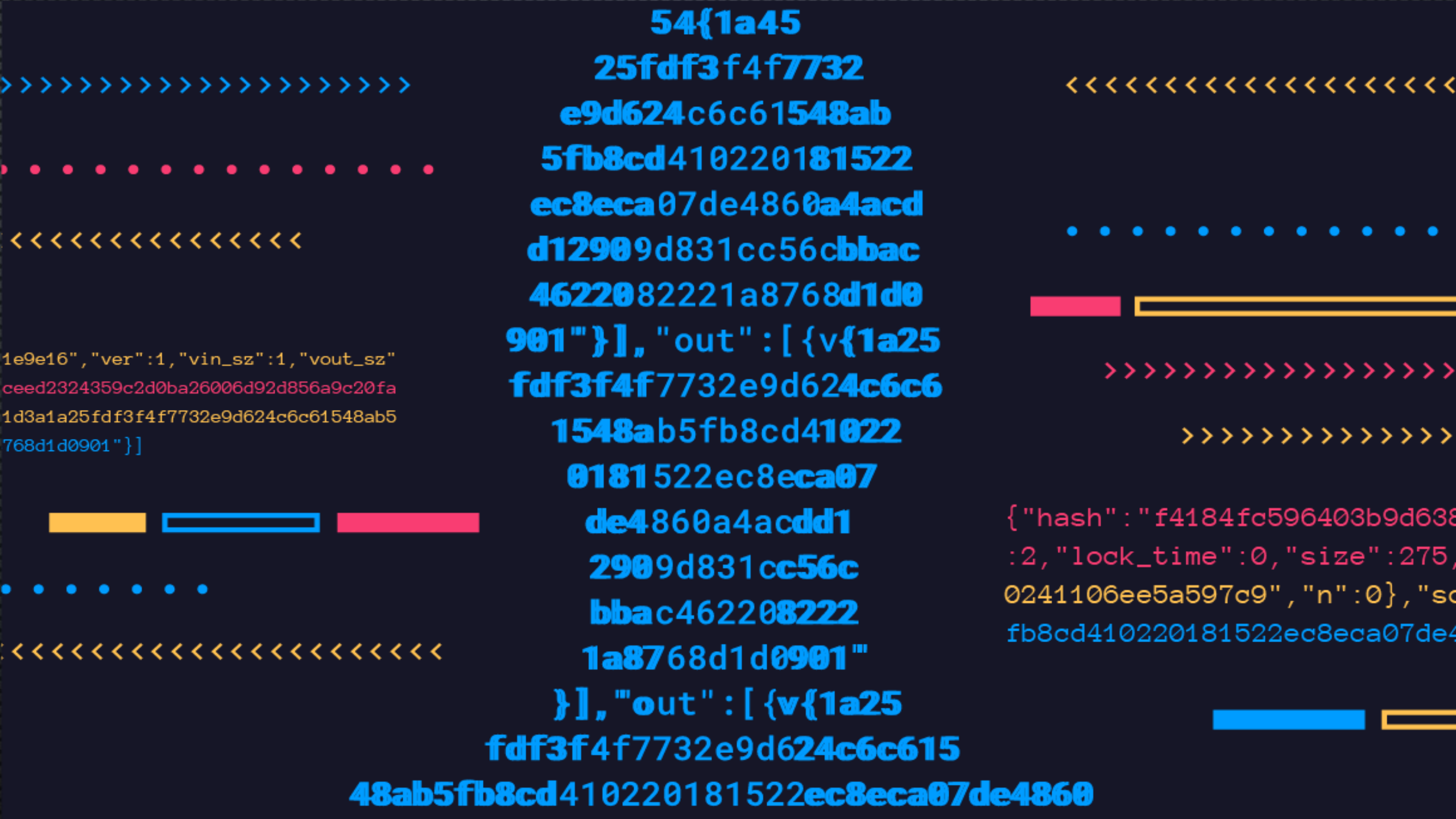
HASH

COD1F1C4NDO D4DO$

J1y4n y4R1


54{1a45
25fdf3f4f7732
e9d624c6c61548ab
5fb8cd410220181522
ec8eca07de4860a4acd
d12909d831cc56cbbac
462208222 1a8768d1d0
901"}],"out":[{v{1a25
fdf3f4f7732e9d624c6c6
1548ab5fb8cd41022
0181522ec8eca07
de4860a4acdd1
2909d831cc56c
bbac46220822 2
1a8768d1d0901"
}],"out":[{v{1a25
fdf3f4f7732e9d624c6c615
48ab5fb8cd410220181522ec8eca07de4860
1e9e16","ver":1,"vin_sz":1,"vout_sz"
ceed2324359c2d0ba26006d92d856a9c20fa
1d3a1a25fdf3f4f7732e9d624c6c61548ab5
768d1d0901"}]
{"hash":"f4184fc596403b9d638
:2,"lock_time":0,"size":275,
0241106ee5a597c9","n":0},"sc
fb8cd410220181522ec8eca07de4

H4SH

HASH são caminhos de mão única (funções matemáticas) que são utilizados para se criar códigos.

Qualquer coisa que se codifica com eles torna-se impossível (matematicamente) de se recuperar.

H4SH

Uma função **HASH** é um algoritmo que mapeia dados de comprimento variável para dados de comprimento fixo.

Os valores retornados por uma função **HASH** são chamados valores **HASH**, códigos **HASH**, somas **HASH** (**HASH** sums), **CHECKSUMS** ou simplesmente **HASHES**.

H4SH

CARACTERÍSTICAS:

- NÃO POSSUEM CHAVE, OU SEJA, GERAR O HASH NÃO ENVOLVE NENHUM SEGREDO QUE DEVA SER COMPARTILHADO;

H4SH

Características:

- Tamanho único, logo não importa o quão grande é a informação original, o hash será sempre de um mesmo tamanho.

Exemplo:

O MD5 tem 128 bits de tamanho, mesmo que seja o HASH de um único caractere (8 bits) ou de um arquivo de 1GB;

H4SH

Um **hash** (ou escrutínio) é uma sequência de bits geradas por um algoritmo de dispersão, que permite a visualização em letras e números, representando um nibble cada.

Exemplo:

Hash MD5 "aula de segurança": **4b12894024c2b347b70f821f2b8a8cae**

Hash MD5 "teste": **698dc19d489c4e4db73e28a713eab07b**

H4SH

Nibble (ou nyble) Sucessão de quatro cifras binárias (bits):

