \leftarrow \{0,1\}^K$.

- Etiquetación de bloques. El DO calcula $W_i = v \parallel i$ y etiqueta cada bloque, $T_{i,m} = (h(W_i)g^m)^d mod\, N$.

- Generación de prueba. Para cada bloque $1 \leq j \leq c$ el auditor calcula un índice $i_j = \pi_{k_1}(j)$ y un coeficiente $a_j = f_{k_2}(j)$. Envía $(c, k_1, k_2, g_s)$ como desafío al servidor. El servidor calculará $T = \prod_{j=1}^{c} T_{i,j}^{a_j}$ y $p = H(g_s^{\sum_{j=1}^{c} a_j m_{ij}} mod\, N)$, y retorna como prueba $V = (T, p)$.

- Verificación de prueba. Para $1 \leq j \leq c$ el auditor calcula $ij = \pi_{k_1}(j)$, $W_{ij} = v \parallel ij$, y $a_j = f_{k_2}(j)$, $\tau = \frac{T^e}{h(W_{ij})^{a_j}} mod\, N$. La correcta posesión de datos se valida si $H(\tau^s mod\, N) = p$.

Estos esquemas permiten al servidor probar la posesión de bloques selectos de un archivo $F$

mediante pruebas probabilísticas. El muestreo aleatorio ayuda a reducir la carga de trabajo en el servidor. Las etiquetas homomórficas no requieren ser almacenadas por el auditor sino que éstas son almacenadas en el servidor junto con los datos, además, el número de auditorías permitidas para un mismo dato no se encuentra limitado, por tanto, bajo este esquema un TPA puede ser implementado. Sin embargo, a pesar de las ventajas, deben ser considerados los siguientes inconvenientes: el tamaño de las etiquetas RSA es largo y el costo de cómputo para generarlas también es alto.

Shah *et al.* [43] proponen una solución que preserva privacidad y habilita el uso de un auditor para verificar la integridad de datos. Durante la fase de inicialización, el servidor almacena los datos cifrados $E_K(D)$ y la llave $K$. Así también, hace público al auditor el valor hash $H(E_K(D))$ y $g^K mod\, p$ que representa la forma enmascarada de la llave, $g$ es el generador de $Z_p^*$, y $p$ un número primo largo. El auditor precalcula una lista de pares de desafío-respuesta cada uno de los cuales consta de un número aleatorio $R_i$ y un HMAC que puede calcularse sólo conociendo $R_i$ y los datos cifrados enteros. En la verificación, el auditor selecciona y envía un desafío al servidor y espera por la respuesta. El inconveniente del esquema es que el número de veces en que un dato puede ser auditado está limitado por la cantidad de HMACs fijados de antemano puesto que no pueden ser reutilizados con el fin de evitar malos usos por parte del servidor. Es importante también considerar los costos de almacenamiento del lado del auditor al mantener distintos HMACs por cada archivo.

Juels y Kaliski [44] definen un esquema POR para archivos estáticos de gran tamaño, cada archivo es dividido en bloques de datos y cada bloque utiliza códigos de corrección de errores para garantizar no sólo la correcta posesión de los datos, sino también su recuperabilidad. En el esquema que proponen, un archivo es cifrado y un conjunto de bloques de verificación denominados centinelas se incrustan en él. Durante la auditoría, el verificador desafía al servidor especificando las posiciones de un subconjunto de centinelas y solicitándole devuelva los valores centinelas asociados. El inconveniente de este esquema es que el número de veces en que los datos pueden ser auditados

se encuentra limitado en relación al número de centinelas que se incrustan durante la fase de preprocesamiento de los datos.

Por otra parte, Schacham y Waters [45] presentan un esquema POR de verificación privada con base en (*Pseudorandom functions*) (PRFs) y MACs, y otro esquema POR que es públicamente verificable utilizando firmas BLS [46]. En orden de compensar sobrecarga de almacenamiento y longitud de respuesta proponen dividir un archivo en bloques de datos, y éstos a su vez en sectores. Para el esquema basado en firmas BLS, sea $e : G \times G \rightarrow G_T$ un mapa bilineal (véase Definición 9) con $G's$ grupos con soporte en $\mathbb{Z}_p$, la llave privada es $x \in \mathbb{Z}_p$ y la pública es $v = g^x \in G$, junto con un generador $u \in G$. Cada bloque $i$ es firmado $\sigma_i = [H(i)u^{m_i}]^x$. Al recibir la consulta $Q = (i, v_i)$ el servidor calcula $\sigma \leftarrow \prod_{(i,v_i)\in Q} v_i m_i$ y $\mu \leftarrow \sum_{(i,v_i)\in Q} v_i m_i$. El auditor verifica mediante la ecuación $e(\sigma, g) = e(\prod_{(i,v_i)\in Q} H(i)^{v_i} u^{\mu}, v)$. Este esquema soporta un número ilimitado de auditorías, además las etiquetas homomórficas basadas en BLS (170 bits) son más cortas que las etiquetas homomórficas RSA. El inconveniente del método consiste en que puesto que el servidor requiere enviar hacia el auditor las combinaciones lineales de los bloques de datos que están siendo auditados es posible que el auditor deduzca información, por tanto, la privacidad no es preservada.

**Definición 8.** ***Mapa Bilineal*:**

*Sean $G_1, G_2$ y $G_T$ tres grupos cíclicos multiplicativos de orden primo $p$, $g_1$ el generador de $G_1$ y $g_2$ el generador de $G_2$, tenemos que $e : G_1 \times G_2 \rightarrow G_T$ es un mapa bilineal con las siguientes propiedades:*

1. *Computable. Existe un algoritmo eficiente para calcular $e$.*
2. *Bilineal. Para todo $u \in G_1, v \in G_2$ y $a, b \in \mathbb{Z}_p$ tenemos que $e(u^a, v^b) = e(u, v)^{ab}$.*
3. *No degenerado. $e(g_1, g_2) \neq 1$.*

Dado el problema de privacidad presente en el esquema basado en BLS [45], Wang *et al.* [47]

proponen un enfoque extendido combinando autenticadores homomórficos con enmascaramiento aleatorio. En este nuevo protocolo, el servidor no requiere enviar directamente al auditor la combinación lineal de los bloques de datos. En su lugar, el servidor utiliza un número aleatorio el cual combina con los bloques de datos elegidos. Posteriormente, el servidor cifra el número aleatorio y lo envía al auditor junto con los bloques de datos elegidos.

### 5.2.2 Discusión

A lo largo de este sección han sido descritos una serie de distintos esquemas de verificación de integridad de datos, en la Tabla 5.5 se presenta un resumen sobre el análisis realizado.

Dicho servicio de auditoría debe cumplir en la mayor medida con los siguientes requerimientos:

- Privacidad de los datos. Debe asegurarse que el auditor no sea capaz de extraer información a partir de los datos recolectados durante el proceso de auditoría.
- Número ilimitado de auditorías. El esquema de auditoría debe ser flexible para asegurar que el auditor disponga de un número ilimitado de distintas formas de realizar verificaciones sobre un mismo dato.
- Verificación pública. Cualquiera que utilice el protocolo de verificación, no solamente el propietario de los datos, puede verificar la integridad de los datos almacenados.

