Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática
Programa de Mestrado Profissional em
Matemática em Rede Nacional

INTRODUÇÃO A SISTEMAS CRIPTOGRÁFICOS E O USO DE GERADORES DE SEQUÊNCIAS DE NÚMEROS ALEATÓRIOS E PSEUDO-ALEATÓRIOS

EDUARDO CÍCERO VIEIRA BORGES JUNIOR

Brasília

2014

Ficha catalográfica elaborada pela Biblioteca Central da Universidade de Brasília. Acervo 1019073.

Borges Junior, Eduardo Cícero Vieira.
B732i Introdução a sistemas criptográficos e o uso de geradores de sequências de números aleatórios e pseudo-aleatórios / Eduardo Cícero Vieira Borges Junior. -- 2014.
54 f. : il. ; 30 cm.

Dissertação (mestrado) - Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Matemática, 2014.
Orientação: Rui Seimetz.
Inclui bibliografia.

1. Criptografia - Matemática. 2. Variáveis aleatórias. I. Seimetz, Rui. II. Título.

CDU 681.188

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Matemática
Programa de Mestrado Profissional em Matemática em Rede Nacional

INTRODUÇÃO A SISTEMAS CRIPTOGRÁFICOS E O USO DE GERADORES DE SEQUÊNCIAS DE NÚMEROS ALEATÓRIOS E PSEUDO-ALEATÓRIOS

Autor: Eduardo Cícero Vieira Borges Junior

Orientador: Prof. Dr. Rui Seimetz

Monografia apresentada como requisito parcial para a obtenção do grau de Mestre em Matemática Profissional pelo Departamento de Matemática da Universidade de Brasília.

Brasília

2014

EDUARDO CÍCERO VIEIRA BORGES JUNIOR

INTRODUÇÃO A SISTEMAS CRIPTOGRÁFICOS E O USO DE GERADORES DE SEQUÊNCIAS DE NÚMEROS ALEATÓRIOS E PSEUDO-ALEATÓRIOS

Monografia apresentada como requisito parcial para a obtenção do grau de Mestre em Matemática Profissional pelo Departamento de Matemática da Universidade de Brasília.

Área de Concentração: Matemática

Prof. Dr. Rui Seimetz – Presidente
Universidade de Brasília

Prof. Dr. Mário José de Souza
Universidade Federal de Goiás

Prof. Dr. Hélder de Carvalho Matos
Universidade de Brasília

Brasília

2014

À minha esposa, Cláudia Borges, mulher da minha vida.

DEDICO

Agradecimentos

À Sociedade Brasileira de Matemática (SBM) por meio do PROFMAT, porque propicia uma melhor capacitação dos educadores no ensino da matemática.

Aos professores da Universidade de Brasília (UnB) que participaram do curso e contribuíram para uma melhor formação dos alunos.

Ao professor Dr. Rui Seimetz, coordenador do PROFMAT na UnB, pela dedicação na condução do curso, bem como na colaboração, paciência e ensinamentos que permitiram a elaboração deste trabalho.

Aos meus amigos Lopes, David e Felipe, pela amizade, e presença nos momentos difíceis.

Aos colegas de turma, e principalmente, os amigos do Colégio Pódion, pelo auxílio e companheirismo ao longo de todo o curso.

Aos familiares, pais, irmão, esposa, pelo apoio, compreensão e incentivo para que tudo desse certo.

A Deus, por tornar isso possível.

"No que diz respeito ao desempenho, ao compromisso, ao esforço, à dedicação; não existe meio-termo. Ou se faz uma coisa bem feita, ou então não se faz"

Ayrton Senna.

Resumo

Neste trabalho iremos abordar alguns tópicos referentes à criptografia elementar, de sistemas criptográficos simétricos e assimétricos, apresentados de modo a expor a interpretação matemática acerca dos principais métodos utilizados para a encriptação de dados.

Constam, ainda, neste trabalho, estudos sobre os principais métodos geradores de números aleatórios e métodos auxiliares a estes geradores, assim como sua necessidade e envolvimento com a criptografia. Estes estudos auxiliares, como será mostrado, promovem a elaboração de um sistema mais robusto, de difícil quebra de código.

Abstract

In this paper we discuss some topics related to elementary cryptography, symmetric and asymmetric cryptosystems, presented to expose the mathematical interpretation about the main methods used for data encryption.

Also in this work are presented studies on the major methods of random number generators and auxiliary generators to these methods, as well as their need and relation with encryption. These auxiliary studies, as will be shown, promote the development of a more robust system, more difficult to break the code.

Sumário

1 Introdução

1.1 CONTEXTO

A segurança dos sistemas de informação é obtida por meio de sistemas criptográficos destinados a modificar o código escrito para outro não legível a quem não for endereçada tal informação.

Esta segurança obtida por muitos sistemas criptográficos depende da geração de dados não correlatos, independentes, que não se podem prever. Como exemplos podem ser citados a sequência chave do *one-time-pad*, a chave do algoritmo de encriptação *DES*, os primos p e q do algoritmo *RSA* e dos esquemas de assinaturas digitais.

Em todos esses casos os dados devem ser aleatórios e sua quantidade deve ser grande o suficiente, de maneira que a probabilidade de se quebrar o código seja mínima e que a sua obtenção não seja tão custosa.

Em termos práticos, pode-se perceber que a quantidade de chaves possíveis para o *DES* é de 2^{56}, se uma chave k for selecionada de um gerador verdadeiramente aleatório. Um inimigo pode tentar até 2^{55} chaves antes de descobrir, com probabilidade $\frac{1}{2}$, a chave correta k. Mas se esta tiver sido criada da expansão de 16 bits verdadeiramente aleatórios em 56 bits, por um gerador pseudo-aleatório, o conjunto de tentativas cai para 215. Isto demonstra o dual entre segurança e facilidade de implementação. Assim o presente trabalho é desenvolvido no âmbito da análise matemática dos principais métodos criptográficos, simétricos e assimétricos, bem como da exposição de alguns dos sistemas utilizados para a geração de números aleatórios e pseudo-aleatórios.

1.2 OBJETIVOS

O principal objetivo do trabalho é introduzir a criptografia como um todo, bem como expor alguns dos métodos criptográficos, uma vez que estes são aplicações diretas e práticas de conhecimentos matemáticos e computacionais elementares.

Em segundo plano pode ser dito que este trabalho esclarece o leitor em relação às utilizações e a importância da geração de sequências de números aleatórios para a criptografia, sem contar a importância da criptografia em si.

2 CRIPTOGRAFIA

A criptografia está intimamente ligada ao cunho deste trabalho. O objetivo deste trabalho é discutir alguns dos principais métodos utilizados para a encriptação de mensagens, sejam estes de criptografia simétrica, ou assimétrica, bem como apresentar alguns mecanismos utilizados para a geração de sequências de números aleatórios e pseudo-aleatórios.

O envolvimento dos números aleatórios nos processos criptográficos, acima de tudo, contribui para a confiabilidade da comunicação, tenta eliminar as chances de rompimento de código por pessoas alheias e quando isso não for possível, adia ao máximo o conhecimento da verdadeira informação transmitida, até que outro meio mais confiável seja estabelecido. A criptografia se compromete a fornecer serviços de segurança da informação. Alguns deles são:

- Serviço confidencial: confiança em que a mensagem possa ser lida apenas pelo receptor desejado.
- Autenticação da origem: garantia sobre quem originou a mensagem.
- Integridade: confiança em que a mensagem não tenha sido modificada desde o momento da criação.
- Não-retratação: quem originou a mensagem não pode negar tê-lo feito.

Por estas questões, é válido salientar aspectos de comprometimento de código e possíveis ataques aos códigos estabelecidos. Estes esclarecimentos constam no final deste capítulo.

A seguir apresentamos os tipos de sistemas de criptografia.

2.1 SISTEMAS CRIPTOGRÁFICOS SIMÉTRICOS E ASSIMÉTRICOS

Existem dois tipos de criptografia, a criptografia simétrica e a assimétrica.

A criptografia simétrica opera com a premissa de que a chave é de conhecimento unicamente do emissor e do receptor.

Na criptografia assimétrica, ou de chave pública, cada um dos usuários do meio criptográfico possui um material chave, dividido em duas porções: a componente pública e a componente privada, cada uma gerando uma respectiva transformação.

Na criptografia simétrica aplica-se um conjunto pré-determinado (pela chave) de cifras ao texto direto. Seja ela a *transformação invertível* **F**, aplicada à mensagem **X** utilizando-se a chave **Z**. Dessa forma tem-se:

$$Y = F(X, Z) = F_Z(X)$$

Nesta situação, o resultado **Y** é a mensagem criptografada.

Na decriptação cabe ao elemento receptor da mensagem aplicar a transformação inversa em Y de modo a se ter o texto inicial. Para isso faz-se:

$$F^{-1}(Y, Z) = F_Z^{-1}(Y) = F_Z^{-1}[F_Z(X)] = X$$

A chave **Z** é responsável pela escolha do modo particular de se fazer tanto a encriptação como a decriptação.

Essa chave deve ser entregue por um canal seguro, considerando-se que a segurança do sistema está embasada na natureza secreta da chave.

Na criptografia assimétrica, ou de chave pública, cada um dos usuários do meio criptográfico possui um material chave, dividido em duas porções: a *componente pública* e a *componente privada*, cada uma gerando uma respectiva transformação. Sua finalidade é a comunicação de usuários em um canal inseguro, com distribuição de mútuas chaves.

Seu funcionamento básico requer um conjunto de transformações invertíveis (algoritmos).

Assim, a partir de uma combinação das componentes pública e privada das chaves, gerando a chave **Z**, é estabelecido, de modo análogo à criptografia simétrica, o processo de encriptação, E_Z, e decriptação, D_Z.

$$E_Z : f_Z(x) = y$$

$$D_Z : f_Z^{-1}(y) = x$$

Para que o processo ocorra, como esperado, de forma ágil e segura, deve-se garantir que:

- $f_Z(x)$ seja facilmente computada;
- $f_Z^{-1}(y)$ seja extremamente difícil de ser encontrada por um inimigo;
- $f_Z^{-1}(y)$ seja fácil de ser escrita para o receptor em posse da chave, porém que seja impraticável a partir dela encontrar a chave privada.

Após essa explicação sobre os tipos de sistemas criptográficos descrevemos os algoritmos correspondentes a cada sistema.

2.2 ALGORITMOS DE SISTEMAS SIMÉTRICOS

2.2.1 ONE-TIME-PAD

Método criado em 1917 por Gilbert Sandford Vernam (com o nome de Cifra de Vernam) e aperfeiçoado por Joseph Mauborgne.

