PROJETO 2: λ − CÁLCULO E CRIPTOGRAFIA

Adrielle A. Carvalho[1], Fernando F. L. Neto[1], João L. L. Melo[1], Thiago V. S. Andrade[1]

[1]Bacharelado em Ciência da Computação – Universidade Federal da Bahia (UFBA)
Salvador – BA – Brasil

adrielle.andrade@ufba.br, fernando.franco@ufba.br, joaollm@ufba.br, thiago.vieira@ufba.br

Abstract. *This project aims to develop a formalization of three encryption methods, namely a transposition encryption method and two different substitution encryption methods, and their implementations in Haskell, in order to protect patient data in multiple health systems.*

Resumo. *Esse projeto visa desenvolver uma formalização de três métodos de ciframento, sendo eles um método de ciframento por transposição e dois métodos distintos de ciframento por substituição e suas implementações em Haskell, com o objetivo de proteger dados de pacientes em múltiplos sistemas de saúde.*

1. Informações Gerais

De acordo com a Tese de Church-Turing, todos os problemas computáveis devem ser capazes de ser representados em máquinas de Turing. Ademais, Alan Turing também provou que existe uma equivalência entre máquinas de Turing e λ − cálculo. Portanto, todo problema que é computável em um desses modelos deve ser computável no outro. Dessa forma, o projeto foi desenvolvido utilizando as cifras Autokey, Vigenere e Rail Fence, seguindo as orientações dadas.

Nesse sentido, a cifra Rail Fence é uma cifra clássica de transposição. Esse tipo de cifra realiza a reorganização das letras no texto simples a fim de criptografá-lo. O modelo Rail Fence é um método bastante antigo e não muito seguro, no qual o texto deve ser escrito sem espaços diagonalmente em diferentes linhas do texto, chamadas de *rails*. Quando o texto chega na última *rail,* ele deve passar a ser escrito na diagonal em direção ao *rail* superior e, em seguida, diagonalmente até o *rail* inferior repetidas vezes até que o texto esteja todo dessa forma.

Diferente das cifras de transposição que apenas rearranjam as unidades do texto em uma ordem diferente sem as modificar, as cifras de substituição mantém a ordem original e operam com sistemas pré-definidos para substituição dos elementos no texto, podendo ser letras isoladas, pares ou grupos. As cifras Autokey e Vigenère pertencem ao segundo tipo.

A cifra de Vigenère é uma cifra polialfabética, a qual faz uso de diversas cifras de César de forma sequencial, com vários valores de deslocamento determinados por uma palavra-chave. É uma versão mais simples da cifra de substituição polialfabética inventada por Leon Alberti em 1945. Essa cifra possui um grau maior de segurança, podendo ser vista como indecifrável à primeira vista. Na cifra de César, cada letra deve ser descolada um número fixo de posições, enquanto na cifra de Vigenère cada letra da palavra-chave indica a linha da tabela que deve ser utilizada para cifrar um elemento do texto original, sendo essa tabela formada pelas 26 possíveis cifras de César.

Já a cifra Autokey, ou Autochave, é também uma cifra polialfabética e é considerada bastante semelhante a cifra de Vigenère, todavia, seu método de escolha de chaves é diferente. Nela, a chave não é escolhida e sim gerada automaticamente através da seleção de

algumas letras no texto ou adicionando uma chave no começo. Em seguida, utiliza-se a mesma técnica usada na cifra de Vigenère, todavia, não é necessário realizar uma repetição da chave. Nesse sentido, essa cifra possui um grau de segurança mais alto, sendo bem mais segura do que outras cifras de substituição polialfabética devido a não repetição da chave.

2. Funcionamento das cifras

Nesta seção, serão exibidos exemplos de como cada cifra funciona, baseando-se na construção do código.

2.1. Rail Fence

Serão testadas três entradas de texto claro a serem criptografadas. Uma vez submetidos ao processo, os textos retornados serão usados como entradas para descriptografia. Para cada entrada, será atribuída uma quantidade distinta de trilhas.