Como podemos observar existen ciertos esquemas que no cumplen con el requerimiento de privacidad, lo cual se debe a que su forma de auditoría demanda una combinación lineal de bloques de muestreo para ser expuesta al auditor con lo cual información podría llegar a ser revelada. Por otra parte los esquemas basados en el método MAC no permiten un número ilimitado de auditorías debido a que el número de veces en que un dato puede ser auditado está limitado por la cantidad de llaves MAC que son fijadas de antemano. Los esquemas que no cumplen con el requerimiento de auditoría pública es debido a que el propio diseño de dichos esquemas demanda que los metadatos

| | [39] | [40] | [41] | [42] | [43] | [44] | [45] | [45] | [47] |
|---|---|---|---|---|---|---|---|---|---|
| **Método** | RSA-HHV | RSA-HHV | RSA-HT | HVT | MAC | MAC | MAC | BLS | BLS |
| **Consultas ilimitadas** | √ | √ | √ | √ | × | × | × | √ | √ |
| **Auditoría pública** | × | × | × | √ | √ | × | √ | √ | √ |
| **Privacidad** | √ | √ | × | × | √ | √ | × | × | √ |

Métodos: **RSA-HT (RSA-based homomorphic tag), RSA-HHV (RSA-based homomorphic hash value), HVT, MAC, BLS.**

**Tabla 5.5:** Análisis de distintos esquemas de verificación de integridad de datos

para verificación sean almacenados por el propietario de datos incurriendo en una sobrecarga de almacenamiento del lado del cliente. Como consecuencia de lo anterior, para el desarrollo del servicio de integridad de la arquitectura propuesta se opta por el esquema de Wang *et al.* [47] ya que cumple con los requerimientos deseables para el despliegue de un *Third Party Auditor* (TPA). En la siguiente sección se describe con más detalle el esquema elegido.

### 5.2.3 Esquema de integridad

El esquema de auditoría pública propuesto por Wang *et al.* [47] retomado para la arquitectura propuesta, utiliza como método de firma al esquema de firmas BLS [46], que son firmas cortas de 170 bits basadas en mapas bilineales (véase Definición 9) sobre curvas elípticas. Este esquema se basa en un muestreo aleatorio de bloques de datos y está constituido por cinco algoritmos que son descritos a continuación:

1. **Generación de llaves (KeyGen)**. Eligir aleatoriamente $x \leftarrow \mathbb{Z}_p, u \leftarrow \mathbb{G}_1$, calcular $v = g_2^x$, $w = u^x$, generando como resultado una llave secreta $sk = (x)$ y los parámetros públicos $pk = (v, w, g, u)$. Los parámetros $pk$ deben ser enviados hacia el TPA.

2. **Generación de firmas (SigGen)**. Dado un archivo $F = (m_1, ..., m_n)$, donde $n \in \mathbb{Z}_p$ y es de orden primo $p$, obtener la firma $\sigma_i$ para cada bloque de datos $m_i$: $\sigma_i \leftarrow (H(i)u^{m_i})^x$, siendo

$H$ una función hash. Se denota $\phi = \{\sigma_i\}_{1 \leq i \leq n}$ como el conjunto de firmas, éste y el archivo $F$ son enviados hacia el servidor.

3. **Desafío (GenChal)**. El TPA elige un subconjunto de elementos aleatorios $I = \{s_1, ..., s_c\}$ del conjunto [1,n]. Para cada elemento $i \in I$ el TPA elige un valor aleatorio $v_i$, con ello conforma el desafío $chal = \{(i, v_i)\}$ que se envía al servidor.

4. **Generación de prueba (GenProof)**. El servidor elige un elemento aleatorio $r \leftarrow \mathbb{Z}_p$ y calcula $R = (w)^r$. Calcula la combinación lineal de los bloques de muestreo $\mu' = \sum_{i \in I} v_i m_i$ y su forma enmascarada $\mu = \mu' + rh(R) \in \mathbb{Z}_p$, y una firma agregada $\sigma = \prod_{i \in I} \sigma_i^{v_i}$. Retorna al TPA $\{\mu, \sigma, R\}$.

5. **Verificación (VerifyProof)**. El TPA verifica la respuesta mediante : $e(\sigma(R^{h(R)}), g_2) = e(\prod_{i=s_1}^{s_c} H(i)^{v_i} u^{\mu}, v)$.

### *Fases del servicio de integridad*

La aplicación del esquema [47] a la arquitectura propuesta se dividió en las siguientes dos fases:

1. **Fase de inicialización.** En esta fase el DO a través del **Módulo Generador de Llaves** ejecuta el algoritmo KeyGen y transfiere los parámetros públicos $pk$ hacia el TPA. Dada la colección de documentos cifrados $\widetilde{C} = \{[\widetilde{K}_1, \widetilde{D}_1], ..., [\widetilde{K}_n, \widetilde{D}_n]\}$ que el DO posee, para cada $\widetilde{D} \in \widetilde{C}$ procede a obtener el conjunto de firmas de cada archivo acorde al algoritmo SigGen

2. **Fase de auditoría.** En esta segunda fase, el TPA a través del algoritmo GenChal y la programación de auditoría definida por el DO, emitirá un desafío ante el CSP por cada archivo a auditar. El CSP atiende cada uno de los desafíos utilizando el algoritmo GenProof y retorna la respuesta al TPA, quien procede a validar mediante el algoritmo VerifyProof. El DO a través del **Módulo de Reporte de Resultados** podrá consultar los resultados obtenidos de la auditoría de sus archivos.

La Figura 5.5 ilustra la interacción entre las entidades de la arquitectura durante las dos fases mencionadas.

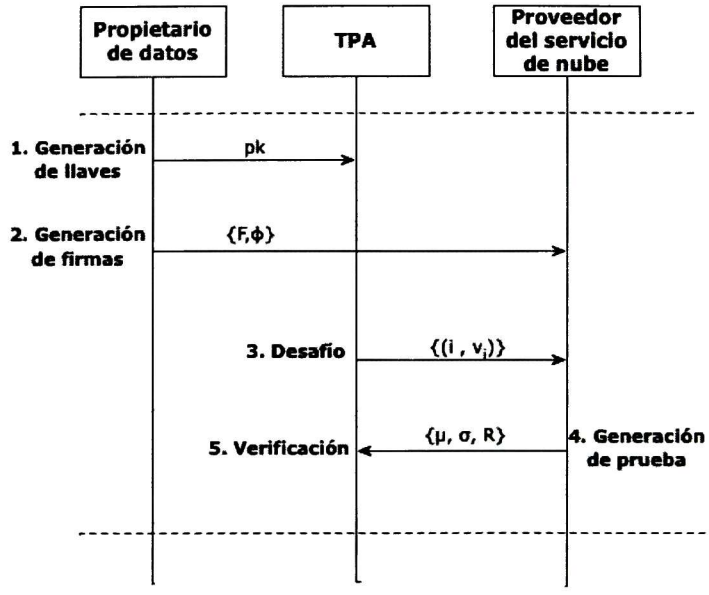



Figura 5.5: Fases del servicio de integridad

# 6

# Resultados

*En este capítulo se presentan los resultados obtenidos durante la evaluación de la arquitectura de servicios de seguridad propuesta. Particularmente, se describen los detalles de implementación, el conjunto de datos y la plataforma de pruebas utilizada, la cual incluye un equipo cliente y una infraestructura de nube. Los experimentos realizados se enfocaron en evaluar el rendimiento de cada uno de los esquemas de seguridad propuestos.*

## 6.1 Herramientas de software utilizadas

La propuesta de solución descrita en el Capítulo 4 y el Capítulo 5 fue implementada en lenguaje de programación Java. Para efectos de experimentación, el conjunto de datos de prueba que se utilizó para hacer el análisis y evaluación de resultados se limitó al soporte de ciertos formatos de archivos (TXT, Word y PDF) con contenido en idioma inglés.

Las herramientas de desarrollo que fueron utilizadas son las siguientes:

- **MySQL** [1].

  Es un sistema de código abierto para la gestión de bases de datos. Se utilizó para almacenar y manipular la distinta información (información personal y de sesión de los usuarios, por e.j.) generada en la arquitectura.

