CRIPTOGRAFIA APLICADA À INDUSTRIA 4.0

Gabriel Ribeiro Paula (IC) e Mario Sergio Correa dos Santos (Orientador)

Apoio:PIBIC Mackenzie

RESUMO

Com a constante inovação tecnológica, a cada momento surgem novas ideias e, consequentemente, novos conhecimentos tecnológicos em todas as áreas. Dentre essas inovações, surgiu a indústria 4.0, com o propósito de integrar internet, tecnologia e sistema de produção. Concomitantemente com os benefícios propiciados por esses avanços, surgiram desafios referente à segurança de dados. Esta pesquisa teve como objetivo a criação de um novo método aplicável às indústrias, capaz de possibilitar a comunicação segura, rápida e eficaz entre diversos equipamentos, e setores, de modo comprovadamente invulnerável. Este artigo está dividido em etapas, a princípio independentes, que consistem na base do programa, as quais são integradas para a realização de uma função específica. A base do programa foi desenvolvida a partir da cifra de César, determinada por um deslocamento padrão na posição das letras do alfabeto. Com o progresso no desenvolvimento do programa, foi aplicado o método de criptografia, conhecido como bloco de cifras de uma única vez, o qual consiste em uma forma mais complexa da cifra de César. Ao final da pesquisa, foi obtido um método, e o mesmo se mostrou capaz de criptografar dados, de forma, já comprovada matematicamente, e o mesmo se mostrou capaz de criptografar dados, de forma, já comprovada matematicamente como sendo indecifrável, a partir de uma fonte aleatória de dados.

Palavras-chave: Criptografia. Indústria 4.0. Segurança de dados.

ABSTRACT

With constant technological innovation, new ideas and, consequently, new technological knowledge arises in every area. Among all these innovations, the 4.0 industry has the purpose of integrating the internet, technology and production system. Concurrently with the benefits of these advances, data security challenges have emerged. This research aimed at the creation of a new method applicable to the industries, able to ensure safety, speedy and effectiveness communication among several equipment, sectors, and in a way that is proven to be invulnerable. This article is divided into independent steps, at first independent, which are the basis of the program, that are integrated to perform a specific function. The basis of the program was developed from the Caesar cipher, determined by a standard displacement in the position of the letters of the alphabet. As the program progressed, a new encryption

method was applied, known as the one-time pad, which consists of a more complex form of the Caesar cipher. At the end of the research, a method was obtained and it was able to encrypt data, already proven mathematically, perfectly safe, from a random source of data.

Keywords: Cryptography. 4.0 industry. Data security.

1. INTRODUÇÃO

O termo “Industria 4.0” – faz alusão à uma suposta quarta revolução industrial – foi criado para descrever técnicas que, com base na internet industrial, permitam aprimorar a eficiência de um sistema de produção ou de uma cadeia produtiva.

> “O termo Indústria 4.0 foi primeiramente utilizado durante a Hannover Fair, em 2011, onde foi proposta uma nova tendência industrial com o desenvolvimento de “smart factories”. As ditas “smart factories” relacionam e articulam sistemas virtuais e físicos que, combinados a redes e plataformas digitais com viabilidade de abrangência globais, proporcionam cadeias de valor revolucionárias.” (TADEU, 2016)

Uma das principais características da indústria 4.0 é a automação e produção flexível, já que a indústria do futuro funcionará com a integração de sistemas digitais e mecânicos, tendo como controvérsia um desafio ainda maior relacionado à segurança de dados.

Esta pesquisa tem como finalidade a criação de um novo método, através de um programa, que permitirá a comunicação segura, rápida e eficaz entre diversos equipamentos, setores, e de modo comprovadamente invulnerável.

2. REFERENCIAL TEÓRICO

Esta seção apresenta o referencial teórico do trabalho, a primeira parte aborda o tema da criptografia e como os números aleatórios podem ser utilizados para codificar mensagens. A segunda parte aborda o tema do teste estatístico utilizado para a verificação da aleatoriedade, além de algumas definições importantes para o contexto, as quais são esclarecidas a diante.

Criptografa

A criptografia consiste em um mecanismo de segurança, responsável por tornar informações inatingíveis, sendo que apenas o destinatário autorizado, dotado da chave, deve ser capaz de extrair o conteúdo da mensagem da sua forma cifrada.

Uma chave criptográfica consiste em um valor específico, que interage com o algoritmo de encriptação. Como exemplo, podemos comparar a fechadura de uma porta, a qual possuí uma série de pinos, sendo que cada um desses pinos pode ocupar múltiplas posições possíveis. Ao posicionar a chave na fechadura, cada um dos pinos é posicionado em uma posição específica. Caso as posições ocupadas pela chave coincidam com as que a

fechadura precisa para ser aberta, ela abre, caso contrário, não. O mesmo ocorre com o conteúdo cifrado, podendo este ser ou não decodificado, dependendo da chave usada.