Todas as mensagens serão convertidas para sentenças com caracteres maiúsculos. A nível de implementação, o algoritmo em haskell aceita mensagens com espaços em branco. No entanto, para os testes, iremos concatenar as palavras do texto claro. As mensagens retornadas após as operações de criptografia e descriptografia serão representadas por caracteres alfabéticos sem espaços em branco, respeitando a capitalização do texto passado como entrada para as operações.

As mensagens a serem criptografadas serão:

1. “meet me after the toga party”
2. “test me”
3. “making my own plans”

2.1.1. “meet me after the toga party”

Alocando os caracteres em duas trilhas, temos a seguinte disposição:

```
m e m a t r h t g p r y
  e t e f e t e o a a t
```

Nesse caso, após o processo de criptografia em duas trilhas do texto claro, o texto criptografado corresponde a “mematrhtgpryetefeteoaat”. A descriptografia do texto retornado deve corresponder a “meetmeafterthetogaparty”.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros o inteiro *2* como quantidade de trilhas e o texto claro *“meetmeafterthetogaparty”*, obtemos como retorno:

```
ghci> encode 2 "meetmeafterthetogaparty"
"mematrhtgpryetefeteoaat"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros o inteiro *2* como quantidade de trilhas e o texto claro *“mematrhtgpryetefeteoaat”*, obtemos

como retorno:

```
ghci> decode 2 "mematrhtgpryetefeteoaat"
"meetmeafterthetogaparty"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.1.2. “test me”

Alocando os caracteres em duas trilhas, temos a seguinte disposição:

```
t s m
 e t e
```

Nesse caso, após o processo de criptografia em duas trilhas do texto claro, o texto criptografado corresponde a “testme”. A descriptografia do texto retornado deve corresponder a “tsmete”.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros o inteiro *2* como quantidade de trilhas e o texto claro *“testme”*, obtemos como retorno:

```
ghci> encode 2 "testme"
"tsmete"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros o inteiro *2* como quantidade de trilhas e o texto claro *“tsmete”*, obtemos como retorno:

```
ghci> decode 2 "tsmete"
"testme"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.1.3. “making my own plans”

Alocando os caracteres em três trilhas, temos a seguinte disposição:

```
m   n   o   l
 a i g y w p a s
  k   m   n   n
```

Nesse caso, após o processo de criptografia em duas trilhas do texto claro, o texto criptografado corresponde a “makingmyownplans”. A descriptografia do texto retornado deve corresponder a “mnolaigywpaskmnn”.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros o inteiro *3* como quantidade de trilhas e o texto claro *“makingmyownplans”*, obtemos como retorno:

```
ghci> encode 3 "makingmyownplans"
"mnolaigywpaskmnn"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros o inteiro *3* como quantidade de trilhas e o texto claro *"mnolaigywpaskmnn"*, obtemos como retorno:

```
ghci> decode 3 "mnolaigywpaskmnn"
"makingmyownplans"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.2. Cifra de Vigenère

Serão testadas três entradas de texto claro a serem criptografadas. Uma vez submetidos ao processo, os textos retornados serão usados como entradas para descriptografia. Para cada entrada, será atribuída uma chave distinta.

Todas as mensagens serão convertidas para sentenças com caracteres maiúsculos. Para os testes, iremos concatenar as palavras do texto claro. As mensagens retornadas após as operações de criptografia e descriptografia serão representadas por caracteres alfabéticos maíusculos sem espaços em branco.

