# Grupo 1

## Alínea 1.1

A expressão

(?:(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])\.){3}(?:25[0-5]| 2[0-4]\d|[01]?\d\d|\d)(?:\/\d{1,2})?

Representa a correspondência de endereços IP com uma notação CIDR opcional.

As partes mais importantes são as seguintes:

* (?:\d|[01]?\d\d|2[0-4]\d|25[0-5]: corresponde aos três primeiros segmentos do endereço IP, garantindo que cada segmento seja um número válido entre 0 e 255,
* (?:\/\d{1,2})?: Esta parte opcionalmente corresponde a uma notação CIDR, onde \/ representa o caractere de barra seguido por um ou dois dígitos.

Sugestões de melhoria:

* Adicione âncoras à expressão regular para garantir que ela corresponda à *string* evitando a correspondência de endereços IP em cadeias maiores.
* Tornar a correspondência da notação CIDR mais restritiva, como por exemplo limitar o intervalo do sufixo CIDR.

Por baixo fica uma alternativa à expressão acima

^(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])(?:\.(?:\d|[01]?\d\d|2[0-4]\d|25[0-5])){3}(?:\/\d{1,2})?$

Nesta versão alternativa:

* Adicionado ^ no início e $ no final para corresponder à *string*,
* Simplifica a expressão, removendo grupos de não captura sempre que possível

## Alínea 1.2

### Pergunta a)

Usando a aplicação ***cyberchef*,** passando a seguinte mensagem binária

01010100 01110101 01110010 01101101 01100001 00100000 01100100 01100101 00100000 01010011 01010010 01000011 00100000 01100100 01100101 00100000 00110010 00110011 00101111 00110010 00110100

Obtemos a seguinte mensagem descodificada

“Turma de SRC de 23/24”

Na imagem abaixo é possível de ver um exemplo do resultado desta aplicação

Uma imagem com texto, file, captura de ecrã

Descrição gerada automaticamente

### Pergunta b)

Abaixo é possível de ver as diferenças dos diferentes conceitos

**ASCII**

* padrão de codificação de caracteres que originalmente usava 7 bits para codificar caracteres, fornecendo 128 caracteres possíveis.
* Inclui letras do alfabeto latino, dígitos, sinais de pontuação e caracteres de controle.
* Não suporta caracteres de outros idiomas ou símbolos especiais além do conjunto básico.

Unicode:

* padrão de codificação de caracteres que tem como objetivo abranger todos os caracteres de todos os sistemas de escrita do mundo.
* Atribui a cada caractere um ponto de código único, geralmente representado em notação hexadecimal (exemplo, U+0041 para a letra maiúscula "A" latina).
* Pode ser implementado usando diferentes esquemas de codificação, como UTF-8, UTF-16 ou UTF-32.

UTF-8 (Unicode Transformation Format 8-bit):

* Esquema de codificação de caracteres de largura variável para Unicode.
* Usa unidades de código de 8 bits para codificar caracteres, com o número de bytes por caractere variando de 1 a 4 bytes.
* Utilizado na internet e em sistemas de computação devido à sua compatibilidade com ASCII e eficiência na representação de caracteres.

UTF-16 (Unicode Transformation Format 16-bit):

* Esquema de codificação de caracteres de largura variável para Unicode.
* Usa unidades de código de 16 bits para codificar caracteres, com o número de unidades de código por caractere variando de 1 a 2.
* Os caracteres do Plano Multilíngue Básico (BMP), que incluem a maioria dos caracteres usados comumente, são codificados com uma única unidade de 16

Bits.

* Caracteres fora do BMP são codificados usando pares de unidades de 16 bits (pares substitutos).
* Usado em plataformas onde caracteres fora do BMP são frequentemente encontrados, como em idiomas asiáticos.

Em suma, ASCII é um padrão de codificação de caracteres básico, estando limitado ao alfabeto inglês e alguns caracteres especiais.

Unicode é um padrão mais amplo que abrange caracteres de todos os sistemas de escrita.

UTF-8 e UTF-16 são esquemas de codificação para Unicode, sendo UTF-8 mais popular na internet devido à sua compatibilidade com ASCII e eficiência, enquanto UTF-16 é frequentemente usado em ambientes que exigem suporte para caracteres fora do Plano Multilíngue Básico.

## Alínea 1.3

### Pergunta a)

Alínea i)

Usando a aplicação ***cyberchef*,** passando a seguinte mensagem em base64(devido ao facto de na última parte da mensagem, existem dois iguais ‘==’)

TmFkYSBkZSBjb3BpYXIgZGEgTmV0IG91IGRlIHVzYXIgbyBjaGF0R1BUIQ==

Obtemos a seguinte mensagem:

“Nada de copiar da Net ou de usar o chatGPT!”

