

UNIVERSIDADE PAULISTA – UNIP

**GABRIEL DE ALMEIDA BATISTA
FELIPE RAMOS DA SILVA
FELIPE DA SILVA BORGES NEVES
FELIPE CORNIANI DE GENARO
LUIS HENRIQUE GIUSEPIN ALONSO**

**AS TÉCNICAS CRIPTOGRÁFICAS:
CONCEITOS, USOS E APLICAÇÕES**

**SÃO PAULO
2015**

**GABRIEL DE ALMEIDA BATISTA
FELIPE RAMOS DA SILVA
FELIPE DA SILVA BORGES NEVES
FELIPE CORNIANI DE GENARO
LUIS HENRIQUE GIUSEPIN ALONSO**

**TÉCNICAS CRIPTOGRÁFICAS:
CONCEITOS, USOS E APLICAÇÕES**

Atividade prática supervisionada e apresentada ao curso Ciência da Computação, para fins de conhecimento na área.

Orientador: Prof. Álvaro A. Colombero Prado.

**SÃO PAULO
2015**

DEDICATÓRIA

Dedicamos este trabalho primeiramente a Deus por ter nos dados vida até este presente momento e aos nossos pais que nos educou, com força e amor incondicionais.

AGRADECIMENTOS

Agradecemos em primeiro lugar a Deus por ser a base das nossas conquistas;

Aos nossos pais, por acreditar e terem interesse em nossas escolhas, apoiando-nos e esforçando-se junto a nós, para que supríssemos todas elas;

Ao professor Álvaro A. Colombero Prado, pela dedicação em suas orientações prestadas na elaboração deste trabalho, nos incentivando e colaborando no desenvolvimento de nossas ideias;

Ao nosso amigo Clederson Bispo da Cruz, por ter nos auxiliado, desde o princípio, na correção gramatical.

*“A lógica é o denominador comum das
ciências especiais”.*

(Willard Van Orman Quine)

RESUMO

A criptografia é quase tão antiga quanto à escrita, pois, desde que as pessoas começaram a escrever, passaram a sentir a necessidade de esconder ou de não permitir que qualquer um entendesse o que foi escrito. O objetivo da criptografia na transmissão de dados é codificar os dados a serem transmitidos de forma que fiquem ininteligíveis em caso de espionagens. O objetivo deste trabalho é avaliar de forma clara o conceito de criptografia, sua utilização e sua aplicabilidade, por meio de uma técnica em um software desenvolvido na linguagem C. O trabalho realizado teve caráter qualitativo. Fundamentado em obras publicadas por estudiosos especialistas, que já apontaram os conceitos, usos, aplicações, vulnerabilidades e falhas de certas técnicas criptográficas. A criptografia é um conceito totalmente necessário e importante nos meios de comunicação e informação que utilizamos hoje, já que há um crescente aumento no uso da internet e uma frequente troca de dados entre servidores e usuários que transmitem dados pessoais e necessitam de sigilo e segurança.

Palavras-chave: Criptografia; Software; Linguagem C.

ABSTRACT

Encryption is almost as old as writing, because as long as people began to write, began to feel the need to hide or not to allow anyone to understand what was written. The purpose of encryption in data transmission is to encode the data to be transmitted so that they are unintelligible in the event of espionage. The objective of this study is to evaluate clearly the concept of encryption, its use and its application through a technique in software developed in C. The work was qualitative. Based on works published by expert's scholars, who have pointed out the concepts, uses, applications, vulnerabilities and flaws of certain cryptographic techniques. Encryption is a totally necessary and important concept in the media and information we use today, as there is a steady increase in Internet use and frequent exchange of data between servers and users who transmit personal data and require confidentiality and security.

Keywords: Cryptography; Software; Language C.

LISTA DE ILUSTRAÇÕES

Imagem 01 – Criptografia Simétrica.	Pag. 37.
Imagem 02 – Criptografia Assimétrica.	Pag. 40.
Imagem 03 – Ron Rivest.	Pag. 43.
Imagem 04 – Modo Fluxos de Cifras.	Pag. 45.
Imagem 05 – Modelo de RC4.	Pag. 47.
Imagem 06 – Encapsulamento WEP.	Pag. 52.
Imagem 07 – Trecho Algoritmo <i>main</i> .	Pag. 60.
Imagem 08 – <i>Main</i> em C.	Pag. 61.
Imagem 09 – Função Menu.	Pag. 62.
Imagem 10 – Função KSA.	Pag. 62.
Imagem 11 – Função FRGA.	Pag. 63.
Imagem 12 – Função <i>ChiperText</i> .	Pag. 63.
Imagem 13 – Printscreen da Home.	Pag. 65.
Imagem 14 – Printscreen da Home.	Pag. 66.
Imagem 15 – Printscreen da Mensagem Simples.	Pag. 66.
Imagem 16 – Printscreen da Mensagem Simples.	Pag. 66.
Imagem 17 – Printscreen da Chave Simétrica.	Pag. 67.
Imagem 18 – Printscreen da Chave Simétrica.	Pag. 67.
Imagem 19 – Printscreen da Chave Simétrica.	Pag. 68.
Imagem 20 – Printscreen da Chave Simétrica.	Pag. 68.
Imagem 21 – Printscreen da Mensagem Criptografada.	Pag. 69.
Imagem 22 – Printscreen da Mensagem Criptografada.	Pag. 69.
Imagem 23 – Printscreen da Mensagem Criptografada.	Pag. 70.
Imagem 24 – Printscreen da Mensagem Criptografada.	Pag. 70.
Imagem 25 – Mensagem Descriptografada.	Pag. 71.
Imagem 26 – Mensagem Descriptografada.	Pag. 71.
Imagem 27 – Gabriel de Almeida Batista.	Pag. 81.
Imagem 28 – Felipe Ramos da Silva.	Pag. 82.
Imagem 29 – Felipe da Silva Borges Neves.	Pag. 83.
Imagem 30 – Felipe Corniani de Genaro.	Pag. 84.
Imagem 31 – Luís Henrique Giusepin Alonso.	Pag. 85.

LISTA DE TABELAS

Tabela 01 – Semântica dos Símbolos.	Pag. 33.
Tabela 02 – Tabela-verdade XOR.	Pag. 48.
Tabela 03 – Tabela-verdade bi-implicação.	Pag. 48.

LISTA DE ABREVIATURA E SIGLAS

XOR – OU Exclusivo.

RSA – Rivest, Shamir e Adleman.

K – Chave Secreta.

PR¹ – Chave Privada do usuário 1.

PU¹ – Chave Pública do usuário 1.

D – Descriptografia.

E – Criptografia.

Y – Texto Cifrado.

X – Texto Simples.

DES – *Data Encryption Standard*.

IBM – *International Business Machines*.

3DES – *Data Encryption Standard* Triplo.

NIST – *National Institute of Standards and Technology*.

MIT – Instituto de Tecnologia de Massachusetts.

ECB – *Electronic Code Book*.

EBC – Encadeamento de Blocos de Cifras.

OFB – *Output Feedback*.

FC – Fluxo de Cifras.

MC – Modo Contador.

RC2 – *Rivest Ciphers* 2.

RC4 – *Rivest Ciphers* 4.

RC5 – *Rivest Ciphers* 5.

RC6 – *Rivest Ciphers* 6.

ISEP – Instituto Superior de Engenharia do Porto.

KSA – *Key-scheduling algorithm*.

SSL / TLS – *Secure Sockets Layer / Transport Layer Security*.

WEP – *Wired Equivalent Privacy*.

WPA – *Wi-Fi Protected Access*.

CRC-32 – *Cyclic Redundancy Checks*.

CC – Cifra Caótica.

SUMÁRIO

1	INTRODUÇÃO.....	21
2	OBJETIVOS	25
2.1	Objetivos Gerais	25
2.2	Objetivos Específicos	25
3	CRIPTOGRAFIA.....	27
3.1	Dois princípios fundamentais.....	29
3.1.1	Redundância.....	29
3.1.2	Atualidade.....	29
3.2	Criptoanálise.....	29
3.2.1	Ataque do texto cifrado conhecido.....	30
3.2.2	Ataque do texto em claro conhecido.....	30
3.2.3	Ataque do texto em claro escolhido.....	30
3.2.4	Ataque do texto em claro escolhido com adaptação	31
3.2.5	Ataque do texto cifrado escolhido.....	31
3.2.6	Ataque do texto escolhido	31
3.2.7	Ataque da chave escolhida.....	31
3.2.8	Rubber-hose cryptanalysis	31
3.2.9	Ataque da compra da chave.....	31
3.2.10	Ataque de força bruta.....	31
3.3	Esteganografia.....	32
4	TÉCNICAS CRIPTOGRÁFICAS MAIS UTILIZADAS	33
4.1	Cifras de Substituição	33
4.2	Cifras de Transposição	34
4.3	Modos de Cifra.....	35
4.3.1	O modo Electronic Code Book.....	35
4.3.2	Modo de Encadeamento de Blocos de Cifras.....	35
4.3.3	Modo de Feedback de Cifra	35
4.3.4	Modo Fluxo de Cifras.....	36
4.3.5	Modo Contador.....	36
4.4	Chave Simétrica.....	36
4.4.1	Data Encryption Standard.....	38
4.4.2	Data Encryption Standard Triplo.....	38
4.4.3	Advanced Encryption Standard	39
4.5	Chave Assimétrica.....	39
4.5.1	Rivest Shamir Adleman	41
4.5.2	Algoritmo da Mochila	42
5	RIVEST CIPHERS.....	43
5.1	Estrutura, conceito e fundamentação.....	45
5.1.1	Tabela-Verdade	48
5.2	Benefícios em relação às técnicas anteriores	49
5.2.1	Adaptabilidade	49
5.2.2	Chave	49
5.2.3	Memória.....	49
5.2.4	Simplicidade	50
5.2.5	Modo de Cifra	50