Uma chave é gerada na forma de uma sequência de bits gerados aleatoriamente. Com esta chave é realizada a operação ou *exclusivo (xor)* entre a chave e a mensagem. A resposta a esta operação constitui a mensagem cifrada.

Mensagem	011010001
Chave	010010110
Cifra	001000111

Encriptação *xor*

Para recuperar a mensagem original basta efetuar o novamente a operação ou *exclusivo (xor)* da cifra com a chave.

Cifra	001000111
Chave	010010110
Mensagem	011010001

Decriptação *xor*

Esse método é bem simples e eficiente, desde que a chave seja tão longa quanto a mensagem e que a chave seja realmente aleatória. Essas condições, sobretudo em razão da dificuldade de se gerar números aleatórios, nem sempre podem ser atendidas, o que restringe seu uso.

2.2.2 DES (DATA ENCRYPTION STANDARD)

Criado pela IBM em 1970, este algoritmo trabalha com blocos de 64 bits de entrada e de saída, chaves de 56 bits com paridade, e é fundamentalmente baseado na permutação de elementos. É fácil de implementar, mas é vulnerável a ataques de força bruta e a transmissão da chave, como é comum nos algoritmos de criptografia simétrica, necessita de máxima segurança. Essa chave é o componente aleatório do algoritmo. O mecanismo do DES consiste em dividir a mensagem (bloco de 64 bits) em duas partes

(32 bits para cada lado), a parte esquerda L e a parte direita R. A chave consiste em 64 bits, mas apenas 56 são utilizados efetivamente no algoritmo, sendo os 8 bits restantes usados somente para checar paridade sendo descartados depois.

No processo de encriptação, os lados, esquerdo e direito, permutam da seguinte forma:

- $L_i = R_{i-1}$ i variando de 1 a 16, uma vez que ocorrerão 16 iterações.
- $R_i = L_{i-1} \oplus F(R_{i-1}, Z_i)$ é uma função invertível pré-definida, denominada função de Feistel.

O símbolo $\oplus$ significa a operação de ou *exclusivo (xor).*

A função F atua na metade do bloco, junto a uma chave. O resultado é combinado à outra metade do bloco, e as metades trocam de "lado".

Esse processo de cruzar e alternar é conhecido como esquema de Feistel, e assegura que o processo de encriptar e decriptar sejam similares. A única diferença é que as chaves de cada estágio deverão ser aplicadas em ordem inversa para decriptar.
A chave Z_i de 48-bits atua em R_{i-1} expandido também para 48 bits para realizar a função que efetuará um *xor* com o lado esquerdo, resultando em R_i . A expansão de é feita da seguinte maneira:

Seja R_i uma sequência de bits:

$$R_i = r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 ... r_{29} r_{30} r_{31} r_{32}$$

Esta sequência é dividida em blocos de 4 bits resultando em R_i':

$$R_i^* = r_1 r_2 r_3 r_4 \_\_\_\_\_ r_5 r_6 r_7 r_8 \_\_\_\_\_ \cdots \_\_\_\_\_ r_{29} r_{30} r_{31} r_{32}$$

Então, à frente de cada bloco é repetido o bit anterior e ao final, o posterior.

$$R_i' = r_{32} r_1 r_2 r_3 r_4 r_5 \_\_\_\_\_ r_4 r_5 r_6 r_7 r_8 r_9 \_\_\_\_\_ \cdots \_\_\_\_\_ r_{28} r_{29} r_{30} r_{31} r_{32} r_1$$

Dessa forma, o resultado R_i' possui 48 bits.

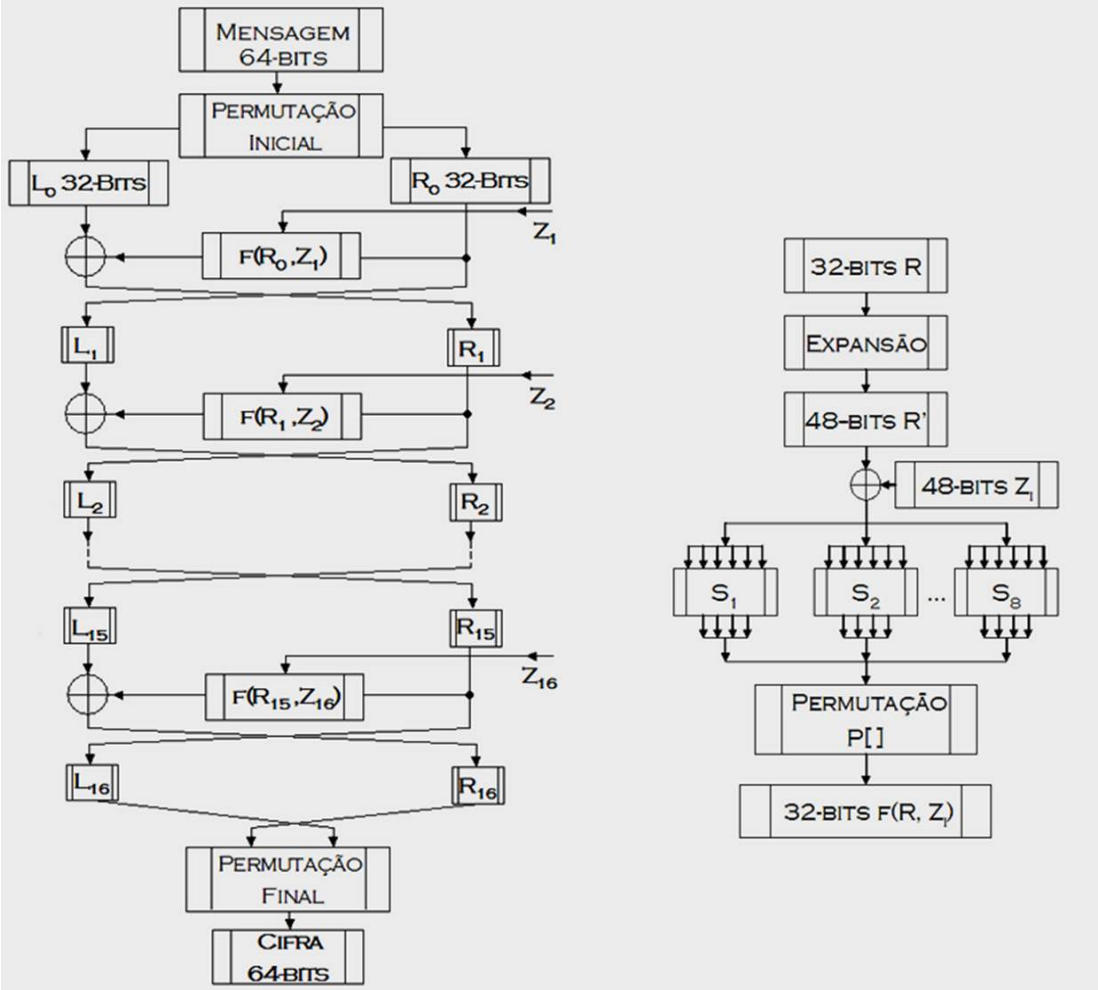


Figura 2.2.2.1 – Algoritmo DES à esquerda e a função de Feistel à direita

Depois de realizado o *xor* entre a chave e o bloco expandido, os 48 bits resultantes passam por um processo de substituição. Essas substituições são feitas em 8 "caixas"com 6 bits de entrada e 4 de saída, cada caixa. É feito isso para que este resultado recupere o tamanho de 32 bits.

O processo de decriptação é feito seguindo os passos inversos ao da encriptação.

Atualmente, este processo não é mais utilizado, porém ele serve de base para a aplicação de versões mais modernas, como o triple DES, mais seguro.

2.2.3 TRIPLE DES (3DES)

O Triple DES é uma cifra de blocos que tem como base o DES.

A sua vantagem em relação ao DES é que o tamanho da chave foi aumentado sem precisar modificar o algoritmo.

Utiliza três etapas, e opera com três chaves: K_1, K_2, K_3.

Este algoritmo realizada a composição da função DES às três chaves, ou seja:

$$C = 3DES(M) = DES\{K_3, DES[K_2, DES(K_1, M)]\}$$

Nesse caso, M é o bloco de mensagem e C é o resultado cifrado.

Abaixo está representado o diagrama de funcionamento simplificado do Triplo DES.

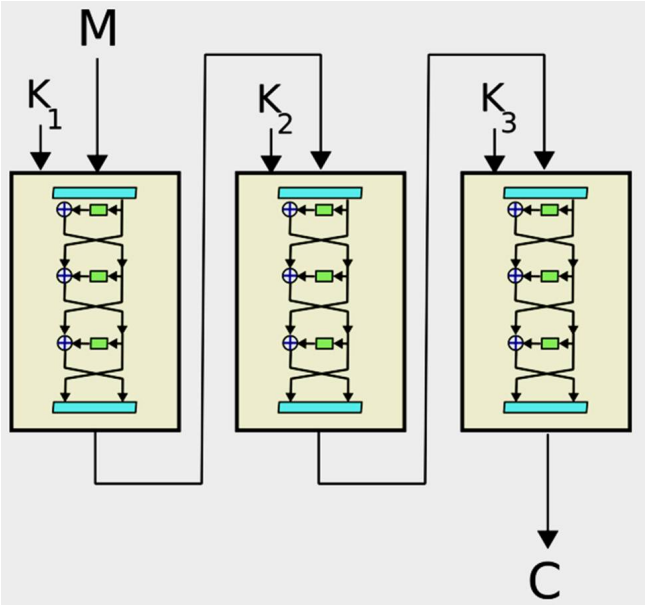


Figura 2.2.3.1 – Algoritmo Triplo DES

2.2.4 RC4

Funciona com base em cifras sequenciais, com operações orientadas a bytes, seu algoritmo faz uso de permutações pseudo-aleatórias da chave. Faz parte de protocolos

usuais de segurança, como o SSL, que protege o tráfego de informações na Internet, e o WEP, que faz parte de rede wireless.

Seu mecanismo é semelhante ao do one-time-pad, uma operação *xor* é aplicada entre uma sequência pseudo-aleatória chaveada e os dados, mesma chave para cifrar e decifrar. Seu tamanho de código é 1/10 do DES.

A chave, que é formada por uma sequência de bits, é independente do texto a ser cifrado.

De acordo com a literatura atual, não foi encontrado nenhum ataque prático ao algoritmo da sequência pseudo-aleatória, mas a maneira em que ele é inicializado é considerada uma forma insegura.

De acordo com o que foi visto, o processo fundamental deste algoritmo está na obtenção do conjunto aleatório. Depois desse processo, o algoritmo se reduz à realização da função *xor* com o texto principal.