Na imagem abaixo é possível de ver o exemplo criado

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Alínea ii)

O tipo de codificação usado nessa mensagem é Base64.

É usado para codificar dados binários em caracteres ASCII.

É frequentemente utilizada em cenários onde dados binários precisam ser transmitidos por meios que são projetados para lidar com dados textuais, como sistemas de email ou codificação de dados binários em URLs.

A codificação Base64 alcança isso representando dados binários usando um conjunto de 64 caracteres ASCII (daí o nome Base64), que são seguros para transmitir em protocolos baseados em texto.

No cenário para descodificar a mensagem, era necessário um descodificador Base64, onde convertia a *string* para a sua mensagem original.

Alínea iii)

Usando o mesmo código, obtemos o seguinte exemplo com os membros do grupo

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

### Pergunta b)

Alínea i)

Usando a aplicação ***cyberchef*,** passando a seguinte mensagem em base58.

wTBQfebwJvCy9Txw4o3NG3FZgPD3i73zpRTynyCBLC96NuQxhbF2K

Obtemos a seguinte mensagem

“Segurança em Redes de Computadores 2324”

Na imagem abaixo é possível de ver um exemplo feito

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Alínea ii)

O tipo de codificação usado é a base58.

### Pergunta c)

Base64 e Base58 são dois métodos de codificação comumente usados na computação para representar dados binários em texto legível.

**Tamanho do alfabeto**:

Base64 utiliza um alfabeto de 64 caracteres, composto por letras maiúsculas e minúsculas, dígitos numéricos e caracteres especiais, como '+', '/'.

Base58 utiliza um alfabeto de 58 caracteres. Isso exclui alguns caracteres que podem ser confundidos, como '0' (zero) e 'O' (letra O), '1' (um) e 'l' (letra L), e outros caracteres que podem causar confusão.

**Caracteres excluídos**:

Base58 exclui certos caracteres do alfabeto Base64 que podem ser confundidos entre si ou que são considerados problemáticos em certos contextos. Por exemplo, os caracteres '+' e '/' são comumente usados em URLs e sistemas de arquivos, então eles são excluídos no alfabeto Base58.

**Tamanho da saída**:

Devido ao seu alfabeto maior, a codificação Base64 geralmente produz uma saída mais compacta do que Base58 para a mesma entrada. Isso significa que Base64 é mais eficiente em termos de espaço quando a compactação é importante.

### Pergunta d)

**Uso:**

Base64 é amplamente usado em uma variedade de contextos, incluindo comunicações na web, armazenamento de dados binários em bancos de dados e muitos outros.

Base58 é comumente usado em cripto moedas, como Bitcoin, para representar endereços públicos de forma legível e compacta.

### Pergunta e)

Não. A criptografia envolve transformar dados em um formato que seja ininteligível sem a chave de criptografia apropriada, enquanto a codificação simplesmente representa dados em um formato diferente sem necessariamente ocultar seu significado.

Ambas as sequências estão codificadas com diferentes codificações, uma de base64 e outra de base58.

A criptografia, por outro lado, envolve a conversão de texto simples em texto cifrado usando algoritmos criptográficos e chaves. Sem a chave de criptografia, o texto original não pode ser facilmente recuperado. A criptografia é usada para garantir a segurança dos dados enquanto são armazenados ou transmitidos, enquanto a codificação é frequentemente usada para fins de representação e transformação de dados.

## Alínea 1.4

### Pergunta a)

Usando a aplicação ***cyberchef*,** passando a seguinte mensagem em base58.

37 55 71 6e 78 43 4a 4d 54 7a 4a 66 68 47 6d 4d 62 37 44 32 50 75 55 50 61 33 71 77 33 31 34 6d 7a 70 38 6a 71 41 56 38 6d 4a 44 76 5a 76 64 6a 33 62 47 66 53 79 34 68 4c 59 68 6e 69 79 43 75 5a 57 6d 6e 72 39 78 63 44 6d 55 53 53 75 52 52 47 67 76 6d 57 47 46 67 79Obtemos a seguinte mensagem

7UqnxCJMTzJfhGmMb7D2PuUPa3qw314mzp8jqAV8mJDvZvdj3bGfSy4hLYhniyCuZWmnr9xcDmUSSuRRGgvmWGFgy

Na imagem abaixo é possível de ver um exemplo feito

Uma imagem com texto, Tipo de letra, software, file

Descrição gerada automaticamente

### Pergunta b)

O tipo de codificação é hexadecimal.