5.3	Aplicações que fazem ou fizeram uso do RC4.....	51
5.4	Discussão Comparativa	52
5.4.1	RC5 X RC4.....	52
5.4.2	DES X RC4.....	53
5.4.3	IDEA X RC4.....	53
5.4.4	AES X RC4.....	53
5.4.5	RSA X RC4.....	54
5.4.6	Sosemanuk X RC4.....	54
5.4.7	Serpent X RC4.....	55
5.4.8	One-Time-Pad X RC4.....	55
5.4.9	Twofish X RC4.....	55
5.5	Vulnerabilidade e Falhas.....	56
5.6	Melhorias propostas ou implementadas	56
5.6.1	Melhorias Implementadas.....	57
5.6.2	Melhorias Propostas.....	57
6	PROJETO.....	59
6.1	Main.c	59
6.2	Rciv.h.....	61
7	APRESENTAÇÃO.....	65
7.1	Printscreen da Home.....	65
7.2	Printscreen da Mensagem Simples.....	66
7.3	Printscreen da Chave Simétrica.....	67
7.4	Printscreen da Mensagem Criptografada.....	69
7.5	Printscreen da Mensagem Descriptografada.....	71
8	CONCLUSÃO.....	73
	REFERÊNCIA BIBLIOGRÁFICA	77
9	FICHAS DE ATIVIDADES PRÁTICAS SUPERVISIONADAS	79
	ANEXOS.....	85

1 INTRODUÇÃO

Tanenbaum e Wetherall (2011, p. 481) explicam que a palavra criptografia vem de origem grega que significam ‘escrita secreta’.

Carissimi, Rochol e Granville (2009, p. 346) definem criptografia como um conceito fundamental empregado em segurança de redes de computadores para prover confidencialidade, autenticidade e integridade das informações. Na realidade, a criptografia por si só é uma ciência bastante complexa, fundamenta-se em conhecimentos de matemática, e que tem sido utilizada pela humanidade há muitos séculos.

Sousa (2002, p. 203) evidencia o objetivo da criptografia na transmissão de dados. Codificar os dados a serem transmitidos de forma que fiquem ininteligíveis em eventuais espionagens (ou “grampos”) que possam ocorrer ao longo do percurso na rede, evitando assim que a informação seja capturada e entendida por estranhos.

Desde os primórdios as redes de computadores, nunca tiveram tantos usuários ativos como hoje. Usuários que fazem transações bancárias, compras e devolução de impostos, são alvos principais de pessoas mal-intencionadas. Nessas condições, surge um ponto fraco atrás de outro, e a segurança tornou-se um problema de grandes proporções.

A segurança é um assunto abrangente e inclui inúmeros tipos de problemas. Em sua forma mais simples, preocupa-se em impedir que pessoas mal-intencionadas leiam ou modifiquem secretamente mensagens enviadas a outros destinatários. Outra preocupação da segurança são as pessoas que tentam ter acesso a serviços remotos que não estão autorizadas a usar. Ela também lida com meios para saber se uma mensagem supostamente verdadeira é um trote. A segurança trata de situações em que mensagens legítimas são capturadas e reproduzidas, além de lidar com pessoas que tentam negar o fato de ter enviados certas mensagens.

A partir destas considerações, visa-se responder a seguinte pergunta: Quais técnicas criptográficas são aconselháveis?

Partindo-se da hipótese, que assimilando o tempo e o conteúdo de estudo com as informações obtidas através de pesquisas é possível, de forma estratégica, concretizar as técnicas criptográficas mais conhecidas, buscar falhas e melhorias, portanto, são relevantes as práticas em si.

O presente trabalho busca contribuir para a sociedade criptográfica mostrando solução para os problemas apresentados, por meio de uma técnica de criptografia.

Com pesquisas, avaliações e benefícios, intenciona evidenciar falhas e vulnerabilidades que as técnicas apresentam, contribuindo na implementação de melhorias em uma técnica específica e destacando suas vantagens em relação às outras técnicas que serão apresentadas, resultando tudo isso em um *software* de criptografia desenvolvido na linguagem C.

Peixinho, Fonseca e Lima (2013, p. 142) promovem a importância das técnicas criptográficas que surgiram pela necessidade de esconder segredos de pessoas específicas. Os antigos generais precisavam transmitir informações para seus exércitos sem o perigo de ter suas mensagens interceptadas e traduzidas pelo inimigo.

Stallings (2007, p. 15) afirma que a ferramenta automatizada mais importante para a segurança de redes e das comunicações é a criptografia.

A necessidade de proteger os canais de comunicação entre pessoas de uma mesma comunidade vem desde os primórdios da civilização. Essa preocupação vai muito além da ideia prioritária de só visar proteção aos meios de comunicação, mas também, resguardo do próprio conteúdo da mensagem. Este é um dos motivos que levaram à criação das técnicas criptográficas.

O trabalho intenciona esclarecer conceitos, funções e aplicações das técnicas criptográficas aos leigos. Para tanto, apresentando as técnicas mais usadas e conhecidas e selecionando uma delas, resultando na criação de um *software*, na linguagem C, que servirá como demonstração.

Fundamentado em obras publicadas por estudiosos especialistas, como Sousa (2002), Stallings (2005, 2007), Carissimi, Rochol e Granville (2009), Tenenbaum e Wetherall (2011), Peixinho, Fonseca e Lima (2013), que já apontaram os conceitos, usos, aplicações, vulnerabilidades e falhas de certas técnicas criptográficas, a presente pesquisa busca unificar e transmitir os principais conceitos de suas obras. A junção de doze obras, de ano e autores diferentes, resulta neste trabalho.

No primeiro capítulo, reunimos conceitos gerais de criptografia. Nesta etapa consideramos as definições de onze autores dos que foram pesquisados, a saber, Caruso e Steffen (1999), Sousa (2002), Tâmega (2003), Carvalho (2005), Stallings

(2005, 2007), Nakamura e Geus (2007), Carissimi, Rochol e Granville (2009), Tenenbaum e Wetherall (2011), White (2012), Peixinho, Fonseca e Lima (2013).

No segundo capítulo, são apresentadas as técnicas mais utilizadas e conhecidas, com base nas obras consultadas.

O terceiro capítulo será totalmente focado em uma técnica de criptografia, apontando sua estrutura, fundamentos, benefícios em relação às técnicas citadas anteriormente, aplicações que fizeram uso, vulnerabilidade, falhas e melhorias propostas ou implementadas. O livro dos autores Abe, Scalzitti e Filho (2002), foi fundamental e fortemente usado para a compreensão da lógica por trás da técnica criptográfica trabalhada.

O quarto capítulo descreverá a estrutura, instruções e uso do programa desenvolvido na linguagem C.

No quinto capítulo, será apresentando o relatório do programa, juntamente com o seu código fonte.

O sexto capítulo apresentará o programa em funcionamento em um computador, apresentando todas as funcionalidades extras.

2 OBJETIVOS

Pesquisar e avaliar de forma clara o conceito de criptografia, sua utilização e suas aplicações. Para que um leigo possa entender e se familiarizar com o tema.

Desenvolver um *software* na linguagem C.

2.1 Objetivos Gerais

Contribuir para a sociedade criptográfica com solução para os problemas apresentados por uma técnica de criptografia. Utilizando um *software* desenvolvido na linguagem C.

2.2 Objetivos Específicos

Visando atingir o objetivo principal, alguns objetivos específicos são requeridos, entre eles:

- Pesquisar e entender o conceito, utilização e aplicação das técnicas criptográficas.
- Avaliar as principais técnicas usadas e selecionar uma para a elaboração da dissertação.
- Buscar os benefícios que a técnica selecionada proporciona.
- Evidenciar as falhas e as vulnerabilidades da técnica trabalhada.
- Divulgar aplicações que fazem ou fizeram uso da técnica.
- Implementar melhorias na técnica.
- Desenvolver um *software* em cima da técnica selecionada.
- Realizar uma discussão dos resultados oriundos do processo de descoberta de conhecimento.
- Apresentar o funcionamento do *software* e suas instruções.

3 CRIPTOGRAFIA

Tâmega (2003, p. 31) reconhece que a criptografia se origina do grego: *kryptos*, ou seja, secreto, e *grapho*, escrita. A criptografia nada mais é do que escrever secretamente, comunicar-se de maneira que nenhuma outra pessoa, a não ser o remetente e o destinatário, possa decifrar a mensagem.

Sousa (2002, p. 203) evidencia o objetivo da criptografia na transmissão de dados. Codificar os dados a serem transmitidos de forma que fiquem ininteligíveis em eventuais espionagens.

A criptografia é quase tão antiga quanto à escrita, pois, desde que as pessoas começaram a escrever, passaram a sentir a necessidade de esconder ou de não permitir que qualquer um entendesse o que foi escrito.

Tanenbaum e Wetherall (2011, p. 481) levantam que quatro grupos de pessoas utilizaram e contribuíram para a criptografia: os amantes, os diplomatas, os militares e as pessoas que gostam de guardar memórias. Entre eles os militares tiveram o papel mais importante e definiram as bases para a tecnologia.

Nas organizações militares as mensagens eram criptografadas e entregues habitualmente a auxiliares mal remunerados, que se encarregavam de criptografá-la e transmiti-las. Uma das restrições era criptografar e transmitir a mensagem no campo de batalha e com pouco equipamento. Outra restrição era a dificuldade de alternar os métodos criptográficos rapidamente, pois isso exigia a repetição do treinamento de um grande número de pessoas. Com o advento dos computadores isso tudo mudou e impulsionou a criptografia.