1 Nibble = 4 bits;

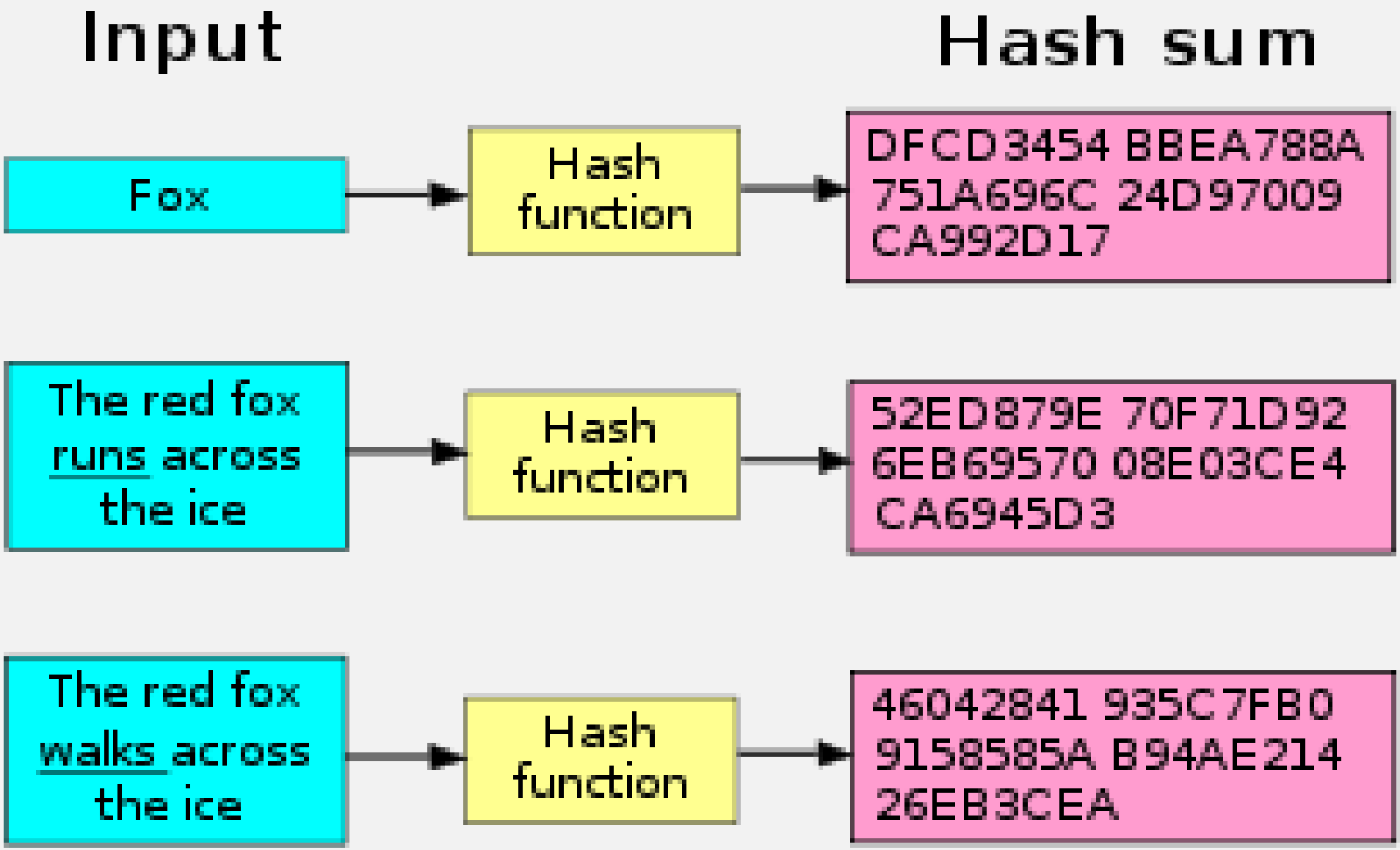
2 Nibble = 1 Byte = 8 bits;

4 Nibble = 1 Word = 2 Bytes = 16 bits.

Sistema de numeração muito utilizado na Informática, assim como na Matemática, e em sistemas digitais eletrônicos.

Padrão de codificação decimal utilizado em muitos circuitos integrados.

0000 0100


Input
Hash sum
Fox
Hash function
DFCD3454 BBEA788A 751A696C 24D97009 CA992D17
The red fox runs across the ice
Hash function
52ED879E 70F71D92 6EB69570 08E03CE4 CA6945D3
The red fox walks across the ice
Hash function
46042841 935C7FB0 9158585A B94AE214 26EB3CEA

H4SH

PROBLEMAS

A SEQUÊNCIA DO HASH É LIMITADA, POR EXEMPLO 64, 128 OU 512 BITS, LOGO EXISTEM COLISÕES (SEQUÊNCIAS IGUAIS PARA DADOS DIFERENTES).

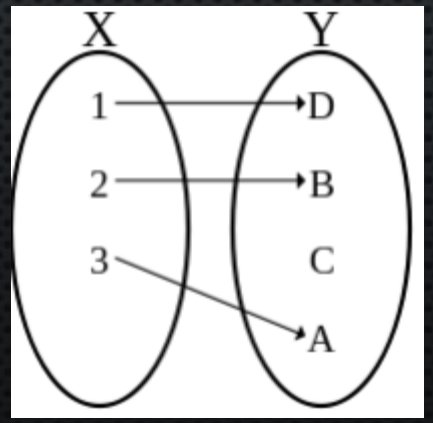
QUANTO MAIOR FOR A DIFICULDADE DE SE CRIAR COLISÕES INTENCIONAIS, MELHOR É O ALGORITMO.

