Tutorial #1

João Freitas up202100373
Rui Gonçalves up202103077

O presente relatório tem como objetivo descrever o processo seguido para a resolução da ficha `Tutorial #1`, disponibilizada no âmbito da disciplina de Criptografia Aplicada. As seções numeradas em baixo representam cada um dos exercícios resolvidos.

Cifras clássicas

A temática do presente problema devolve-se no uso de cifras clássicas para esconder o conteúdo original de mensagens. Para tal é proposto a decifração de três mensagens cifradas usando métodos estatísticos e Vigenère:

1) Substituição Mono Alfabética

A primeira mensagem cifrada corresponde a uma "simples" substituição mono alfabética (Cifra de Caesar), sendo o alfabeto tradicional utilizado (`abcdefghijklmnopqrstuvwxyz`) e a língua portuguesa na criação da mensagem. Nesta substituição primeiramente é "baralhado" o alfabeto, de forma a criar outro alfabeto não reconhecido pela mente humana. Seguidamente, é aplicado o alfabeto produzido à mensagem original, substituíndo as letras da mesma conforme estas estão posicionadas no alfabeto original, tal como demonstrado na Figura 1.

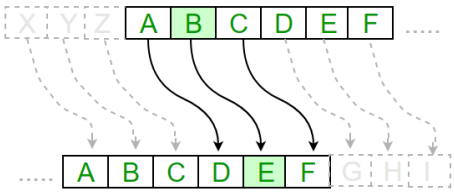

Figura 1 - Exemplo de substituição das letras do alfabeto (Cifra de Caesar)

Para recuperar a mensagem original, é preciso obter o alfabeto produzido para baralhar a mensagem. Como sabemos que a mensagem foi escrita sobre a língua portuguesa, então é possível inferir o alfabeto produzido ao determinar o número de ocorrências de cada uma das letras da mensagem cifrada e comparar com o número de ocorrências das letras num dicinário português. Assim, comparando as ocorrências, obtém-se com grande confiança qual letra cifrada corresponde a cada letra do alfabeto original.

```
# /** Help functions can be found in the original source file (inside .zip file)**/

cipher_msg_alphabet_list = list(map(to_ascii_map, msg1))

cipher_msg_alphabet_map = dict(list(map(lambda x: to_freq_tuple(x, cipher_msg_alphabet_list), cipher_msg_alphabet_

cipher_msg_alphabet_map_sort_desc = sorted(cipher_msg_alphabet_map.items(), key=sort_desc)

ptdict_letters_map_sort_desc = sorted(Stats.items(), key=sort_desc)

letters_table_sort_desc = table(rows=list(map(tuple_to_row, ptdict_letters_map_sort_desc)), header_row=["Letter (P

message_table_sort_desc = table(rows=list(map(tuple_to_row, cipher_msg_alphabet_map_sort_desc)), header_row=["Lett

msg1_as_string = reduce(to_ascii_fold, map(to_ascii_map, msg1))

replace = [('x', 'a'), ('r', 'e'), ('p', 'o'), ('q','r'), ('u','s'),('d','p'),('f','t'),('w','d'), ('v','h'),
           ('a','i'), ('y','n'), ('o','k'), ('c','m'), ('t','u'), ('b','g'),('n','j'),('s','c'),('h','v'),('g','f'
           ('l','l'),('j','b'),('i','q'),('z','x'),('k','z')]

msg1_decoded = reduce(to_ascii_fold, map(lambda x: replace_char(x, replace), msg1_as_string))

print("[*] FREQUÊNCIAS DO ALFABETO DA LINGUA PORTUGUESA\n")
display(letters_table_sort_desc)

print("[*] FREQUÊNCIAS DO ALFABETO DA MENSAGEM (Msg2)\n")
```

```
display(message_table_sort_desc)

print("[*] Mensagem 1 (descodificada):\n")
print(msg1_decoded)
```

Recorrendo à linguagem de programação Python e ferramenta SageMath, foi desenvolvido um *script* que carrega a mensagem cifrada e ocorrências das letras num dicionário Português, de forma a gerar as tabelas de ocorrências das letras nos dois alfabetos em questão. Primeiramente procurou-se mostrar no ecrã a tabela da frequência de ocorrência das letros do alfabetos, ordenadas pelo número de ocorrências (funções `display`). Desta forma foi possível realizar associações entre cada letra de uma forma muito mais fácil.