> "(...) criptografia, derivada da palavra grega kriptos, que significa "oculto". O objetivo da criptografia não é ocultar a existência de uma mensagem, e sim esconder o seu significado – um processo conhecido como encriptação. Para tornar a mensagem incompreensível, o texto é misturado de acordo com um protocolo especifico, que já foi estabelecido previamente por ambos transmissor e receptor. Assim, o receptor da mensagem pode reverter o protocolo misturador e tornar a mensagem compreensível. "(SINGH, 2007)

O processo inverso da criptografia é denominado descriptografia, o qual ocorre quando um texto já encriptado, é decodificado para o texto original através de uma chave previamente conhecida, sendo assim decifrado.

A seção abaixo expõe uma técnica antiga para a realização do processo de criptografar e descriptografar, a qual foi fundamental para o desenvolvimento do programa proposto.

Cifra de César

A cifra de César consiste na substituição de cada letra do alfabeto por aquela localizada um determinado número de posições adiante, como demonstrado na tabela 1, onde pode ser observado um alfabeto codificado com a cifra de Cesar e um deslocamento de 3 posições.

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**
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

Tabela 1 – Exemplo de alfabeto codificado com a cifra de Cesar.
Fonte: Elaborado pelo autor.

A cifra de César, entretanto, pode ser facilmente decodificada, por ser monoalfabética existem apenas 25 combinações possíveis que podem ser testadas em pouco tempo. A próxima seção descreve uma técnica que apesar de ser parecida com a cifra de César permite melhorar sua segurança.

Bloco de cifras de uma única vez

Além do método de criptografia de César, existe também a técnica do “bloco de cifras de uma única vez”. Essa técnica consiste na utilização de um alfabeto exclusivo para cada letra contida na mensagem, mesmo que esta se repita, o seu código não será o mesmo. Ou seja, a técnica é semelhante à utilizada na cifra de Cesar, porém com uma quantidade diferente de deslocamentos para cada uma das letras da mensagem.

Como exemplo temos:

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
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

Tabela 2 – Exemplo de alfabeto codificado com o método bloco de cifras de uma única vez, com deslocamento de 5 letras.
Fonte: Elaborado pelo autor.

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
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

Tabela 3 – Exemplo de alfabeto codificado com o método bloco de cifras de uma única vez, com deslocamento de 100 letras, equivalente ao de 22 letras.
Fonte: Elaborado pelo autor.

Através dessa técnica é possível alcançar a segurança perfeita, já comprovada matematicamente, para isso a chave utilizada deve ser perfeitamente aleatória. A próxima seção apresenta um teste estatístico usado para verificar a aleatoriedade da chave.

Distribuição de Benford

A distribuição de Benford teve como base estudos de Simon Newcomb, astrônomo e matemático, responsável pela publicação de um artigo em 1881, no qual evidenciou um padrão no desgaste das bordas presentes em um livro de logaritmos. Identificando uma diminuição característica desse desgaste com o decréscimo das páginas.

Apenas em 1938, Frank Benford, físico, publicou um artigo descrevendo o mesmo fenômeno descrito anteriormente por Newcomb. Benford relatou sobre mais de 20 mil dados observados e analisados por ele em diferentes amostras de diversas fontes – como distâncias entre rios, estatísticas de beisebol, números de endereços, entre outras (HILL, 1996, 1998, 1999).

A distribuição de Benford se dá nos ditos dígitos significativos, isto é, todos os algarismos com exceção dos zeros localizados a esquerda, independente do tamanho do número analisado. Ao analisar o primeiro algarismo significativo de alguma amostras, Benford notou experimentalmente que, a probabilidade dos números naturais de 1 a 9 em certas distribuições não era como o esperado, de um para nove (11,11%), pois o número 1 tinha a maior probabilidade de ocorrer (30%), enquanto o número 2 menos (17%), e assim consecutivamente, como demonstrado na tabela 4. Esses valores representados graficamente formam uma curva logarítmica decrescente.

Dígito	Probab. 1º dígito	Probab. 2º dígito	Probab. 3º dígito	Probab. 4º dígito
0	–	0,119679	0,101784	0,100176
1	0,301030	0,113890	0,101376	0,100137
2	0,176091	0,108821	0,100972	0,100098
3	0,124939	0,104330	0,100573	0,100059
4	0,096910	0,100308	0,100178	0,100019
5	0,079181	0,096677	0,099788	0,099980
6	0,066947	0,093375	0,099401	0,099941
7	0,057992	0,090352	0,099019	0,099902
8	0,051153	0,087570	0,098641	0,099863
9	0,045757	0,084997	0,098267	0,099824

Tabela 4 - Distribuição de probabilidade dos primeiros quatro dígitos, conforme a Lei de Benford.
Fonte: CYMROT, R., ROCHA, F. R., FERREIRA, D. S. Análise dos dígitos industriais baseada na lei de Benford e sua aplicação utilizando rotinas computacionais.

Essa distribuição ficou conhecida como distribuição de Benford ou Lei de Benford, aplicada por meio da análise dos dígitos.