Os profissionais diferenciam cifras e códigos. Tanenbaum e Wetherall (2011, p. 481) descrevem cifra como uma transformação de caractere por caractere ou de bit por bit, sem levar em conta a estrutura linguística da mensagem e código substitui uma palavra por outra palavra ou símbolo. Os códigos não são mais utilizados, embora tenham uma história gloriosa.

A cifra opera na sintaxe (símbolos) da mensagem, enquanto um código geralmente opera na semântica (significado). Um código é armazenado como um mapeamento num livro de códigos, enquanto cifras transformam símbolos individuais conforme um algoritmo.

Tanenbaum e Wetherall (2011, p. 481) relatam que o código mais bem-sucedido foi inventado e usado pelas forças armadas dos Estados Unidos durante a

Segunda Guerra Mundial no Pacífico. Ela simplesmente tinha índios navajos como transmissor e receptor, que usavam palavras em Navajo específicas para termos militares, como *chay-dagahi-nail-tsaidi* (literalmente, matador de cágado) para indicar uma arma antitanque. A linguagem navajo é altamente tonal, extremamente complexa, e não tem nenhuma forma escrita. Além disso, nem uma única pessoa no Japão conhecia alguma coisa sobre ela.

Peixinho, Fonseca e Lima (2013, p. 124) explicam que basicamente, um processo criptográfico envolve a aplicação de três conceitos elementares: a mensagem / texto, o algoritmo e a chave. A mensagem consiste, pura e simplesmente, na informação que se deseja transmitir de forma segura; o algoritmo é a forma que será utilizada para cifrar e decifrar uma mensagem; e a chave, em alguns modelos computacionais, pode ser entendida como o segredo compartilhado que deve ser conhecido apenas pelas duas partes envolvidas no processo de comunicação. A garantia da confidencialidade está em esconder do atacante o algoritmo ou a chave utilizada.

O algoritmo criptográfico define a forma como a mensagem será cifrada e decifrada. A definição prévia do algoritmo pelas partes envolvidas (transmissor e receptor) é um dos fatores fundamentais no processo de comunicação seguro. (PEIXINHO, FONSECA E LIMA, 2013, p. 124).

Stallings (2007, p. 18) defini que uma mensagem original é conhecida como texto claro ou *plaintext*, enquanto uma mensagem codificada é chamada de texto cifrado ou *ciphertext*. Descreve também, que um processo de converter texto claro em texto cifrado é conhecido como cifragem ou criptografia; restaurar o texto claro a partir do texto cifrado é a decifragem ou descriptografia.

Os muitos esquemas utilizados para a criptografia constituem a área de estudo conhecida como criptologia. Esse esquema é conhecido como sistema criptográfico ou cifra. As técnicas empregadas para decifrar uma mensagem sem qualquer conhecimento dos detalhes de criptografia estão na área da criptoanálise. [...]. As áreas da criptografia e criptoanálise juntas são chamadas de criptologia. (STALLINGS, 2007, p. 18).

Carissimi, Rochol e Granville (2009, p. 346) fornecem que os algoritmos de cifragem e de decifração são de conhecimento público. A garantia da confidencialidade da informação reside no conhecimento da chave de cifragem e decifração. É justamente em função da chave de cifragem que os métodos de criptografia são classificados em duas categorias: chave simétrica e chave assimétrica.

3.1 Dois princípios fundamentais

Tanenbaum e Wetherall (2011, p. 487) descrevem os dois princípios fundamentais da criptografia e alertam que podemos correr o risco de violá-los.

Vale ressaltar que existem dois tipos de intrusos, ativo e passivo. Intruso ativo, aquele que é capaz de interferir o protocolo, atrasando, eliminando ou alterando a mensagem. Intruso passivo, aquele que não interfere o protocolo, mas tem acesso às mensagens trocadas.

3.1.1 Redundância

O primeiro princípio é que todas as mensagens criptografadas devem conter alguma redundância, ou seja, informações que não são necessárias para a compreensão da mensagem.

3.1.2 Atualidade

O segundo princípio criptográfico é tomar algumas medidas para assegurar que cada mensagem recebida possa ser confirmada como uma mensagem atual, isto é, enviada muito recentemente. Essa medida é necessária para impedir que intrusos ativos reutilizem mensagens antigas.

3.2 Criptoanálise

Tâmega (2003, p. 36) fornece a explicação de criptoanálise e diz que a criptoanálise consiste em estratégias matemáticas com o objetivo de quebrar

algoritmos de criptografia. Como não há um meio matemático de se provar que esse algoritmo é ou não seguro, então, ele é seguro até que se prove o contrário.

Tâmega (2003) ainda descreve a maior dificuldade que um criptoanalista encontra ao tentar quebrar um algoritmo, está no número de chaves que este utiliza, porque quanto maior o número de chaves, maior a dificuldade de se conseguir obter sucesso na criptoanálise de um determinado sistema.

Caso não seja possível quebrar um sistema ele fica conhecido como sistema de segurança perfeita.

Nakamura e Geus (2007, p. 313) citam alguns ataques baseados na criptoanálise e ainda a definem como uma ciência de recuperar uma informação cifrada sem o acesso direto à chave de criptografia, de forma que ela pode recuperar a mensagem original ou a chave de criptografia.

3.2.1 Ataque do texto cifrado conhecido

O atacante possui diversas mensagens, todas cifradas com o mesmo algoritmo criptográfico. O objetivo é recuperar a mensagem original ou deduzir a chave, que pode ser usada para decifrar as mensagens.

3.2.2 Ataque do texto em claro conhecido

O atacante possui o conhecimento das mensagens cifradas e também do seu equivalente em claro. O objetivo é deduzir as chaves utilizadas ou um algoritmo para decifrar qualquer mensagem cifrada com a mesma chave.

3.2.3 Ataque do texto em claro escolhido

O atacante escolhe um texto em claro e faz a análise de acordo com o texto cifrado obtido. O objetivo é deduzir as chaves ou um algoritmo para decifrar as mensagens.

3.2.4 Ataque do texto em claro escolhido com adaptação

Esse é um caso especial do ataque de texto em claro escolhido, no qual o atacante escolhe e modifica o texto em claro escolhido de acordo com os resultados que vem obtendo. O objetivo é deduzir as chaves ou um algoritmo para decifrar as mensagens.

3.2.5 Ataque do texto cifrado escolhido

O atacante escolhe diferentes textos cifrados para serem decifrados e tem o acesso aos textos decifrados. Esse ataque é usado mais contra algoritmos de chaves pública. Com isso, ele pode deduzir a chave utilizada.

3.2.6 Ataque do texto escolhido

Composição dos ataques de texto em claro escolhido e de texto cifrado escolhido.

3.2.7 Ataque da chave escolhida

O atacante usa o conhecimento sobre relações entre diferentes chaves.

3.2.8 Rubber-hose cryptanalysis

Ataques baseados em ameaça, chantagem ou tortura, para que o usuário entregue a chave criptográfica.

3.2.9 Ataque da compra da chave

Ataque baseado em suborno.

3.2.10 Ataque de força bruta

Ataque em que todas as combinações de chaves possíveis são testadas.

3.3 Esteganografia

A origem de esteganografia remonta a Grécia antiga e significa “escrita coberta”. Sendo um dos métodos estudados pela criptologia, que basicamente funciona como uma maneira de camuflar a mensagem a ser passada, e não a torna totalmente ininteligível como a criptografia costuma fazer.

Tâmega (2003, p. 25) explica que esteganografia esconde as mensagens por artifícios, como, imagens ou um texto que tenha sentido, mas que sirva apenas para disfarçar o real conteúdo. Funciona da mesma forma que o *barn code*: mesclar a mensagem numa outra, em que apenas determinadas palavras devem ser lidas para descobrir o texto camuflado. Ao contrário da criptografia, que procura esconder a informação da mensagem, a esteganografia procura esconder a mensagem.

4 TÉCNICAS CRIPTOGRÁFICAS MAIS UTILIZADAS

Como citado anteriormente, é justamente em função da chave de cifragem que os métodos de criptografia são classificados em duas categorias: chave simétrica e chave assimétrica.

Para que o leitor possa entender de forma clara o conceito apresentado neste capítulo é aconselhável que o mesmo se familiarize com a semântica e expressão dos seguintes símbolos:

Tabela 01 – Semântica dos Símbolos.

Símbolo	Significado
K	Chave Secreta
PR ¹	Chave Privada do usuário 1
PU ¹	Chave Pública do usuário 1
D	Descriptografia
E	Criptografia
Y	Texto Cifrado
X	Texto Simples

Fonte: Própria, 2015.

4.1 Cifras de Substituição

Tanenbaum e Wetherall (2011, p. 483) explicam que em uma cifra de substituição, cada letra ou grupo de letras é substituído por outra letra ou grupo de letras, de modo a criar um “disfarce”, mas mantendo a ordem dos símbolos.

Uma das cifras mais antigas é a cifra de César, atribuída a Júlio César.

Tanenbaum e Wetherall (2011) explicam que nesse método *a* se torna D, *b* se torna E, *e* se torna F e *z* se torna C. Por exemplo, bola passaria a ser EROD.

Logo abaixo segue o alfabeto simples em letra minúscula e o alfabeto cifrado em letra.

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

Este sistema geral é chamado de cifra de substituição monoalfabética, sendo a chave a *string* de 26 letras corresponde ao alfabeto completo.

Vale ressaltar que não é obrigatório pular 4 letras, o valor do salto pode ser maior ou menor e o valor do mesmo será a chave secreta (K).

4.2 Cifras de Transposição

Tanenbaum e Wetherall (2011, p. 483) explicam que ao contrário das cifras de substituição que preservam a ordem dos símbolos no texto simples, mas os disfarçam. A cifra de transposição reordena as letras, mas não as disfarçam.

