CENTRO UNIVERSITÁRIO FEI

GUILHERME ALBERTINI DE OLIVEIRA

**CONVERSÃO UMA BASE DE SENHAS INSEGURA**

São Bernardo do Campo

2018

Guilherme Albertini de Oliveira

**CONVERSÃO UMA BASE DE SENHAS INSEGURA**

Trabalho de Segurança no Desenvolvimento de Software apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel Ciência da Computação. Orientado pelo Prof. Dr. Ricardo Destro.

São Bernardo do Campo

2018

Guilherme Albertini de Oliveira

**CONVERSÃO UMA BASE DE SENHAS INSEGURA**

Trabalho de Segurança no Desenvolvimento de Software, apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel em Ciência da Computação.

**AGRADECIMENTOS**

-

**RESUMO**

Este trabalho tem como objetivo aprimorar uma base de senhas disponibilizada com uma criptografia simples, para uma base mais segura, utilizando 3 algoritmos de HASH.

Palavras-chave: Criptografia. Segurança. Hash Digest.

**ABSTRACT**

This paper looks forward to accomplishing a full conversion of an insecure password base that uses a simple cryptography algorithm, to a secure base using at least 3 distinct hash algorithms.

Keywords: Cryptography. Secure Software. Hash Digest.

**LISTA DE ILUSTRAÇÕES**

-

**LISTA DE TABELAS**

-

**SUMÁRIO**

[1 INTRODUÇÃO](#Indicador1) 11

[2 METODOLOGIA](#Indicador6) 11

[2.1 Cifra](#Indicador7) de césar 11

[2.2 MD5](#Indicador7) 11

[2.3 hash SHA1](#Indicador7) 11

[2.4 hash SHA256](#Indicador7) 11

[3 IMPLEMENTAÇÃO](#Indicador11) 12

[3.1 Equipamentos](#Indicador12) e linguagem de programação 12

[3.1 Metodologia](#Indicador12) para contagem de tempo 12

[4 RESULTADOS](#Indicador16) E CONCLUSÃO 12

[4.1 PERGUNTAS](#Indicador12) 12

[4.2 TEMPO](#Indicador12) DE RESPOSTA DOS ALGORITMOS 13

[REFERÊNCIAS](#Indicador26) 14

# [1 Introdução](#Indicador32)

Com o avanço da tecnologia, e também do hardware computacional, cada vez mais é necessária uma cautela maior em relação à segurança em softwares. Em um cenário onde IoT é uma realidade, os dados estão cada vez mais suscetíveis à invasões e vazamentos, logo, é inviável desenvolver um novo sistema sem que este contemple o básico de segurança de dados.

Neste presente trabalho é apresentada uma base de dados que contém senhas de usuários de um sistema qualquer. A base apresentada contempla uma criptografia de senhas simples, a Cifra de César. Embora o resultado do algoritmo Cifra de César pareça indecifrável para os seres humanos, é uma criptografia bastante simples para computadores.

Este trabalho implementa um simples *console application* que converte esta base fornecida para uma base mais segura.

# [2 Metodologia](#Indicador32)

A metodologia apresentada neste trabalho consiste em criar uma aplicação para converter a base de senhas apresentada com o algoritmo de Cifra de César, para uma base utilizando MD5, uma base utilizando SHA1 e outra utilizando SHA256. Além disto será apresentado o tempo para a conversão da base para cada um dos algoritmos, e também o tempo para a validação de uma senha na nova base.

Os tópicos seguintes comentam sobre cada um dos algoritmos utilizados neste trabalho.

# [2.1 Cifra de césar](#Indicador32)

A Cífra de César como dito anteriormente, é um algoritmo bastante simples de criptografia, o mesmo consiste em deslocar o caractere em questão X posições na tabela ASCII. Por exemplo, supondo que temos que criptografar o caractere ‘A’ com Shift de 5 posições, basta que peguemos a posição do caractere ‘A’ na tabela ASCII e somemos 5 posições, no caso, iremos obter o caractere ‘F’. Para decriptografar basta subtrair 5 posições ao invés de somar.

O resultado da Cifra de César parece bem aceitável a olho nu, porém é muito frágil se submetido a um teste de força bruta em um computador, por este motivo o algoritmo é considerado inseguro.