A comparação dos dados obtidos com a distribuição de Benford deve ser realizada através do teste de aderência quiquadrado, utilizando a equação abaixo:

$$\chi_0^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i} = \sum_{i=1}^{k} \frac{O_i^2}{E_i} = -n$$

Dessa forma, Ei valor da frequência esperada pela distribuição de Benford, Oi valor da frequência observada na amostra, k valor do número total possível de dígitos (9 para o primeiro e 10 para os demais dígitos significativos) e n igual ao número de observações na amostra. O valor encontrado é comparado ao da distribuição quiquadrado, em um nível de significância estabelecido com (k – 1) graus de liberdade (SIEGEL; CASTELLAN JR., 2008).

Sendo assim, uma vez que os valores gerados forem submetidos ao teste estatístico acima citado e sendo comprovado como aleatórios, o programa, usando-os como chave

encriptadora, codificará dados de forma matematicamente impossível de serem decifrados para aqueles que não possuírem a chave correta. Em posse do teste estatístico torna-se necessário uma fonte de dados viável para a formação chave. A seção abaixo apresenta algumas opções a serem submetidas ao teste.

Aleatoriedade presente no ambiente industrial

Em uma fábrica existem diversas fontes de dados que podem ser candidatas a geradoras de números aleatórios, como por exemplo os dados provenientes de vibração de máquinas, de temperatura de peças ou componentes, e mesmo dados provenientes da captação de ruído branco gerado por interferências das diversas máquinas e equipamentos.

O ruído branco é explicado como um ruído que apresenta componentes em todo o espectro de frequência, sendo que um sinal aleatório é considerado ruído branco.

Nos estudos de sistemas de telecomunicações considera-se que um ruído é branco quando este apresenta componentes frequências em toda a onda passante utilizada para transmissão. O som de um ruído branco é semelhante ao de um televisor fora do ar, ou da pronúncia das letras “ch” associadas.

3. METODOLOGIA

O objetivo desta pesquisa foi o desenvolvimento de um programa capaz de cifrar dados de forma matematicamente perfeita. O desenvolvimento do programa foi feito em diferentes versões, sendo que a final consiste em um resultado aprimorado das anteriores.

A primeira consistiu na realização de tarefas independentes, mas integradas, sendo estas responsáveis pelas funções básicas do programa. Posteriormente, foi elaborado um programa responsável pela implementação da cifra de César com a troca de uma posição, concluindo a segunda etapa. Por fim, na terceira e última fase, foi desenvolvido um programa capaz de executar a técnica do bloco de cifras de uma única vez.

Primeira versão

Foram realizadas determinadas tarefas, a princípio independentes, que seriam a base do programa, as quais posteriormente serão integradas para a realização de uma função específica. Dentre elas, a construção de um programa responsável por contabilizar a quantidade de letras dentro de uma *string. String* é um tipo de variável dentro da linguagem

utilizada – *Virtual Basic for Applications* (VBA), a qual armazena caracteres de texto. Para atingir esse objetivo foi utilizada a função *len*.

O próximo passo foi a construção de um programa capaz de trocar uma letra por outra dentro de uma *string,* a seguir outro objetivo foi estabelecido, a elaboração de um programa responsável pela inversão da ordem das letras de uma *string*, etapas importantes para formação da *string* que contém a chave.

A última parte dessa primeira versão é constituída de um programa capaz de abrir um arquivo externo do formato .txt, editá-lo e salvar em outro arquivo .txt, resolvida com a parte de código a seguir:

```
ArquivoEntrada = FreeFile
  CaminhoArquivoEntrada = "C:\LOCAL DO ARQUIVO .TXT CONTENDO O TEXTO A SER CIFRADO”
  ArquivoSaida = FreeFile
  CaminhoArquivoSaida = "C:\LOCAL DO ARQUIVO .TXT CONTENDO O TEXTO JÁ CIFRADO"
  Open CaminhoArquivoEntrada For Input As ArquivoEntrada
```

Onde ArquivoEntrada é a variável que armazena os caracteres de texto do arquivo entrada.

```
  Do While Not EOF(ArquivoEntrada)
    Line Input #ArquivoEntrada, TextoProximaLinha
    TextoProximaLinha = TextoProximaLinha & vbCrLf
    TextoArquivo = TextoArquivo & TextoProximaLinha
    linhas = linhas + 1
  Loop
  vetorLinhas = Split(TextoArquivo, vbCrLf)
```

No trecho acima a variável vetorLinhas armazena o texto do arquivo .txt de entrada separando-o em linhas.

```
Close ArquivoEntrada
  Open CaminhoArquivoSaida For Output As ArquivoSaida
      For i = 0 To linhas - 1
            Print #ArquivoSaida, trocado
            Next i
```

```
Close ArquivoSaida
```

Onde trocado é a variável que armazena o texto que será futuramente cifrado com o desenvolvimento do programa, enquanto que ArquivoSaida é a variável que se representa o arquivo de texto, onde será gravado o texto cifrado.