O exemplo mostra uma cifra de transposição muito comum, a de colunas. Tanenbaum e Wetherall (2011) explicam que a cifra se baseia em uma chave que é uma palavra ou frase que não contém letras repetidas e que o objetivo da chave é numerar as colunas de modo que a coluna 1 fique abaixo da letra da chave mais próxima do início do alfabeto e assim por diante. O X é escrito horizontalmente, em linhas. O Y é lido em colunas, a partir da coluna cuja letra da chave seja mais baixa. Nesse exemplo a chave é GABRIEL.

G	A	B	R	I	E	L
<u>4</u>	<u>1</u>	<u>2</u>	<u>7</u>	<u>5</u>	<u>3</u>	<u>6</u>
f	e	l	i	z	a	q
u	e	l	e	q	u	e
t	r	a	n	s	f	e
r	e	o	q	u	e	s
a	b	e	e	a	p	r
e	n	d	e	o	q	u
e	e	n	s	i	n	a

No exemplo acima o texto simples é felizaquelequetransfereoquesabeeaprendeoqueensina e o texto cifrado é EEREBNELLAOEDNAUFEPQNFUTRAEEZQSUAOIQEESRUAIENQEES.

Tanenbaum e Wetherall (2011, 484) explicam que para quebrar uma cifra de transposição, o criptoanalista deve primeiro estar ciente de lidar com tal cifra,

segundo ele deve fazer uma estimativa do número de colunas e por último ordenar as colunas.

4.3 Modos de Cifra

Tanenbaum e Wetherall (2011) citam alguns modos de cifra usados em chave simétrica e assimétrica.

4.3.1 O modo *Electronic Code Book*

O modo ECB (*Electronic Code Book*), funciona em dividir o X em blocos consecutivos de 8 bytes (64 bits) e codifica-los uns após outros com a mesma chave. O último fragmento de texto simples é completado até 64 bits, se necessário.

4.3.2 Modo de Encadeamento de Blocos de Cifras

O modo EBC (Encadeamento de Blocos de Cifras) encadeia cada bloco de cifras e realiza uma operação XOR com o bloco de texto cifrado anteriormente, antes de ser codificado. Como consequência, o mesmo bloco de texto simples não é mais mapeado para o mesmo bloco de texto cifrado, e a criptografia não é mais uma grande cifra de substituição monoalfabética.

4.3.3 Modo de Feedback de Cifra

O modo Output Feedback (OFB) transforma uma cifra de bloco num gerador de números pseudoaleatórios. O texto cifrado realimenta a cifra de bloco e este processo é repetido para produzir um fluxo de bits pseudorrandômicos. O fluxo de bits é totalmente determinado pelo algoritmo, pela chave, por um vetor de inicialização e pelo número de bits que realimentam a cifra em cada etapa. O fluxo de bits pode então servir para fazer uma operação XOR com o texto claro a fim de produzir o texto cifrado, transformando efetivamente a cifra de bloco numa cifra de fluxo.

4.3.4 *Modo Fluxo de Cifras*

O modo FC (Fluxo de Cifras) funciona pela codificação de um vetor de referência com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para obter um segundo bloco de saída. Em seguida, esse bloco é codificado para obter um terceiro bloco e assim por diante. A sequência de bloco de saída é tratada como uma chave única e submetida a uma operação XOR com X para obter Y.

4.3.5 *Modo Contador*

O MC (Modo Contador) não codifica o X diretamente. Em vez disso, o vetor de referência somado a uma constante é codificado, e o Y resultante é submetido a um XOR com o X. Aumentar o vetor de referência em uma unidade a cada novo bloco facilita a decodificação de um bloco em qualquer lugar no arquivo sem que primeiro seja preciso decodificar todos os seus predecessores.

4.4 **Chave Simétrica**

Stallings (2007, p. 17) explica que a chave simétrica ou criptografia simétrica é uma forma de criptossistema em que a criptografia e a descryptografia são realizadas usando a mesma chave. Ela é conhecida também como criptografia convencional.

A criptografia simétrica transforma o texto claro em texto cifrado, usando uma chave secreta e um algoritmo de criptografia. Usando a mesma chave e um algoritmo de descryptografia, o texto claro é recuperado a partir do texto cifrado. Ela usa cifra de substituição ou de transposição.

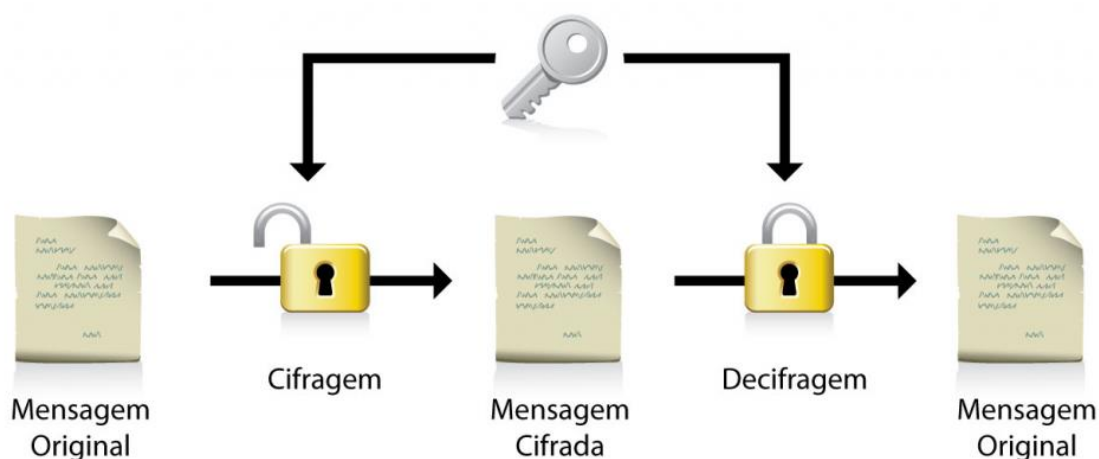
A vantagem da criptografia de chave simétrica é sua simplicidade, o que reduz sua complexidade computacional para as operações de cifragem e decifração, embora ela apresente duas desvantagens importantes. A primeira é que o remetente e o destinatário devem conhecer a chave secreta, o que implica um problema: como divulgar o valor da chave entre os interessados sem comprometer o seu sigilo. [...]. A segunda desvantagem é praticamente uma consequência da primeira: a maneira de

gerenciar e compartilhar as chaves para cada par de usuários garantindo seu segredo dos demais, considerando-se ainda, potencialmente, uma quantidade importante de chaves. [...]. (CARISSIMI, ROCHOL e GRANVILLE, 2009, p. 391).

Peixinho, Fonseca e Lima (2013, p. 124) também citam as vantagens da criptografia simétrica: velocidade e algoritmos rápidos, facilidade de implementação em hardware e chaves pequenas e simples geram cifradores robustos. Juntamente com as suas desvantagens: dificuldade do gerenciamento de chaves, não permite a autenticação e não permite o não repúdio do remetente.

A imagem 01 ilustra o funcionamento de uma criptografia simétrica.

Imagem 01 – Criptografia Simétrica.



Fonte: <http://biblioo.info/37ertificação-digital/>, 2015.

A criptografia simétrica também chamada segundo Stallings (2015, p. 381) de chave única apresentada na imagem 01 é dívida em dois passos. No primeiro passo o remetente pega a chave única, geralmente secreta, criptografa e envia a mensagem cifrada para o receptor. No segundo passo o receptor pega a mensagem cifrada e com a mesma chave usada para criptografar a mensagem, ele é capaz de decifrar a mesma.

4.4.1 *Data Encryption Standard*

Data Encryption Standard (DES) desenvolvida pela IBM foi adotada em 1977 pelo governo dos Estados Unidos como seu padrão oficial para as informações não confidências. Foi amplamente adotada pelo setor de informática para uso em produtos de segurança.

Tanenbaum e Wetherall (2011, p. 489) dizem que “Em sua forma original, ela já não é mais segura; no entanto, em uma forma modificada ela ainda é útil”.

Stallings (2005, p. 383) diz que o motivo do DES na sua forma original não ser mais seguro é a sua chave de 56 bits, porque foi apenas uma questão de tempo até que a velocidade de processamento dos computadores o tornasse obsoleto.

Tanenbaum e Wetherall (2011) explicam que o DES criptografa um X em blocos de 64 bits, produzindo 64 bits de Y. O algoritmo, parametrizado por uma chave de 56 bits, tem 19 estágios. O primeiro deles é transposição independente da chave no texto simples de 64 bits. O último estágio é exatamente o inverso dessa transposição. O penúltimo estágio troca os 32 bits mais à esquerda pelos 32 bits mais à direita. Os 16 estágios restantes são funcionalmente idênticos, mas são parametrizados por diferentes funções de chave. O algoritmo foi projetado para que a mesma chave de decodificação fosse a mesma chave de codificação, uma característica necessária em qualquer algoritmo de chave simétrica.

4.4.2 *Data Encryption Standard Triplo*

No início de 1979, a IBM percebeu que o tamanho da chave do DES era muito pequeno e criou uma forma de aumentá-lo usando a criptografia tripla. Com isso tornou-se o padrão internacional 8732. Com isso foi criado o *Data Encryption Standard Triplo* (3DES).

Segundo Stallings (2005, p. 383). O 3DES envolvia a repetição básica do DES três vezes, mediante duas ou três chaves únicas, para um tamanho de chave de 112 ou 168 bits.

Tanenbaum e Wetherall (2011, p. 491) explicam que no primeiro estágio o X é criptografado com K^1 da maneira usual do DES. No segundo estágio, o DES é executado no modo de descriptografia, com o uso de K^2 como chave. Por fim, outra criptografia DES é feita com K^1 .