De seguida, optou-se por substituir todas as letras da mensagem que não se soubesse ainda qual a letra do alfabeto original que a representa pelo caractér "-" (função `replace_char`), de forma que a nossa mente percebesse mais fácilmente a mensagem original. Desta forma e através de um processo iterativo *trial and error*, foi-se substituindo cada uma das letras mais prováveis até se chegar a um consenso da representação das **8** primeiras letras mais frequentes no alfabeto produzido (`xrpqudfw`). Tendo a base do alfabeto produzido e mais uma vez seguindo o mesmo processo iterativo, conseguiu-se encontrar o alfabeto original e decifrar a mensagem.

```
ohojeforemrecebidospeloprimeiroministrodemitterrandosecologistastencionamaceitarasdesculpasesaudarasuapromessa
deapoioaumareservamundialnaantartidamasquenadajustificaaintencaodecontinuarcomasexperienciasdematerialatomicop
elofactodegeorgebushvoltaraapresentarsetradicaonopartidentrecatolicoseprotestantesnairlandadonorteaspalavrassu
bstituemhojeasbalaseosdisparosdemorteiroosinimigoscentenariosvaoconversarsobamediacaodepeterbrookeoministrodel
ondrestransformadoemheroitemdezsemanasparaencontrarumasolucaoqueacabecomumaguerracivildedecadasministrobritani
coparaairlandadonortedesdevercaixabrookevaitentarcomquatropartidosdaprovinciachegaraacordoquantoaumanovaformad
eadministracaodairlandadonortequeexcluaogovernodelondresborisieltsincandidatoapresidenciadarepublicarussaemele
icoesmarcadasparadejunhochegouontemasiberiaocidentalparatentarconvencerosmineirosdokuzbassasuspenderemagrevequ
edurahadoismesesaomesmotempoemmoscovomilpessoasmanifestaramachuvaapoioasuacandidaturaapesardascriticasdealgumasint
```

2) Cifra de Vigenère

A segunda mensagem foi cifrada com a cifra de Vigenère. Esta cifra é mais complexa do que a Caesar, pois utiliza uma substituição polialfabética, ou seja, multiplos alfabetos. Para a utilizar, primeiro temos que construir uma matriz que contém os alfabetos repetidos *N* vezes, sendo N o número de letras do alfabeto. Depois, é produzido uma chave de tamanho menor ou igual ao tamanho da mensagem a cifrar, que irá servir de guia da cifracação. O alfabeto descrito horizontalmente na matriz (i.e., rows) representa o alfabeto original, sendo que o alfabeto descrito verticalmente representa o alfabeto da chave. Por final, para cada uma das letras do alfabeto e chave, é encontrado a letra na matriz que corresponde à letra da mensagem cifrada.

Uma vez que não se conhece a chave usada na cifra, nem o seu tamanho, não é possível decifrar a mensagem ao "refazer" o algoritmo da cifra. Contudo é possível aplicar o teste de Friedman e determinar o período, ou seja, o tamanho da chave. Através de shifts sucessivos na mensagem original e representar os valores do teste Friedman num gráfico de barras, reparamos que a cada 5 shifts, o valor destaca-se dos restante. Com isto podemos afirmar que a chave tem um período/tamanho de 5 letras.

Uma vez conhecido o período da chave, podemos dividir o texto em colunas do tamanho da mesma. Neste caso o texto dividiu-se em 5 colunas. Tendo as colunas, ao determinarmos a letra que com o maior número de ocorrências, podemos afirmar que essa letra faz parte da chave usada para cifrar a mensagem.

A chave obtida através das colunas é `gqsfg`. Aplicando a mesma sobre uma função que decifra a cifra Vigenère, obtemos a seguinte mensagem decifrada:

```
OHOJEFOREMRECEBIDOSPELOPRIMEIROMINISTRODEMITTERRANDOSECOLOGISTASTENCIONAMACEITARASDESCULPASESAUDARASUAPROMESSADEAP
OIOAUMARESERVAMUNDIALNAANTARTIDAMASQUENADAJUSTIFICAAINTENCAODECONTINUARCOMASEXPERIENCIASDEMATERIALATOMICOPELOFACTO
DEGEORGEBUSHVOLTARAAPRESENTARSETRADICAONOPARTIDENTRECATOLICOSEPROTESTANTESNAIRLANDADONORTEASPALAVRASSUBSTITUEMHOJE
ASBALASEOSDISPAROSDEMORTEIROOSINIMIGOSCENTENARIOSVAOCONVERSARSOBAMEDIACAODEPETERBROOKEOMINISTRODELONDRESTRANSFORMA
DOEMHEROITEMDEZSEMANASPARAENCONTRARUMASOLUCAOQUEACABECOMUMAGUERRACIVILDEDECADASMINISTROBRITANICOPARAAIRLANDADONORT
EDESDEVERCAIXABROOKEVAITENTARCOMQUATROPARTIDOSDAPROVINCIACHEGARAACORDOQUANTOAUMANOVAFORMADEADMINISTRACAODAIRLANDAD
ONORTEQUEEXCLUAOGOVERNODELONDRESBORISIELTSINCANDIDATOAPRESIDENCIADAREPUBLICARUSSAEMELEICOESMARCADASPARADEJUNHOCHEG
OUONTEMASIBERIAOCIDENTALPARATENTARCONVENCEROSMINEIROSDOKUZBASSASUSPENDEREMAGREVEQUEDURAHADOISMESESAOMESMOTEMPOEMMO
SCOVOMILPESSOASMANIFESTARAMACHUVAAPOIOASUACANDIDATURAAPESARDASCRITICASDEALGUMASINTERVENC
```

O código desenvolvido para concluir a decifração encontra-se no caderno Jupyter `week1.ipynb`.