Segunda versão

Nessa versão, o objetivo foi a construção de um programa responsável pela implementação da cifra de César com a troca de uma posição. Para tornar possível a implementação da cifra de César foi necessária a criação de um alfabeto armazenado em uma *string* com formato de vetor, onde cada letra ocupava uma posição, sendo assim 26 posições.

Também foi necessária a criação de um vetor contendo um alfabeto com uma posição deslocada. Por exemplo, com um deslocamento A se torna B, B se torna C, C se torna D e assim por diante. Foi utilizado um “loop” com “For”, que consiste em um “laço”, algo que o programa repete determinadas vezes, alterando assim o vetor do alfabeto original e salvando o novo alfabeto com uma posição deslocada em outro vetor, denominado como alfabetoC.

Após a criação dos alfabetos modificados, foi necessária a contagem da quantidade de caracteres presentes no texto a ser cifrado, esse número foi armazenado em uma variável, chamada de k, do tipo *Integer* a qual é utilizada para armazenar valores numéricos em VBA.

A próxima etapa consistiu em realizar uma verificação do texto original para poder alterar a posição de cada letra individualmente. Cada letra do texto a ser cifrado é comparada a letras alfabeto não modificado, até que se encontre sua posição correspondente, por exemplo: a letra C do texto original, não será correspondente a letra A ou B, e sim a letra C do alfabeto original, a qual se encontra na terceira posição. Ao acontecer essa correspondência o programa armazena na *string* trocado, a letra que estiver na mesma posição, mas no alfabeto cifrado. Dessa forma a terceira posição do alfabeto cifrado, o qual tem suas letras deslocadas uma posição, armazena a letra D.

Pensamento ilustrado com o trecho de código abaixo:

```
For z = 0 To linhas
    For i = 0 To k
        For j = 1 To 26
            If Mid(vetorLinhas(z), i + 1, 1) = alfabeto(j) Then
                trocado = trocado + alfabetoC(j)
```

```
        End If
    Next j
Next i
Next z
```

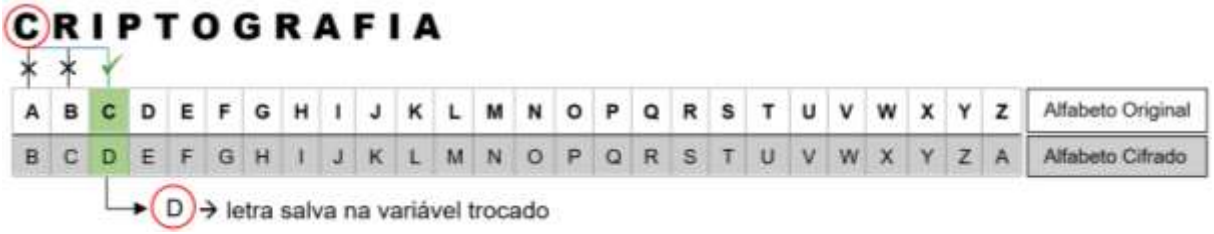

Imagem 1 – Exemplificação da codificação da letra C pelo programa. Esquema que seguirá o mesmo padrão para as letras seguintes da palavra “criptografia”.
Fonte: Elaborado pelo autor.

Em seguida, foi necessária a elaboração de programa capaz de descriptografar a cifra de César com a troca de uma posição. Para isso, é necessário construir o caminho da volta utilizando a mesma chave que foi usada para criptografar, nesse caso a chave é o deslocamento em uma posição, ou seja, o número um.

Dessa forma, semelhante ao procedimento citado anteriormente, cada letra do texto já cifrado foi comparada ao alfabeto cifrado, a variável alfabeto. Ao haver coincidência, a letra era substituída por sua posição correspondente no alfabeto original e armazenada na variável trocado, que no fim do código será ordenada a sua gravação no arquivo de saída, o qual dessa vez irá conter a mensagem original, por se tratar do processo de descriptografia.

Ilustrado no pedaço do código abaixo:

```
If Mid(vetorLinhas(z), i + 1, 1) = alfabetoC(j) Then
trocado = trocado + alfabeto(j)
End If
```

Após a construção de raciocínio referente ao deslocamento unitário, fez-se necessária a criação de um programa encarregado pela implementação da cifra de César com um valor n de troca de posições, a ser introduzido pelo usuário.

O valor de n corresponde ao deslocamento de letras, como citado no deslocamento unitário, n seria equivalente a um. Se n for igual a três a cifra trocará todas as letras A pela letra D no texto cifrado. No programa a variável n foi introduzida na hora de construir o alfabeto cifrado.

Para determinar o valor de n, a seguinte pergunta foi feita ao usuário, contida no código abaixo:

```
n = InputBox("Qual o valor de n?")
```

Terceira versão

A versão final teve como propósito a elaboração de um programa capaz de implementar a cifra de César com o valor n de troca de posições, localizado em um outro arquivo externo em formato .txt.

Para realização do objetivo estabelecido, foi necessária a introdução de um trecho de código responsável pela localização de um arquivo .txt externo, com o código contendo o valor da chave a ser utilizado. Esse valor era utilizado para a construção do alfabeto cifrado.