# [2.2 MD5](#Indicador32)

# O algoritmo MD5 para criação de hashes, desenvolvido pela RSA. Este algoritmo foi considerado ruim para o armazenamento de senhas pois na verdade foi desenvolvido para outra finalidade, a assinatura digital. O algoritmo é extremante rápido para calcular hashes de arquivos grandes, porém falha em ser seguro em armazenar dados sensíveis como senhas.

# [2.3 HASH SHA1](#Indicador32)

# [O algorimo SHA1 (Secure Hash Algorithm) foi desenvolvido pela NSA, como um algoritmo seguro. Porém após muito tempo de seu uso, foi notado que o algoritmo não é extremamente difícil de se quebrar, e é possível fazer engenharia reversa de uma hash e encontrar seu dado de origem. O SHA1 é significativamente mais rápido que o SHA256 e é também utilizado, como o MD5, para criar assinaturas para arquivos muito grande.](#Indicador32)

# [2.4 HASH SHA256](#Indicador32)

# 

# [Assim como o SHA1, o SHA256 (Secure Hash Algorithm) também foi desenvolvido pela NSA. O algoritmo cria uma Hash de 256 bits para sua entrada, e diferentemente de seu antecessor é possível ser utilizado para proteger dados sensíveis como senhas. O SHA256 é significativamente mais lento que seu antecessor mas tem uma capacidade bem maior de assegurar os dados.](#Indicador32)

# [3 Implementação](#Indicador42)

# [3.1 Equipamentos e linguagem de programação](#Indicador32)

A aplicação foi desenvolvida na linguagem C# utilizando o framework .NET. O equipamento utilizado foi um MacBook Air Apple Inc., com um processador Intel Core i5 de 1.8 GHz com dois núcleos e uma memoria RAM de 8GB.

# [3.2 metodologia para contagem do tempo](#Indicador32)

Para a contagem do *Elapsed Time* de cada uma das conversões, e também das validações, foi utilizado a classe ‘Stopwatch’ contida no namespace ‘System.Diagnostics’[1] do framework .NET.

# [4 Resultados e conclusão](#Indicador42)

Foi possível notar que o mais rápido dos algoritmos foi o MD5, e sendo o mais lento o SHA256. Apesar de ser mais devagar processando em média 118 MB/s, o SHA256 apresenta uma HASH muito mais segura do que o MD5, que apresenta uma média de 411 MB/s.

O código fonte para o aplicativo desenvolvido se encontra no anexo e também no repositório do github no endereço:

https://github.com/Albertini96/CryptographyTest

# [4.1 PERGUNTAS](#Indicador32)

P: Existem duas strings com tamanho máximo de 10 caracteres que possuem o mesmo hash digest MD5? E quanto ao SHA-1 e SHA-256 Justifique sua resposta.

R: Não. O MD5 consegue gerar 3,402823669209385e38, isto é 2ˆ128, combinações únicas, e uma string de 10 caracteres representados pela tabela ASCII tem 1,180591620717411e21, isto é 128ˆ10, combinações distintas. Sendo 2ˆ128>128ˆ10, é improvável existir uma colisão, porém, se dois prefixos com o mesmo hash forem construídos, um sufixo comum pode ser adicionado a ambos para tornar uma colisão mais provável. Deste modo é possível que duas strings diferentes produzam o mesmo hash. Em todos os algoritmos de Hash com uma saída de tamanho fixo será possível uma colisão.

 P: A partir de qual tamanho de string podemos afirmar que certamente existirão duas strings com mesmo hash digest MD5?

R: Segundo meus cálculos, com 19 caracteres da tabela ASCII é possível gerar 1,088903574147003e40, isto é 128ˆ19, combinações diferentes. Como o MD5 consegue gerar

# 2ˆ128 combinações diferentes e 2^128 < 128ˆ19, logo é possível concluir que com 19 caracteres, sendo que cada caractere é representado por 8 bits, é impossível não acontecer colisão no MD5 em algum momento.

# [4.2 TEMPO DE RESPOSTA DOS ALGORITMOS](#Indicador32)

A seguir é apresentado o tempo de resposta dos algoritmos em uma amostra de 30 execuções por algoritmo.

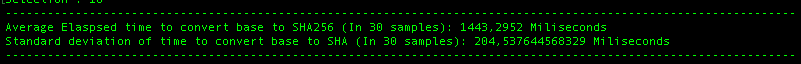
MD5:



SHA1:



SHA256:



# [REFERÊNCIAS](#Indicador27)

[1] <https://msdn.microsoft.com/pt-br/library/system.diagnostics.stopwatch(v=vs.110).aspx> (Acessado em 1/6/2018).