Stallings (2005) explica que as principais desvantagens do 3DES é que o software do algoritmo é relativamente lento e que tanto o DES quanto o 3DES usam um tamanho de bloco de 64 bits. Por questão de eficiência e segurança, um tamanho de bloco maior é desejável.

4.4.3 *Advanced Encryption Standard*

As vidas úteis dos 3DES e do DES começaram a se aproximar do fim. Em 1997 NIST (*National Institute of Standards and Technology*), convidou pesquisadores do mundo inteiro para proporem um novo padrão criptográfico para uso não confidencial que seria chamado de AES (*Advanced Encryption Standard*).

Em 2001, o *Rijndael* se tornou o padrão AES do governo dos Estados Unidos. *Rijndael* é a junção dos sobrenomes de seus criadores: Rijmen + Daemen.

Tanenbaum e Wetherall (2011) explicam que o *Rijndael* admite tamanhos de chaves e tamanhos de blocos desde 128 bits até 256 bits em intervalos de 32 bits. O comprimento da chave e do bloco pode ser escolhido independentemente. Porém o AES especifica que o tamanho do bloco de cifra pode ser 128 bits e o comprimento da chave deve ser 128, 192 ou 256.

Tanenbaum e Wetherall (2011, p. 492) dizem. “DES, o *Rijndael* utiliza substituição e permutações, e também emprega várias rodadas”.

4.5 Chave Assimétrica

Carissimi, Rochol e Granville (2009, p. 348) descrevem que as principais características da chave assimétrica é empregar para a cifragem e a decifração duas chaves distintas: uma chave pública e uma chave privada (secreta).

Chave assimétrica é também, segundo Peixinho, Fonseca e Lima (2013, p. 125), conhecida como criptografia de chave pública.

O algoritmo de chave pública normalmente empregado é o de 1977, definido pelos pesquisadores Ron Rivest, Adi Shamir e Leonard Adleman, mais conhecido como RSA.

Essa característica resulta em duas vantagens em relação à criptografia simétrica. A primeira vantagem é a redução da quantidade de chaves

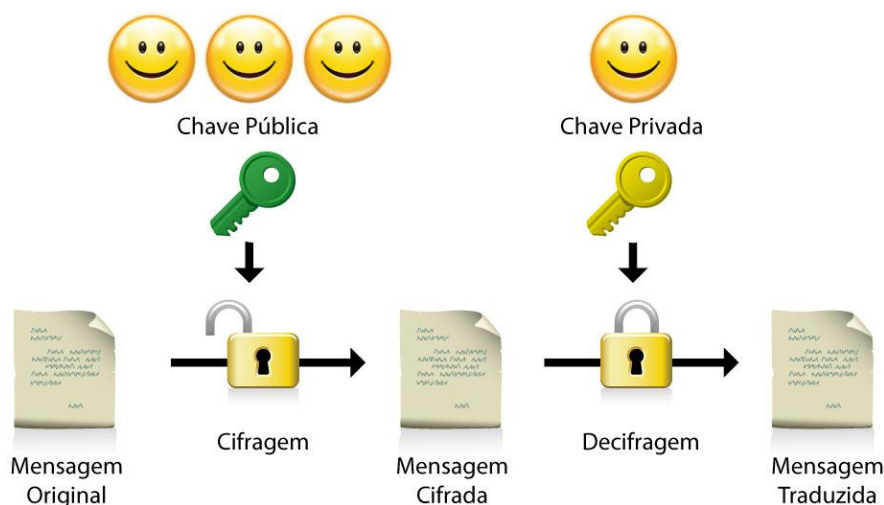
necessárias para n partes comunicarem entre si, já que cada parte utiliza seu par de chaves independentemente de quem é seu correspondente. A segunda vantagem é em relação à divulgação e o conhecimento das chaves entre as partes envolvidas. Por definição, a chave privada não é divulgada nem conhecida pelos demais; já a chave pública, justamente por se pública, é livremente distribuída. [...] (CARISSIMI, ROCHOL e GRANVILLE, 2009, p. 349).

Além das vantagens Carissimi, Rochol e Granville (2009, p. 349) citam duas desvantagens. A primeira é que a chave pública e a chave privada são matematicamente relacionadas de forma a prover a característica de que o que é cifrado por uma é decifrado apenas pela outra. A segunda é garantir que uma chave pública realmente pertence ao correspondente com qual se deseja comunicar. Essa preocupação é oriunda do fato que alguém mal-intencionado poderia se passar por um dos pares de uma comunicação.

Peixinho, Fonseca e Lima (2013, p. 125) também citam as vantagens da criptografia simétrica: gerenciamento de chaves, implantação de não repúdio do remetente e pode ser utilizada para garantir a confidencialidade, a autenticidade ou ambos. Juntamente com a sua desvantagem: desempenho, pois é muito mais lenta que a criptografia simétrica.

A imagem 02 ilustra o funcionamento de uma criptografia simétrica.

Imagem 02 – Criptografia Assimétrica.



Como ilustrado na imagem 02, é necessário que exista duas chaves, uma pública e uma privada. A chave pública ela é distribuída para os indivíduos que o dono da mesma deseja trocar mensagens. A privada ela não é distribuída para ninguém, porque apenas o dono das duas chaves pode saber as duas, já que uma é necessariamente obrigatória para o entendimento da mensagem cifrada. O remetente que possuir a chave pública do receptor poderá escrever uma mensagem simples e usando a chave pública cifrar a mesma e envia. O receptor deverá ser o único que possui a chave privada e usá-la-á para decifrar a mensagem.

4.5.1 *Rivest Shamir Adleman*

RSA (Rivest Shamir Adleman) foi um dos primeiros algoritmos chave pública desenvolvido em 1977 por Rin Rivest, Adi Shamir e Len Adleman da MIT (Instituto de Tecnologia de Massachusetts).

Segundo Tenenbaum e Wetherall (2011, p. 498) o RSA sobreviveu a todas as tentativas de rompimento por mais de um quarto de século. Grande parte da segurança prática se baseia nele.

A principal desvantagem, relatada por Tenenbaum e Wetherall (2011) é a exigência de chaves de pelo menos 1.024 bits para manter um bom nível de segurança (em comparação com 128 bits para os algoritmos de chave simétrica), e isso o torna bastante lento.

A Criptografia de Chave pública é mais segura contra análise criptográfica que a criptografia simétrica. [...]. Na realidade, a segurança de qualquer esquema de criptografia depende (1) do tamanho da chave e (2) do trabalho computacional envolvido na violação de uma cifra. [...]. Em princípio, não há nada sobre a criptografia simétrica ou a de chave pública que trone uma superior a outra do ponto de vista de resistência a criptoanálise. [...]. Um segundo engano é acreditar que a criptografia de chave pública é uma técnica de fim geral, que tornou a criptografia simétrica obsoleta, ao contrário, devido ao *overhead* computacional dos atuais esquemas de criptografia de chave pública, não parece haver qualquer probabilidade de que a criptografia de chave simétrica seja abandonada num futuro próximo. (STALLINGS, 2005, p. 392).

Stallings (2005, p. 394) explica que primeiro é selecionado dois números primos, p e q , e calculado seu produto n , que é o módulo para criptografia e de descryptografia. Em seguida, é necessário a quantidade $\Theta(n)$, denominada o quociente de Euler de n , que é o número de inteiros positivos menores que n e relativamente primos com n . Depois, é selecionado um inteiro e que seja relativamente primo com $\Theta(n)$, ou seja, o maior divisor comum de e e $\Theta(n)$ é 1. Finalmente, calcule d tal que $ed \equiv 1 \pmod{\Theta(n)}$.

Stallings (2005) explica que o único método para vencer o algoritmo de RSA é o ataque de força bruta, mas a maioria dos debates sobre a análise criptográfica do RSA tem focalizado na tarefa de desmembrar seus fatores primos.

4.5.2 Algoritmo da Mochila

Tenenbaum e Wetherall (2011, p. 500) explica que além do RSA, existiu um algoritmo que a ideia era que alguém possui um grande número de objetos, cada objeto com um peso diferente. O dono dos objetos codifica a mensagem selecionando secretamente um subconjunto dos objetos e colocando-os na mochila. O peso total dos objetos na mochila torna-se público, e o mesmo acontece com a lista de todos os objetos possíveis. A lista de objetos da mochila é mantida em segredo. Com outras restrições, o problema de descobrir uma lista de objetos possíveis com o peso fornecido foi considerado computacionalmente inviável e formou a base do algoritmo de chave pública.

O inventor do algoritmo, Ralph Merkle, estava bastante seguro de que esse algoritmo não poderia ser decifrado. O algoritmo foi quebrado pelos mesmos criadores do RSA.

5 RIVEST CIPHERS

Desenvolvido em 1987 por Ron Rivest para a empresa *RSA Data Security, Inc.*, especializada em sistema de criptografia. Foi, durante um tempo, um segredo comercial bem protegido, popular, e utilizado largamente em *software*, como *Lotus Notes*, *Apple Computer's AOCE*, *Oracle Secure SQL*, *Internet Explorer*, *Netscape* e *Adobe Acrobat*.

Imagem 03 – Ron Rivest.



Fonte: <http://people.csail.mit.edu/43ivest/>, 2015.

Em setembro de 1994 foi postado um código fonte em um mailing *list* dedicada à criptografia (*Cypherpunks*) supostamente equivalente ao RC4. Espalhou-se rápido pela rede e foi confirmada a compatibilidade com o RC4. A implementação não é comercial e é referida como ARC4 (*Alleged* RC4).

O RC4 é uma melhoria do RC2, desenvolvido também por Ron Rivest. RC2 é uma cifra de bloco, similar ao DES. RC4 é uma cifra de corrente, onde o algoritmo

produz uma corrente de números pseudoaleatórios que são cifrados através de uma operação lógica XOR com a própria mensagem.