- **Java Pairing Based Cryptography Library (jPBC)** [2].

  Es una biblioteca diseñada para implementar criptosistemas basados en emparejamientos bilineales. Mediante ésta, se implementó el esquema de auditoría pública de datos [47] y el esquema de firmas cortas [46].

- **Bouncy Castle Crypto APIs** [3].

  Consiste en una colección de APIs para criptografía. Se utilizó en la PKI para desarrollar las funcionalidades de emisión y revocación de certificados digitales estándar X.509.

- **Apache PDFBox Java PDF Library** [4].

  Es una biblioteca de código abierto que permite la creación de nuevos documentos PDF, y manipular y extraer el contenido de documentos PDF existentes. Fue utilizada para la extracción de texto de documentos PDF.

- **Apache POI - HWPF Java API** [5].

  Es una API para manipular (leer y escribir) archivos en formato MS Word. Fue utilizada para la extracción de texto de documentos MS Word.

- **Ciphertext-Policy Attribute-Based Encryption** [6].

  Es una biblioteca que implementa el esquema CP-ABE.

[1] http://www.mysql.com/
[2] http://gas.dia.unisa.it/projects/jpbc/
[3] http://www.bouncycastle.org/java.html
[4] http://pdfbox.apache.org/
[5] http://poi.apache.org/hwpf/index.html
[6] http://hms.isi.jhu.edu/acsc/cpabe/

## 6.2 Escenario de pruebas

En esta sección se describe el escenario de pruebas utilizado durante la experimentación realizada y que se presenta a lo largo de este capítulo. A partir de este punto, es importante mencionar que cada uno de los experimentos fue ejecutado 31 veces de acuerdo al teorema del límite central [7].

### 6.2.1 Plataforma de hardware

La experimentación realizada consistió en evaluar dos tipos de procesos: los que son llevados a cabo por el propietario de datos y/o usuarios (DO/U) y los que son llevados a cabo por el proveedor de servicios de nube (CSP). Por tanto, para la plataforma de hardware y en relación a los dos tipos de procesos se considera un equipo cliente y una infraestructura de nube, respectivamente.

#### *Equipo cliente*

El equipo cliente que se utilizó fue una computadora portátil con las siguientes características:

- Procesador: 3rd Gen Intel® CoreTM i7-3632QM 2.20 GHz.
- Memoria RAM: 8 GB DDR3 SDRAM.
- Disco duro: 1 TB (5400 rpm).
- Arquitectura: 64 bit.
- Sistema operativo: Windows 8.

[7] Si el tamaño muestral es lo bastante grande (mayor a 30) y se calcula su promedio, dicho promedio seguirá una distribución normal.

### *Infraestructura de nube*

La infraestructura de nube sobre la cual se realizaron los experimentos correspondió a un modelo de despliegue de una nube privada, sus características particulares son ilustradas en la Figura 6.1 y descritas a continuación:

- Sistema operativo de nube: OpenStack [8] versión Essex o 2012.1.
- Módulos de OpenStack utilizados:
  - Keystone. Provee autenticación y autorización para todos los servicios de OpenStack.
  - Glance. Provee un catálogo y repositorios de discos de imágenes virtuales utilizadas por Nova.
  - Nova. Proporciona y administra grandes redes de máquinas virtuales e interactúa con el hypervisor [9].
  - Horizon. Permite a los usuarios interactuar con los servicios de OpenStack a través de una interfaz web, para lanzar una instancia, asignar una dirección IP, configurar controles de acceso, entre otros.
- Sistema operativo: Ubuntu Server 12.04.2 LTS basado en Linux.
- Tipos de nodos:
  - Nodo controlador. En este nodo se ejecutan los servicios de control, tales como base de datos, colas de mensajes, y la API de servicios de Keystone, Glance y Nova.
  - Nodo cómputo. En este tipo de nodo se ejecuta el hypervisor donde las máquinas virtuales residen.
- Hypervisor: *Kernel Based Virtual Machine* (KVM).

[8] http://www.openstack.org/
[9] Monitor de Máquina Virtual (VMM por sus siglas en inglés).

- Redes: dos redes físicas (interna y externa) y una red virtual para las instancias de las máquinas virtuales.

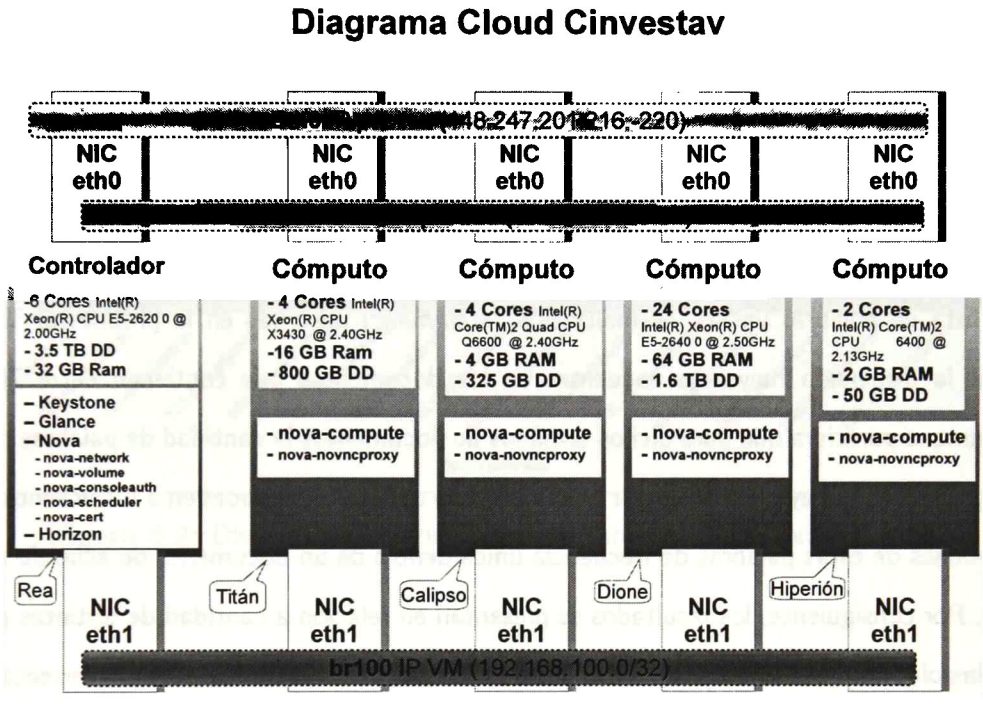



**Figura 6.1:** Características de la nube privada

## 6.2.2 Conjunto de datos de prueba

Para la realización de las pruebas funcionales y experimentos de evaluación del esquema propuesto fue necesario disponer de un conjunto de datos que tuviera un volumen considerable con el fin de contar con un escenario más cercano a lo que es el enfoque del uso del almacenamiento en la nube, es decir, un almacenamiento generalmente masivo. El conjunto de datos utilizado fue tomado de la base de datos de *Request for Comments* (RFC) [10] integrada por documentos técnicos y organizacionales

[10]http://www.rfc-editor.org/download.html

acerca del Internet y por especificaciones técnicas y documentos de política producidos por el *Internet Engineering Task Force* (IETF), ésta contiene alrededor de 6889 archivos de texto plano que hacen un volumen total de 341 MB.

