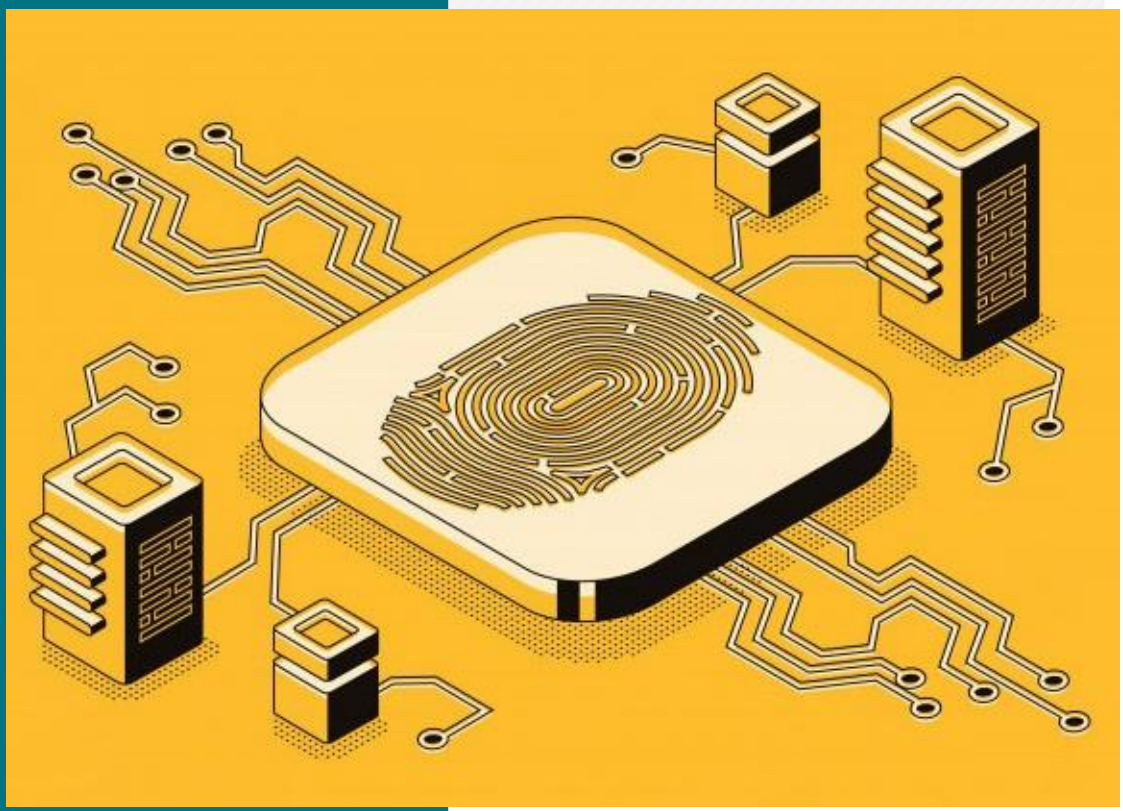
AGENDA 15

CRIPTOGRAFIA EM DADOS PHP



MERGULHANDO NO TEMA...

Criptografia?

Criptografia em Sistemas Computacionais é basicamente um estudo que aplica formas para uma comunicação e armazenamento seguro e privado de dados.

A definição da palavra Criptografia vem do grego “Kryptós”, “oculto” e “Graphein”, “escrita”. Então a criptografia de dados, basicamente oferece a transformação de dados em formatos que todos tenham acessão ao seu conte- údo ou semântica, para um formato não entendível por pessoas/sistemas que não foram autorizados, usando um algotimo de criptografia.

Imagem 02 - freepik.com

Obs.: algoritmo de criptografia é um método matemático criado para encriptar e/ou desencriptar dados com o uso das chaves de criptografia, chave dados ou conjundo de dados utilizados para encriptar um texto, ou desencriptar um texto já encriptado. (DEVMEDIA, 2018)

O foco desta agenda será em confidencialidade que significa que a informação não está disponível para pessoas e processos que não tenham autorização de acesso ou utilização de sua semântica, ou seja, somente pessoas/processos autorizados terão acesso à informação armazenada.

Vamos programar em PHP

O PHP pode ajudá-lo com várias extensões que lhe oferecerão algoritmos de “mãounica” ou one-way. Ao codificar o texto, esses algoritmos não permitem que textos já codificados retornem aos textos originais. A outra alternativa é mão dupla, o que possibilita a codificar e decodificar o texto. Claro que vamos esses algoritmos aprender implementando-os!