## Alínea 1.5

### Pergunta a)

Usando a aplicação ***cyberchef*,** passando a seguinte mensagem em base64.

U1JDKE8gc2VncmVkbyBkaXN0byDDqSBuw6NvIHNlIGRlc2lzdGlyIGZhY2lsbWVudGUuIFNlciBwZXJzaXN0ZW50ZS4pCg==

Obtemos a seguinte mensagem

SRC(O segredo disto é não se desistir facilmente. Ser persistente.)

Na imagem abaixo é possível de ver um exemplo feito

Uma imagem com texto, software, Página web, Website

Descrição gerada automaticamente

### Pergunta b)

Transformação de texto para base64.

## Alínea 1.6

### Pergunta a)

Usando o website dCode, com a cifra de cipher, com as seguintes mensagens

1. ("NMX(Dnoj nzmqz kvmv oznoz yz phv xdamv hpdoj xjiczxdyv.")
2. POZ(Jxfp al jbpjl. X afcfzriaxab fox pryfkal klp bubozfzflp moxqfzlp.)

Para a primeira mensagem, obtemos o seguinte resultado

“("SRC(Isto serve para teste de uma cifra muito conhecida.")”

Na imagem abaixo é possível de ver o processo

Uma imagem com texto, captura de ecrã

Descrição gerada automaticamente

Para a mensagem em 2), obtemos a seguinte mensagem

SRC(Mais do mesmo. A dificuldade ira subindo nos exercicios praticos.)

Na imagem abaixo, é possível de ver o resultado

Uma imagem com texto, captura de ecrã, Página web, comida

Descrição gerada automaticamente

Para ambos os casos, foi necessária fazer o processo 25 vezes.

## Alínea 1.7

### Pergunta a)

Usando a mensagem, vai ser codificado em MD5.

"SRC(Vamos ver os hashes.)”

Obtém se o seguinte valor

d7cf6746ebf88b48e5223429df2121fd

É possível de ver na imagem seguinte o exemplo aplicado

Uma imagem com captura de ecrã, texto, software

Descrição gerada automaticamente

### Pergunta b)

Usando a mensagem SRC(Vamos ver os hashes.)

SRC(Vamos ver os hashes.)

Obtém se o seguinte resultado

38924ba0d78463eabb547c16a8ebbe43

É possível de ver na imagem seguinte o exemplo aplicado

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Comparando os dois resultados, é possível de ver que os hashes MD5 são diferentes.

Essa discrepância ocorre devido a pequenas diferenças nas entradas de cada frase, como espaços, pontuação ou letras maiúsculas.

Nesse caso, os hashes são diferentes, embora as *strings* de entrada pareçam idênticas.

Isto demonstra a sensibilidade do MD5 a alterações em cada frase, razão pela qual o MD5 não é recomendado para fins criptográficos devido à sua vulnerabilidade a ataques de colisão.

No entanto, para fins não criptográficos, (somas de verificação ou pesquisas em tabelas hash) o MD5 ainda pode ser adequado.

### Pergunta c)

Usando o dcode, obtemos os seguintes resultados

Uma imagem com texto, Tipo de letra, captura de ecrã

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Como é possível de ver, os resultados são idênticos usando o MD5 em ambos os casos.

Experimentando com o algoritmo MD4 obtemos o seguinte resultado

Uma imagem com texto, Tipo de letra, número, captura de ecrã

Descrição gerada automaticamente

Alterando o algoritmo é possível de ver que a mensagem codificada é diferente, quando usando diferentes algoritmos. Apenas é de notar que o começo de ambas as mensagens é a mesma( “3ª…”).

### Pergunta d)

Tentado descodificar ambas as mensagens na alínea a), é possível de ver que não é possível voltar para a mensagem original.

Tal facto deve-se ao facto de o MD5 não conseguir "descodificar" a mensagem de volta para o original. MD5 é uma função hash unidirecional, o que significa que ela converte dados de entrada em uma sequência de bytes de tamanho fixo (no caso do MD5, 128 bits ou 16 bytes), mas não é possível reverter esse processo para obter os dados originais a partir do hash.

A função MD5 é projetada para ser computacionalmente difícil de reverter. Isso significa que, dada uma sequência de hash, é extremamente difícil (teoricamente impossível em uma prática viável) encontrar uma entrada que gere esse hash específico. Portanto, o MD5 é usado principalmente para verificar a integridade dos dados ou para fins de verificação, não para criptografia ou segurança de dados.

### Pergunta e)