H4SH

Função Injetora

Uma função é injetora se e somente se quaisquer que sejam X1 e X2 e (pertencentes ao domínio da função), X1 é diferente de X2 implica que f(X1) é diferente de f(X2).

Pode haver elemento no conjunto de chegada sem associação com o conjunto de saída.

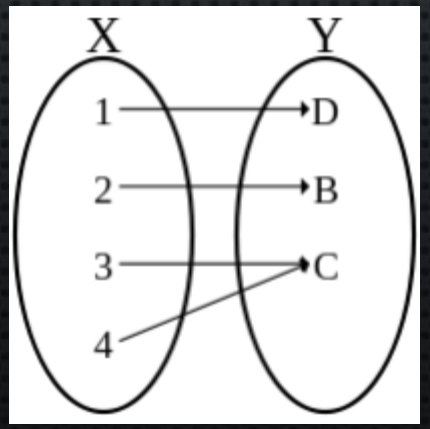


H4SH

Função Sobrejetora

Uma função é sobrejetora se e somente se quando o conjunto imagem coincide com o contradomínio da função.

Dois (02) elementos do conjunto de partida podem ter uma mesma imagem no conjunt de chegada.

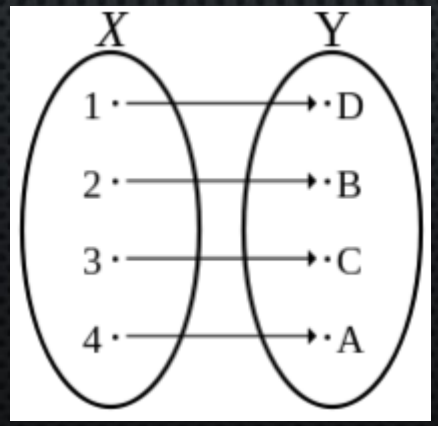


H4SH

Função Bijetora

Uma função é bijetora se e somente se correspondência biunívoca ou bijeção.

É uma função injetora e sobrejetora ao mesmo tempo.



H4SH

É DESEJÁVEL QUE UM ALGORITMO DE HASH SEJA BIJETORA, OU SEJA, PARA CADA ENTRADA HAJA SOMENTE UMA SAÍDA, EM OUTRAS PALAVRAS, PARA CADA TEXTO LIMPO HAJA SOMENTE UM E NÃO MAIS DO QUE UM ÚNICO CÓDIGO HASH.

A NÃO OCORRÊNCIA DESTA CONDIÇÃO É CONHECIDA COMO **COLISÃO**.

H4SH

Os algoritmos de hash mais usados são os de 16 bytes MD4 e MD5 ou o SHA-1, de 20 bytes.

Os principais algoritmos de hash são:

- MD (MD4, MD5);
- SHA-1 (128) - SHA-2 (224, 256, 384 e 512);
- RIPMED;
- WIRLPOOL;
- OpenSSL.

MD4

MD4

Desenvolvido em 1990/91 por Ron Rivest, vários ataques foram detectados, o que fez com que o algoritmo fosse considerado frágil.

Descrito na RFC 1320

MD5

MD5

O MD5 (Message-Digest algorithm 5) é um algoritmo de hash de 128 bits unidirecional desenvolvido pela RSA Data Security, Inc.

descrito na RFC 1321.

Muito utilizado por softwares com protocolo par-a-par (P2P), ou Peer-to-Peer, em inglês), verificação de integridade e logins.

Existem alguns métodos de ataque divulgados para o MD4 e MD5.

MD5

Exemplo:

$ nano teste

$ md5sum teste
60ed36ed0b29ed1af96fb7a5d6e21b75 teste

O MD5 Gera um hash com 32 caracteres.

