

Implementación del Hash del cuco para búsqueda y encriptación de contraseñas para una base de datos

Anthony Bautista¹, Cristopher García², Rosa Limachi³ y Julio Rosales⁴

^{1 2 3} Ciencias de la Computación ⁴ Matemática

Universidad Nacional de Ingeniería, Lima, Perú.

{¹abautista1, ³rlimachip}@uni.pe {²crisebas100, ⁴julio845}@gmail.com

Resumen—En este informe propone una forma de obtener una búsqueda de usuarios de una base de datos en tiempo constante, así como asegurar sus contraseñas, empleando el lenguaje python para la implementación y el algoritmo de hash del cuco para la búsqueda y la encriptación.

I. INTRODUCCIÓN

Para un administrador de la base de datos de una institución siempre surge el problema de identificar de una manera rápida quién y cómo se usa los recursos del sistema. Una solución ha este problema sería la creación de un sistema de autenticación de usuarios donde se genera y almacena un nombre de usuario y su respectiva contraseña. Aunque esta solución es válida sigue habiendo el problema de la suplantación de usuarios, por el acceso no autorizado de las contraseñas en la base de datos donde se almacenaba, y el aumento de tiempo en la búsqueda de usuarios a medida que estos aumenten. Por ello se tendría que encriptar las contraseñas de una forma que no se pueda obtener la real a partir de la encriptada y emplear un algoritmo que encuentre a los usuarios mediante una operación constante.

I-A. OBJETIVO

- Búsqueda de usuarios y encriptación de sus contraseñas en una base de datos a partir del algoritmo de hash del cuco.

I-B. DEFINICIONES

- Base de Datos: Colección de información organizada de un modo específico para que su contenido pueda ser tratado y analizado de manera rápida.
- Diccionario: Estructura de datos para almacenar un conjunto de elementos que admite tres operaciones básicas: Búsqueda (x) que devuelve verdadero si x está en el conjunto actual, y falso en caso contrario; Insertar (x) que agrega el elemento x al conjunto actual si aún no está presente; Eliminar (x) que elimina x del conjunto actual si está presente [4].
- Función Hash: Función que a partir de una entrada que suele ser una cadena, genera una salida (cadena) de longitud fija, esta tiene información.
La colisión hash se produce cuando entradas distintas generan el mismo valor en una función hash.

- Tabla Hash: Estructura de datos que relaciona claves y valores para cada elemento que guarda. Utilizaremos una función hash para transformar la clave de un dato e identificar el lugar que ocupará en la tabla.
- Colisiones Hash: Situación donde al aplicar una función hash a dos entradas distintas generan igual valor.
- Hash del Cuco: Algoritmo que permite resolver las colisiones hash ofreciendo para un valor x la posibilidad de tomar la posición $h1(x)$ o $h2(x)$. Esto nos permitirá buscar un elemento observando solo dos posiciones en la matriz. Al insertar un nuevo elemento x, por supuesto, todavía puede ocurrir que no haya espacio, ya que tanto la posición $h1(x)$ como la posición $h2(x)$ pueden ocuparse. Esto se resuelve imitando los hábitos de anidamiento del cuco europeo: ¡Deseche el ocupante actual de la posición $h1(x)$ para dejar espacio! Si la posición alternativa para $h1(x)$ está vacante, esto no es un problema. De lo contrario, $h1(x)$, siendo víctima de la crueldad de x, repite el comportamiento de x y arroja al ocupante. Esto se continúa hasta que el procedimiento encuentra una posición vacante o ha tardado demasiado. En este último caso, se eligen nuevas funciones hash y se reconstruye toda la estructura de datos [4].