2.3 ALGORITMOS DE SISTEMAS ASSIMÉTRICOS

2.3.1 DIFFIE-HELLMAN

Este algoritmo faz parte do protocolo *"key exchange"*, que permite que duas partes, que nunca haviam tido contato, compartilhem uma chave secreta através de um meio inseguro de comunicação. Essa chave, posteriormente, pode encriptar mensagens, estabelecendo um sistema simétrico de criptografia.

O algoritmo Diffie-Hellman foi inventado em 1976, sendo o primeiro algoritmo de chave pública.

Este algoritmo é fundamentalmente baseado na dificuldade de se calcular logaritmos discretos. A função logaritmo discreto é a inversa da função exponencial modular, definida como $y = g^x (\mathrm{mod}\ p)$, onde p e g são primos.

Em seu funcionamento, para a distribuição exponencial discreta temos:

$$y \equiv g^x (\mathrm{mod}\ p), \quad \text{para} \quad 1 \leq x \leq p-1$$

Nesse caso, g também é chamado de primitiva no módulo de p.

O retorno de x é calculado conforme abaixo:

$$x \equiv (\log d_g y)(\bmod p), \quad \text{para} \quad 1 \leq y \leq p-1$$

Os usuários têm conhecimento de g e p.

O usuário i seleciona um número aleatório x_i do conjunto $\{1, 2, ..., p\}$ que será sua chave privada. Dessa forma, a informação que será guardada num diretório público com os dados do usuário i é dada por

$$y_i \equiv g^{x_i} (\bmod p)$$

Outro usuário j realiza o mesmo procedimento.

A comunicação entre i e j é feita da seguinte maneira:

O usuário i adquire informações de j, y_j, do diretório público, para formar, a partir de sua chave privada individual, x_i , a chave privada comum a ambos, k_{ij} ou k_{ji} a partir das operações a seguir:

$$k_{ij} \equiv [y_j]^{x_i} \, (\bmod p)$$

$$k_{ij} \equiv [(g^{x_j})]^{x_i} \, (\bmod p)$$

$$k_{ij} \equiv g^{x_j . x_i} (\bmod p)$$

Analogamente, a partir das propriedades de exponenciação modular, tem-se que

$$k_{ij} \equiv k_{ji}$$

A última equação significa que i e j em comum acordo definem k_{ij} ou k_{ji} como a chave secreta em seu sistema criptográfico de comunicação.

Para um inimigo criptoanalista que tenha acesso à informação de i e j contida no diretório, seria necessário encontrar a chave

$$k_{ij} \equiv \left[(y_j)^{logdy_i}\right] (\mathrm{mod}\ p)$$

A necessidade do cálculo deste logaritmo discreto modular, quando se tem um valor de p elevado (tipicamente superior a 100 casas decimais), torna impraticável a determinação da chave secreta comum de i e j, levando em consideração as capacidades computacionais atuais.

É possível perceber, que este algoritmo, bem como o RSA e o de El Gamal, discutidos a seguir, para sua eficiência em termos de segurança, necessitam de números primos com um número elevado de casas decimais. Tal característica tem acentuado ainda mais a discussão e o interesse a respeito do estudo e melhor compreensão dos números primos.

Uma variedade de autenticações criptográficas incorpora o Diffie-Hellman como algoritmo inicial, sobretudo como base para a troca de chaves entre dois ou mais pontos de comunicação, de modo que se possa estabelecer uma comunicação segura mútua.

2.3.2 RSA

O **RSA** é um algoritmo de encriptação de dados, que deve o seu nome a três professores do Instituto MIT *(Massachussets Institute of Tecnology)*, os quais são fundadores da atual empresa RSA Data Security, Inc; Ron **R**ivest, Adi **S**hamir e Len **A**dleman, que inventaram este algoritmo.

A segurança do algoritmo se baseia pelo fato da fatoração de um número grande ser muito difícil, em termos computacionais, uma vez que ainda não se estabeleceu um método capaz de realizar esta operação de modo mais eficiente que os métodos usualmente apresentados. O algoritmo usa a teoria dos números na formulação matemática.

Pelos processos matemáticos que a mensagem deve passar e pela própria característica de algoritmo assimétrico, o processo usualmente se torna lento, tipicamente cerca de 100 vezes mais lento que o DES.

É necessária a computação dos seguintes parâmetros:

- Seleção de primos: p e q;
- Realiza-se a multiplicação dos primos: parâmetro $n = p.q$.

A partir destes parâmetros, aplica-se a função φ de Euler ao valor n.

Deve ser lembrado que a função $\varphi(n)$ de Euler retorna o número de inteiros positivos menores que n com os quais n é primo entre si, ou seja, com os quais n não possui divisores comuns. Dessa forma, para um número primo qualquer, p, este número seria primo entre si com todos os inteiros positivos até $p-1$, assim, o retorno da função seria

$$\varphi(p) = p - 1$$

Outra propriedade que deve ser destacada é a multiplicidade referente a números primos entre si, ou seja, se dois números a e b são primos entre si, a função de Euler aplicada ao seu produto seria dada por

$$\varphi(a.b) = \varphi(a).\varphi(b)$$

Claro que dois números primos distintos serão primos entre si, portanto, para o valor da função de Euler aplicada ao parâmetro n citado anteriormente, tem-se

$$\varphi(n) = \varphi(p.q) = (p-1).(q-1)$$

O teorema de Euler diz que, para todo a e n primos entre si, tem-se que

$$a^{\varphi(n)} \equiv 1 (\mathrm{mod}\ n)$$

Observe-se que para a particularização de $n = p$ primo, tem-se o Pequeno Teorema de Fermat, ou seja

$$a^{\varphi(p)} \equiv a^{p-1} \equiv 1 (\mathrm{mod}\ p) \Rightarrow a^p \equiv a (\mathrm{mod}\ p)$$

A seguir, são estabelecidos os parâmetros e e d:

- Escolhe-se um valor (também denominado expoente de encriptação) $e < \varphi(n)$ (inteiro positivo), tal que e e $\varphi(n)$ sejam primos entre si.
- Encontra-se o valor (por sua vez, usualmente denominado expoente de decriptação) $d < \varphi(n)$ tal que

$$e.d \equiv 1(\text{mod } \varphi(n))$$

Este número d pode ser calculado utilizando-se o algoritmo de Euclides estendido, e representa o inverso multiplicativo modular de e.

A encriptação no algoritmo é feita pela função exponenciação discreta, ou modular. Seja x a mensagem e y o texto cifrado. É estabelecida a operação

$$y \equiv x^e(\text{mod } n)$$

A chave pública, de conhecimento geral, consiste no par (n, e) e a chave privada, de conhecimento restrito às partes comunicantes, no par de primos (n, d). . A chave privada deve ser, naturalmente, mantida em sigilo, bem como os valores de p, q e $\varphi(n)$, uma vez que a partir destes pode ser calculado o valor de d.

Após o recebimento da mensagem y, que pode ser transmitida por um meio inseguro de comunicação, a parte receptora da mensagem, a quem esta se destina, realiza a decriptação através da operação

$$y^d \equiv (x^e)^d \equiv x^{ed}(\text{mod } n)$$

A partir da relação pré-estabelecida entre e e d, tem-se

$$e.d \equiv 1(\text{mod } \varphi(n)) \Leftrightarrow e.d = Q.\varphi(n) + 1; \quad Q \in \mathbb{Z}$$

Aplicando-se este resultado na equação anterior, tem-se

$$y^d \equiv x^{ed} (\mathrm{mod}\ n)$$
$$\equiv x^{Q.\varphi(n)+1} (\mathrm{mod}\ n)$$
$$\equiv \left\{ \left[\left(x^{\varphi(n)} \right)^Q \right] .x \right\} (\mathrm{mod}\ n)$$

Pelo Teorema de Euler:

$$x^{\varphi(n)} \equiv 1 (\mathrm{mod}\ n)$$

$$y^d \equiv \left\{ \left[\underbrace{\left(x^{\varphi(n)} \right)}_{1}^{Q} \right] .x \right\} (\mathrm{mod}\ n)$$

$$x \equiv y^d (\mathrm{mod}\ n)$$

Dessa forma, com o conhecimento (privado) de d, o destinatário consegue retornar à mensagem original.

Ajustando-se de maneira um pouco distinta, o algoritmo RSA também pode ser usado em assinaturas digitais, nas quais deseja-se assegurar a identidade do emissor da mensagem.

O RSA sofre possibilidades de ataque dos tipos força bruta, ataques matemáticos e ataques de temporização. Estes últimos, mais sofisticados, se baseiam em diferenças de tempo no cálculo de uma operação (exponenciação modular, por exemplo) com diferentes operandos. Dentre as formas de se evitar este ataque, pode se realizar a exponenciação num período de tempo constante ou aplicar atrasos aleatórios.

2.4 ATAQUES AOS SISTEMAS CRIPTOGRÁFICOS

Criptoanálise é o nome dado à metodologia que estuda as formas de se desfazer os efeitos da criptografia, quebrando-se cifras ou forjando-se sinais que sejam aceitos como genuínos.

Alguns tipos de ataques de um criptoanalista são descritos a seguir.

No **ataque do tipo normal** (*ciphertext-only*), o inimigo só tem acesso à parte ou todo do criptograma, ou mensagem cifrada. Todo sistema criptográfico deve resistir a pelo menos esse tipo de ataque.

Uma técnica usual é o estudo da frequência de caracteres, por exemplo, na língua portuguesa a frequência em que ocorre a letra "a"é maior que a frequência de "y". Com a comparação das frequências dos caracteres cifrados, sendo a mensagem realizada numa determinada língua, é possível inferir alguma informação do texto cifrado.

No **ataque à informação** conhecida (*known-plaintext*) o inimigo tem conhecimento de alguns criptogramas e suas correspondentes mensagens originais, formados pela chave atual.

O **ataque à informação escolhida** (*chosen-plaintext*) é caracterizado pelo fato do inimigo ser capaz de submeter qualquer mensagem à criptografia e receber o criptograma correto com a chave atual utilizada.

O objetivo desse ataque é obter alguma informação que reduza a segurança do esquema criptográfico, e na pior das hipóteses, revelar o esquema de produção da chave.

Esse tipo de ataque tem maior importância quando se trata de criptografia assimétrica, onde a chave é pública e um inimigo pode encriptar qualquer mensagem de seu interesse.

O **ataque do tipo informação cifrada escolhida** (*chosen-ciphertext*) é um tipo de ataque no qual o inimigo pode escolher um criptograma arbitrário e obter o resultado correto para sua decriptação. Alguns algoritmos famosos na sua versão mais antiga eram suscetíveis a este ataque, como o RSA.