Ilustrado abaixo:

```
'Configura o arquivo de entrada de n
   ArquivoValor = FreeFile
   CaminhoArquivoValor = "C:\LOCAL DO ARQUIVO .TXT"
'Abre o arquivo de entrada para leitura de n
   Open CaminhoArquivoValor For Input As ArquivoValor
'Lê o conteúdo do arquivo de entrada linha a linha
   Do While Not EOF(ArquivoValor)
      Line Input #ArquivoValor, n
   Loop
```

A alteração de maior relevância consistiu na formação da chave, visto que, anteriormente consistia em apenas um valor de n e posteriormente um texto. Esse fato evidencia a importante mudança da cifra de César para a técnica do bloco de cifras de uma única vez. Ao invés de utilizar apenas um alfabeto cifrado para todo o texto, seria possível a aplicação de um alfabeto para cada letra do texto original, tornando-o matematicamente indecifrável sem a detenção da chave correta.

A confiabilidade do bloco de cifras de uma única vez é resultado de uma chave aleatória. A chave injeta uma incerteza no texto cifrado e, se o texto é aleatório, ele não possui padrões, ou estruturas, nada que um criptoanalista possa usar como base. Pode ser provado matematicamente que é impossível para um criptoanalista quebrar uma mensagem cifrada com um bloco de cifras de uma única vez. Ou seja, não se acredita meramente que o bloco seja indecifrável, existem comprovações de que ele realmente fornece segurança absoluta. Segundo Singh (2004) “O bloco de cifra de uma única vez oferece a garantia do segredo: o Santo Graal da criptografia.” (SINGH, 2004, p. 141)

Dessa forma, foi necessária a criação de um código apenas para formação da chave. Como explicado no item anterior, após a abertura do arquivo localizado externamente ocorre um processo similar ao feito com o arquivo que contém o texto a ser cifrado. O arquivo é aberto e separado linha a linha, seguido pela contagem de caracteres presentes na chave. Por último, é realizada a troca das letras por sua posição correspondente no alfabeto, formando uma sequência de números a partir das letras contidas na chave.

Por exemplo, se estivesse contido no arquivo chave o texto, abcdg, o programa ao realizar a troca das letras por números que correspondem à sua posição no alfabeto ficaria da seguinte forma, 12347.

De acordo com a sequência numérica criada o texto do arquivo entrada será cifrado com o número de deslocamentos correspondente do número apresentado. Em caso de texto maior que o tamanho da chave, a mesma se repete.

Após todas as etapas citadas acima integradas, o programa foi concluído. Sendo esse capaz de cifrar dados de forma matematicamente perfeita e impenetrável, somente com a condição de que a chave gerada seja perfeitamente aleatória.

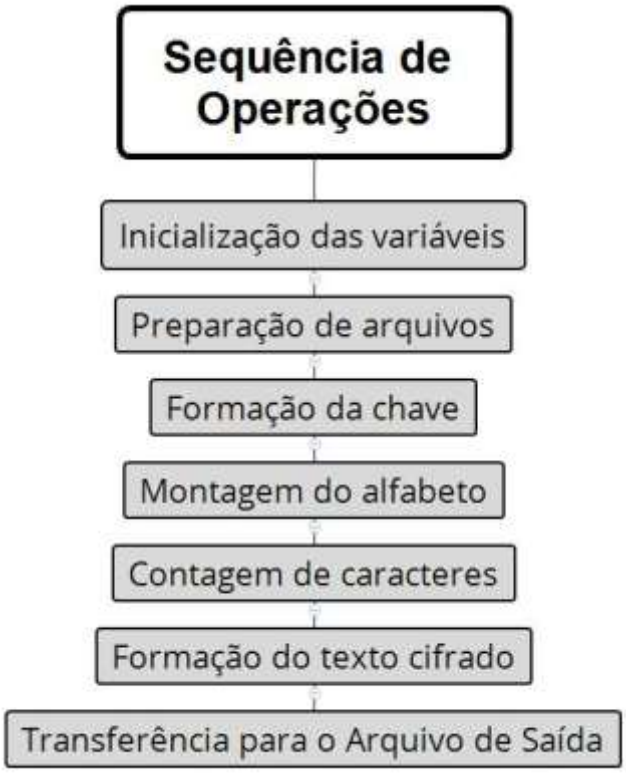

Imagem 2 – Fluxograma com a sequência de operações realizadas pelo programa já finalizado.
Fonte: Elaborado pelo autor.

4. RESULTADO E DISCUSSÃO

Durante o desenvolvimento do programa, um empecilho encontrado foi a forma como o mesmo reconheceria os espaços em branco. Como resolução mais viável, o espaço foi considerado como mais um caractere do texto e foi adicionado como uma letra a mais do alfabeto. Após esta alteração, o vetor alfabeto passou a conter 27 caracteres e o programa se tornou capaz de cifrar letras como se fossem espaço e vice-versa.

Outra dificuldade encontrada ocorreu quando o arquivo a ser cifrado continha quebras de linha. Ao realizar a criptografia, o programa repetiu o trecho anterior à quebra, problema o qual não foi identificada ainda uma solução.