MD5

O MD5 é, provalvelmente, o algoritmo mais utilizado em php, principalmente em tutoriaise vídeoaulas.Trata-se de um algoritmo de hash (forum.iMasters.com.br[1])de 128 bits. De forma clara e objetiva, o md5 gera uma string alfa-numérica de 32 caracteres.

O que deve ficar claro é que não importa se você tá gerando o MD5 de um texto com cinco ou seis letras ou de um texto de que contenha mais de mil palavras, o MD5 sempre te entregará uma string de 32 caracteres.

Obs: Lembre-se de quando for armazenar essas informações no Banco de Dados, você devecolocar espaço suficiente para 32 caracteres.

Vamos começar? Para isso é necessário criar um arquivo PHP e salvar com o nome de “mdr” com a extensão php. Não se esqueça de criar os delimetadores do PHP para depois iniciar a programação. Veja a codificação a seguir:

[1] *MATERA – disponível em http://www.matera.com/blog/post/criptografia-hash-assinatura-digital-qual-diferenca. Acesso em 30/10/2018.*

```php
<?php
    $texto = 'técnico em desenvolvimento de sistemas';
    $codificado = md5($texto);
    echo "Resultado do texto codificação usando md5: " . $codificado;
?>
```

Criamos duas variáveis: uma chamada “texto”, na qual atribuímos um texto qualquer, e outra chamada “codificado”, em que será atribuído o resultado da codificação MD5 no texto armazenado na variável “texto”. O resultado, caso você tenha utilizado o mesmo texto desse exemplo, será: “d187e7685130ea461e79e342e2a81558” que também pode ser obervado na imagem a seguir.

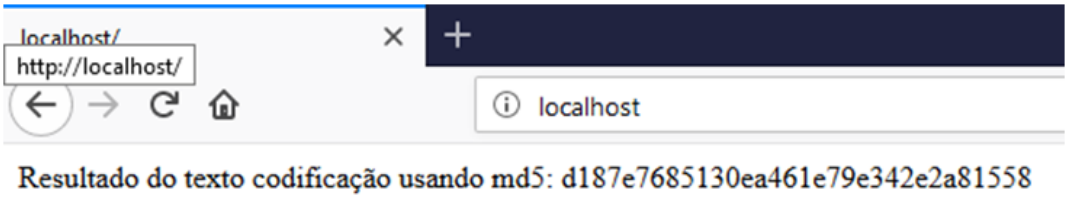


Imagem 03 - Resultado no navegado do algoritmo MD5

Obs.: É possível MD5 gerar 2 strings iguais, mas isso é considerado raríssimo.

SHA1

Trata-se de outra hash de mão única. Segue basicamente o proposto pelo MD5, porém ela possui 160 bits, o que resulta emuma string maior formada por 40 caracteres alfa-numéricos.

Vamos implementar?

Para isso é necessário criar um arquivo PHP e salvar com o nome de “sha1” com a extensão php. Não se esqueça de criar os delimetadores do php!

Para utilizar sha1 no PHP é exatamente a mesma coisa que o MD5, apenas trocando o nome das função de mdr para sha1:

```php
<?php
    $texto = 'técnico em desenvolvimento de sistemas';
    $codificado = sha1($texto);
    echo "Resultado do texto codificação usando sha1: " . $codificado;
?>
```

Obs.: Vale ressaltar que pelo fato do algoritmo Hash ter uma chavede 160 bits e retornar uma string maior (40 caracteres), a chance de encontrar duas strings codificadas iguais é bem mais rara que numa chave de 128 bits. O resultado no navegador será:

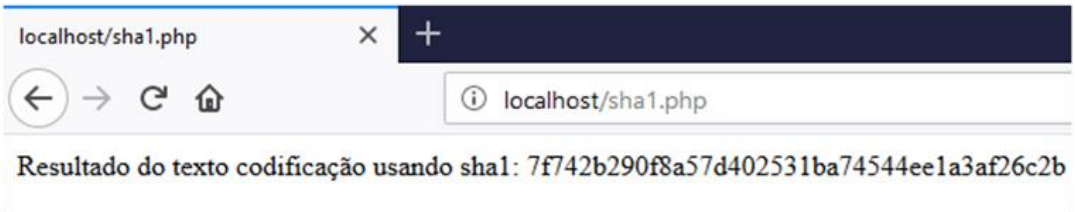


Imagem 04 - Resultado no navegado do algoritmo SHA1

BASE64

Trata-se de um método para codificação dos dados, com a principal característica de ser uma codificação de mão dupla. Logo ela é dividida em duas funções: a primeira para codificar a string base64_encode e a outra

para decodificar a string voltando ao seu estado original: base64_decode.