No **ataque de força bruta** (*brute-force*) o inimigo tenta todos os valores possíveis de chave. Este tipo de ataque só é executado quando os outros não forem eficazes.

Nesse tipo de ataque, o número esperado de tentativas antes da chave correta ser encontrada é a metade do tamanho da chave. Cifras simétricas com chaves de tamanho superior a 64 bits já foram quebradas através de ataques de força bruta. A possibilidade de quebra por esse ataque aumenta com o passar do tempo e a evolução tecnológica, com sua força computacional cada vez mais rápida.

Alguns tipos de encriptação não podem ser quebrados pelo método da força bruta, devido às suas propriedades matemáticas. Um exemplo é o *one-time-pad*, onde cada bit possui seu correspondente bit chave. Em relação ao ataque de força bruta, o *one-time-pad* apenas revela uma decodificação correta, mas qualquer outra possível combinação de bits pode resultar na mesma codificação, ou seja, não é possível distinguir uma chave verdadeira de outra falsa que resulte no mesmo texto cifrado.

A seguir apresentamos algumas das aplicações diretas de sequências de números aleatórios em criptografia.

2.5 UTILIZAÇÃO DE NÚMEROS ALEATÓRIOS EM CRIPTOGRAFIA

Dentre as aplicações que os números aleatórios podem ter, no que se refere ao campo da criptografia, pode-se enumerar:

- Chaves de seções e de mensagens de cifras simétricas, como o *one-time-pad*, o DES, o 3DES.
- Números que se combinam com senhas, para confundir programas de vírus que deduzem senhas.
- Inicialização de vetores de cifras de blocos.
- Chaves aleatórias relacionadas a protocolos.

Dessa maneira, no intuito de aumentar a segurança de sistemas criptográficos, deve-se analisar de maneira mais atenciosa algumas das maneiras conhecidas de se gerar números aleatórios, e de que forma estes podem ser aplicados em sistemas práticos. Esta análise é o objeto de discussão do capítulo a seguir.

3 GERADORES DE NÚMEROS ALEATÓRIOS

Apesar de ter se falado em número aleatório, este termo só adquire sentido quando visto dentro de um conjunto, uma vez que não é possível dizer se um número é mais ou

menos aleatório sem conhecer a sequência a que ele pertence e o número de amostras nessa sequência.

A geração aleatória em criptografia configura uma das fontes de segredo que consta nos sistemas criptográficos, porém, além disso, deve-se usar um bom algoritmo de encriptação e processos robustos de manejo de chaves para fortalecer ainda mais os algoritmos.

Na geração de chaves, a busca por sequências aleatórias é muito importante, principalmente no sentido de produção de chaves fortes. A probabilidade de ser gerada uma chave fraca é pequena. No algoritmo DES (Data Encription Standard) por exemplo possui 16 chaves fracas de um total de 2^{56}. Mas já que a chave não é de conhecimento geral, a fraqueza da chave não gera tanto perigo de enfraquecimento do sistema criptográfico, a não ser que haja falha no sistema de transmissão dos dados. A geração de chaves aleatórias para os sistemas criptográficos de chave pública é mais difícil, pois além de um rigor maior em relação à confirmação de aleatoriedade, estas chaves possuem características matemáticas específicas.

Algumas fontes físicas de geradores aleatórios são descritas a seguir.

3.1 FONTES ALEATÓRIAS

3.1.1 LANÇAMENTO DE MOEDA

De maneira mais teórica até uma sequência de zeros e uns obtida pelos lançamentos de uma moeda cuja face cara tenha aparecido basta para a obtenção de uma sequência, assumindo que esta moeda não seja viciada, ou seja, a probabilidade em infinitos lançamentos é igual a 1/2 . É claro que esta forma de geração de bits aleatórios não tem nenhum valor prático, até porque a taxa de geração não é suficiente e esta forma é ineficiente.

3.1.2 RUÍDO DO DIODO

O ruído de um diodo pode servir para produzir sequências de números aleatórios. Isso acontece por causa de defeitos inerentes nos materiais semicondutores do diodo e seu comportamento na região limite de polarização.

A única maneira de saber se esta forma de geração realmente promove números aleatórios é realizando testes estatísticos de aleatoriedade. Em geral, os diodos são fontes de sequências binárias que tem uma distribuição estatística uniforme, o que promove uma maneira eficiente de geração de sequências aleatórias, porém nem sempre a taxas que atendam determinados sistemas.

3.1.3 MATERIAL RADIOATIVO

Os materiais radioativos também podem funcionar como um gerador de números aleatórios, uma vez que a radioatividade decai aleatoriamente.

Um contador Geiger, conectado a um computador, pode ser usado para transformar estes dados em informação útil e administrável em criptografia.

Essa geração é rara e cara devido aos custos de manutenção e segurança dos dispositivos normalmente utilizados.

3.1.4 T12RNG

O T12RNG é um dispositivo capaz de gerar números aleatórios através de um hardware, que pode ser conectado à uma porta serial de computador. A origem da sua aleatoriedade na geração é devido ao ruído branco, propriedade dos semicondutores instalados no circuito.

Pode ser usado em senhas, chaves criptográficas, *one-time-pads*, etc.

Tem uma taxa de geração de 60 bytes?segundo, o que é considerado muito lento em aplicações práticas.

Como as fontes verdadeiramente aleatórias são originadas usualmente por fenômenos físicos, e estes são de difícil captação e transformação em dados utilizáveis a uma frequência alta o suficiente para garantir a comunicação eficiente, opta-se por utilizar esses raros números aleatórios como sementes de geradores de números pseudo-aleatórios. Estas sequências são suficientemente aleatórias desde que a semente não seja conhecida. A apresentação de alguns dos métodos utilizados por esses geradores de números pseudo-aleatórios é o objeto de estudo do capítulo a seguir.

4 GERADORES DE NÚMEROS PSEUDO-ALEATÓRIOS

De acordo com o uso específico das sequências de números pseudo-aleatórios, estas possuem diversas definições, algumas mais restritivas e outras mais abrangentes. Aqui se apresentam algumas definições abrangentes.

Uma sequência de números pseudo-aleatórios se caracteriza por ser uma sequência cujo período é longo o bastante para que uma sequência finita retirada desta, de comprimento razoável (de acordo com o uso que será feito dela) não seja periódica.

Para o propósito de auxílio aos processos criptográficos, o gerador de sequência de números pseudo-aleatórios deve ter a seguinte característica: assemelhar-se a uma sequência aleatória, ou seja, esta sequência deve passar em todos os testes estatísticos de aleatoriedade possíveis.

Todos os geradores pseudo-aleatórios são periódicos, mas estes períodos podem chegar a 2^256 bits ou mais, daí a sua utilidade em criptografia. Sendo mais prático de se implementar e possuindo um período tão longo como este, aumenta-se a facilidade de obtenção e de gerenciamento de chaves pseudo-aleatórias, características dos processos criptográficos.

Dos métodos de geração de chaves utilizados em criptografia, principalmente do tipo simétrica, os geradores de sequência de números pseudo-aleatórios são aqueles de aplicação mais importante. Para ser criptograficamente segura, uma sequência aleatória gerada deve ter as seguintes propriedades:

- Ser imprevisível – De modo que seja computacionalmente impraticável prever o próximo bit aleatório da sequência, conhecendo todo o algoritmo, hardware gerador ou toda a sequência anterior de bits.
- Não pode ser reproduzida fielmente – Se um gerador for iniciado com a mesma entrada já utilizada, a saída deve ser uma sequência aleatória completamente descorrelatada da primeira.

Mesmo possuindo diversas vantagens, este método de geração de sequências só pode ser utilizado em conjunto com o método inteiramente aleatório.

A seguir apresentamos alguns dos métodos de geração de sequências aleatórias e pseudo-aleatórias.

4.1 GERAÇÃO DE SEQÜÊNCIAS DE NÚMEROS ALEATÓRIOS UTILIZANDO SOFTWARES

Apesar da prática de utilização de números aleatórios gerados por compiladores em sistemas, por exemplo, de jogos, não é recomendado utilizar os compiladores presentes no mercado como fonte geradora porque eles não são suficientemente seguros para propósitos práticos de encriptação, não estabelecendo a segurança devida aos sistemas criptográficos, já que a criptografia é extremamente sensível às propriedades dos geradores de números aleatórios.

Ao usar um gerador de números aleatórios fraco pode-se obter uma sequência aleatória com certo grau de correlação que irá influenciar no resultado final da encriptação.

É impossível produzir algo verdadeiramente aleatório em um computador, ou seja, por software, isto porque as operações que decorrem neste são de natureza determinística, impedindo a ocorrência de aleatoriedade.

Os algoritmos realizados em computador para a obtenção de números aleatórios contribuem um sistema criptográfico quando a sequência gerada possui um período muito grande, mas ainda não são considerados aleatórios, apenas pseudo-aleatórios, pois qualquer sequência periódica não é aleatória e por ser periódica se torna previsível.

No âmbito da mecânica quântica defende-se que fenômenos verdadeiramente aleatórios só existem na natureza, no mundo real, enquanto que mecanismos computacionais e algoritmos só podem trazer resultados pseudo-aleatórios.

4.2 ALGORITMOS DE GERAÇÃO DE SEQÜÊNCIAS DE NÚMEROS PSEUDO-ALEATÓRIOS

O estudo de algoritmos de geração pseudo-aleatórios serve para criar mecanismos auxiliares para a implementação de geradores de sequências de números aleatórios.

Como a geração de números verdadeiramente aleatórios é difícil e muitas vezes

a taxa de geração não atende aos critérios desejados de um projeto, os algoritmos pseudo-aleatórios auxiliam ao atendimento desses critérios da seguinte maneira: esses números aleatórios podem ser usados como *sementes* (dados de iniciação do gerador) de geradores pseudo-aleatórios, que possuem taxas de geração de bits maior.

A utilização de mais de um tipo de algoritmo pseudo-aleatório pode ser utilizada na tentativa de aumentar o período da sequência pseudo-aleatória.

É por isso que será explicitado um conjunto satisfatório de algoritmos de geradores pseudo-aleatórios.

4.2.1 MÉTODO CONGRUENTE LINEAR

Muitos geradores de números pseudo-aleatórios utilizados hoje em dia são modificações do método linear congruente. Ele se baseia na seguinte relação de recorrência:

$$x_{n+1} = (a.x_n + b)(\text{mod } m), n \geq 0$$

O valor x_0 inicial é chamado semente, a é o multiplicador, b o incremento, m é a quantidade de número diferentes que se deseja gerar, n é o índice dos números. A escolha adequada desses valores é fundamental para a determinação do período da sequência de saída. Pela simples observação constata-se que o período será sempre menor que m, e que para se ter uma sequência maior deve-se aumentar o valor de m.