5. CONSIDERAÇÕES FINAIS

O objetivo do presente trabalho foi o desenvolvimento de um programa capaz de cifrar dados de forma matematicamente perfeita, e o mesmo foi atingido, uma vez que se obtenha uma fonte de números perfeitamente aleatórios. Apesar algumas restrições de funcionamento, tais como presença de letras maiúsculas, sinais de pontuação gráfica e quebras de linha, o método desenvolvido cumpriu com o objetivo estipulado.

A garantia de funcionamento ideal do programa é definida pelo maior grau de segurança possível, portanto, letras ou espaços devem ser introduzidas no arquivo chave. Dessa maneira, será possível a formação de 27 alfabetos distintos. Se fossem introduzidos números, a formação seria reduzida apenas à variação unitária, de zero a nove; nessa condição, existem apenas dez alfabetos a serem alternados durante a codificação.

Uma vez escolhida a fonte dos dados, os mesmos precisam ser analisados para verificar se de fato são aleatórios, tal analise é feita através da ferramenta estatística utilizada na verificação da confiabilidade dos valores de amostras baseada na distribuição de Benford. Ao serem comprovadamente aleatórios perfeitos, torna-se possível implementar na empresa uma política ou rotina, na qual, com uma determinada frequência, as chaves geradas sejam distribuídas manualmente, entre as maquinas, setores ou fábricas que terão de comunicar entre si.

6. REFERÊNCIAS

ALLEN, B.O. Pseudo-Random vs True Random. Denver, 2012. Disponível em: <http://boallen.com/random-numbers.html>. Acesso em 01 de abr. de 2017.

ARKTIS. A quarta revolução da indústria. Sorocaba, 2015. Disponível em: <http://arktis.com.br/a-quarta-revolucao-da-industria/>. Acesso em 01 de abr. de 2017.

BERGLUND, M., KARLTUN, J., (2007). Human, technological and organizational aspects influencing the production scheduling process. Int. J. Prod. Econ. 110, 160-174.

CYMROT, R., ROCHA, F. R., FERREIRA, D. S. Análise dos dígitos industriais baseada na lei de Benford e sua aplicação utilizando rotinas computacionais. Revista Mackenzie de Engenharia e Computação, v. 12, n. 1, p. 125-145

FILHO, S. N. Fundamentos sobre Ruídos. Revista Backstage, vol. 8, 2002.

FOLEY, Louise; WILSON, S. Analysis of an on-line random number generator. Project Report, Illinois, 2001.

FRIAS. Industria 4.0 exige foco maior em segurança. IT Security Conference. São Paulo, 2016. Disponível em: <http://itsa-brasil.com.br/hotsite/noticias/industria-4-0-exige-foco-maior-em-seguranca/>. Acesso em 01 de abr. de 2017.

HE, K., JIM, M. Cyber-Physical System for maintenance in industry 4.0. Jönköping University – School of Engineering, 2016.

HILL, T. P. A Statistical Derivation of the Significant-Digit Law. Statistic Science, v. 10, p. 354-363, 1995. Disponivel em: <http://www.tphill.net/publications/BENFORD%20PAPERS/statisticalDerivationSigDigitLaw1995.pdf >. Acesso em: 17 abr. 2008.

HILL, T. P. The first digit phenomenon. The American Scientist, v. 86, n. 4, p. 358-363, 1996. Disponível em: <http://www.tphill.net/publications/BENFORD%20PAPERS/TheFirstDigitPhenomenonAmericanScientist1996.pdf>. Acesso em: 17 abr. 2008.

HILL, T. P. The difficulty of faking data. Chance, v. 26, p. 8-13, 1999. Disponivel em: <http://www.math.gatech.edu/~hill/publications/cv.dir/faking.pdf>. Acesso em: 17 abr. 2008.

IMAM. Industria 4,0. Revista Logística. São Paulo, 2015 Disponível em:<http://www.imam.com.br/consultoria/artigo/pdf/industria-4.0.pdf> Acesso em 01 de abr. de 2017.

KLEINSCHMIDT, J. H. Segurança da Informação – Criptografia simétrica. Santo André, 2013. Disponível em <http://professor.ufabc.edu.br/~joao.kleinschmidt/aulas/seg2013/aula_02-1_seg.pdf>.Acesso em 01 de abr. de 2017.

SANTOS, P. R. Indústria 4.0 – Sistemas inteligentes para manufatura do futuro. Revista Ferramental. Vol. 2. Florianópolis, 2016.

SINGH, S. O livro dos códigos - A ciência do sigilo do antigo Egito à criptografia quântica. 6a ed. Rio de Janeiro: Record, 2007.

SOUZA, P. A., BRANQUINHO, M. KIEFER, A., SANTOS, C., VIDEIRA, E. Cyber Security para Sistemas de Automação de Energia – como a defesa em profundidade pode aumentar a segurança cibernética em instalações críticas. XI Simpósio de automação de sistemas elétricos. Campinas, 2015.