Para los experimentos se seleccionó únicamente 1000 archivos del conjunto total, el tamaño de éstos oscila entre 15-330 KB con un volumen total de 118 MB. En la gráfica de la Figura 6.2 se ilustra la dispersión de los documentos de acuerdo a la cantidad de palabras distintas encontradas en cada uno de ellos; los resultados obtenidos consideran la aplicación de las técnicas de preprocesamiento de texto (lista de palabras vacías, normalización, *stemming*) descritas en la propuesta. Tal como se observa, la dispersión mayor se encuentra en los documentos que contienen entre 700-1000 palabras, aunque pareciera que para dichos tamaños de documentos la cantidad de palabras distintas encontradas es mucha, hay que considerar que la mayoría de éstas corresponden a tecnicismos, siendo también muchas de estas palabras de frecuencia única dentro de un documento de acuerdo a lo que se observó. Por consiguiente, los resultados se presentan en relación a cantidad de distintas palabras dentro de la colección y no en relación a tamaños de documentos, ya que se desea hacer notar que el tamaño de la colección de documentos no determina la cantidad de palabras distintas que contendrá la colección.

## 6.3 Evaluación del esquema de control de acceso y confidencialidad

El esquema de control de acceso y confidencialidad propuesto y descrito en la Sección 4.3.4 tiene como fundamento el uso de dos primitivas criptográficas: un esquema de cifrado simétrico (SE) para cifrar los documentos y un esquema de cifrado basado en atributos (ABE) para cifrar las llaves simétricas.

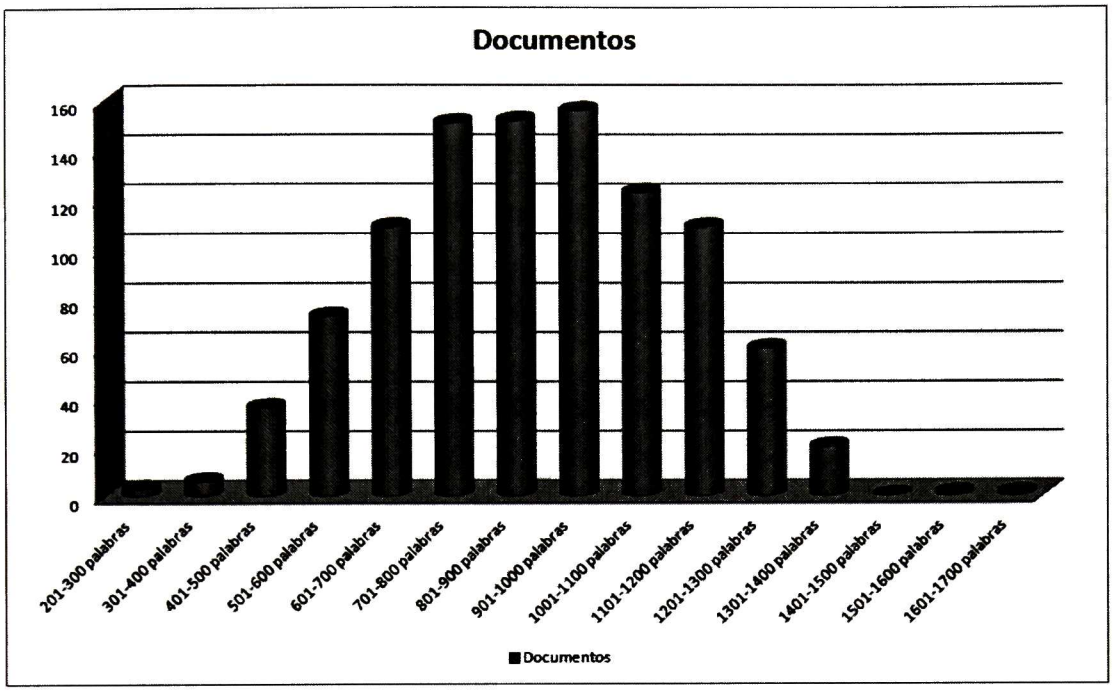



**Figura 6.2:** Dispersión de documentos por cantidad de palabras distintas

Entre los esquemas SE existen opciones como AES, Triple DES, IDEA, entre otros; particularmente se utilizó AES por ser un estándar bien conocido y uno de los más populares actualmente. Los esquemas SE poseen la característica de ser muy eficientes para cifrar grandes volúmenes de datos, por ello cualquier otra opción podría haber sido utilizada. Se eligió un nivel de seguridad de $2^{80}$ bits para AES, por tanto, las llaves simétricas que se utilizaron fueron de 128 bits.

Particularmente, en esta sección se realza el interés de evalúar el desempeño del esquema ABE por ser más demandante en sus operaciones que un esquema SE, el esquema ABE que se utilizó fue CP-ABE. Primeramente se evaluó la operación de generación de llaves secretas de los usuarios, operación que es ejecutada por el DO. En la Tabla 6.1 se presentan los resultados de ésta, tomando como métricas el tamaño y tiempo de generación de la llave secreta. En el experimento, se fue variando el número de atributos de usuario, parámetro de entrada del algoritmo de generación de

llaves. De acuerdo a lo observado, el tamaño y tiempo de generación de la llave se encuentra en relación al número de atributos proporcionados.

| # de atributos | Tamaño de llave (KB) | Tiempo (s) |
|---|---|---|
| 3 | 0.95 | 0.574 |
| 6 | 1.77 | 0.917 |
| 9 | 2.58 | 1.290 |
| 12 | 3.40 | 1.642 |
| 15 | 4.20 | 1.990 |

**Tabla 6.1:** Generación de llaves secretas en CP-ABE

Posteriormente, se evaluaron las operaciones de cifrado y descifrado en CP-ABE, los resultados se presentan en la Tabla 6.2. En la primer columna de la tabla se indica el número de nodos hojas del árbol de acceso, es decir, la política de acceso definida para un documento, y en la segunda columna el tiempo de cifrado para cifrar un mensaje, donde, para nuestro esquema el mensaje corresponde a una llave simétrica de 128 bits. La tercer columna refleja el tamaño de salida del texto cifrado, y la cuarta el tiempo medido en segundos de la operación de descifrado.

| # de nodos | Tiempo de cifrado (s) | Tamaño texto cifrado (KB) | Tiempo de descifrado (s) |
|---|---|---|---|
| 3 | 0.903 | 1.15 | 0.664 |
| 6 | 1.388 | 1.99 | 0.919 |
| 9 | 1.862 | 2.83 | 1.176 |
| 12 | 2.323 | 3.67 | 1.430 |
| 15 | 2.765 | 4.49 | 1.695 |

**Tabla 6.2:** Cifrado y descifrado en CP-ABE

Con base a estos resultados, se observa que los tiempos de cifrado y descifrado, y el tamaño del texto cifrado están en función del número de nodos especificados en la política de acceso. En general, también observamos que los tiempos de descifrado son mucho más pequeños que los tiempos de cifrado para un mensaje dado. Cabe mencionar que el cifrado se realiza una única vez por el DO, mientras que la operación de descifrado se realiza cada que un usuario descarga un documento del servicio de nube.

## 6.4 Evaluación del esquema de indización

En esta sección se presenta la evaluación del esquema de indización propuesto en la Sección 5.1.5, proceso que es llevado a cabo por el propietario de los datos e involucra los siguientes pasos: la extracción y preprocesamiento del texto de cada uno de los documentos a indizar, los cálculos de ponderación palabra-documento (TF-IDF) y la construcción del índice de búsqueda seguro. Particularmente, nos enfocaremos en evaluar el rendimiento de la parte que implica las operaciones criptográficas, es decir, la construcción del índice de búsqueda seguro que se constituye por vocabulario y listas invertidas.