Este algoritmo é extremamente simples, porém atende muito bem aos requisitos de geradores de sequências de números pseudo-aleatórios, desde que seja realizada a escolha adequada dos parâmetros, como já foi citado.

Para obter máxima eficiência do algoritmo, a escolha dos parâmetros deve atender:

i. O módulo de m deve ser grande. Uma vez que os valores de x estarão entre 0 e $m-1$, o período nunca será maior do que m;

ii. Para que a computação de *mod* m seja eficiente, m deve ser uma potência de 2, isto é, 2^k. Neste caso, o *mod* m poderá ser obtido truncando-se o resultado à direita por k bits.

iii. Se b for diferente de zero, o máximo período possível m é obtido se e somente se:

   a. os inteiros m e b são primos entre si;

   b. todo número primo que é um divisor de m, é também um divisor de $a - 1$;

   c. $a - 1$ é um múltiplo de 4, se o inteiro m é múltiplo de 4.

iv. Se $b = 0$, e m é potência de 2, o maior período P possível será $P = m/4$, considerando que: x_0 (semente) seja um número ímpar e o multiplicador a seja dado por $a = 8k + 3$ ou $a = 8k + 5$, para algum k inteiro não-negativo.

Estas condições foram apresentadas em resumo e sem demonstração, mas para um estudo mais aprofundado, estes resultados são apresentados nas referências [Jain (1991)], [Banks (1996)], [Law (1991)] e [Dudewicz (1985)].

4.2.2 LFSR (LINEAR FEEDBACK SHIFT REGISTER)

Esse algoritmo tem sido utilizado na criptografia militar dos Estados Unidos desde o começo da eletrônica.

O LFSR é formado por duas partes: um deslocador à direita e uma função de retorno na formação de bits.

Em períodos de tempo definidos, um bit (bit final) é requisitado e o bloco restante é deslocado para direita e um novo bit é inserido à esquerda. Este bit inserido é resultado de uma determinada operação realizada com os demais bits.

A saída utilizada para formar uma sequência é o conjunto de bits mais à direita, que saem do registrador a cada ciclo de deslocamento. O período de um registrador como este é o tamanho da sequência de saída até que a sequência comece a repetir.

O LFSR é o tipo mais simples de registrador e o mais comum usado em criptografia. A função de retorno correspondente a este algoritmo é a função *xor* de bits remanescentes no registrador.

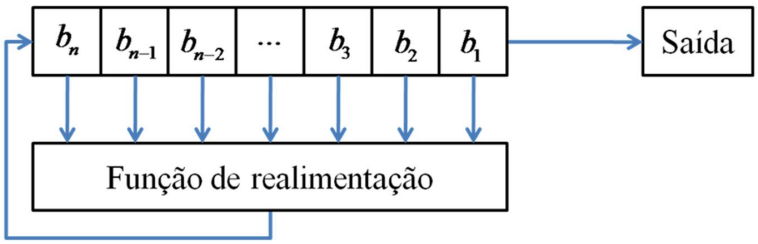


Figura 4.2.2.1 – Esquema geral do LFSR

O registrador acima se comporta da seguinte $b_1 \oplus b_4$ maneira para uma semente igual a 0001.

Registrador	$b_1 \oplus b_4$	Saída
0001	1	1
1000	1	0
1100	1	0
1110	1	0
1111	0	1
0111	1	1
1011	0	1
0101	1	1
1010	1	0
1101	0	1
0110	0	0
0011	1	1
1001	0	1
0100	0	0
0010	0	0
0001	1	1
1000	1	0
1100	1	0

Tabela 4.2.2.1 – Esquema de funcionamento do LFSR

É possível observar que seu período é de tamanho $15.(2^4 - 1)$.

Por motivo óbvio a semente nunca pode ser nula.

Um gerador LFSR pode gerar uma sequência pseudo-aleatória de período até $2n-1$, sendo n o número de bits que o registrador comporta.

Para que um LFSR tenha período máximo, o polinômio formado pela sequência tap (lista de bits remanescentes no registrador, também chamada configuração de Fibonacci) mais um deve ser um *polinômio primitivo em módulo 2*.

Esta definição de *polinômio primitivo em módulo 2* é estudada no âmbito os "campos de Galois", aspectos matemáticos que saem do escopo deste trabalho.

4.2.3 GERADOR GEFFE

Este gerador de sequência-chave usa três LFSRs, combinados de uma maneira não linear. Dois deles são entradas de um multiplexador, e o terceiro controla a saída do multiplexador.

Seja a_1, a_2 e a_3 as saídas desses três LFSRs, então a saída b do gerador Geffe é:

$$b = (a_1 \wedge a_2) \oplus [(\sim a_2) \wedge a_3]$$

Na equação anterior, os símbolos $\wedge$ e $\sim$ representam as operações lógicas de adição ("e" lógico) e negação, respectivamente.

Em forma de fluxograma, tem-se:

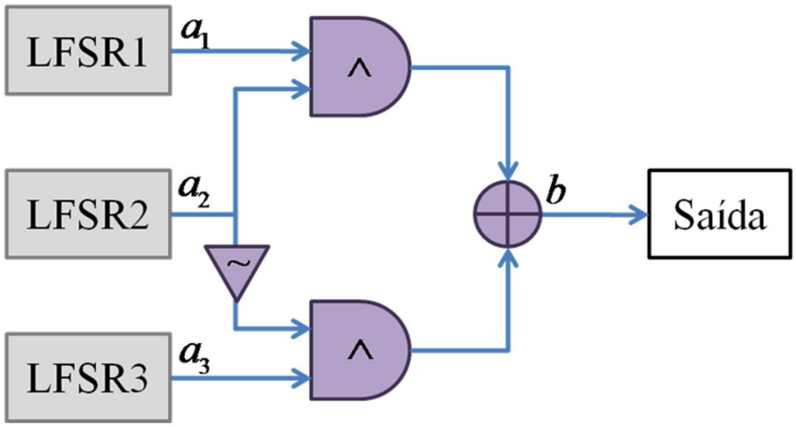


Figura 4.2.3.1 – Gerador Geffe

Se os LFSRs possuem os seguintes períodos n_1, n_2 e n_3, o período do gerador Geffe resultante n_{GEFFE} é o mínimo múltiplo comum desses períodos:

$$n_{GEFFE} = mmc(n_1, n_2, n_3)$$

Apesar deste gerador ser vulnerável ao teste de correlação a sua análise motiva a criatividade do leitor para o pensamento de quantos algoritmos mais possam existir baseadas em ideias simples como esta ou com o agrupamento de várias ideias simples.

É possível também implementar a ideia deste gerador para uma função de várias entradas e para cada uma delas um LFSR.

5 TRANSPOSIÇÃO DIDÁTICA

5.1 SUGESTÕES DE AULA

Tendo em vista a importância dos assuntos tratados nos capítulos anteriores, pode-se dizer que a introdução desses temas em atividades de sala de aula, principalmente para alunos de ensino médio, é de grande interesse, principalmente quando se observa que os algoritmos utilizados em sistemas criptográficos, notadamente aqueles do tipo assimétrico (ou de chave pública), configuram, em sua maioria, uma aplicação direta dos conhecimentos de Matemática Elementar, em particular relacionados à Aritmética Modular. Dessa forma, a introdução adequada da aplicação da Matemática na Criptografia pode, além de agregar maior conhecimento referente a uma atividade recorrente no dia a dia dos alunos, sob a ótica da aplicação da Criptografia nos sistemas de segurança de informação, trazer maior interesse nestas e em outras aplicações dos conceitos de Aritmética Modular. Este capítulo, portanto, faz algumas sugestões de como abordar alguns desses temas, de modo que alunos com conhecimento elementar das operações modulares, não se sintam embaraçados ao serem apresentados, não somente à idéia geral de como os sistemas criptográficos do tipo assimétrico funcionam, mas também da abordagem matemática a ele inerente.

5.1.1 DIFFIE-HELMAN

Para a introdução do algoritmo de *"key exchange"* de Diffie-Helmman, a abordagem didática mais usual faz uma analogia entre a troca de chaves e a mistura de cores. Esta analogia tem ainda maior semelhança com o algoritmo tendo em vista que o processo de misturar duas ou mais cores é bastante simples, entretanto, separá-las em busca das cores iniciais é uma tarefa que pode ser considerada árdua, de modo semelhante ao que se espera de funções de encriptação em geral. Tal analogia é mostrada no diagrama a seguir. As três principais etapas do processo são descritas fazendo a comparação entre o método de mistura de tintas e o algoritmo Diffie-Hellman delineado no capítulo 2.3.1 deste trabalho.

Na etapa 1, os dois usuários A e B têm conhecimento de uma tinta comum, que será misturada com as tintas que possuem, referentes às suas cores secretas, gerando duas cores distintas das suas cores secretas, a serem trocadas através de um meio de comunicação não-seguro. Esta etapa é equivalente às operações

$$y_A \equiv g^{x_A} (\mathrm{mod}\ p)$$

$$y_B \equiv g^{x_B} (\mathrm{mod}\ p)$$

onde g e p são valores conhecidos de ambos, e possivelmente de um usuário criptoanalista interessado em intervir na comunicação de A e B. A analogia direta entre estes valores se faz com o uso da tinta comum, que serve para mudar o valor das chaves privadas individuais, ou das cores secretas, representadas pelos valores x_A e x_B.

