Tecnológico de Monterrey

TC3005B

Desarrollo e Implantación de Sistemas de Software

Modulo 3. Desarrollo Avanzado de Aplicaciones Web

Dr. Frumencio Olivas Alvarez

frumen@tec.mx

Tecnológico de Monterrey

Contenido

- Node.js
- Crypto
- Hash
- Cipher
- Decipher
- Práctica

Node.js

Es un entorno en tiempo de ejecución basado en el lenguaje de programación JavaScript, muy utilizado en la capa del servidor (back-end).

Es de código abierto, multiplataforma que ejecuta código en JavaScript de lado del servidor, lo que permite modificar una pagina web antes de ser enviada al usuario.

Tecnológico de Monterrey

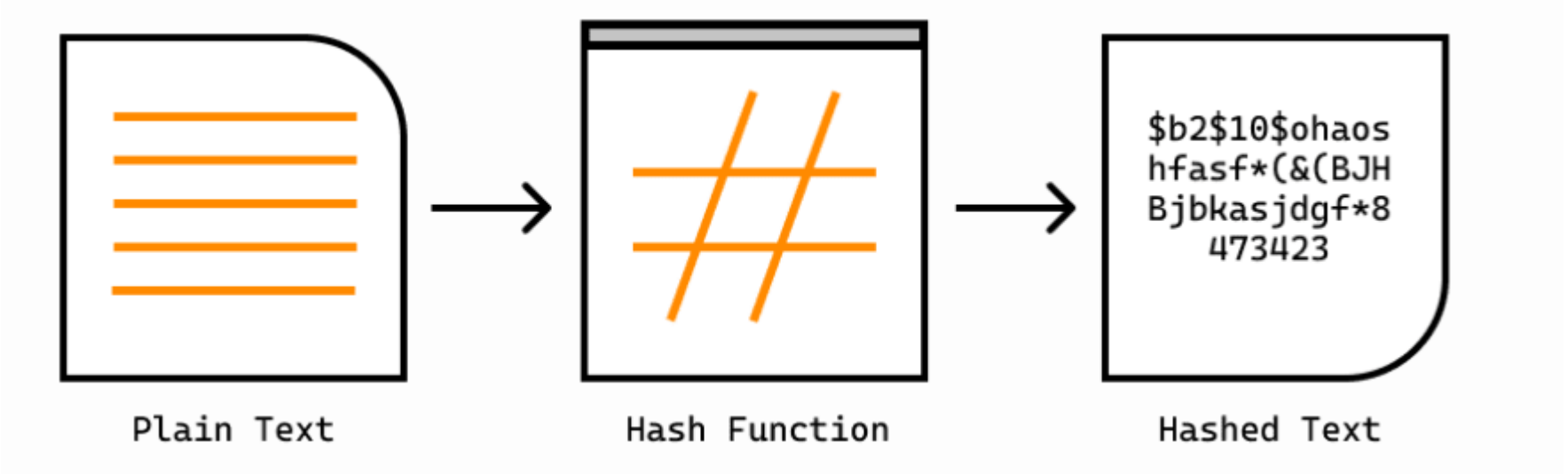
Crypto

Node.js nos provee de la librería **crypto** para utilizar algoritmos de hashing, con los cuales podemos ocultar/encriptar las contraseñas de los usuarios o generar un token para el acceso a nuestra API.

Dentro de esta librería existen diversos métodos para realizar la encriptación, en nuestro caso usaremos createHash().

Los algoritmos de hashing disponibles son lo que se encuentran en **OpenSSL** de código abierto.

Algoritmo Hash



Tecnológico de Monterrey

Ejemplo simple con Crypto

```
import crypto from "crypto";

let msg = "Hola";

const hashing = crypto.createHash("sha512");
// md4, md5, sha1, sha224, sha256, sha384, sha512

const hash = hashing.update(msg).digest("base64url");
// binary, hex, base64, base64url

console.log(hash);
```

Tecnológico de Monterrey

Hackers

Ataques:

- Fuerza bruta.
- Diccionarios.
- Tablas Arcoíris.

Sal y Pimienta
(Salt and Pepper)

Una forma de hacer más seguros los hash de contraseñas ante ataques de fuerza bruta, diccionarios y/o tablas arcoíris es agregando sal y pimienta a la contraseña antes de calcular su hash.

En lugar de que la contraseña sea **123456** pasaría a ser **N45x17owr9TGietQiolUBwxc123456** donde los 24 caracteres del inicio son la sal o pimienta agregada a la contraseña.

Sal
(Salt)

- Para cada contraseña se debe generar sal aleatoria.
- Por lo que dicha sal debería ser guardada, junto con el hash generado separado por algún símbolo que no existe al convertir el hash a texto, por ejemplo los dos puntos.
- De tal manera que cada contraseña que se ingrese, debe ser concatenada a la sal antes de calcular su hash.
- Esto implica que para comparar contraseñas, primero se debe separar la sal de la contraseña guardada para concatenarla a la contraseña ingresada.