Vamos implementar?

Para isso é necessário criar um arquivo PHP e salvar com o nome de “base64” com a extensão php. Não se esqueça de criar os delimetadores do PHP antes de programar. Para utilizar Base64 no PHP o procedi- mento é exatamente o mesmo realizado para o md5 e para o sha1, trocando apenas o nome da função para base64_encode para codificar e base64_decode para decoficar a string. O código deve ficar dessa forma:

```php
<?php

    $texto = 'Técnico em Desenvolvimento de Sistemas';
    $criptografada = base64_encode($texto);
    echo "Resultado da criptografia usando base64: " . $criptografada."<br>";

    $textoOriginal = base64_decode($criptografada);
    echo "Resultado da decodificação usando base64: " . $textoOriginal;
?>
```

Obs.: Essa forma de criptografia é muito utilizada para codificar url de site.

O resultado no navegador será:

Imagem 05 - Resultado no navegado do algoritmo Base64_encode e Base64_decode

HASH()

Trata-se de uma função que possibilita a escolha do algoritomo hash de mão única. Basicamente esta fun- ção precisa de dois parâmetros:

- Algoritmo a ser usado para codificação;
- String para codificação.

Há também um terceiro parâmetro que pode ser utilizado, mas sem obrigatoriedade, trata-se de um valor booleano. Caso o valor seja verdadeiro, a string retornada é codificada em binário, ou seja, é retornada com uma cadeia de números composta apenas por 0 e 1, quando valor falso a string retornada é em caracteres alfa numéricos. O valor falso é o padrão, caso for utilizá-lo não é necessário passa-lo como parâmetro.

Vamos implementar?

Para isso é necessário criar um arquivo PHP e salvar com o nome de “hash” com a extensão php. Nãose esqueça de criar os delimetadores do PHP antes de programar. Para utilizar hash no PHP deve utilizar hash (parâmetro 1, parâmetro 2); sendo o parâmetro 1– o nome do algoritmo a ser utilizado, e parâmetro 2– o texto a ser criptrografado.

Um possível exemplo de codificação seria esse:

```php
<?php

        $texto = 'Técnico em Desenvolvimento de Sistemas';
        $criptografada = hash('ripemd160', $texto);
        echo "Resultado da criptografia usando função Hash com o algoritmo ripem160: " .
$criptografada."<br>";

        $texto = 'Técnico em Desenvolvimento de Sistemas';
        $criptografada = hash('sha256', $texto);
        echo "Resultado da criptografia usando função Hash com o algoritmo sha256: " .
$criptografada."<br>";

    ?>
```

Obs: Neste exemplo foram utilizados dois algoritmos: ripemd160 e sha256 – uma versão do algoritmo sha1, mas com uma palavra de 256 bits.

Veja o resultado no navegador:

Imagem 06 - Resultado no navegado da função hash utilizando dois algoritmos diferentes

Para se ter uma noção de quantos algoritmos é possível utilizar e qual o tamanho de cada um, vamos fazer um códido e como resultado teremos a codificação com cada uma das opções de altoritmos disponíveis na função hash. (http://php.net/manual/ [2]). O código deve ficar dessa forma:

```php
<?php

    $texto = "Técnico em Desenvolvimento de Sistemas";

    foreach (hash_algos() as $v) {
            $codificado = hash($v, $texto, false);
            echo "Algoritmo: ".$v."<br>";
            echo "Tamanho: ".strlen($codificado)."<br>";
            echo "Texto Codificado: ".$codificado."<br>";
            echo "<br>";
    }
?>
```

[2] *PHP Manual – disponível em http://php.net/manual/pt_BR/function.hash.php. Acesso em 30/10/2018*

O resultado no navegador será esse:

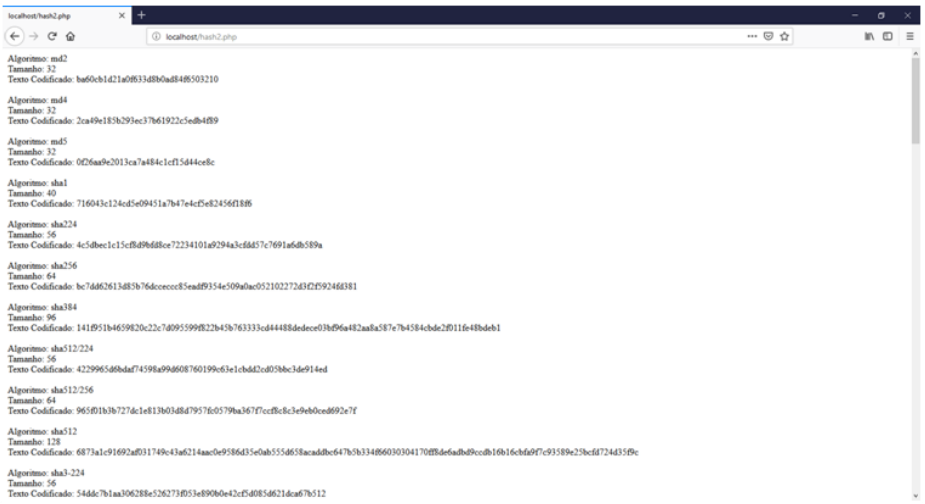


Imagem 07 - Resultado no navegador da função hash utilizando todos os algoritmos diferentes

PasswordHashing()

Trata-se de uma API que fornece um conjunto de funções que facilitam muito a utilização de hash focado em senhas. Essas funções são capazes de ajudar o desenvolvedor a criar hashes seguros. Basicamente essas funções fazem todo o trabalho pesado, de forma segura e com alto grau de confiabilidade. Assim, precisamos trabalhar com duas funções: password_hash e password_verify. A primeira criará a hash e a segunda fará a verificação da hash.

Vamos implementar?

Para isso é necessário criar um arquivo PHP e salvar com o nome de “passwordHash” com a extensão php. Não se esqueça de criar os delimetadores do PHP antes de programar. Para utilizar password_hash no PHP devemos utilizar password_hash (“texto”, PASSWORD_DEFAULT), sendo o primeiro parâmetro:o texto a ser codificado e o segundo parâmetro: as configurações de custo e salt.

- O salt é utilizado para evitar que duas senhas idênticas produzam hashes idênticos, o que facilita e muito a descoberta de senhas.
- O custo é ovalor quedetermina o quão complexo o algoritmo deve ser e, portanto, o quanto demorará a geração do hash.

A utilização do PASSWORD_DEFAULT como configuração garantirá a aleatorieada na configuração. O código deve ficar assim:

```php
<?php

$texto = "Técnico em Desenvolvimento de Sistemas";
$codificado =   password_hash($texto, PASSWORD_DEFAULT);
echo "Texto Codificado: ".$codificado."<br>";
?>
```

O resultado no navegador será este:

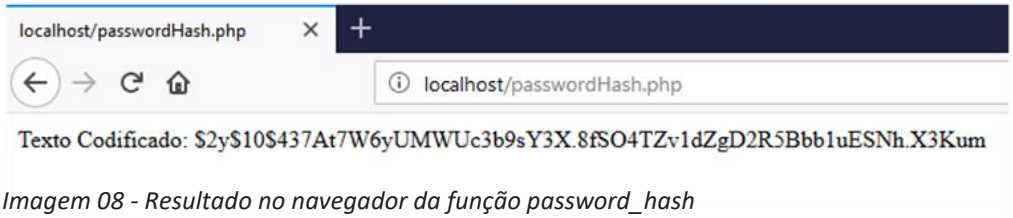


Imagem 08 - Resultado no navegador da função password_hash

Você pode estar se perguntando: “Porque o meu resultado não está igual a imagem 9?”. A resposta é simples: para cara execução da função password_hash, o resultado será um texto codificado de forma diferente, o que com certeza dificulta a quebra e a descoberta dessa hash.

Como fazer para verificar se a senha digitada será a correta, se para cada utilização da função será criada uma novastringhash? Novamente a resposta é simples: basta utilizar a função password_verify. Veja como ficará a codificação:

```php
<?php

    $texto = "Técnico em Desenvolvimento de Sistemas";
    $codificado =   password_hash($texto, PASSWORD_DEFAULT);
    echo "Texto Codificado: ".$codificado."<br>";

    if (password_verify("Técnico em Desenvolvimento de Sistemas", $codificado)) {
        echo 'Texto Correto';
      } else {
        echo 'Texto Incorreto';
      }
?>
```

O resultado no navegador será assim:

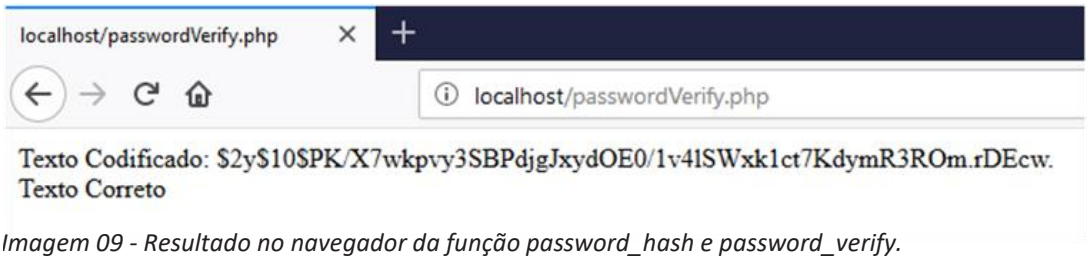
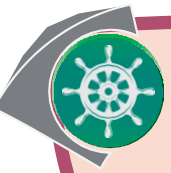


Imagem 09 - Resultado no navegador da função password_hash e password_verify.