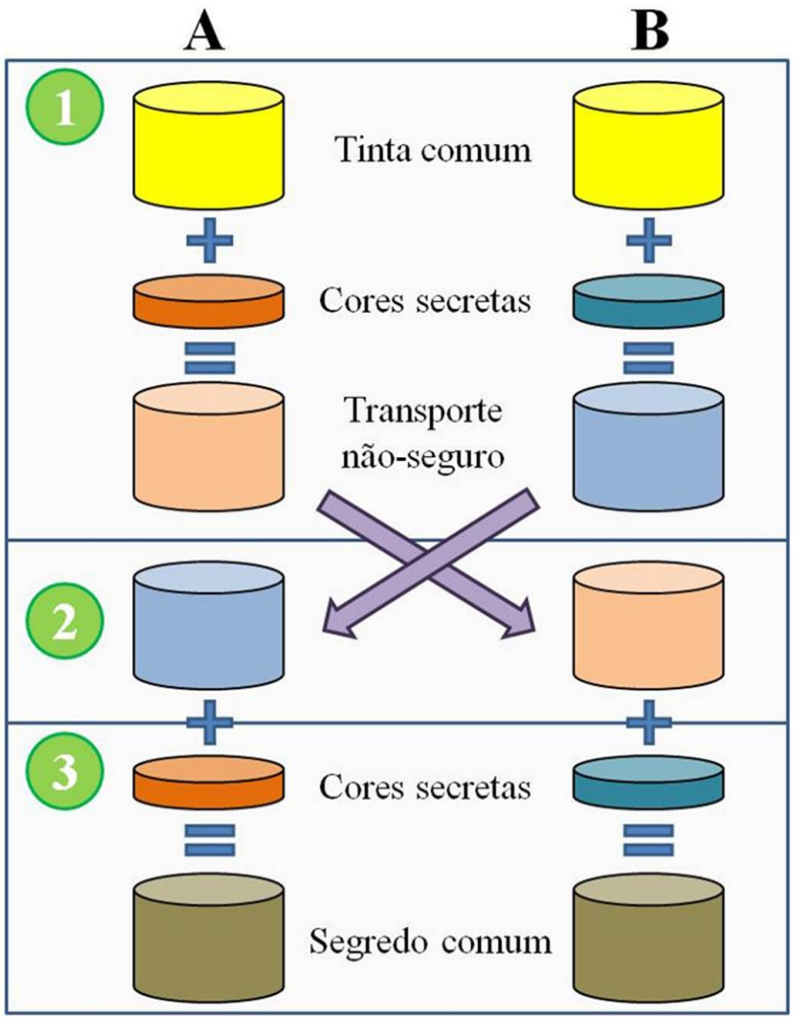


Figura 5.1.1.1 – Esquema de operação – Diffie-Hellman

As operações previamente descritas correspondem à mistura entre as cores secretas e a tinta comum. Estas operações são simples de serem realizadas, bem como a mistura de duas cores de tinta, entretanto, a partir do seu resultado, ou seja, dos valores y_A e y_B é extremamente difícil retornar aos valores iniciais x_A e x_B, tal qual é a dificuldade de retornar às cores secretas utilizadas inicialmente.

Neste ponto, cabe-se ressaltar que a dificuldade em se retornar aos valores x_A e x_B consiste basicamente no enorme trabalho computacional requerido em se realizar operações modulares de retorno (no caso, de logaritmo discreto modular), principalmente naquela envolvendo números primos p de valores elevados, tipicamente da ordem de centenas de casas decimais.

Na etapa 2, o material misturado por A é enviado a B e vice-versa, através de um meio que pode ou não ser seguro o suficiente para a comunicação de interesse. Esta etapa é idêntica à idéia do algoritmo Diffie-Hellman, ou seja, os valores y_A e y_B são trocados entre A e B, utilizando-se um meio de comunicação que pode ou não ser seguro.

Na etapa 3, tanto A como B misturam o material recebido com suas cores secretas, tendo, naturalmente, como resultado as mesmas cores finais. Isto obviamente ocorre pois as cores misturadas por eles ao longo do processo são as mesmas, sendo apenas a ordem em que elas foram misturadas distintas. Esta mistura corresponde às operações:

- Para o usuário A

$$k_{AB} \equiv [y_B]^{x_A} \,(\mathrm{mod}\ p)$$
$$k_{AB} \equiv [(g^{x_B})]^{x_A} \,(\mathrm{mod}\ p)$$
$$k_{AB} \equiv g^{x_B \cdot x_A} (\mathrm{mod}\ p)$$

- Para o usuário B

$$k_{BA} \equiv [y_A]^{x_B} \,(\mathrm{mod}\ p)$$
$$k_{BA} \equiv [(g^{x_A})]^{x_B} \,(\mathrm{mod}\ p)$$
$$k_{BA} \equiv g^{x_A \cdot x_B} (\mathrm{mod}\ p)$$

Assim, o resultado obtido para estas operações é idêntico, como esperado, pois

$$k_{AB} = k_{BA} \Leftrightarrow g^{x_A \cdot x_B} \equiv g^{x_B \cdot x_A} (\mathrm{mod}\ p)$$

Após as três etapas, os usuários possuem uma cor de tinta idêntica, que corresponde ao seu segredo comum, assim como as chaves $k_{AB} = k_{BA}$. Esse valor é usualmente utilizado como chave secreta para sistemas criptográficos simétricos, como o DES ou o *one-time-pad*.

Neste ponto, pode ser interessante apresentar um exemplo numérico de como o algoritmo Diffie-Hellman pode ser aplicado. Para tanto, sejam escolhidos os valores de g, p, x_A e x_B:

$$g = 3$$

$$p = 17$$

$$x_A = 15$$

$$x_B = 13$$

Assim, aplicando-se as operações adequadas, tem-se

$$y_A \equiv 3^{15} \equiv 14.348.907 \equiv 6(\text{mod } 17)$$

$$y_B \equiv 3^{13} \equiv 1.594.323 \equiv 12(\text{mod } 17)$$

Após a troca entre A e B desses valores, são realizadas as operações do protocolo:

- Para o usuário A

$$k_{AB} \equiv [1.594.323]^{15}(\text{mod } 17)$$

$$k_{AB} \equiv [(3^{13})]^{15}(\text{mod } 17)$$

$$k_{AB} \equiv 3^{13.15} \equiv 10(\text{mod } 17)$$

- Para o usuário B

$$k_{BA} \equiv [14.348.907]^{13}(\text{mod } 17)$$

$$k_{BA} \equiv [(3^{15})]^{13}(\text{mod } 17)$$

$$k_{BA} \equiv 3^{15.13} \equiv 10(\text{mod } 17)$$

Dessa forma, o valor da chave privada comum a ambos é

$$k_{AB} \equiv k_{BA} \equiv 10(\text{mod } 17)$$

É importante acrescentar que o valor de $p = 17$ foi utilizado a efeito de exemplo, e não constitui um valor tipicamente utilizado, tendo em vista que o cálculo de x_A ou x_B poderia ser realizado em tempo reduzido a partir do valor de p e de y_A e y_B.

Entretanto, para valores elevados de p, da ordem de centenas de casas decimais, não é possível determinar estes valores em um intervalo de tempo razoável.

5.1.2 RSA

O estudo do algoritmo RSA, sob a óptica de alunos de Ensino Médio, geralmente apresenta dificuldades em alguns conceitos tipicamente apresentados em cursos de nível superior, e dentre eles, pode-se citar a aplicação da função $\varphi(n)$ de Euler, também referida como função *totiente* ou *tociente* de Euler, e do Teorema de Euler da Teoria dos Números, ou Teorema do Tociente.

Com relação à função $\varphi(n)$ a introdução de sua definição não parece apresentar grandes acréscimos aos conceitos adquiridos previamente, e sua exposição pode ser feita de modo geral como *"a quantidade de números naturais menores ou iguais a n com os quais n é primo entre si"*. Abordando com uma linguagem um pouco mais matemática, pode-se escrever:

$$\varphi(n) = \#\left\{k \in \mathrm{N} \mid \left[k \leq n\right] \wedge \left[\mathrm{mdc}\left(k, n\right) = 1\right]\right\}$$

Esta ideia é melhor estabelecida quando alguns valores numéricos são apresentados. Por exemplo, utilizando os valores de n iguais a 8, 9, 11 e 13. As tabelas a seguir apresentam os valores menores que n e com os quais estes números são primos entre si. Naturalmente, o valor computado para a função é o total de valores destacados em verde, resultando no total destacado em azul.

$n = 8$	Primo entre si
1	SIM
2	NÃO
3	SIM
4	NÃO
5	SIM
6	NÃO
7	SIM
8	NÃO
$\varphi(8)$	4

$n = 9$	Primo entre si
1	SIM
2	SIM
3	NÃO
4	SIM
5	SIM
6	NÃO
7	SIM
8	NÃO
9	NÃO
$\varphi(9)$	5

Tabelas 5.1.2.1 e 5.1.2.2 – Função $\varphi(n)$ aplicada a $n = 8$ e $n = 9$

$n = 11$	Primo entre si
1	SIM
2	SIM
3	SIM
4	SIM
5	SIM
6	SIM
7	SIM
8	SIM
9	SIM
10	SIM
11	NÃO
$\varphi(11)$	10

$n = 13$	Primo entre si
1	SIM
2	SIM
3	SIM
4	SIM
5	SIM
6	SIM
7	SIM
8	SIM
9	SIM
10	SIM
11	SIM
12	SIM
13	NÃO
$\varphi(13)$	12

Tabelas 5.1.2.3 e 5.1.2.4 – Função $\varphi(n)$ aplicada a $n = 11$ e $n = 13$

A partir destes últimos dois valores ($n = 11$ e $n = 13$), é interessante apontar uma propriedade interessante da função em estudo, que refere-se à sua aplicação em números primos. É natural perceber que qualquer número primo p terá função $\varphi(p)$ tal que

$$\varphi(p) = p - 1$$

Esta propriedade não deve apresentar maior dificuldade de compreensão tendo em vista a própria definição de número primo, a partir da qual um número primo somente será dividido por 1 e por si próprio. Dessa forma, em conjunto com a definição da função totiente de Euler, tem-se o resultado apresentado na equação anterior. Deve-se destacar a importância deste resultado, principalmente no que se refere ao estudo do RSA, quando aplicada com outra propriedade da função $\varphi(p)$, aquela que se refere à sua multiplicidade, ou seja:

$$\varphi(a.b) = \varphi(a).\varphi(b)$$

sendo que a e b são primos entre si.

No que se refere à demonstração deste resultado, este provavelmente deve ser omitido dos estudantes de Ensino Médio, pois apresenta um nível de dificuldade acima daquele usualmente apresentado a eles. Entretanto, sua aplicação ao produto de nú-

meros primos é fundamental ainda no estudo do RSA. Dessa forma, para o produto de dois primos p e q, tem-se que

$$\varphi(p.q) = \varphi(p).\varphi(q) = (p-1).(q-1)$$

Acrescente-se a este último resultado o Teorema de Euler da Teoria dos Números, cuja demonstração também deve ser omitida. Este teorema estabelece que

$$a^{\varphi(n)} \equiv 1(\bmod\ n)$$

onde a é primo entre si com n. Novamente, para efeito didático, pode ser interessante apresentar um exemplo numérico desses últimos conceitos. Assim, sejam escolhidos os valores de a, p e q.

$$a = 2$$

$$p = 5$$

$$q = 7$$

A partir destes valores pode-se calcular

$$n = p.q = 5.7 = 35$$

$$\varphi(n) = (p-1).(q-1) = 4.6 = 24$$

Assim, substituindo-se os valores conforme o proposto, para verificação, tem-se

$$a^{\varphi(n)} = 2^{24} = 16.777.216$$

$$a^{\varphi(n)} = 2^{24} = 35.(479.349) + 1$$

$$a^{\varphi(n)} \equiv 2^{24} \equiv 1(\bmod\ 35)$$

$$a^{\varphi(n)} \equiv 1(\bmod\ n)$$

sendo esta última a equação do Teorema de Euler, conforme esperado. Para a utilização em criptografia, deve-se adotar um procedimento idêntico àquele descrito anteriormente, seja ele:

- Escolhe-se um valor (também denominado expoente de encriptação) $e < \varphi(n)$ (inteiro positivo), tal que e e $\varphi(n)$ sejam primos entre si. Para exemplificação numérica, escolha-se $e = 13$.