Pimienta (Pepper)

- La pimienta es una sola cadena aleatoria de caracteres que se le concatena a cada contraseña.

- Dicha pimienta debería ser guardada tal vez como variable en el archivo .env de tal manera que cada contraseña que se ingrese, debe ser concatenada a la pimienta antes de calcular su hash.

Sal y Pimienta (Salt and Pepper)

La sal es diferente para cada contraseña y genera un hash distinto con contraseñas iguales, pero se guarda junto al hash en la BD.

La pimienta es única, lo que genera el mismo hash con la misma contraseña.

Tecnológico de Monterrey

Ejemplo 1 con mejor encriptación

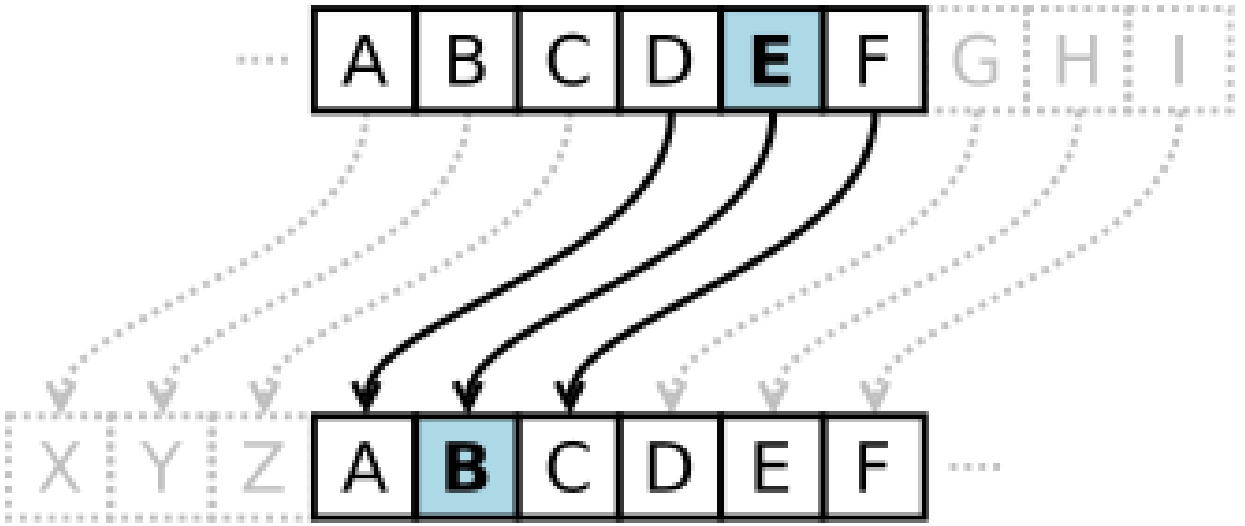
```
import crypto from "crypto";

let msg = "Hola";

const pepper = process.env.PEPPER;


const newMsg =  pepper + msg;

const hashing =  crypto.createHash("sha512");
// md4, md5, sha1, sha224, sha256, sha384, sha512

const hash = hashing.update(newMsg).digest("base64url");
// binary, hex, base64, base64url

console.log(hash);
```

Tecnológico de Monterrey

Ejemplo 2 con mejor encriptación

```
import crypto from "crypto";

let msg = "Hola";

const salt = crypto.randomBytes(24);


const newMsg =  salt.toString(("base64url") + msg;

const hashing =  crypto.createHash("sha512");
// md4, md5, sha1, sha224, sha256, sha384, sha512

const hash = hashing.update(newMsg).digest("base64url");
// binary, hex, base64, base64url

console.log(salt + ":" + hash);
```

Cifrado y Descifrado



Tecnológico de Monterrey

Ejemplo con cifrado

```
import crypto from "crypto";

const texto = "hola mundo";

const encryption_key = "byz9VFNtbRQM0yBODcCb1lrUtVVH3D3x"; // 32 chars
const initialization_vector = "X05IGQ5qdBnIqAWD"; // 16 chars

const cipher = crypto.createCipheriv(
  "aes-256-cbc",
  Buffer.from(encryption_key),
  Buffer.from(initialization_vector)
);
let crypted = cipher.update(texto, "utf8", "hex");
crypted += cipher.final("hex");

console.log(crypted);
```

Tecnológico de Monterrey

Ejemplo con descifrado

```
…

const decipher = crypto.createDecipheriv(
  "aes-256-cbc",
  Buffer.from(encryption_key),
  Buffer.from(initialization_vector)
);
let dec = decipher.update(crypted, "hex", "utf8");
dec += decipher.final("utf8");

console.log(dec);
```

Práctica

- Mejora el proyecto del Backend para que:
  - Las contraseñas se guarden de manera segura.
  - La comparación al iniciar sesión sea correcta.

Tecnológico de Monterrey

Dudas ¿?

Tecnológico de Monterrey