SHA-1

SHA-1

(*Secure Hash Algorithm*)

Desenvolvido pelo NIST e NSA.

Já foram exploradas falhas no SHA-1.

SHA-1

Exemplo:

$ sha1sum teste
afb3f71d0e172a2f9001657f7b2d8e35db33c125 teste

O SHA-1 gera um hash de 40 caracteres.

SHA-1

Exemplo:

$ sha224sum teste
9a49efcc115e2b4ac6df619a59b9c8e031c3a618b1222910
72646c 5c teste

O SHA-224 gera um hash de 56 caracteres.

SHA-1

Exemplo:

$ sha256sum teste
77dbf8829dd04f33f847272c15777f52e521e4baa635a83b9
bc28a 74493efe31 teste

O SHA-256 gera um hash de 64 caracteres

SHA-1

Exemplo:

$ sha384sum teste
28dbb76eb9b839d3baa3f02634acff343b79668279a035f75
6327 5dad5579b990290fd3b4f31bf25fd9e82363e7f38b0
teste

O SHA-384 gera um hash de 96 caracteres.

SHA-1

Exemplo:

$ sha512sum teste
8cf95d98e9f3a3effb6d4274de9e77a54c330289dfacf96f7c
3c8069b829b254d8e472ad56c9762e76723f5974acaade09
3e246f9f 017bd82ebe5062cc97f847 teste

O SHA-512 gera um hash de 128 caracteres

Algoritmo	Tamanho de saída (bits)	Tamanho dos blocos (bits)	Comprimento (bits)	Tamanho das palavras (bits)	Varreduras	Operações	Colisão
SHA-0	160	512	64	32	80	+,and,or,xor,rotl	Sim
SHA-1	160	512	64	32	80	+,and,or,xor,rotl	Com falhas
SHA-256/224	256/224	512	64	32	64	+,and,or,xor,shr,rotr	Não
SHA-512/384	512/384	1024	128	64	80	+,and,or,xor,shr,rotr	Não

RIPMD

RIPMD

Criada por Hans Dobbertin, Antoon Bosselaers, e Bart Preneel em um projeto chamado RIPE (Race Integrity Primitives Evaluation) entre 1988 e1992.

Produz uma saída de 160 bits.

RIPMD

Exemplo:

$ openssl rmd160 teste
RIPEMD160(teste)=788e595dcadb4b75e20c1dbf54a18a23cf233787

O RIPMD gera um hash de 40 caracteres

WHIRLPOOL

Função criptográfica de hash desenvolvida por Paulo S. L. M. Barreto e por Vincent Rijmen (co-autor do AES).

A função foi recomendada pelo projeto NESSIE (Europeu).

Foi também adotado pelo ISO e IEC como parte do padrão internacional ISO 10118-3.

WHIRLPOOL

Exemplo:
$ sudo apt-get install md5deep
$ whirlpooldeep teste
f13697ecb3e10789449ed839f224376b633eadbe3739c07c7
843bf91a86f4374d3697924e3c396cfeb777b56d38700c41e
032c21c4fce52d5f59024969536c74 /home/aluno/teste

O whirlpool gera um hash de 128 caracteres

OPENSSL

O OpenSSL também é capaz de fazer Hash de arquivos.

Exemplo:

```
$ openssl dgst -sha1 teste
SHA1(teste)=f3b31f8f0152e016db457d8eb19f06d249b05ee2
```

```
$ openssl dgst -sha512 teste
SHA512(frederico.txt)=
1a5f2f548823adc2a6928a77c2b075d179cbf2c4b967e27d8a5582120317a687
```

COLISAO

O IDEAL DE UMA FUNÇÃO HASH É RETORNAR UMA POSIÇÃO DIFERENTE NA TABELA PARA CADA REGISTRO, MAS PODE OCORRER A OBTENÇÃO DE VALORES REPETIDOS, CHAMADOS COLISÕES.