O RC4 é uma técnica de cifra amplamente utilizada no mundo. Embora já exista uma nova versão do algoritmo (RC6) e, atualmente, outras técnicas mais seguras e velozes.

O RC5 é uma melhoria do RC4, se caracterizasse por uma grande flexibilidade e possibilidade de parametrização. Com o RC5 os blocos de entrada podem ter qualquer dimensão pré-determinada, a chave também pode ter qualquer comprimento, e também se pode pré-determinar o número de iterações do algoritmo. Como se pode deduzir ele é muito flexível, estando sujeito a uma serie de parâmetros que devem ser ajustados às necessidades particulares de cada caso.

Segundo o Instituto Superior de Engenharia do Porto (ISEP) a mensagem original é fornecida ao algoritmo sob a forma de dois blocos de w bits, correspondendo ao alinhamento mais conveniente para o sistema em causa, os valores típicos para w são: 16 32 e 64. A mensagem cifrada possui uma forma idêntica.

Outro parâmetro importante é o número de aplicações do algoritmo, pode variar de 1 a 255. Para aplicar r vezes o algoritmo, vai ser gerada a partir da chave uma tabela com $t = 2 \cdot (r + 1)$ blocos de w bits. A chave é especificada pelos parâmetros b e k , b especifica o número de bytes (octetos) que constitui a chave e k é a chave propriamente dita. É habitual usar a notação RC5- $w / r / b$ para especificar uma implementação particular RC5. Podemos dizer que o RC5-32 / 16 / 7 é equivalente ao DES. Os valores mais usados são RC5-32 / 12 / 16 e RC5-32 / 16 / 8. (ISEP)

O RC6 é derivado do RC5. Foram desenvolvidas por Ron Rivest, Matt Robshaw, Ray Sidney e Yiqun Lisa Yin para atender as exigências do *Advanced Encryption Standar* (AES). É um cifrado para chaves de 128 bits, mas que também pode utilizar chaves de 192 e 256 bits. Segundo Semente (2011, p. 23) o funcionamento utiliza as seguintes operações básicas:

- $A + B$: Adição modular de inteiros;
- $A - B$: Subtração modular de inteiros;
- $A \text{ XOR } B$: OU-Exclusivo bit-a-bit de palavras de w -bits;

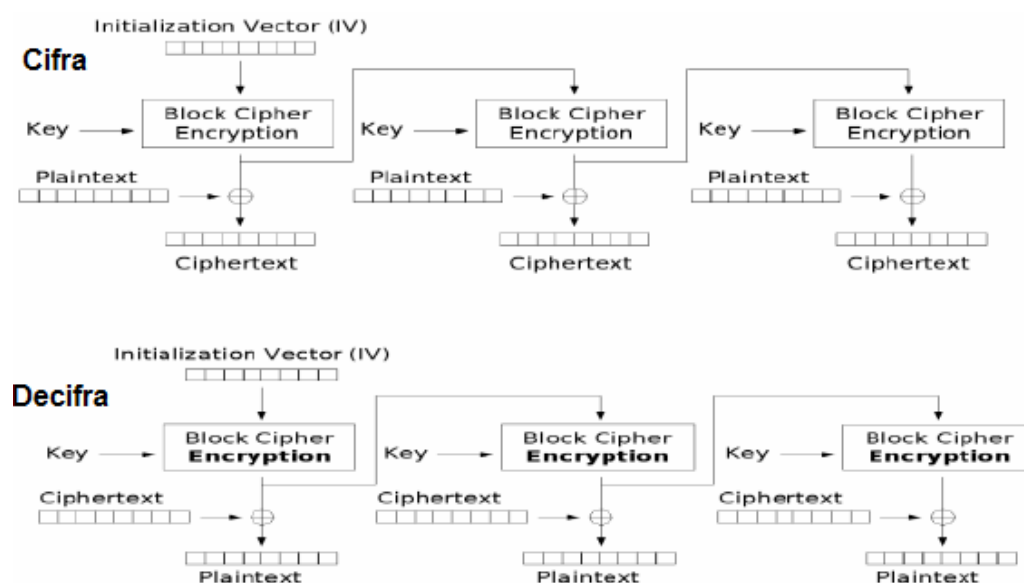
- $A * B$: Multiplicação modular entre inteiros;
- $A \ll B$: Deslocamento de uma palavra A de w -bit para a esquerda por uma quantidade de bits dado pelo valor dos logs (w) bits menos significativos de B;
- $A \gg B$: Deslocamento de uma palavra A de w -bits para a direita por uma quantidade de bits pelo valor dos logs (w) bits menos significativos de B.

Semente (2011, p. 25) observa que o algoritmo do RC6 foi desenvolvido para microprocessador HCS08.

5.1 Estrutura, conceito e fundamentação

O RC4 usa o modo de fluxo de cifras que funciona pela codificação de um vetor de referência com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para obter um segundo bloco de saída. Em seguida, esse bloco é codificado para obter um terceiro bloco e assim por diante. A sequência de bloco de saída é tratada como uma chave única e submetida a uma operação XOR com texto simples para obter texto cifrado.

Imagem 04 – Modo Fluxo de Cifras.



Fonte: http://www.projetoderedes.com.br/artigos/artigo_cifras_em_bloco_cifras_de_fluxo.php,

As transformações no algoritmo RC4 são lineares, não são necessários cálculos complexos, já que o sistema funciona basicamente por permutações e somas de valores inteiros. Seu funcionamento é simples, ele tem uma fase de inicialização da chave chama de *key-scheduling algorithm* (KSA). O RC4 aceita chaves de 1 a 256 bits. Utiliza um vetor S que é primeiramente inicializado em sua totalidade, conforme mostra um trecho, abaixo, de um código em C.

```
for (*i = 0; *i < 256; (*i)++)
{
    s[*i] = *i;
}
```

Após, o vetor S é processado com 256 iterações do algoritmo geração pseudoaleatória e, ao mesmo tempo, combina o resultado do algoritmo geração pseudoaleatória aos valores da chave. O código, abaixo, em C mostra a segunda parte do processo de inicialização do RC4. Sendo *tamanhoKay* o tamanho da chave.

```
*j = 0;
for (*i = 0; *i < 256; (*i)++)
{
    *j = (*j + s[*i] + key[*i % *tamanhoKey]) % 256;
    troca(s, i, j);
}
```

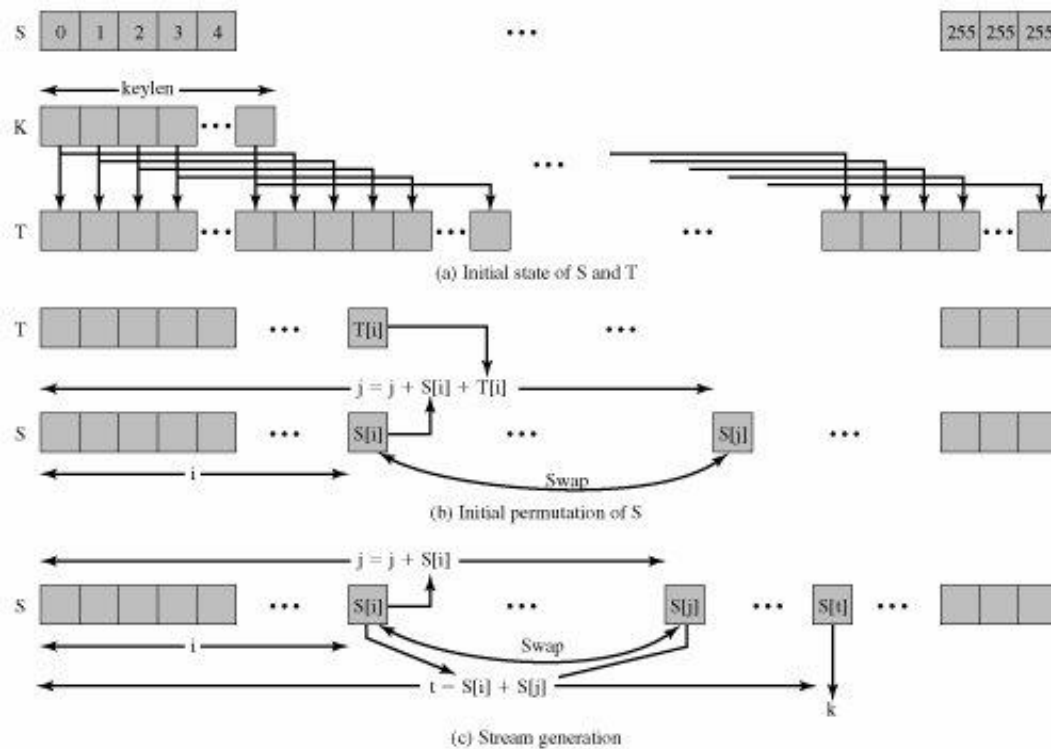
O fluxo principal de cifração utiliza o algoritmo FRGA. A cada iteração, o algoritmo geração pseudoaleatória modifica o seu estado interno e gera um byte de saída. O FRGA permanece ativo até que todos os valores do texto plano sejam processados. A saída é o resultado de uma operação XOR entre o byte de entrada e o módulo de 256 do valor do vetor S, na posição $S[i] + S[j]$. Cada valor da posição $S[i]$ é trocado com a posição $S[j]$, a cada 256 iterações. Segundo Formolo (2009, p. 42) essa troca não é uma regra fixa, existem variações do RC4 que fazem a troca com mais frequência, o que aumenta a segurança do algoritmo. O trecho, abaixo, de um código em C mostra a cifração do RC4.

```

for (aux = 0; aux < *tamanhoPlainText; aux++)
{
    *i = (*i + 1) % 256;
    *j = (*j + s[*i]) % 256;
    troca(s, i, j);
    result[aux] = (s[(s[*i] + s[*j]) % 256]) ^ *(plain + aux);
}
chiper=(unsigned char*)calloc((*tamanhoPlainText-1),(sizeof(unsigned
char)));
strcpy(chiper, result);

```

Imagem 05 – Modelo de RC4.