### 6.4.1 Vocabulario

La seguridad en el vocabulario de palabras es aplicada mediante el uso una función pseudoaletoria y una llave secreta, de forma implementada se calcula el valor HMAC de cada palabra perteneciente al vocabulario. La seguridad del algoritmo HMAC depende de la seguridad de la función hash subyacente, del tamaño de su salida hash y de la fuerza de seguridad de la llave. Específicamente se utilizó una llave de 128 bits y una función SHA-1, la cual otorga una salida de 160 bits. En la gráfica de la Figura 6.3 se presentan los tiempos de construcción del vocabulario medidos en términos de milisegundos (ms) y en la gráfica de la Figura 6.4 los espacios de almacenamiento medidos en términos de Kbytes (KB) para distintos valores dentro de un rango de 16,948-90,795 palabras dentro del vocabulario obtenidas del conjunto de datos prueba a partir de 100-1000 documentos bajo incrementos de 100. Con base a estos resultados se puede establecer que esta parte del proceso de indización demanda mínimos requerimientos de tiempo consumido para la construcción del vocabulario.

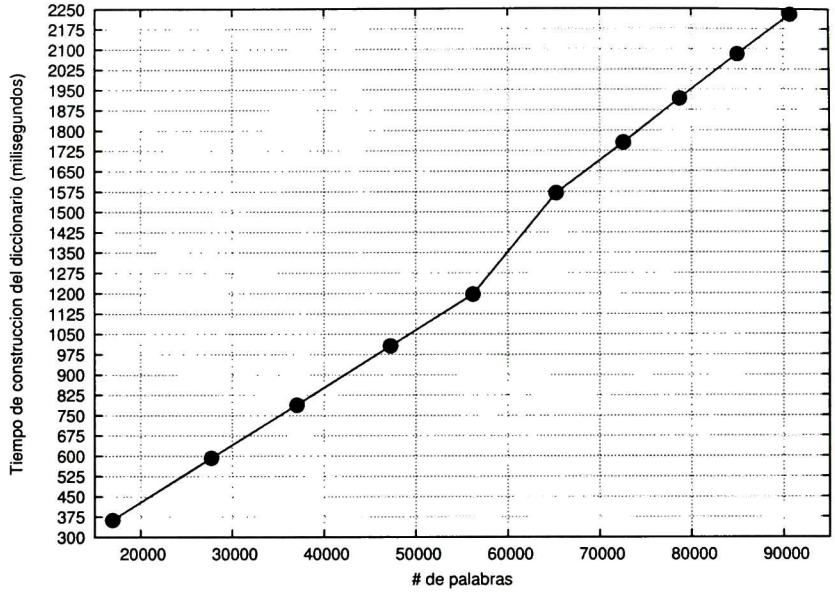



**Figura 6.3:** Tiempo de construcción del vocabulario de palabras

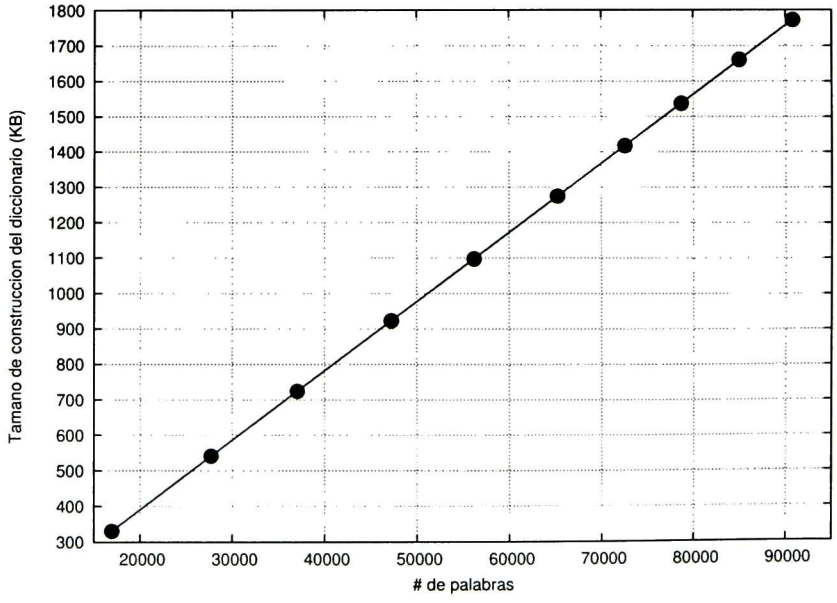



**Figura 6.4:** Espacio de almacenamiento del vocabulario

### 6.4.2 Listas invertidas

La seguridad para las listas invertidas consiste en cifrar cada una de las entradas del índice correspondientes a las puntuaciones de relevancia palabra-documento mediante el criptosistema homomórfico aditivo, concretamente se utilizó el criptosistema Okamoto-Uchiyama con una llave de 1024 bits. En la gráfica de la Figura 6.5 se presenta el costo de tiempo para la construcción del índice invertido, a manera de optimización se precalculó un conjunto de valores que determinan la aleatoriedad del cifrado dentro del criptosistema, para nuestro caso un total de $2^{16}$, aunque esta cantidad podría ser determinada a conveniencia.

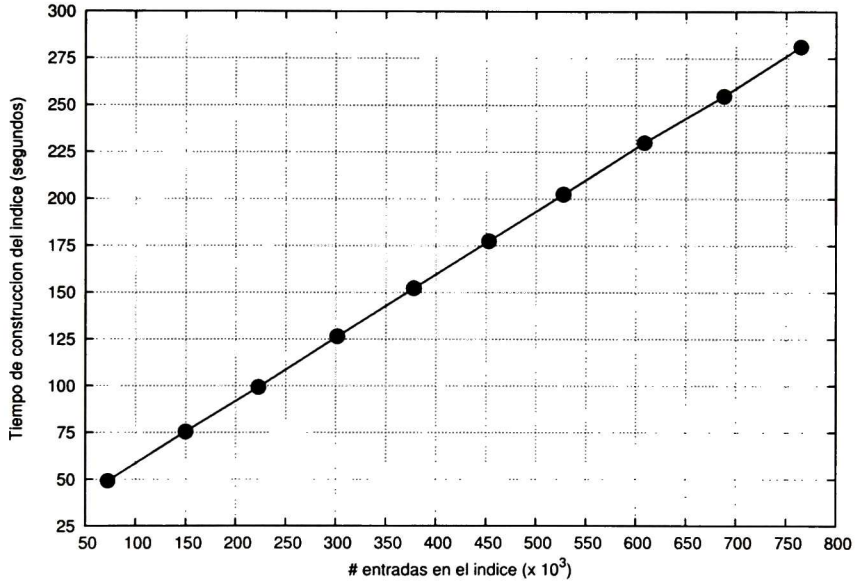



**Figura 6.5**: Tiempo de construcción del índice invertido

Por otra parte, en la gráfica de la Figura 6.6 se presentan los espacios de almacenamiento para el índice invertido. Note que el factor determinante del tamaño del índice lo determina la cantidad de palabras únicas dentro de cada documento. Como se puede observar, tanto el costo en tiempo como en almacenamiento incrementan linealmente conforme el número de entradas dentro del índice aumenta.

