Découverte

Hachage de données

Rédigé par

David ROUMANET
Professeur BTS SIO

Changement

Date	Révision

Sommaire

Nomenclature :

- **Assimiler** : cours pur. Explication théorique et détaillée (globalement supérieur à 4 pages).
- **Décoder** : fiche de cours, généralement inférieure à 5 pages.
- **Découvrir** : Travaux dirigés. Faisable sans matériel.
- **Explorer** : Travaux pratiques. Nécessite du matériel ou des logiciels.
- **Mission** : Projet encadré ou partie d'un projet.
- **Voyager** : Projet en autonomie totale. Environnement ouvert : Vous êtes le capitaine !

A Introduction

Le hachage est une méthode de chiffrement non réversible : il est théoriquement impossible de retrouver un message une fois chiffré par ce moyen. On parle alors d'une empreinte.

A.1 Présentation

Cette courte activité présente le fonctionnement du hachage et quelques résultats à avoir en tête.

A.2 Prérequis

Connaître les méthodes de hachage.

B Codage

B.1 Hachage simple MD5

Le premier code permet de coder une chaine en une empreinte MD5.

Pour cela, nous utiliserons la librairie **crypto-js**, en version 4.0 téléchargeable sur : https://www.cdnpkg.com/crypto-js/file/crypto-js.js/

Ce framework est peu documenté, mais simple à comprendre. Voici comment l'utiliser dans une simple page HTML/JS :

```
<script type="text/javascript" src="path-to/crypto-js/crypto-js.js"></script>
<script type="text/javascript">
    let encrypted = CryptoJS.AES(...);
    let encrypted = CryptoJS.SHA256(...);
</script>
```

Notre premier code doit pouvoir fournir une empreinte MD5 d'un message saisi.

hachage_message_md5.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Générateur d'empreinte MD5</title>
    <script src="./crypto-js.js"></script>
  </head>
  <body>
    <input type="text" id="message" placeholder="Saisir message">
    <div id="result"></div>

    <script>
      let oMessage = document.getElementById("message");
      let oResult = document.getElementById("result");

      function updateHash() {
        let message = oMessage.value;
        let hashType = oHashType.value;
        let hash = CryptoJS.MD5(message);
        oResult.textContent = hash.toString();
      }

      oMessage.addEventListener("input", updateHash);
    </script>
  </body>
</html>
```

Ce code permet donc d'afficher en temps réel, le résultat du hachage MD5 sur le message saisi.

L'ajout d'espace ou caractère invisible conduit systématiquement à une génération d'une empreinte différente. Cependant, il existe d'autres méthodes de hachage.

Pourriez-vous ajouter les choix pour SHA1, SHA256 et SHA512 ?

B.2 Hachage avec différents algorithmes

La proposition suivante permet d'ajouter les choix demandés :

hachage_message.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Générateur d'empreinte</title>
    <script src="./crypto-js.js"></script>
  </head>
  <body>
    <input type="text" id="message" placeholder="Saisir message">
    <select id="hashType">
      <option value="MD5">MD5</option>
      <option value="SHA1">SHA1</option>
      <option value="SHA256">SHA256</option>
      <option value="SHA512">SHA512</option>
    </select>
    <div id="result"></div>

    <script>
      let oMessage = document.getElementById("message");
      let oHashType = document.getElementById("hashType");
      let oResult = document.getElementById("result");

      function updateHash() {
        let message = oMessage.value;
        let hashType = oHashType.value;
        let hash;

        switch(hashType) {
          case "MD5":
            hash = CryptoJS.MD5(message);
            break;
          case "SHA1":
            hash = CryptoJS.SHA1(message);
            break;
          case "SHA256":
            hash = CryptoJS.SHA256(message);
            break;
          case "SHA512":
            hash = CryptoJS.SHA512(message);
            break;
        }

        oResult.textContent = hash.toString();
      }

      oMessage.addEventListener("input", updateHash);
      oHashType.addEventListener("change", updateHash);
    </script>
  </body>
</html>
```

Ce code peut donc générer des empreintes MD5, SHA1 à SHA-512 : on note que les empreintes les plus longues sont les plus sécurisées.