COLISÃO OCORRE QUANDO 2 REGISTROS GERAM O MESMO HASH CODE. PARA SER BOA, UMA FUNÇÃO HASH DEVE GERAR UM CÁLCULO RÁPIDO E COM POUCAS COLISÕES.

UMA FUNÇÃO HASH IDEAL NÃO DEVE GERAR NENHUMA COLISÃO.

COLISAO

TEORIA DAS PROBABILIDADES (PARADOXO DO ANIVERSÁRIO)

DADO UM GRUPO DE 23 (OU MAIS) PESSOAS ESCOLHIDAS ALEATORIAMENTE, A CHANCE DE QUE DUAS PESSOAS TERÃO A MESMA DATA DE ANIVERSÁRIO É DE MAIS DE 50%.

PARA 57 OU MAIS PESSOAS, A PROBABILIDADE É MAIOR DO QUE 99%, ENTRETANTO, ELA NÃO PODE SER EXATAMENTE 100% EXCETO QUE SE TENHA PELO MENOS 367 PESSOAS (CONSIDERANDO QUE O ANOS PODE SER BISSEXTO).

COLISÃO

É MAIS FÁCIL CALCULAR A PROBABILIDADE P(N) DE QUE TODOS OS N ANIVERSÁRIOS SEJAM DIFERENTES SE N > 365.

PELO **PRINCÍPIO DA CASA DOS POMBOS** ESTA PROBABILIDADE É 1.

n	$p(n)$
10	12%
20	41%
23	50.7%
30	70%
50	97%
100	99.99996%
200	99.99999999999999999999999998%
300	$(1 - 7\times10^{-73}) \times 100\%$
350	$(1 - 3\times10^{-131}) \times 100\%$
367	100%

COLISÃO

Princípio da Casa dos Pombos É a afirmação de que se n pombos devem ser postos em m casas, e se n > m, então pelo menos uma casa irá conter mais de um pombo.

Matematicamente falando, isto quer dizer que se o número de elementos de um conjunto finito A é maior do que o número de elementos de um outro conjunto B, então uma função de A em B não pode ser injetora.

COLISÃO

EXEMPLO:

QUANTAS PESSOAS SÃO NECESSÁRIAS PARA SE TER CERTEZA QUE HAVERÁ PELO MENOS DUAS DELAS FAZENDO ANIVERSÁRIO NO MESMO MÊS?