STALLINGS, W. Criptografia e segurança de redes. 4ª Ed. Pearson, 2008

TADEU, H. F. B. O que seria a indústria 4.0? – Pesquisa sobre digitalização. São Paulo, 2016. Disponível em: <https://www.fdc.org.br/professoresepesquisa/nucleos/Documents/inovacao/digitalizacao/boletim_digitalizacao_fevereiro2016.pdf>. Acesso em 01 de abr. de 2017.

Apêndice

Código completo:

```
Option Explicit
   Dim ArquivoChave As Integer ''define o arquivo que contém a chave
   Dim ArquivoEntrada As Integer ''define o arquivo que contém a entrada
   Dim ArquivoSaida As Integer ''define o arquivo que contém a saída
   Dim CaminhoArquivoChave As String ''define o caminho que deve ser seguido para encontrar o arquivo que contém a chave
   Dim CaminhoArquivoEntrada As String ''define o caminho que deve ser seguido para encontrar o arquivo que contém a entrada
   Dim CaminhoArquivoSaida As String ''define o caminho que deve ser seguido para encontrar o arquivo que contém a saída
   Dim TextoArquivo As String ''armazena o texto contido no arquivo entrada, antes dele ser transformado em vetor
   Dim TextoArquivoChave As String ''armazena o texto contido no arquivo chave, antes dele ser transformado em vetor
   Dim TextoProximaLinha As String ''armazena o texto contido em uma linha do arquivo entrada
   Dim TextoProximaLinhaChave As String ''armazena o texto contido em uma linha do arquivo chave
   Dim vetorLinhas() As String      ''declara um vetor com dimensões variaveis para armazenar o texto do arquivo entrada
   Dim vetorLinhasChave() As String ''declara um vetor com dimensões variaveis para armazenar o texto do arquivo chave
   Dim vetorChave() As String       ''declara um vetor com dimensões variaveis para armazenar o valor correspondente de cada letra da chave
   Dim p As Integer      ''usada para recomeçar a chave, na hora da cifragem
   Dim x As Integer      ''usada como auxiliar na cifragem, representando a posição da letra a ser cifrada
   Dim y As Integer      ''indica qual alfabeto será usado na cifragem
   Dim i As Integer      ''auxilia na formação do vetorLinhasChave()
   Dim j As Integer      ''auxilia na formação do vetorLinhasChave()
   Dim k As Integer      ''armazena a quantidade de caracteres presentes no vetorLinhas()
   Dim kc As Integer     ''armazena a quantidade de caracteres presentes no vetorLinhasChave()
   Dim z As Integer      ''representa a linha do vetorLinhasChave()
   Dim linhas As Integer ''para salvar o numero de linhas no arquivo
   Dim linhasChave As Integer ''quantidade de linhas contidas no arquivo chave
   Dim espaco As Integer ''armazena resposta do usuario referente ao espaço
   Dim nChave As String ''armazena o número correspondente de cada letra da ordem alfabética
   Dim trocado As String ''armazena o texto já cifrado
```

```
   Dim n As String ''é o numero extraido de nChave, que indica qual alfabeto será usado para a cifragem
   Dim alfabeto(1 To 27) As String ''vetor que contém o alfabeto em sua ordem normal
   Dim alfabetoC(1 To 27) As String ''vetor que contém o alfabeto na ordem da cifra desejada
Public Sub principal()
Call inicializa_variaveis
Call prepara_arquivos
Call forma_chave
Call monta_alfabeto
Call conta_caracteres
Call troca_texto
Call fecha_arquivos
End Sub
Sub inicializa_variaveis()
   ArquivoChave = 0
   ArquivoEntrada = 0
   ArquivoSaida = 0
   CaminhoArquivoChave = ""
   CaminhoArquivoEntrada = ""
   CaminhoArquivoSaida = ""
   TextoArquivo = ""
   TextoArquivoChave = ""
   TextoProximaLinha = ""
   TextoProximaLinhaChave = ""
   p = 0
   x = 0
   y = 0
   i = 0
   j = 0
   k = 0
   kc = 0
   z = 0
   linhas = 0
   linhasChave = 0
   espaco = 0
   nChave = ""
   trocado = ""
   n = ""
   For i = 1 To 27
     alfabeto(i) = ""
     alfabetoC(i) = ""
   Next
   i = 0
End Sub
Sub prepara_arquivos()
   ArquivoChave = FreeFile
   CaminhoArquivoChave = " C:\LOCAL DO ARQUIVO .TXT "

   Open CaminhoArquivoChave For Input As ArquivoChave
   Do While Not EOF(ArquivoChave)
      Line Input #ArquivoChave, TextoProximaLinhaChave
      TextoProximaLinhaChave = TextoProximaLinhaChave & vbCrLf
      TextoArquivoChave = TextoArquivoChave & TextoProximaLinhaChave
```

```
      linhasChave = linhasChave + 1
   Loop
   vetorLinhasChave = Split(TextoArquivoChave, vbCrLf)
   ArquivoEntrada = FreeFile
   CaminhoArquivoEntrada = " C:\LOCAL DO ARQUIVO .TXT "
   ArquivoSaida = FreeFile
   CaminhoArquivoSaida = " C:\LOCAL DO ARQUIVO .TXT "
   Open CaminhoArquivoEntrada For Input As ArquivoEntrada
   Do While Not EOF(ArquivoEntrada)
      Line Input #ArquivoEntrada, TextoProximaLinha
      TextoProximaLinha = TextoProximaLinha & vbCrLf
      TextoArquivo = TextoArquivo & TextoProximaLinha
      linhas = linhas + 1
   Loop