Constatez les caractères utilisés dans chaque empreinte.

C Salage

Le salage est une technique extrêmement simple pour renforcer le hachage d'un mot de passe.

Concrètement, si deux utilisateurs ont le même mot de passe dans une base de données (tous les deux ont appelé leur chien "Médor"), la compromission du mot de passe de l'un entraine la compromission de l'autre.

Pour éviter cela, on peut hacher le message avec une autre partie de données (préfixe ou suffixe), par exemple, la date de naissance, l'adresse de messagerie ou la date d'inscription sur le site. Il est nécessaire de stocker cet élément propre à l'utilisateur dans la base. Ainsi, il y a très peu de chance d'avoir la même empreinte, pour une chaine de mot de passe généralement entre 8 et 13 caractères…

C.1 Exemple de hachage avec salage

salage_mot_de_passe.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Générateur d'empreinte avec salage</title>
    <script src="./crypto-js.js"></script>
  </head>
  <body>
    <input type="text" id="message" placeholder="Saisir mot de passe">
    <input type="date" id="birthdate" placeholder="Saisir date naissance (jj/mm/aaaa)">
    <div id="result"></div>

    <script>
      let oMessage = document.getElementById("message");
      let oBirthdate = document.getElementById("birthdate");
      let oHashType = document.getElementById("hashType");
      let oResult = document.getElementById("result");

      function updateHash() {
        let message = oMessage.value;
        let birthdate = oBirthdate.value;
        let saltedMessage = message + birthdate; // Combiner le mot de passe et la date de naissance
        let hash = CryptoJS.SHA512(saltedMessage).toString();
        oResult.innerHTML = hash
      }

      oMessage.addEventListener("input", updateHash);
      oBirthdate.addEventListener("change", updateHash);
    </script>
  </body>
</html>
```

D Gestion des collisions

Le hachage en MD5 est peu fiable, car le domaine de collision est plus élevé. Qu'est-ce qu'une collision ?

C'est quand deux messages différents donnent le même résultat. Cette probabilité n'est pas nulle, puisqu'on peut considérer qu'il existe une infinité de messages à hacher ;

Vous trouverez en bas de ce site deux images à télécharger : https://natmchugh.blogspot.com/2015/02/create-your-own-md5-collisions.html

Testez-les, à l'aide du programme suivant :

hachage_md5_fichier.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>MD5 File Hash Calculator</title>
    <script src="./crypto-js.js"></script>
    <style>
        #hashHistory {
            margin-top: 20px;
            border: 1px solid #ccc;
            padding: 10px;
        }
    </style>
</head>
<body>
    <h1>MD5 File Hash Calculator</h1>
    <input type="file" id="fileInput">
    <div id="result"></div>
    <div id="hashHistory">
        <h3>Hash History:</h3>
        <ul id="historyList"></ul>
    </div>

    <script>
        const fileInput = document.getElementById('fileInput');
        const result = document.getElementById('result');
        const historyList = document.getElementById('historyList');
        const hashHistory = [];
        fileInput.addEventListener('change', handleFileSelect);

        function handleFileSelect(event) {
            const file = event.target.files[0];
            if (!file) return;
            const reader = new FileReader();
            reader.onload = function(e) {
                const fileContent = e.target.result;
                const hash = CryptoJS.MD5(CryptoJS.lib.WordArray.create(fileContent)).toString();
                displayResult(file.name, hash);
                addToHistory(file.name, hash);
            };
            reader.readAsArrayBuffer(file);
        }

        function displayResult(fileName, hash) {
            result.innerHTML = `<p><strong>File:</strong> ${fileName}</p><p><strong>MD5 Hash:</strong>
${hash}</p>`;
        }

        function addToHistory(fileName, hash) {
            hashHistory.unshift({ fileName, hash, timestamp: new Date() });
            updateHistoryDisplay();
        }

        function updateHistoryDisplay() {
            historyList.innerHTML = '';
            hashHistory.forEach(item => {
                const li = document.createElement('li');
                li.textContent = `${item.hash} - ${item.fileName}  ($
{item.timestamp.toLocaleString()})`;
                historyList.appendChild(li);
            });
        }
    </script>
</body>
</html>
```