- Encontra-se o valor (por sua vez, usualmente denominado expoente de decriptação) $d < \varphi(n)$ tal que

$$e.d \equiv 1(\text{mod } \varphi(n))$$

Neste ponto, pode-se determinar o valor de d a partir da definição anterior e da congruência indicada, ou seja:

$$e.d \equiv 1(\text{mod } \varphi(n))$$

$$e.d = k.\varphi(n) + 1$$

$$d = \frac{k.\varphi(n) + 1}{e}$$

onde k é um número inteiro a ser determinado, em geral, pelo algoritmo de Euclides estendido.

Adotando, para esta nova análise, os seguintes valores

$$p = 19$$

$$q = 23$$

Pode-se calcular

$$n = p.q = 19.23 = 437$$

$$\varphi(n) = (p-1).(q-1) = 18.22 = 396$$

Para o cálculo de d, aplicando-se diretamente, por inspeção, $k = 2$, tem-se que

$$d = \frac{k.\varphi(n) + 1}{e}$$

$$d = \frac{2.396 + 1}{13}$$

$$d = 61$$

Assim, tendo calculado estes termos, pode-se estabelecer para a situação de teste apresentada, os valores privados e públicos que poderiam ser utilizados para essa comunicação teórica. As tabelas a seguir consolidam estes valores.

Valores privados	
p	19
q	23
$\varphi(n)$	396
d	61

Valores públicos	
n	437
e	13

Tabelas 5.1.2.5 e 5.1.2.6 – Valores públicos e privados do exemplo apresentado

Suponha-se, então, que um usuário B queira enviar uma mensagem a um usuário A utilizando o método RSA como forma de criptografia dessa comunicação, com base nos valores apresentados. Assim, considere-se que os valores públicos são de conhecimento de A e B (assim como de qualquer usuário da rede, inclusive de um criptoanalista C), entretanto, os valores privados são de conhecimento exclusivo de A, a quem se destina a mensagem.

Para aplicação didática, é possível visualizar esta situação da seguinte maneira: o usuário A disponibiliza um cadeado aberto em um diretório público. Este representa seu segredo público. Qualquer pessoa pode anexar uma mensagem a este cadeado, uma vez que ele é de domínio público. Entretanto, quando B vincula uma mensagem ao cadeado, antes de enviá-lo novamente a A ele realiza o fechamento do cadeado. Esta etapa do processo caracteriza bem o que a função de encriptação deve realizar, uma operação simples de realizar, o fechamento do cadeado, entretanto, sua abertura é de difícil operação sem a posse da chave.

Dessa forma, o cadeado fechado pode ser enviado em segurança em uma rede de comunicação, ainda que esta não seja segura, pois considera-se que sua abertura, sem a posse da chave é extremamente difícil.

Nesse ponto, ao receber o cadeado, o usuário A, que possui a chave privada, representada pela chave do cadeado, pode abri-lo facilmente, tendo acesso à mensagem decriptada.

Muitas vezes questiona-se a possibilidade de um usuário mal-intencionado usar este cadeado aberto e enviar uma mensagem, passando-se por B. Tal problema pode ser resolvido estabelecendo-se, entre A e B o uso de assinaturas digitais, as quais podem ser geradas utilizando-se o método de Diffie-Hellman, com pequenas alterações.

O esquema a seguir delineia o funcionamento do RSA previamente descrito.

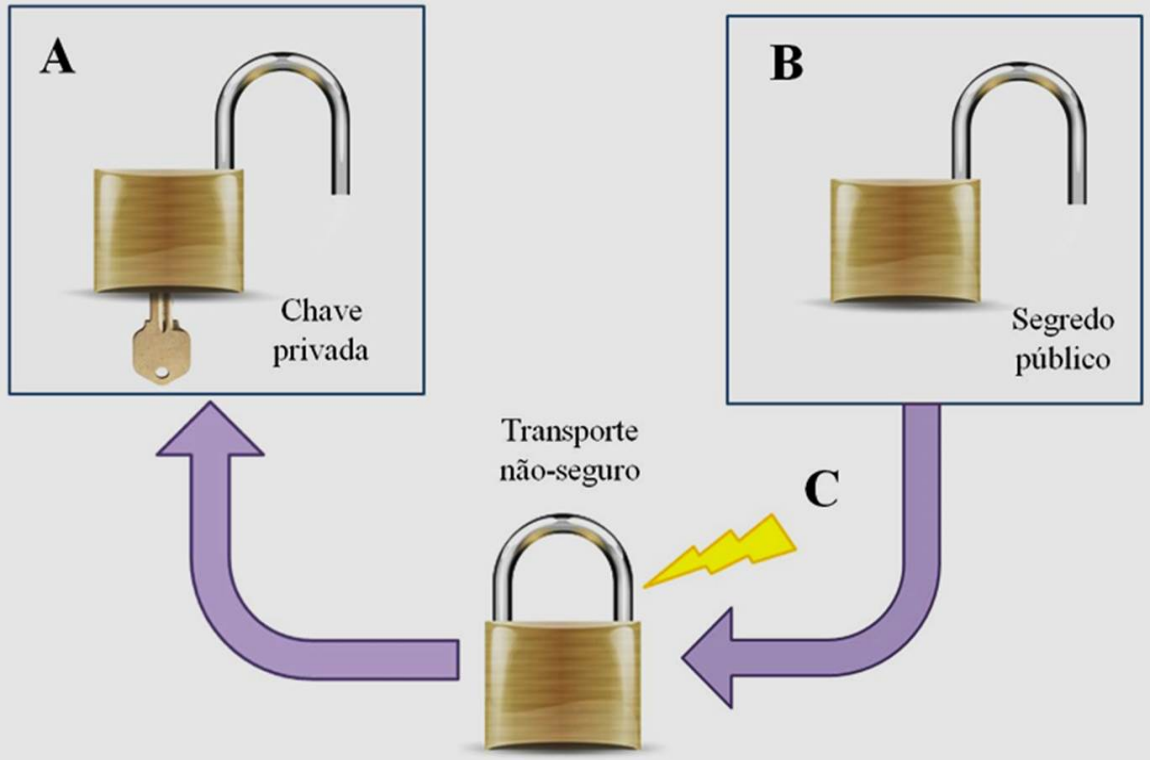


Figura 5.1.2.1 – Esquema de operação – RSA

Naturalmente que neste caso, os valores públicos e e n constituem em conjunto o segredo público, ou como usualmente referido, a chave pública e os valores d e n constituem em conjunto a chave privada, que servirá para abrir o cadeado. Utilizando-se o exposto anteriormente, tem-se que a operação

$$y \equiv x^e (\mathrm{mod}\ n)$$

representa o ato de fechar o cadeado com a mensagem x, tendo em vista que a partir do conhecimento de y, e e n é extremamente difícil obter o valor de x, ou abrir o cadeado. Assim, y mimetiza o cadeado fechado com a mensagem. Ao receber o valor y proveniente de B, o usuário A realiza a operação

$$y^d \equiv (x^e)^d \equiv x^{e.d} (\text{mod } n)$$

A relação entre e e d fora estabelecida de tal forma que

$$e.d \equiv 1 (\text{mod } \varphi(n)) \Leftrightarrow e.d = k.\varphi(n) + 1; k \in Z$$

Esta relação aplicada à equação anterior leva a

$$\begin{aligned} y^d &\equiv x^{e.d} (\text{mod } n) \\ &\equiv x^{k.\varphi(n)+1} (\text{mod } n) \\ &\equiv \left\{ \left[\left(x^{\varphi(n)} \right)^k \right] .x \right\} (\text{mod } n) \end{aligned}$$

que pelo teorema de Euler retorna

$$\begin{aligned} x^{\varphi(n)} &\equiv 1 \left(\text{mod } n\right) \\ y^d &\equiv \left\{ \left[\underbrace{\left(x^{\varphi(n)} \right)^k}_{1} \right] \cdot x \right\} \left(\text{mod } n\right) \\ x &\equiv y^d \ \left(\text{mod } n\right) \end{aligned}$$

que constitui a mensagem original de B.

Observe-se que os valores de p e q, bem como $\varphi(n)$, servem apenas como base para a construção da chave d, e portanto devem também ser mantidos em sigilo.

Retornando ao exemplo original, no qual B gostaria de enviar uma mensagem a A. Suponha-se que a mensagem de interesse seja a palavra "MAT". Para isso, B converte esta mensagem a um valor numérico. Utilizando-se o método mais usual, convertendo-se com base no código ASCII para decimal, seria obtido, para cada letra

Letra	ASCII – Decimal
M	77
A	66
T	84

Tabela 5.1.2.7 – Conversão para ASCII decimal das letras M, A e T

Para efeito de simplicidade no envio, cada letra seria enviada individualmente. Seja a primeira letra, M, a ser enviada. Assim, o valor de x, referente à mensagem seria

$$x_1 = 77$$

Aplicando-se o método RSA descrito, com os valores previamente estabelecidos, $n = 437$, $e = 13$ e $d = 61$, o usuário B efetuaria as operações

$$y_1 \equiv x_1^{\,e} \pmod{n}$$
$$y_1 \equiv 77^{13} \pmod{437}$$
$$y_1 \equiv 3344871416\,1911959408\,89917 \pmod{437}$$
$$y_1 \equiv 248 \pmod{437}$$

Este valor seria enviado a A, que realizaria

$$y_1^{\,d} \equiv 248^{61} \pmod{437}$$
$$y_1^{\,d} \equiv 248 \cdot \left[248^{10}\right]^6 \pmod{437}$$
$$y_1^{\,d} \equiv 248 \cdot \left[8800691718\,6476071872\,1024\right]^6 \pmod{437}$$
$$y_1^{\,d} \equiv 248 \cdot [400]^6 \pmod{437}$$
$$y_1^{\,d} \equiv 1015808000\,000000000 \pmod{437}$$
$$y_1^{\,d} \equiv 77 \pmod{437}$$
$$x_1 \equiv y_1^{\,d} \equiv 77 \pmod{437}$$

retornando, assim, a mensagem enviada por B. O processo é igualmente repetido para os demais valores da mensagem.

