**---今回はデータの改ざん確認の為の変換、ハッシュ化について考えます---**

* **内容**OpenSSLを使ったHash化を行う。
* **目標**利用目的に合わせたHash化方法を学ぶ
* **重要キーワード**・Hash・MD5  
  ・SHA
* **Hash**

複合困難な形式へとデータを変換することをハッシュ化と呼びます。

暗号化に似た特性と活かし、ダウンロードしたデータに改ざんされていないか、また欠落していないかの確認するために使用されています。（確認の為だけの物なので、元の形に戻す必要はありません。）  
また、ほかにもデータベースなどにパスワードなどの個人情報を保存して運用した場合、ログインした際の入力データやデータベースに保存しているパスワードを覗き見られた際に悪用される恐れがあります。

しかし、入力データをハッシュ化して送信、パスワードをハッシュ化してデータベースに保存しておくと、除き見られても変形されたデータは元のデータに戻せない為悪用することを防ぐことが出来ます。

* ログイン認証では入力された値をハッシュ化し、データベースに保存されているハッシュ化されたものと同じかを判断しています。

また授業内ではそれぞれのHash化アルゴリズムの内容については触れないので興味がある学生は個人で調べてみてください。

グラフィカル ユーザー インターフェイス

中程度の精度で自動的に生成された説明

* MD5(Message Digest Algorithm 5)  
  Hashアルゴリズムの1つ。  
  入力データのサイズに関係なく128bitのハッシュ値を出力するアルゴリズムです。128bitなので32桁の16進数となります。  
  ハッシュ値は128bit固定のため、異なる入力データでも同じハッシュ化された値になる可能性があり、暗号化などのセキュリティ面では使用されません。  
  しかし、MD5は高速に計算が行えるためファイルデータの整合性チェックやファイルのチェックサムに使用されます。  
  オンラインゲームではダウンロードしたアセットデータなどにデータ欠損が無いかのチェックなどに利用されます。
* SHA  
  こちらもHashアルゴリズムの1つ。  
  主に利用されているのはSHA-2であり、この中でもいろいろなバリエーションがあります。  
  近年では64bit CPUに最適化されているSHA-512が良く利用されています。  
  こちらはハッシュ化されたデータが512bit(64バイト)の固定長のハッシュ値を生成されます。  
  非常に高いセキュリティをもっており、大量のデータ処理に適した設計になっています。  
  現代のサーバ環境ではSHA-512は高速かつ、効率的に計算を行えるため、セキュリティが重視される場面でよく使用が推奨されています。
* **課題8\_1** OpenSSLのMD5暗号化について  
  MD5暗号化手順は下記の通り。  
  ①**[課題8\_01]** EVP＿MD＿CTXを生成し、初期化。  
  　 生成にはEVP\_MD\_CTX\_new関数を使用。  
  　 初期化にはEVP\_DigestInit\_ex関数を使用。

EVP\_MD\_CTX\* mdctx;

unsigned int md5Length = EVP\_MD\_size(EVP\_md5());// MD5サイズ取得

unsigned char\* md5 = reinterpret\_cast<unsigned char\*>(OPENSSL\_malloc(md5Length));//メモリ確保し初期化

事前にハッシュ化後のサイズ分、OPENSSL\_malloc関数で確保しておく必要があるので注意。

**[課題8\_02]**その後、MD5コンテキストを作成し、ハッシュ化の初期設定を行う。

mdctx = EVP\_MD\_CTX\_new();

EVP\_DigestInit\_ex(mdctx, EVP\_md5(),NULL);

② **[課題8\_03]**EVP\_DigestUpdate関数を使用してハッシュ化したい値を読み取る。

EVP\_DigestUpdate(mdctx, readData.c\_str(), readData.size());

③ **[課題8\_04]**EVP\_DigestFinal\_ex関数を使用してハッシュ化を実行。

EVP\_DigestFinal\_ex(mdctx, md5, &md5Length);

④**[課題8\_05]**値を表示用に16進に変換する。(MD5コンテキストの16進表記用)

hashData = ConvertingToHexNumber(md5, md5Length);

hashData = ConvertingToHexNumber(md5, md5Length);

⑤**[課題8\_06]**EVP\_MD\_CTX\_free関数を使用してメモリ開放する。

　EVP\_MD\_CTX\_free(mdctx);

ここまでで、課題8\_1が完成します。

* **課題8\_2** OpenSSLのSHA512暗号化について

MD5とSHA512で同じ関数を使用します。  
引数が一部、異なっているだけですので引数については各自で調べてみてください。

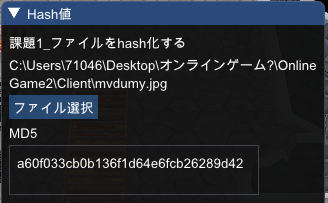
1. EVP＿MD＿CTXを生成し、初期化。  
   生成にはEVP\_MD\_CTX\_new関数を使用。

初期化にはEVP\_DigestInit\_ex関数を使用。

1. EVP\_DigestUpdate関数を使用してハッシュ化したい値を読み取り。
2. EVP\_DigestFinal\_ex関数を使用してハッシュ化を実行。
3. ConvertingToHexNumber 関数を使用して、SHA512コンテキストの16進表記を行う。(引数 sha512)
4. EVP\_MD\_CTX\_free関数を使用してメモリ開放する。

* 課題

1. ファイルを参照し、MD5でハッシュ化を行えるようにすること。  
   MD5でハッシュ化した値が次の手順でハッシュ化したものと同じ値か調べて同じであることを確認すること。

  
  
・MD5のハッシュの確認方法は下記の通り

ファイルを選択するとMD5でハッシュ化された値が表示される。

　⓪画像ファイルをデスクトップに置く。

①VisualStudioから完成した課題を実行、クライアント画面から[Assignment08 Hash化]を選択。

②ウィンドウの[ファイル選択]を選び、デスクトップに置いた画像ファイルを選択。

③MD5のハッシュ値が表示されるので、④からの作業での値と同じか確認する

確認の為、Widowsの機能から確認を行う

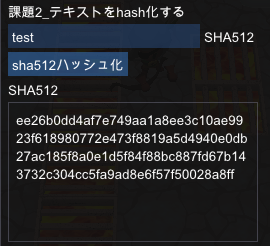
④コマンドプロンプト起動する

MD5変換値

⑤「cd desktop」を入力してエンター

⑥「certutil -hashfileファイル名 MD5」を入力してエンター  
ハッシュ値が表示されるので先ほどの値と同じか確認する。  
テキスト

自動的に生成された説明  
※ ここでは前回課題でダウンロードした画像(mvdumy.jpg)を指定しています。

②テキストを入力し、SHA-512でハッシュ化を行えるようにすること。  
SHA-512でハッシュ化した値が次の手順でハッシュ化したものと同じ値か調べて同じであることを確認すること。  
  
・SHA-512のハッシュ確認方法は下記の通り  
<https://dencode.com/ja/hash/sha512>にアクセス  
ハッシュ化する前のテキストを入力。  
エンコード結果の内容とハッシュ化された値を比較。  


テキストを入力しボタンを押すとsha512でハッシュ化された値が表示される