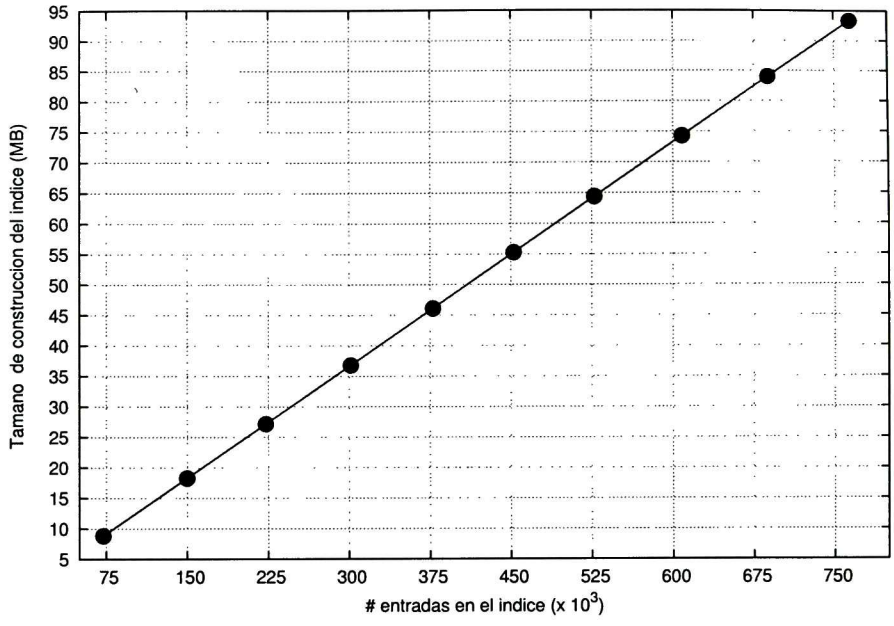



**Figura 6.6:** Espacio de almacenamiento del índice invertido

## 6.5 Evaluación del esquema de búsqueda

En esta sección se evalúa el rendimiento de cada una de las tres fases que constituyen el esquema de búsqueda propuesto descrito en la Sección 5.1.6.

### 6.5.1 Generación de consulta

Para emitir una consulta un usuario autorizado hace uso de la llave secreta que le fue compartida por el propietario de datos y procede a generar la trampilla de cada palabra calculando su valor HMAC tal como en la construcción del vocabulario de palabras. Acorde a lo que se observó el tiempo para generar una consulta es casi despreciable, por ejemplo para una consulta de hasta 50 palabras el tiempo necesario es solamente 1 milisegundo. Finalmente, se indica que el costo de comunicación incurrido durante la transferencia de una consulta hacia el servidor de nube equivale a $l$ bits por

palabra, donde *l* corresponde al número de bits de salida que otorga la función hash utilizada.

### 6.5.2 Búsqueda por relevancia

En esta fase, dada una consulta con múltiples palabras el servidor de nube ejecuta la búsqueda sobre el índice invertido acorde a la función BUSQUEDARELEVANCIA descrita en el Algoritmo 2, para así devolver al usuario la lista de identificadores de documentos junto con su respectiva puntuación de relevancia en forma cifrada. Para efectuar las búsquedas sobre el diccionario de palabras se utilizó como estructura de datos una tabla hash, esto permitió recuperar la lista invertida asociada a una palabra en un tiempo constante. Por el contrario, los cálculos efectuados mediante la propiedad aditiva del criptosistema homomórfico para obtener la puntuación de relevancia de cada documento corresponden al factor dominante de cómputo.

En el experimento, el rendimiento fue evaluado variando de 1 a 10 el número de palabras contenidas en una consulta y el número de resultados otorgados en la búsqueda: de 15 a 300 con incrementos de 15. Además, dado el hecho que las listas invertidas de las palabras no siempre coinciden en longitud se optó por simular la cantidad de cálculos como si las listas invertidas fueran de igual longitud, esto con el objetivo de poder evaluar los escenarios en donde se ejecute la cantidad máxima de operaciones de la propiedad aditiva del criptosistema homomórfico. En la gráfica de la Figura 6.7 se muestran los resultados en términos de tiempo obtenidos bajo el escenario descrito, los resultados son mantenidos en el orden de milisegundos.

Con respecto al costo de comunicación que implica retornar un resultado de búsqueda se indica que es de $N * (h)$ bits, donde $N$ es el número total de identificadores de documentos retornados coincidentes con los de la consulta y $h$ es el tamaño de módulo utilizado en el criptosistema homomórfico.

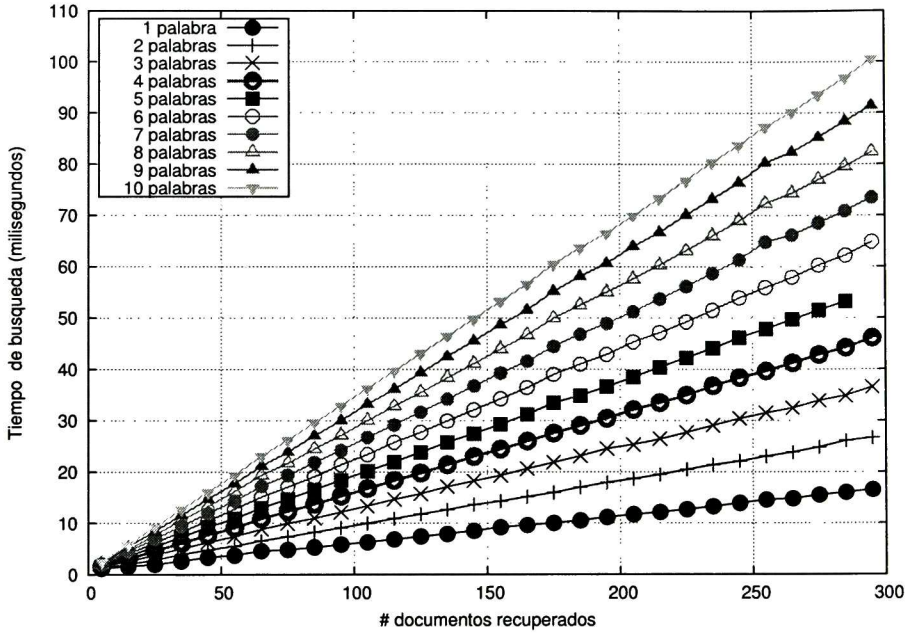



**Figura 6.7:** Proceso de búsqueda

### 6.5.3 Ranking

La fase final del esquema de búsqueda propuesto es llevada a cabo por el usuario (lado del cliente) que emitió la consulta, y consiste en descifrar los resultados retornados por la fase de búsqueda y así obtener los top-$k$ resultados. El costo de cómputo implicado en este proceso es el siguiente:

1) Una operación de descifrado asimétrico (Okamoto-Uchiyama) por cada puntuación de relevancia.

2) El ordenamiento de las puntuaciones de relevancia descifradas.

Para evaluar el rendimiento de esta fase se tomó como métrica el tiempo total requerido para la ejecución de las operaciones de cómputo anteriormente descritas. Los resultados obtenidos son presentados en la gráfica de la Figura 6.8 para diferentes números de valores procesados: de 15 a 300 y con incrementos de 15. Como es posible observar, aún cuando existen operaciones costosas como

lo es el descifrado asimétrico consideramos que los tiempos obtenidos son aceptables. Por ejemplo, para una cantidad de hasta 270 valores procesados el tiempo que se consume en esta fase está en el orden de los milisegundos.