II. ESTADO DEL ARTE

- Cuckoo Hashing
Presentamos un diccionario simple con el tiempo de búsqueda constante en el peor de los casos, que iguala el rendimiento teórico del esquema de hash dinámico clásico de Dietzfelbinger, siendo este más complicado de implementar. Además, se menciona algunos retos a superar. Como el no haberse encontrado una familia de función hash explícita que sea probablemente buena para el esquema y si aumentar tablas mejorara el rendimiento de la memoria.[1]
- Hashing: Técnicas y Hash para la Protección de Datos
Se compara las diversas técnicas de hashing, donde cada una de estas puede presentar colisiones, con un costo computacional alto. Además, muestra sus diversos usos como en el cifrado de contraseñas, en creación de certificados digitales, cuando ocurre un ataque de base de datos .[2]
- Algoritmo hash y vulnerabilidad a ataques
Explica el problemas del ciframiento de los datos al

aplicar encriptación por hash, donde se propone como solución emplear dos algoritmos como el SHA-1 y RIPEMD-160.[3]

III. DISEÑO DEL EXPERIMENTO

El experimento se realizara con el lenguaje python creando 4 módulos:

1. Main
Módulo principal donde se muestra las opciones de registro, validación de usuario y salir. Basta con ejecutar esta función para que el programa funcione.
2. Datos
Almacena las funciones que permiten saber si el usuario que se registra existe, así como validar su contraseña.
3. Cuckoo
Posee la función colocar que se usa tanto para desordenar las contraseñas ingresadas en forma de una sucesión de caracteres, como para almacenar las cuentas de usuarios en las tablas hash, mediante el algoritmo del hash del cuckoo, como se muestra en Algoritmo 2. Ver Figura 1.
4. Recepcion_datos
Permite el almacenamiento de usuario y contraseñas en archivos cvs, encriptando la contraseña mediante el módulo Cuckoo. Además, de verificar si el usuario con su respectiva contraseña existen.

Algorithm 1: funcion_hash(funcion,clave)

```

1 if cont == n then
2   aux.append(clave)
3 for j = 0 to ver do
4   suma = suma + ord(i)
5 switch funcion do
6   case 1 do
7     return suma % n
8   case 2 do
9     return n - (suma % n) - 1

```

Algorithm 2: Colocar(clave, tabla, cont, n)

```

1 if cont == n then
2   aux.append(clave)
3 for j = 0 to ver do
4   pos[i] = funcion_hash(i + 1, clave)
5   if hashTable[tbl][pos[tbl]] != 0 then
6     save = hashTable[tbl][pos[tbl]]
7     hashTable[tbl][pos[tbl]] = clave
8     colocar(save, (tbl + 1) % ver, cont + 1, n)
9   else
10    hashTable[tbl][pos[tbl]] = clave

```

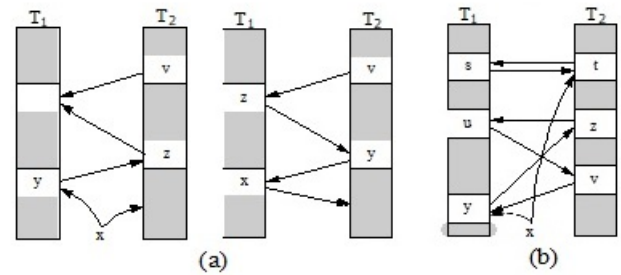


Figura 1. Ejemplos de inserción de cuckoo Hashing. Las flechas muestran posibilidades para mover las teclas. (a) La clave x se inserta con éxito moviendo las teclas z e y de una tabla a la otra. (b) No se puede acomodar la llave x y es necesario un refrito.

IV. PRUEBA

Ver en el cuaderno de jupyter en el siguiente enlace: <https://github.com/crisebas/Cuckoo-Hashing/blob/master/experimentacion.ipynb>

REFERENCIAS

- [1] Rasmus, Pagh., Flemming Friche Rodler. (2001) Cuckoo Hashing. Journal of Algorithms.
- [2] Samuel, Sánchez., Pablo, Domínguez., Luis Velásquez. (2009) Hashing: Técnicas y Hash para la Protección de Datos. Universidad Tecnológica de Panamá.
- [3] Mena Miranda, Yerko. (2009) Algoritmos HASH y vulnerabilidad a ataques. Universidad Mayor De San Andrés.
- [4] Rasmus, Pagh. (2006) Cuckoo Hashing for Undergraduates. IT University of Copenhagen.