As mensagens a serem criptografadas serão:

1. "meet me after the toga party", cuja chave será "pdf"
2. "test me", cuja chave será "test"
3. "making my own plans", cuja chave será "olsen"

Vale ressaltar que a tabela de ciframento da Cifra de Vigenère é dada por:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

2.2.1. “meet me after the toga party”

Consultando a tabela em função da chave “PDF”, temos que a criptografia da mensagem “MEETMEAFTERTHETOGAPARTY” corresponde à “BHJIPJPIYTUYWHYDJFEDWIB”. A descriptogtrafia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“PDF”* e o texto claro *“MEETMEAFTERTHETOGAPARTY”*, obtemos como retorno:

```
ghci> encode "PDF" "MEETMEAFTERTHETOGAPARTY"
"BHJIPJPIYTUYWHYDJFEDWIB"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“PDF”* e o texto criptografado *“BHJIPJPIYTUYWHYDJFEDWIB”*, obtemos como retorno:

```
ghci> decode "PDF" "BHJIPJPIYTUYWHYDJFEDWIB"
"MEETMEAFTERTHETOGAPARTY"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.2.2. “test me”

Consultando a tabela em função da chave “TEST”, temos que a criptografia da mensagem “TESTME” corresponde à “MIKMFI”. A descriptogtrafia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“TEST”* e o texto claro *“TESTME”*, obtemos como retorno:

```
ghci> encode "TEST" "TESTME"
"MIKMFI"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“TEST”* e o texto criptografado “MIKMFI”, obtemos como retorno:

```
ghci> decode "TEST" "MIKMFI"
"TESTME"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.2.3. “making my own plans”

Consultando a tabela em função da chave “OLSEN”, temos que a criptografia da

mensagem “MAKINGMYOWNPLANS” corresponde à “ALCMAUXQSJBADEAG”. A descriptogtrafia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“OLSEN”* e o texto claro *“MAKINGMYOWNPLANS”*, obtemos como retorno:

```
ghci> encode "OLSEN" "MAKINGMYOWNPLANS"
"ALCMAUXQSJBADEAG"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“OLSEN”* e o texto criptografado *“ALCMAUXQSJBADEAG”*, obtemos como retorno:

```
ghci> decode "OLSEN" "ALCMAUXQSJBADEAG"
"MAKINGMYOWNPLANS"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.3. Cifra Autokey

Serão testadas três entradas de texto claro a serem criptografadas. Uma vez submetidos ao processo, os textos retornados serão usados como entradas para descriptografia. Para cada entrada, será atribuída uma chave distinta.

Todas as mensagens serão convertidas para sentenças com caracteres maiúsculos. Para os testes, iremos concatenar as palavras do texto claro. As mensagens retornadas após as operações de criptografia e descriptografia serão representadas por caracteres alfabéticos maiúsculos sem espaços em branco.

As mensagens a serem criptografadas serão:

4. “meet me after the toga party”, cuja chave será “pdf”
5. “test me”, cuja chave será “test”
6. “making my own plans”, cuja chave será “olsen”

Vale ressaltar que a tabela de ciframento da Cifra de Vigenère é dada por:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

2.2.1. “meet me after the toga party”

Consultando a tabela em função da chave “PDF”, aplicado o método de incorporação da chave à mensagem, temos que a criptografia da mensagem “MEETMEAFTERTHETOGAPARTY” corresponde à “BHJFQITRXEWMLVMVKTDGRIY”. A descriptogtrafia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“PDF”* e o texto claro *“MEETMEAFTERTHETOGAPARTY”*, obtemos como retorno:

```
ghci> encode "PDF" "MEETMEAFTERTHETOGAPARTY"
"BHJFQITRXEWMLVMVKTDGRIY"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“PDF”* e o texto criptografado *“BHJFQITRXEWMLVMVKTDGRIY”*, obtemos como retorno:

```
ghci> decode "PDF" "BHJFQITRXEWMLVMVKTDGRIY"
"MEETMEAFTERTHETOGAPARTY"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.2.2. “test me”

Consultando a tabela em função da chave “TEST”, aplicado o método de incorporação da chave à mensagem, temos que a criptografia da mensagem “TESTME” corresponde à “MIKMFI”. A descriptogtrafia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“TEST”* e o texto claro *“TESTME”*, obtemos como retorno:

```
ghci> encode "TEST" "TESTME"
"MIKMFI"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“TEST”* e o texto criptografado “MIKMFI”, obtemos como retorno:

```
ghci> decode "TEST" "MIKMFI"
"TESTME"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

2.2.3. “making my own plans”

Consultando a tabela em função da chave “OLSEN”, temos que a criptografia da

mensagem “MAKINGMYOWNPLANS” corresponde à “ALCMASMIWJTBJOJF”. A descriptografia do texto retornado, por sua vez, deve retornar o texto claro dado como entrada.