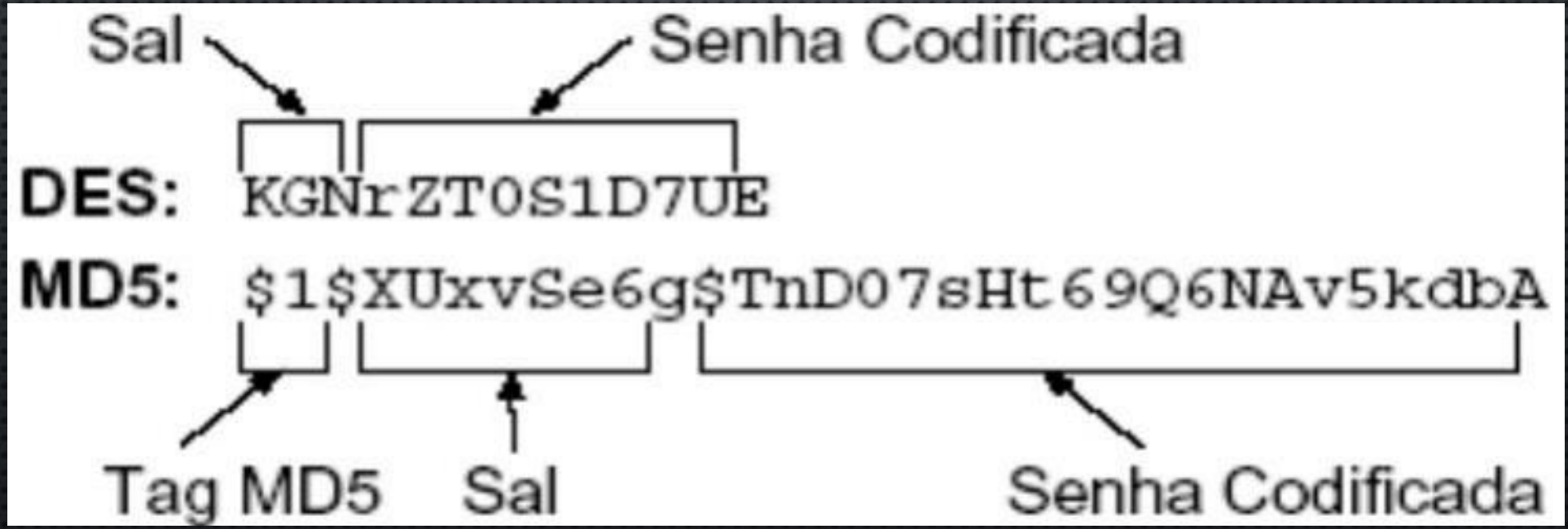
COLISAO

COMO RESOLVER O PROBLEMA DAS COLISÕES?

NÃO HÁ COMO RESOLVER O PROBLEMA, POIS A FUNÇÃO HASH É DE TAMANHO FIXO.

ASSIM, DEVE-SE SE CERTIFICAR DE USAR CADA VEZ MAIS ALGORITMOS QUE USEM MAIOR QUANTIDADE DE BITS (GERAR MAIORES CADEIAS DE CARACTERES).

HASH NO LINUX