Fonte: <http://flylib.com/books/em/3.190.1.63/1/>, 2015.

Embora seja muito utilizado e de construção simples, o RC4 é alvo de diversos ataques criptoanalistas como os citados anteriormente. Segundo Formolo (2009).

O RC4 não utiliza LFSR, que é um elemento comum em cifradores de fluxo. Para gerar símbolos cifrados, ele implementa uma máquina de estados

sobre operações simples, iteradas em um número finito de rodadas. Desta forma, ele utiliza a base de cifradores de fluxo, a qual é gerar símbolos pseudoaleatórios, baseados em um estado inicial e chave secreta, combinados com os símbolos planos. (FORMOLO, 2009, p. 43).

De uma forma geral, o algoritmo consiste em utilizar um *array* que a cada utilização tem os seus valores permutados, e misturados com a chave. Esta chave utilizada na inicialização do *array* pode ter até 256 bytes (2048 bits), embora o algoritmo seja mais eficiente quando é menor, pois a perturbação aleatória induzida no *array* é superior.

5.1.1 Tabela-Verdade

Abe, Scalzitti e Filho (2002) definem a tabela-verdade da disjunção exclusiva (XOR) entre dois operandos que resulta em um valor verdadeiro se e somente se (\leftrightarrow) um dos operadores possui valor verdadeiro e o outro falso.

Tabela 02 – Tabela-verdade XOR.

A	B	$A \vee B$
1	1	0
1	0	1
0	1	1
0	0	0

Fonte: ABE, SCALZITTI E FILHO, 2002.

A tabela-verdade XOR apresentada, anteriormente, é a negação da tabela-verdade da bi-implicação. Abe, Scalzitti e Filho (2002, p. 34) definem a tabela-verdade da bi-implicação entre dois operandos que resulta em um valor verdadeiro se e somente se (\leftrightarrow) as proposições A e B possuem o mesmo valor-verdade. A proposição ($A \leftrightarrow B$) é falsa se e somente se as proposições A e B tiverem valores-verdade trocados.

Tabela 03 – Tabela-verdade bi-implicação.

A	B	$A \leftrightarrow B$
----------	----------	---

1	1	1
1	0	0
0	1	0
0	0	1

Fonte: ABRE, SCALZITTI E FILHO, 2002.

5.2 Benefícios em relação às técnicas anteriores

O RC4 é considerado um algoritmo de chave simétrica – mesma chave cifra e decifra – de cifra de fluxo, pelo fato de o processo de encriptação e deciptação serem independentes do tamanho do texto simples.

O benefício de ser usar um algoritmo de chave simétrica é a sua simplicidade, pois ela apresenta facilidade de uso e rapidez para executar os processos criptográficos e não consome muitos recursos tecnológicos.

5.2.1 Adaptabilidade

A adaptabilidade do RC4 é um ótimo benefício, pois dar a capacidade de regular o tamanho do vetor de inicialização para aumentar a segurança.

5.2.2 Chave

A chave tem que ter no mínimo 1 bit e no máximo 2048 bits, enquanto o AES vai de 128 a 256 bits, o DES é só de 56 bits e o 3DES é de 168 bits. Com chave grande é possível dificultar a criptoanálise.

5.2.3 Memória

Por ser uma chave simétrica, não são necessários muitos recursos para a sua encriptação e deciptação.

5.2.4 Simplicidade

Criptografia de chave simétrica são bem simples de serem implementadas em *software* e *hardware*.

5.2.5 Modo de Cifra

Existe n modos de cifra (ECB, EBC, OFB, FC, MC...), mas alguns são melhores que outros em certas situações.

Tanenbaum e Wetherall (2011) descrevem cinco modos de cifras e suas fraquezas.

O primeiro é o modo *electronic code book* (ECB), sua fraqueza é que para fraudar a mensagem, basta o intruso ativo copiar um bloco de n bytes e colar em outro lugar que tenha a mesma quantidade de bytes. Mesmo sem saber o conteúdo do bloco, o intruso ativo pode facilmente alterar o conteúdo.

O segundo é o modo de encadeamento de blocos de cifras (EBC), usado no RSA e no DES, se a fraqueza do ECB é a facilidade de alterar o conteúdo da mensagem alterando um bloco por outro. O EBC foi desenvolvido para resolver essa fraqueza, mas mesmo o EBC sendo superior ao ECB, tem um ponto fraco que é a exigência de um bloco de 64 bits inteiro para poder iniciar a decodificação.

O terceiro é o modo de *feedback* de cifra (OFC), usado no 3DES e no AES, superior, em certas situações, ao EBC e conseqüentemente superior ao ECB. O OFC tem uma fraqueza, se um bit do texto cifrado for invertido acidentalmente durante a transmissão, os 8 bytes decodificados enquanto o byte defeituoso estiver no registrador de deslocamento serão danificados. Depois que o byte defeituoso for empurrado para fora do registrador de deslocamento, o texto simples correto será gerado mais uma vez.

O quarto é o modo fluxo de cifras (FC) usado no RC4, superior, em certas situações, ao OFC que conseqüentemente é superior ao EBC que é superior ao ECB. Tornando o RC4, em alguns casos específicos, superior ao 3DES, AES, RSA,

O quinto modo contador (MC) é superior, em certas situações, aos três citados anteriormente, porque o EBC, OFC e FC apresentam a impossibilidade de conseguir acesso aleatório a dados codificados.

5.3 Aplicações que fazem ou fizeram uso do RC4

Como visto no capítulo anterior, RC4 mostra algumas vantagens, em certos ambientes, em relações a certas técnicas de criptografia (DES, RC2, 3DES, AES, DES,...). O motivo é a utilização do modo fluxo de cifras (FC) que possui como falha a incapacidade de conseguir acesso aleatório a dados codificados, mas possui como princípio de funcionamento o segredo criptográfico perfeito, ou seja, tamanho da chave diferente do tamanho da mensagem.

RC4 é fortemente usado nos padrões SSL / TLS (*Secure Sockets Layer / Transport Layer Security*), WEP (*Wired Equivalent Privacy*), WPA (*Wi-Fi Protected Access*), *CipherSaber*, *BitTorrent protocol encryption*, *Microsoft Point-to-Point Encryption*, *Secure shell*, *Kerberos* e *MSOffice 2002 / 2003*. Foi durante um longo tempo usado também em navegadores como o *Google Chrome*, *Mozilla Firefox*, *Microsoft Edge* e *Internet Explore*.

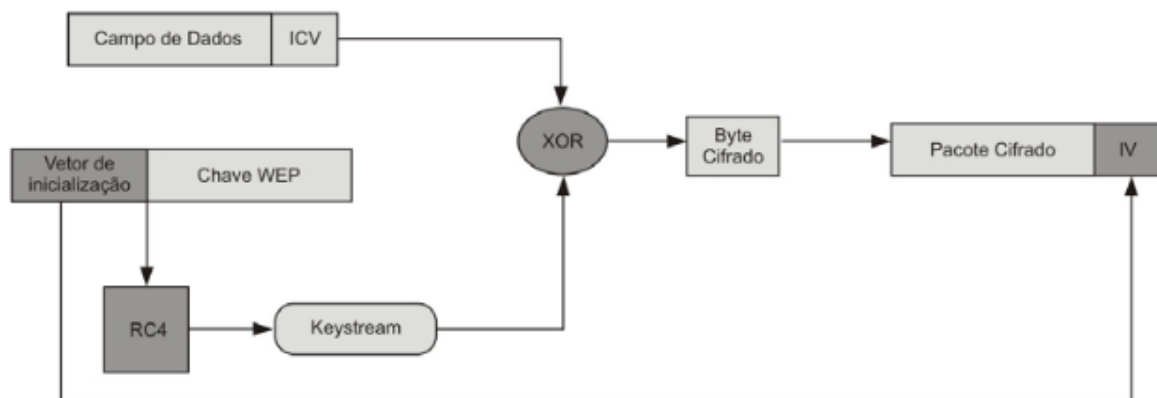
[...] O Google também informou que apenas 0,13% das conexões HTTPS feitas por usuários do navegador usam RC4. [...]. Já a Microsoft planeja desativar o RC4 por padrão para todos os usuários do Microsoft Edge e do Internet Explorer [...]. A empresa também afirmou que a porcentagem de serviços web inseguros que suportam apenas o RC4 é pequeno e está encolhendo. [...]. A Mozilla também compartilhou que cerca de 0,08% dos usuários do Firefox ainda utilizam RC4. (*CANALTECH*, 2015).

SSLT / TLS é um protocolo de segurança que protege as telecomunicações via internet para serviços como e-mail, navegação e outros tipos de transferência de dados.

[...] WEP foi introduzido no padrão IEEE 802.11 em 1999. Ele provê dois métodos de autenticação de dispositivos, utiliza o CRC-32 (*Cyclic Redundancy Checks*) para a verificação da integridade de dados e o algoritmo de criptografia RC4 para prevenir a leitura de dados [...]. (LINHASRES E GONÇALVES, p. 3).

WEP foi o primeiro protocolo de segurança adotado e ainda é muito usado nos dias atuais.

Imagem 06 – Encapsulamento WEP.



Fonte: Linhas e Gonçalves, 2015.

WPA foi desenvolvido para corrigir as fraquezas do sistema WEP. É baseado no RC4 e em um subconjunto de especificações apresentadas em uma versão preliminar do IEEE 802.11i.