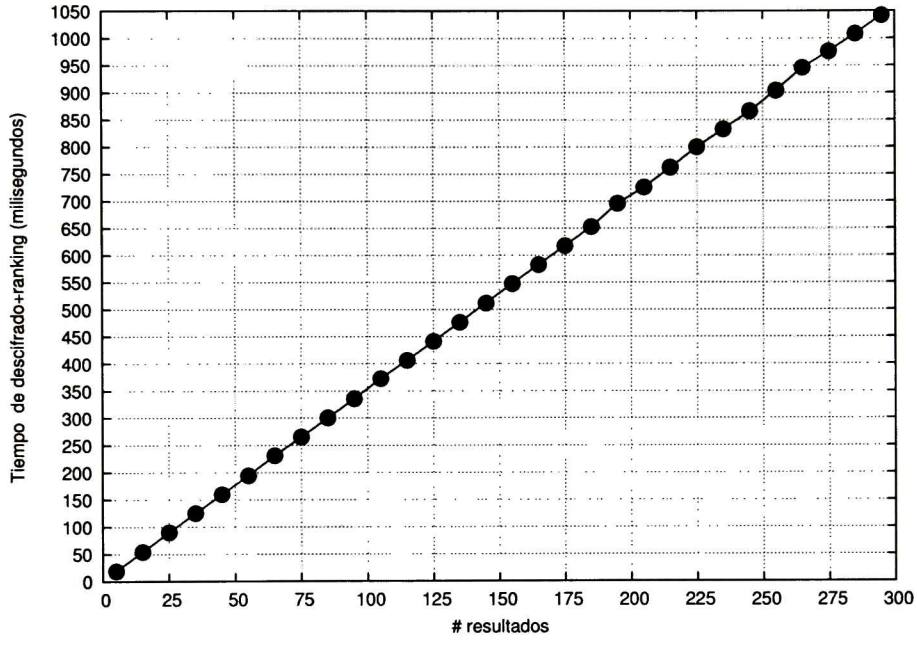



**Figura 6.8:** Tiempo de ranking (lado del cliente) medido en milisegundos.

## 6.6 Evaluación del esquema de integridad

El esquema de auditoría pública de datos que se implementó [47] consta de cinco fases: generación de llaves, generación de firmas, desafío, generación de prueba y verificación de prueba. En el esquema, las primeras dos fases son ejecutadas por el DO, la tercer fase por el TPA, la cuarta por el CSP y la última fase también ejecutada por el TPA. Para evaluar el servicio de integridad se definió un nivel de seguridad $2^{80}$ bits, por tanto, el tamaño de bloques de datos que se utilizó fue de 160 bits.

### *Fase de inicialización*

Esta fase es parte fundamental para la inicialización del servicio de auditoría de datos, es ejecutada por el DO con el fin de generar las firmas o metadatos de verificación que serán utilizados durante la fase de generación de prueba. En la Tabla 6.3 se presentan los resultados medidos en segundos (s) del tiempo requerido para generar el conjunto de firmas de un documento, durante la evaluación se utilizaron documentos con distintos tamaños.

| Tamaño de archivo (KB) | Tiempo (s) |
|---|---|
| 99 | 1.578 |
| 240 | 3.690 |
| 300 | 4.682 |
| 502 | 10.779 |
| 874 | 19.067 |
| 1033 | 22.043 |
| 1491 | 31.982 |
| 2074 | 48.756 |
| 2699 | 68.326 |
| 3052 | 55.421 |

**Tabla 6.3:** Fase de generación de firmas

### *Fase de Auditoría*

El proceso central de auditoría donde interactúan TPA y CSP involucra dos fases: generación y verificación de prueba. En la Tabla 6.4 se muestran los resultados obtenidos para ambas fases y para un muestreo aleatorio de 300 bloques de datos de un documento, parámetro utilizado en [47].

| Fase | Tiempo (ms) |
|---|---|
| Generación de prueba (TPA) | 163 |
| Verificación de prueba (CSP) | 88 |

**Tabla 6.4:** Auditoría pública de datos

# 7

# Conclusiones y trabajo futuro

## 7.1 Conclusiones

El cómputo en la nube es un paradigma reciente que ha cobrado gran auge debido al hecho de ser una alternativa tecnológica que responde a las demandas de cómputo y almacenamiento masivo de las organizaciones, proporcionando ventajas como disponibilidad, escalabilidad, costo-eficiencia, entre otras. Sin embargo, la adopción de este paradigma conlleva diversos desafíos de seguridad, lo cual es motivo de estudio de este trabajo de tesis. Particularmente, se abordó la problemática de desarrollar una arquitectura de servicios de seguridad (autenticación, control de acceso, confidencialidad, integridad) para un servicio de almacenamiento y búsqueda de documentos en la nube.

En este trabajo de tesis, se propuso un esquema de control de acceso de grano fino y confidencialidad que tiene como base principal un Cifrado Basado en Atributos (ABE). Por naturaleza,

los esquemas ABE están compuestos por algoritmos complejos, conjuntamente, los tiempos en los algoritmos de cifrado y descifrado, y el tamaño del texto cifrado y de las llaves están en función de la política de acceso que sea especificada; tal como se observó en la experimentación realizada. En relación a esto, apropiadamente incrementamos la eficiencia de nuestro esquema mediante un enfoque híbrido de cifrado, es decir, un esquema de cifrado simétrico (SE) para el cifrado de los documentos, y ABE para el cifrado de las llaves de SE.

Consideramos como una de las principales aportaciones de esta tesis el esquema seguro de búsqueda por relevancia y múltiples palabras para datos cifrados que aquí se propuso, el cual integra también control de acceso durante la búsqueda. Dentro de la literatura, la problemática de desarrollar un esquema que soportara este tipo de búsquedas había sido estrechamente abordada antes. Nuestro esquema permite a los usuarios recuperar de forma segura los top-$k$ documentos más relevantes, siendo esto un enfoque muy conveniente para el contexto de almacenamiento masivo del cómputo en la nube. La propuesta utiliza técnicas tradicionales de Recuperación de la Información para la construcción del índice de búsqueda, y cifrado homomórfico (HE) como bloque de seguridad principal, con ello fue posible habilitar los cálculos de relevancia durante la búsqueda en el servicio de nube.

Con la finalidad de seleccionar un esquema HE para la propuesta, se realizó un estudio comparativo de distintos criptosistemas homomórficos aditivos, implementando algunos de ellos. El rendimiento de éstos fue evaluado considerando la expansión del mensaje, y la complejidad de las operaciones de cifrado, descifrado así como la de la operación homomórfica. De lo observado, se concluye que la elección de un criptosistema homomórfico depende de los requerimientos de eficiencia y seguridad de la aplicación en particular, y que no existe alguno que sea adecuado para todos los escenarios. Para nuestro caso, el criptosistema Okamoto-Uchiyama fue seleccionado por ser el más apropiado para el esquema propuesto.

Se realizó también un análisis de seguridad y de eficiencia de varios esquemas de verificación de integridad de datos presentando las ventajas y desventajas de cada uno de ellos. Con relación a los requerimientos que se identificaron (muestreo aleatorio, privacidad, número ilimitado de auditorías) para un servicio de auditoría pública de datos a través de un TPA se retomó para la arquitectura uno de los esquemas analizados que cumpliera con dichos requerimientos.

Con el trabajo realizado se ha dado respuesta a la pregunta de investigación motivo de este trabajo de tesis, y se han cumplido los objetivos establecidos al inicio de este documento. Sin embargo, existen algunos aspectos que podrían mejorarse tal como se discute a continuación.

## 7.2 Trabajo futuro

Algunas ideas consideradas como trabajo futuro para la extensión y enriquecimiento de este trabajo de tesis son las siguientes:

- Enfoque paralelo. La inicialización de los distintos servicios de seguridad corre a cargo del propietario de datos (lado del cliente), y conlleva la ejecución de gran cantidad de procesos: cifrado de documentos, cifrado del índice y generación de firmas para verificación de integridad de datos. Dado que en la actualidad la mayoría de las computadoras poseen una arquitectura multiprocesador y que todos estos procesos son altamente paralelizables (es decir, el cifrado de un documento no depende del cifrado de otro documento), una implementación en paralelo de cada uno de ellos podría ayudar a incrementar notablemente la eficiencia en términos de tiempos de ejecución.

- Esquema de indización extendido a otros idiomas. Para ello, la metodología con la que se cuenta puede seguirse tal cual, la única adaptación consistiría en incluir técnicas de procesamiento de texto enfocadas a otros idiomas.