Apesar de os valores encontrados possuírem um número relativamente elevado de casas decimais para grande quantidade das calculadoras de mão, a enorme maioria dos computadores consegue realizar estas operações de maneira relativamente simples e rápida, e é provável que seja ainda mais didático apresentar de que maneira essas operações podem ser realizadas em um computador, preferencialmente expondo o passo a passo destas através de recursos audiovisuais.

Deve-se notar ainda que, para aplicações práticas, os valores de p e q possuem diversas casas decimais, tipicamente da ordem de centenas, de modo a tornar ainda mais difícil o processo de criptoanálise.

5.2 TRABALHOS FUTUROS

Tendo em vista que as análises apresentadas, conforme percebido ao longo do trabalho, envolvem usualmente valores numéricos bastante elevados, a sistematização das operações envolvidas pode ser sobremaneira trabalhosa quando realizadas braçalmente. Entretanto, uma vez que se tenha embasado solidamente os conceitos matemáticos apresentados e exemplificados com valores reduzidos, pode-se perceber a oportunidade de introduzir nesse contexto o estudo e aplicação de linguagens de programação, de modo a utilizar esta ferramenta, que tem se tornado cada vez mais recorrente no dia-a-dia dos estudantes de Ensino Médio.

Naturalmente que para o aluno iniciante em qualquer tipo de linguagem de programação, este processo seria improdutivo, entretanto, para sua aplicação como objetivo final de um curso de duração de um semestre que introduzisse alguma linguagem de programação mais simples e amigável ao usuário, a proposta parece razoável.

Possivelmente os estudantes que realizariam maior procura seriam aqueles com interesses diretamente relacionados à Computação e Matemática, bem como aqueles que costumeiramente participam de Olimpíadas aplicadas a essas áreas, no entanto, a demonstração de que, ao final do curso, estes obteriam conhecimento, ainda que simplificado, nos conceitos relacionados à Criptografia pode trazer ainda maior interesse.

Em paralelo a essa proposta, pode-se sugerir, como continuação do curso previ-

amente citado, outro curso que aplicasse e implementasse sistemas criptográficos de chave simétrica, particularmente o protocolo DES, tendo em vista sua relativa simplicidade. De acordo com o rendimento neste curso, poderia ainda ser aplicada a ideia de geradores de sequências pseudo-aleatórias, bem como sua implementação prática, de modo que as sequências pudessem ser utilizadas como chaves no DES. Este curso deveria, obviamente, ter como pré-requisito o primeiro, de modo a aproveitar os conhecimentos de Criptografia e Programação adquiridos previamente.

Naturalmente que estes cursos, que em conjunto levariam algo em torno de dois semestres, não poderiam tomar um tempo muito elevado dos alunos em sua rotina diária, principalmente por se tratarem de atividades extracurriculares. Aliás, é exatamente esta característica, de ser uma atividade que não faz parte da grade curricular usual aliada ao fato desse estudo se tratar de uma atividade claramente interdisciplinar que pode atrair o interesse de um número maior de alunos.

5.3 Considerações finais

Observa-se que em muitos países o estímulo à participação de atividades extra-curriculares tem trazido resultados extremamente positivos, principalmente no que diz respeito à inovação tecnológica e intelectual, os quais estão diretamente relacionados ao desenvolvimento educacional e econômico.

Este trabalho apresenta, portanto, uma breve introdução aos principais métodos de aplicação de sistemas criptográficos, tanto de chave simétrica como assimétrica, e sugere ao longo deste capítulo, não somente uma forma de abordar estes assuntos em sala de aula como propõe que eles sejam levados adiante, com a execução de um ou mais cursos extracurriculares que realizassem sua implementação computacional, tendo em vista que o crescimento desta e de atividades semelhantes certamente acrescentaria, e muito, no que diz respeito à qualidade educacional, e direcionamento vocacional de nossos alunos.

Referências

[1] BEUTELSPACHER, Albrecht. *Cryptography.* Ed Mathematical Association of America. Segunda Edição.

[2] GARCIA, L. Alberto. *Probability and Random Processes for Electrical Engineerig.* Ed Addison-Wesley Publishing Company. Segunda Edição.

[3] KOBLITZ, Neal. *A course in number Theory and Cryptography.* Ed Springer. Primeira Edição.

[4] MARSAGLIA, G. DIEHARD: a batery of tests of randomness. URL http://stat.fsu.edu/ geo/diehard.html.

[5] MONTGOMERY, C. DOUGLAS; RUNGER, C. G. *Estatística Aplicada e Probabilidade para Engenheiros.* LTC . Segunda Edição.

[6] NIST, National Institute of Standards and Technology. Advanced Encryption Standard. URL http://csrc.nist.gov/CryptoToolkit/aes/.

[7] PAPOULIS, A.; PILLAI, *S. Probability, Random Variables and Stochastic Processes.* McGraw-Hill. Quarta Edição.

[8] RANDOM.C. Linux kernel random number generator. URL . http://www.kernel.org.

[9] SCHEINER, B. Protocols, Algorithms, and Source Code in C. *Applied Cryptography.* Ed. John Wiley e Sons, Ltd. Segunda Edição.

[10] SEDRA, S. ADEL; SMITH, C.K. *Microeletrônica.* Makron Books Ltda, 2000. Quarta Edição.

[11] TANENBAUM, A. S. Computer Networks. Upper Saddle River: Prentice Hall, New Jersey, 1996.

[12] VANSTONE, S.; OORSCHOT, P.van; MENEZES A. *Handbook of Applied Cryptography.* CRC Press, 1996.

[13] Jain, R., *The Art of Computer Systemns Performance Analysis*, Jhon Wiley e Sons, 1991.

[14] Banks, J. Carson, J.S. e NELSON, B.L., *Disrete-Event System Simulation*, Prendice-Hall, Englewood Cliffs, NJ, Segunda Edição, 1996.

[15] Law, A.M. e KELTON, W.D., *Simulation Modeling and Analysis*, McGraw-Hill, NY, Segunda Edição, 1991.

[16] Dudewicz E. J. e KARIAN, Z. A., *Modern Design and Analysis of Discrete-Event Computer Simulations*, IEEE Computer Society Press, 1985.

[17] ZELENOVSKY, R.; MENDONÇA, *A. Eletrônica Digital, Curso Prático e Exercícios*. MZ Editora, 2002.

[18] WIKIPEDIA. Symetric Cryptography - wikipedia, the free enciclopedia, 2014. URL en.wikipedia.org/wiki/ Symetric Cryptography. Acessado em 15/06/14.

[19] WIKIPEDIA. Digital Signature - wikipedia, the free enciclopedia, 2014. URL en.wikipedia.org/wiki/ Digital Signature. Acessado em 15/06/14.

APÊNDICE

NOÇÕES DE LÓGICA MATEMÁTICA E ELETRÔNICA DIGITAL

Esta parte do trabalho foi escrita para que uma revisão nos conceitos de funções lógicas e introdução à eletrônica digital fossem feitos, com o objetivo de o leitor entender melhor os algoritmos presentes neste trabalho, e não ter que recorrer a buscas fora de escopo. Este resumo aborda apenas conceitos restritos de funções lógicas, bem como a conhecimentos introdutórios de eletrônica digital, configurando um conjunto de requisitos para o entendimento dos algoritmos citados ao longo do trabalho.

FUNÇÕES DE VARIÁVEIS BOOLEANAS

Essas funções serão estudadas tendo como entrada e dados de manipulação números binários.

- Função "e" – conjunção lógica ($\wedge$)

  Esta função possui duas entradas e sua saída será igual a 1 se as duas entradas forem iguais a 1.

A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

- Função "*ou*" − conjunção lógica ($\vee$)

  Esta função também possui duas entradas e sua saída será igual a 1 se pelo menos uma das entradas forem iguais a 1.

A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

- Função "*não*" − negação lógica ($\sim$)

  Esta função possui apenas uma entrada, sendo sua saída a negação da entrada, ou seja valores contrários, o que era 1 passa a ser 0 e o que era 0 passa a ser 1.

A	Saída
0	1
1	0

- Função "*xor*"− disjunção exclusiva ($\oplus$)

  Esta função possui duas entradas e tem origem na função ou, sendo que sua saída será igual a 1 se apenas uma das duas entradas for igual a 1. Se ambas forem 1, a saída será zero.

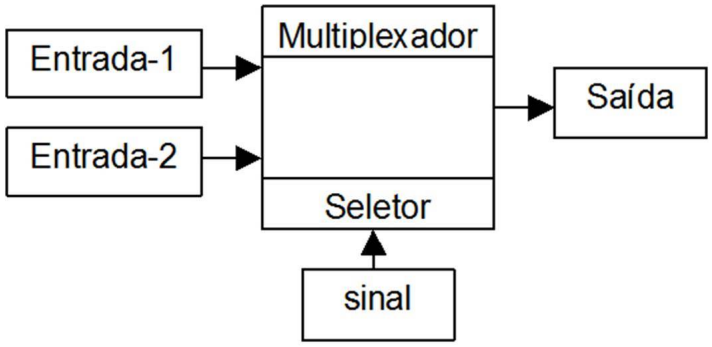
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

Esta última função é muito utilizada em criptografia, sendo que muitos algoritmos utilizam esta função em seus processos.

MULTIPLEXADORES

Um circuito multiplexador consiste basicamente de permitir, de acordo com um sinal, que determinada entrada seja reproduzida na saída.

No caso da figura abaixo, caso o sinal seja nível alto (valor atribuído em bits = 1), a saída reproduzida é a Entrada−1, caso o sinal seja 0, a saída é igual a Entrada−2.



Pela função desempenhada por este sinal, ele recebe o nome de sinal habilitador.

REGISTRADORES DE DESLOCAMENTO E SINAL DE RELÓGIO (CLOCK)

O registrador de deslocamento é um circuito capaz de armazenar e deslocar uma palavra (conjunto de bits) para a direita ou para a esquerda mediante um sinal de relógio.

Semelhante ao sinal habilitador do multiplexador o clock também funciona habilitando a saída do registrador, mas também é responsável pela entrada e pelo deslocamento deste registrador.

Por motivo computacional o clock é, em sua maioria, periódico. Muitas vezes se conjuga este sinal a outro, tornando o sinal habilitador não periódico.