Invocando a função *encode* do algoritmo em haskell, passando como parâmetros a chave *“OLSEN”* e o texto claro *“MAKINGMYOWNPLANS”*, obtemos como retorno:

```
ghci> encode "OLSEN" "MAKINGMYOWNPLANS"
"ALCMASMIWJTBJOJF"
```

Invocando a função *decode* do algoritmo em haskell, passando como parâmetros a chave *“OLSEN”* e o texto criptografado *“ALCMASMIWJTBJOJF”*, obtemos como retorno:

```
ghci> decode "OLSEN" "ALCMASMIWJTBJOJF"
"MAKINGMYOWNPLANS"
```

Como queríamos mostrar, o texto claro foi criptografado e seu retorno foi descriptografado corretamente pelo algoritmo.

3. Conclusões

Através do estudo dos conceitos básicos de criptografia, dos métodos de substituição e transposição, compreensão da linguagem de programação Haskell e do funcionamento do paradigma de programação funcional, foi possível desenvolver as implementações dos algoritmos de criptografia Vigenère, Autokey e Rail Fence.

A partir das implementações do trabalho, conseguimos dialogar com discussões a respeito de λ - cálculo. O paradigma de programação funcional utilizado no desenvolvimento dos algoritmos serve como base para a conversão das funções criadas em termos lambda. Dessa forma, é possível perceber o poder computacional do modelo de λ - cálculo, como discutido em sala.

Por fim, as implementações dos algoritmos de criptografia Vigenère, Autokey e Rail Fence em Haskell deste trabalho contemplam as requisições de paradigma de programação funcional, dialogando com as discussões sobre λ - cálculo, e atendem corretamente os principais conceitos de criptografia de suas respectivas cifras.

4. Referências

Autokey Cipher - Cifras Simétricas, Acervo Lima. Disponível em <http://www.decom.ufop.br/romildo/2014-2/bcc222/>. Acesso em: 18 de novembro de 2022.

Cifra de autochave - Autokey Cipher (2019), Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/Cifra_de_autochave>. Acesso em: 18 de novembro de 2022.

Cifra de Vigenère (2022), Wikipedia. Disponível em: <https://pt.wikipedia.org/wiki/Cifra_de_Vigen%C3%A8re>. Acesso em: 18 de novembro de 2022.

Criptografia e Segurança em Rede Técnicas Clássicas de Encriptação - Universidade de São Paulo. Disponível em: <http://wiki.stoa.usp.br/images/c/cf/Stallingscap2e3.pdf>. Acesso

em: 18 de novembro de 2022.

Hopcroft, J. E.; Ullman, J. D.; Motwani, R. (2002) “Introdução à teoria de autômatos, linguagens e computação”, Editora Campus.

Junior, M. (2014) “Programação Funcional em Haskell”. Disponível em: <http://www.decom.ufop.br/romildo/2014-2/bcc222/>. Acesso em: 14 de novembro de 2022.

Ashutosh, K. Rail Fence Cypher - criptografia e descriptografia, Acervo Lima. Disponível em: <https://acervolima.com/rail-fence-cipher-criptografia-e-descriptografia/>. Acesso em: 19 de novembro de 2022

Sipser, M. (2007) “Introdução à teoria da computação”, Thomson Learning.

Vieira, N. J. (2004) “Linguagens e Máquinas: Uma Introdução aos Fundamentos da Computação”.