End Sub
Sub forma_chave()
   vetorLinhas = Split(TextoArquivo, vbCrLf)
   For i = 0 To linhasChave
      For j = 1 To Len(vetorLinhasChave(i))
         kc = kc + 1
      Next j
   Next i

   For z = 0 To linhasChave
      For i = 0 To kc
         If Mid(vetorLinhasChave(z), i + 1, 1) = "a" Then
            nChave = nChave + "1" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "b" Then
            nChave = nChave + "2" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "c" Then
            nChave = nChave + "3" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "d" Then
            nChave = nChave + "4" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "e" Then
            nChave = nChave + "5" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "f" Then
            nChave = nChave + "6" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "g" Then
            nChave = nChave + "7" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "h" Then
            nChave = nChave + "8" + vbCrLf
         End If
         If Mid(vetorLinhasChave(z), i + 1, 1) = "i" Then
            nChave = nChave + "9" + vbCrLf
         End If
```

```
If Mid(vetorLinhasChave(z), i + 1, 1) = "j" Then
   nChave = nChave + "10" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "k" Then
   nChave = nChave + "11" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "l" Then
   nChave = nChave + "12" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "m" Then
   nChave = nChave + "13" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "n" Then
   nChave = nChave + "14" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "o" Then
   nChave = nChave + "15" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "p" Then
   nChave = nChave + "16" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "q" Then
   nChave = nChave + "17" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "r" Then
   nChave = nChave + "18" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "s" Then
   nChave = nChave + "19" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "t" Then
   nChave = nChave + "20" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "u" Then
   nChave = nChave + "21" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "v" Then
   nChave = nChave + "22" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "w" Then
   nChave = nChave + "23" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "x" Then
   nChave = nChave + "24" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "y" Then
   nChave = nChave + "25" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = "z" Then
   nChave = nChave + "26" + vbCrLf
End If
If Mid(vetorLinhasChave(z), i + 1, 1) = " " Then
```

```
            nChave = nChave + "27" + vbCrLf
          End If
        Next i
    Next z
    vetorChave = Split(nChave, vbCrLf)
End Sub
Sub conta_caracteres()
    For i = 0 To linhas
        For j = 1 To Len(vetorLinhas(i))
            k = k + 1
        Next j
    Next i
End Sub
Sub troca_texto()
    p = 0
    For z = 0 To linhas
        For x = 0 To k
            For y = 1 To 27
                If Mid(vetorLinhas(z), x + 1, 1) = alfabeto(y) Then
                    If p + 1 > UBound(vetorChave) Then
                        p = 0
                    Else: End If
                    n = vetorChave(p)
                    n = CInt(n)
                        For i = 1 To 27
                            If (n + i) <= 27 Then
                              alfabetoC(i) = alfabeto(i + n)
                            ElseIf i <= 27 Then
                              alfabetoC(i) = alfabeto((-27 + n) + i)
                            End If
                        Next i
                    trocado = trocado + alfabetoC(y)
                End If
            Next y
                p = p + 1
        Next x
    Next z
End Sub
Sub monta_alfabeto()
    alfabeto(1) = "a"
    alfabeto(2) = "b"
    alfabeto(3) = "c"
    alfabeto(4) = "d"
    alfabeto(5) = "e"
    alfabeto(6) = "f"
    alfabeto(7) = "g"
    alfabeto(8) = "h"
    alfabeto(9) = "i"
    alfabeto(10) = "j"
    alfabeto(11) = "k"
    alfabeto(12) = "l"
    alfabeto(13) = "m"
```

```
   alfabeto(14) = "n"
   alfabeto(15) = "o"
   alfabeto(16) = "p"
   alfabeto(17) = "q"
   alfabeto(18) = "r"
   alfabeto(19) = "s"
   alfabeto(20) = "t"
   alfabeto(21) = "u"
   alfabeto(22) = "v"
   alfabeto(23) = "w"
   alfabeto(24) = "x"
   alfabeto(25) = "y"
   alfabeto(26) = "z"
   alfabeto(27) = " "
End Sub
Sub fecha_arquivos()
   Close ArquivoEntrada
   Open CaminhoArquivoSaida For Output As ArquivoSaida
   For i = 0 To linhas - 1
      Print #ArquivoSaida, trocado
   Next i
   Close ArquivoSaida
End Sub
```

Contatos: ribeiro0120@gmail.com e mario.santos@mackenzie.br