Portanto, podemos finalizar afirmando que quando a assunto é senha, o melhor é utilizar a APIpassword nativa no php.



VOCÊ NO COMANDO

Utilizando o que foi visto até agora....

1. Crie um uma página PHP que contenha:

a. Formulário com um campo para inserção de texto e um botão.

2. Ao clicar no botão, o texto inserido no campo deverá ser codificado em um algoritmo hash de sua preferência (MD5, SHA1 etc.)

3. Exiba:

• Senha: "Texto Digitado"

• Senha Codificada: "Texto Codificado"

4. No título da página deverá constar o nome do algoritmo utilizado.

Dicas:

• Utilize no Formulário o Método Post;

• Pode utilizar framework W3 css para estilização.

A seguir, confira se você conseguiu resolver o desafio proposto!

A resposta apresentada a seguir é uma das possibilidades para resolver o problema proposto. Qualquer dúvida contate o seu professor mediador!

Arquivo VoceNoComando.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
    <title>MD5</title>
</head>
<body>
<form action="codifica.php" method="post" class="w3-container w3-card-4 w3-black w3-text-green w3-
margin w3-display-middle" style="width: 30%;">
    <input type="hidden" name="nome_form" value="frmLogin" />
    <div class="w3-row w3-section">
    <div class="w3-col" style="width:11%"><i class="w3-xxlarge fa fa-lock"></i></div>
        <div class="w3-rest">
        <input class="w3-input w3-border w3-round-large" name="txtTexto"  placeholder="Insira Texto">
        </div>
```

```
    </div>
    <div class="w3-row w3-section">

            <button name="btnCrip" class="w3-button w3-block w3-margin w3-green  w3-cell w3-round-
large" style="width: 90%;">Criptografar</button>

    </div>
</form>
</body>
</html>
```

Resultado no Navegador

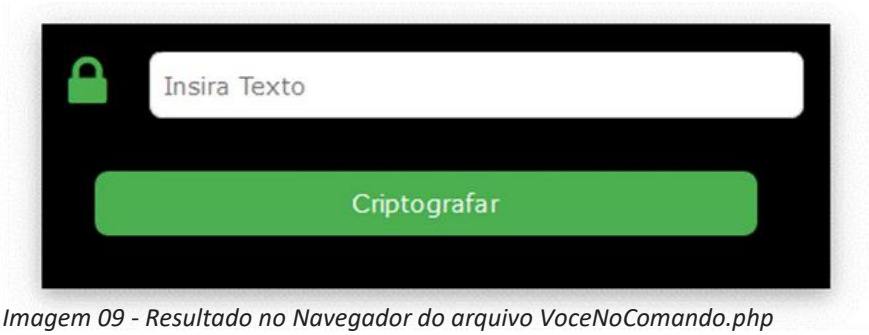


Imagem 09 - Resultado no Navegador do arquivo VoceNoComando.php

Arquivocodifica.php

```
<?php
    $texto = $_POST["txtTexto"];
    $codificado =   md5($texto);
    echo "Senha codificada ".$texto." em MD5: ".$codificado."<br>";
?>
```

Resultado no Navegador

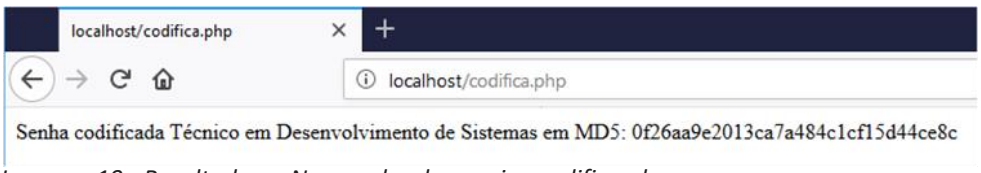


Imagem 10 - Resultado no Navegador do arquivo codifica.php

Obs.: O texto digitado foi Técnico em Desenvolvimento de Sistemas.