- Privacidad de las políticas de acceso y atributos de usuario. Tanto las políticas de acceso de los documentos como los atributos de los usuarios en el esquema de control de acceso de grano fino y búsqueda propuesto requieren ser almacenados en el servicio de nube, esto podría verse como una falta a la privacidad de dicha información. Sin embargo, consideramos que esta falta de privacidad es tolerable puesto que la confidencialidad de los documentos y la privacidad del índice y consultas es preservada. Más sin embargo, sí sería conveniente desarrollar un mecanismo para la protección de dicha información lo cual se deja como trabajo futuro.

# Bibliografía


[1] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on*, pp. 380–383, june 2010.

[2] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on*, vol. 1, pp. 647–651, march 2012.

[3] R. Chow, P. Golle, M. Jakobsson, E. Shi, J. Staddon, R. Masuoka, and J. Molina, "Controlling data in the cloud: outsourcing computation without outsourcing control," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, (New York, NY, USA), pp. 85–90, ACM, 2009.

[4] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, SP '00, (Washington, DC, USA), pp. 44–, IEEE Computer Society, 2000.

[5] R. O. A. D. Boneh, G. Di Crescenzo and G. Persiano, "Public key encryption with keyword search," in *proceedings of Eurocrypt 2004*, pp. 506–22, march 2004.

[6] E.-J. Goh, "Secure indexes." Cryptology ePrint Archive, Report 2003/216, 2003.

[7] P. R. Christopher D. Manning and H. Schütze, "Introduction to information retrieval," 2008.

[8] N. I. of Standards and Technology, "The nist definition of cloud computing." http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf, September 2011.

[9] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1 – 11, 2011.

[10] N. I. of Standards and Technology, "Secure hash standard (shs)." http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf, March 2012.

[11] N. I. of Standards and Technology, "The keyed-hash message authentication code (hmac)." http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf, March 2002.

[12] W. Diffie and M. Hellman, "New directions in cryptography," *Information Theory, IEEE Transactions on*, vol. 22, pp. 644–654, nov 1976.

[13] V. S. Miller, "Use of elliptic curves in cryptography," in *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, (New York, NY, USA), pp. 417–426, Springer-Verlag New York, Inc., 1986.

[14] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, pp. 203–209, Jan. 1987.

[15] N. I. of Standards and Technology, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile." http://tools.ietf.org/html/rfc5280, May 2008.

[16] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 8, pp. 1467–1479, 2012.

[17] M. Chuah and W. Hu, "Privacy-aware bedtree based solution for fuzzy multi-keyword search over encrypted data," in *Distributed Computing Systems Workshops (ICDCSW), 2011 31st International Conference on*, pp. 273–281, june 2011.

[18] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," in *INFOCOM, 2011 Proceedings IEEE*, pp. 829–837, april 2011.

[19] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp. 253–262, june 2010.

[20] S. Sanka, C. Hota, and M. Rajarajan, "Secure data access in cloud computing," in *Internet Multimedia Services Architecture and Application(IMSAA), 2010 IEEE 4th International Conference on*, pp. 1–6, dec. 2010.

[21] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology – EUROCRYPT 2005* (R. Cramer, ed.), vol. 3494 of *Lecture Notes in Computer Science*, pp. 457–473, Springer Berlin Heidelberg, 2005.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, CCS '06, (New York, NY, USA), pp. 89–98, ACM, 2006.

[23] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pp. 321–334, may 2007.

[24] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1–5, march 2010.

[25] C. Liu, L. Zhu, L. Li, and Y. Tan, "Fuzzy keyword search on encrypted cloud storage data with small index," in *Cloud Computing and Intelligence Systems (CCIS), 2011 IEEE International Conference on*, pp. 269–273, sept. 2011.

[26] I. H. Witten, A. Moffat, and T. C. Bell, "Managing gigabytes: Compressing and indexing documents and images - errata," 1996.

[27] R. L. Rivest, L. Adleman, and M. L. Dertouzos, "On data banks and privacy homomorphisms," *Foundations of Secure Computation, Academia Press*, pp. 169–179, 1978.

[28] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 26, pp. 96–99, Jan. 1983.

[29] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proceedings of CRYPTO 84 on Advances in cryptology*, (New York, NY, USA), pp. 10–18, Springer-Verlag New York, Inc., 1985.

[30] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial information," in *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, (New York, NY, USA), pp. 365–377, ACM, 1982.

[31] J. B. Clarkson, "Dense probabilistic encryption," in *Proceedings of the Workshop on Selected Areas of Cryptography*, pp. 120–128, 1994.

[32] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Advances in Cryptology EUROCRYPT '98, International Conference on the Theory and Application of Cryptographic Techniques, Espoo, Finland, May 31 - June 4, 1998, Proceeding*, vol. 1403 of *Lecture Notes in Computer Science*, pp. 308–318, Springer, 1998.

[33] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the 17th international conference on Theory and application of cryptographic techniques*, EUROCRYPT'99, (Berlin, Heidelberg), pp. 223–238, Springer-Verlag, 1999.

[34] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of paillier's probabilistic public-key system," in *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography*, PKC '01, (London, UK, UK), pp. 119–136, Springer-Verlag, 2001.

[35] D. Boneh, E.-J. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in

*Proceedings of the Second international conference on Theory of Cryptography*, TCC'05, (Berlin, Heidelberg), pp. 325–341, Springer-Verlag, 2005.

[36] C. Gentry, *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[37] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme." Cryptology ePrint Archive, Report 2010/520, 2010.

[38] M. F. Porter, "Readings in information retrieval," ch. An algorithm for suffix stripping, pp. 313–316, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997.

[39] D. L. G. Filho and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," 2006. Submitted to SCN'2006 decio@decpp.net 13254 received 16 Apr 2006.

[40] F. Sebe, J. Domingo-Ferrer, A. Martinez-Balleste, Y. Deswarte, and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 20, pp. 1034 –1038, aug. 2008.

[41] G. Yamamoto, S. Oda, and K. Aoki, "Fast integrity for large data," in *Workshop on Software Performance Enhancement for Encryption and Decryption (SPEED 2007)*, pp. 21–32, 2007.

[42] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, (New York, NY, USA), pp. 598–609, ACM, 2007.

[43] M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, HOTOS'07, (Berkeley, CA, USA), pp. 11:1–11:6, USENIX Association, 2007.

[44] A. Juels and B. S. Kaliski, Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, (New York, NY, USA), pp. 584–597, ACM, 2007.

[45] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '08, (Berlin, Heidelberg), pp. 90–107, Springer-Verlag, 2008.

[46] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *J. Cryptol.*, vol. 17, pp. 297–319, Sept. 2004.

[47] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM, 2010 Proceedings IEEE*, pp. 1 –9, march 2010.

# CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL IPN

## UNIDAD TAMAULIPAS

Cd. Victoria, Tamaulipas, a 17 de diciembre de 2013.

Los abajo firmantes, integrantes del jurado para el examen de grado que sustentará la C. Sandra Daniela Hernández de la Rosa, declaramos que hemos revisado la tesis titulada:

**"Servicio seguro de almacenamiento y búsqueda de documentos en un entorno de nube"**

Y consideramos que cumple con los requisitos para obtener el grado de Maestra en Ciencias en Computación.

Atentamente,

**Dr. Arturo Díaz Pérez**

**Dr. Hiram Galeana Zapién**

**Dr. José Luis González Compeán**

**Dr. Víctor Jesús Sosa Sosa**

Km. 5.5 Carretera a Soto la Marina. Ciudad Victoria, Tamaulipas. CP 87130
Parque Científico y Tecnológico TECNOTAM
Tels.: (834) 1070220 y 316 6600
E-mail: admin@tamps.cinvestav.mx