HASH NO LINUX

```
root@xd:/# cat /etc/shadow | grep aluno
aluno:$6$b/j49.Lg$tRje4WWGiUILq6P1Autzd.r3KHeA5E3DYym/gAuBxXjizHSWqirfO
4fxA6c1qDp7ieHDZAhL3dQbLqR1l4UC50:17679:0:99999:7:::
```

HASH NO LINUX

Processo de geração de senha no Linux:

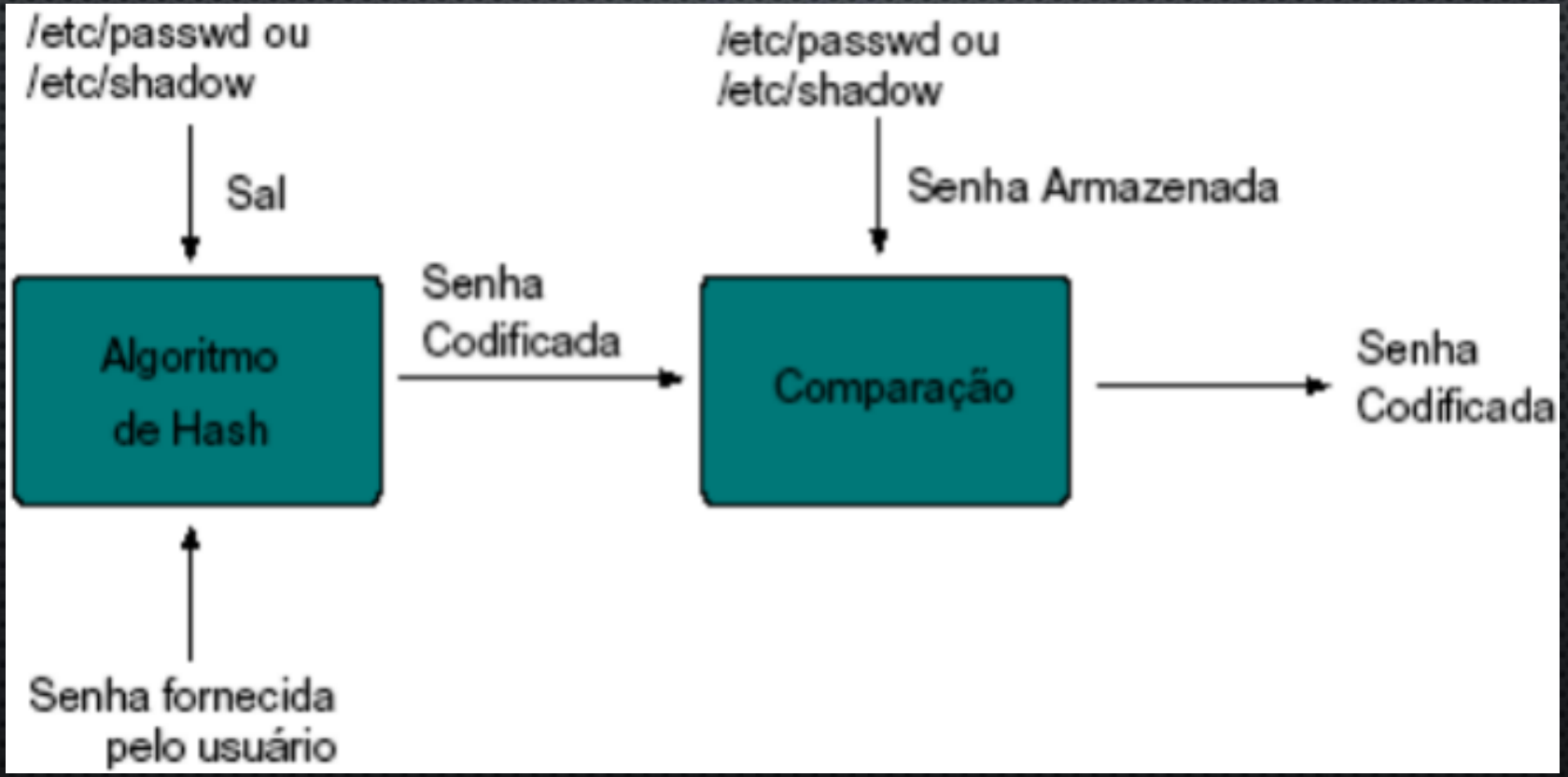
inserir a senha;

mistura com sal (salt-number);

insere-se a mistura no algoritmo hash MD5.

HASH NO LINUX

HASH NO LINUX



HASH NO LINUX

Arquivos de senha no Linux:

/etc/passwd: arquivo que contém informações gerais do usuário;

/etc/shadow: arquivo que possui todas as informações da senha do usuário.

HASH NO LINUX

```
# cat /etc/shadow | grep aluno
aluno:$6$fBkvLfoC$FctFylyYzqo3wSx00DX87neeIBLBRSv3Hg9MrMYyjX
en9qxH50f8XVW.UD6BRX0a8JXG.KssZ6By95L9AUQYM0:16093:0:99999:
7:::
```

HASH NO LINUX

```
$ perl -e 'print crypt("123456","\$6\$ fBkvLfoC \$"), "\n" '

$6$fBkvLfoC$FctFylyYzqo3wSx00DX87neeIBLBRSv3Hg9MrMYyjXen9qx
H50f8XVW.UD6BRX0a8JXG.KssZ6By95L9AUQYM0
```

HASH NO LINUX

Script Perl de “ATAQUE DE SENHAS” para gerar hash MD5 utilizando a função crypt() do Linux:

```
# perl -e 'print crypt("123456","\$1\$ fBkvLfoC \$"), "\n"' $1$ fBkvLfo$eIBzuS6K3d0/H3YnbBFaF.
```

HASH NO LINUX

```
$ perl -e 'print crypt("123456","\$6\$ fBkvLfoC \$"), "\n" '

$6$fBkvLfoC$FctFylyYzqo3wSx00DX87neeIBLBRSv3Hg9MrMY
yjXen9qx H50f8XVW.UD6BRX0a8JXG.KssZ6By95L9AUQYM0

# cat /etc/shadow | grep aluno | cut -f 2 -d:
$6$fBkvLfoC$FctFylyYzqo3wSx00DX87neeIBLBRSv3Hg9MrMY
yjXen9qx H50f8XVW.UD6BRX0a8JXG.KssZ6By95L9AUQYM0
```