É considerado seguro num contexto prático (duração das transações). No entanto, de forma geral segundo *DevMedia* (2015) “ [...] os adeptos da criptografia não consideram o RC4 um dos melhores sistemas criptográficos, e em algumas aplicações são considerados como sendo muito inseguros [...] ”.

5.4 Discussão Comparativa

RC4, como mostrado nos capítulos anteriores, não é a única técnica de criptografia conhecida e muito menos a mais ou menos segura.

No conjunto das chaves simétricas, além do RC4, temos as técnicas DES, 3DES, AES, IDEA, *Serpent*, *Twofish*, entre outras. No conjunto das chaves assimétricas temos o RSA, Mochila, entre outras.

Neste capítulo estaremos comparando o RC4 com várias técnicas criptográficas, citas ou não no trabalho, para chegar a algumas características diferentes.

5.4.1 RC5 X RC4

O algoritmo RC5 foi projetado pelo professor Ronald Rivest do MIT, Estados Unidos, e a primeira publicação foi em 1994. Após sua publicação, muitos

investigadores da comunidade científica se esforçaram para avaliar com precisão a segurança oferecida. Ele é o sucessor do RC4 e apresenta significativas melhorias nos fatores segurança e velocidade.

O RC5 possui um modo de cifra diferente do RC4 e similar ao DES. O algoritmo baseia-se na operação de rotação de um número variável de posições.

5.4.2 DES X RC4

O DES é um sistema de criptografia que já está há muito tempo no mercado, mas isso não quer dizer que ele seja mais seguro que o RC4. DES usa uma chave de 56 bits para criptografar blocos de mensagens com 64 bits onde ocorre uma alteração num alfabeto composto por 264 símbolos, já o RC4 consegue ter chaves que podem ser de 1 a 1024 bits e utilizam o XOR como algoritmo de criptografia dando assim maior segurança aos dados. Há também uma variação, citado no capítulo 4, do DES o chamado Triple-DES que se baseiam na execução da técnica do DES três vezes.

O modo de cifra é muito diferente. DES usa o encadeamento de bloco, explicado no capítulo anterior, enquanto o RC4 usa o modo FC.

5.4.3 IDEA X RC4

O IDEA foi inventado na década de 90 na Suíça, ele segue as mesmas linhas do DES, porém apresenta maior velocidade, ele utiliza uma chave de 128 bits e foi muito utilizado em procedimentos bancários, em contrapartida, o RC4 é um pouco mais antigo que o IDEA e é mais utilizado em protocolos e procedimentos WEB.

5.4.4 AES X RC4

O sistema AES começou no início do séc. XXI, por isso, ela é muito nova e complexa, enquanto sua antecessora (RC4) é mais antiga e simples. Simples ou complexa não determina o seu potencial e sim a sua versatilidade.

A diferença mais significativa entre os dois provavelmente seria seu tipo. AES é uma cifra de bloco que opera em blocos distintos de dados usando uma chave fixa e uma fórmula, enquanto RC4 é uma cifra de fluxo que não tem um tamanho de

bloco discreto. Em vez disso, ele usa um *keystream* de bits pseudoaleatórios que é combinado com os dados usando uma operação OU-exclusivo (XOR). Você pode usar cifras de bloco como cifras de fluxo e vice-versa, de modo que a separação não é muito distinta. Mas é bem conhecido que RC4 não é muito eficaz quando utilizada como um bloco de cifra.

5.4.5 RSA X RC4

O RC4 é um algoritmo simétrico de criptografia de fluxo desenvolvido em 1987, pessoas mais adeptas a criptografia não o consideram por não serem necessários cálculos que acaba por deixando muito simples e rápido.

O RSA é uma técnica mais segura que o RC4 por diversos motivos o principal seria o método usado para a geração de chaves onde ele se baseia na teoria dos números, por causa da dificuldade em fatorar um número em seus componentes primos numa divisão exata, sem números quebrados.

É desta particularidade que vem a segurança do RSA. Na verdade, não é impossível quebrar a criptografia RSA, mas para fazer isso seriam necessários alguns bons anos ou décadas, tornando a ideia inviável.

5.4.6 Sosemanuk X RC4

Sosemanuk é uma cifra de fluxo, funciona por meio da cifra de bloco, baseada na técnica *Snow* e *Serpent*, porém ela tem uma inicialização mais rápida. O comprimento dela varia de 128 a 256 bits, porém a segurança é garantida apenas quando se usa 128 bits, a *Sosemanuk* não é patenteada, logo ela é livre para qualquer utilização.

RC4 também é uma cifra de fluxo e funciona por meio de cifra de bloco, porém ela possui menor dificuldade e tem melhor funcionamento com comprimento abaixo de 256 bits.

5.4.7 *Serpent X RC4*

O *Serpent* é um algoritmo do tipo *block cipher* que utiliza bloco de 128 bits dividido em 4 palavras de 32 bits cada e trabalha com chaves de 128, 192 ou 256 bits.

O algoritmo *Serpent* foi projetado de modo que todas as operações pudessem ser executadas em paralelo, utilizando 32 faixas de 1 bit. Portanto, foi estruturado de forma a maximizar o paralelismo tornando ele mais complexo e mais seguro que o RC4 comum.

O modo de cifra, FC, usado no RC4 leva uma grande vantagem no quesito segurança em comparação ao usado no *Serpent*, BC, mas perde no quesito de acessar aleatoriamente dados codificados.

5.4.8 *One-Time-Pad X RC4*

A técnica *One-Time-Pad*, em português cifra de uso único, foi escrita em 1882 e reinventada em 1917, não pode ser quebrada se usada corretamente, gerando uma chave secreta aleatória que para isso deve ter no mínimo, a mesma quantidade de caracteres do texto simples, utilizado pelo exército e governo britânico. Ela é derivada da cifra de *Vernam*.

RC4 não possui uma função *random* para gerar chaves aleatórias e existem vários ataques eficazes.

5.4.9 *Twofish X RC4*

Twofish foi um dos cinco finalistas do AES (padrão de criptografia avançada), mas não foi selecionado. É uma cifra de bloco, publicado em 1998.

Twofish tem um tamanho de bloco de 128 bits, um tamanho de chave que varia de 128 a 256 bits e é otimizado para processadores de 32 bits. Por ciência que falta – jargão usado, na área de inteligência artificial, para referência poucas informações – e fonte confiável, não foi possível encontrar mais informações sobre o *Twofish*.

5.5 Vulnerabilidade e Falhas

Desde a divulgação pública do algoritmo em 1994, começaram a surgir ataques criptoanálise contra o RC4, porém, o primeiro ataque significativo surgiu em 2001, quando Fluhrer, Martin e Shamir demonstrou que o protocolo RC4 possuía vulnerabilidades em seu algoritmo de agendamento de chaves (KSA) e que seriam possíveis ataques práticos a protocolos de segurança que utilizassem esta cifra, tal como o WEP.

No ano de 2004, um cracker usando apelidado de *KoreK* publicou uma lista de discussão que continha 16 correlações entre os primeiros bytes da chave e os dois primeiros bytes do fluxo pseudoaleatório. Dois anos seguintes Adreas Klein publicou dois novos ataques.

Palma e Pereira explicam que o primeiro demonstra uma nova correlação entre os primeiros bytes da sequência pseudoaleatório e que não requer a existência do vetor S. O segundo dispensa necessidade dos primeiros bytes do fluxo pseudoaleatório.

A função KSA do RC4 é vulnerável, possuindo duas fraquezas significativas. A primeira é a existência de uma larga classe de chaves consideradas fracas, onde uma pequena porção dos bits da chave determina um grande número de bits da permutação inicial S. A segunda é a fraqueza do vetor de inicialização, se um atacante conhecer parte da chave (IV), ele é capaz de derivar a parte secreta da chave analisando o byte inicial do fluxo de byte de saída do RC4 com (relativo) pouco esforço.

5.6 Melhorias propostas ou implementadas

O algoritmo RC4 não é seguro e otimizado como seus predecessores. Por ele se uma técnica simétrica, ele já pode ser considerado inferior a qualquer técnica assimétrica, já que na teoria a assimétrica é mais segura e recomendada do que uma simétrica.

A Criptografia de Chave pública é mais segura contra análise criptográfica que a criptografia simétrica. [...]. Na realidade, a segurança de qualquer esquema de criptografia depende (1) do tamanho da chave e (2) do trabalho

computacional envolvido na violação de uma cifra. [...]. Em princípio, não há nada sobre a criptografia simétrica ou a de chave pública que trone uma superior a outra do ponto de vista de resistência a criptoanálise. [...]. Um segundo engano é acreditar que a criptografia de chave pública é uma técnica de fim geral, que tornou a criptografia simétrica obsoleta, ao contrário, devido ao overhead computacional dos atuais esquemas de criptografia de chave pública, não parece haver qualquer probabilidade de que a criptografia de chave simétrica seja abandonada num futuro próximo. (STALLINGS, 2005, p. 392).

5.6.1 Melhorias Implementadas

Estivermos deixando o programa mais otimizado, para usar o mínimo possível de recursos. E para isso utilizamos as técnicas de alocação dinâmica de memória.

Demos a capacidade de o programa criar, editar, apagar e abrir arquivos.txt para facilitar a leitura da mensagem cifrada.

Criamos uma biblioteca chamada “rciv.h” onde contêm as funções de cifragem e decifragem do algoritmo do RC4. Ela foi criada para diminuir e facilitar a leitura na função *main*, onde a mesma apenas recebe dados e chama as funções que irão fazer o trabalho de criptografia e descriptografia.

5.6.2 Melhorias Propostas

Poderia ser implementado juntamente com o RC4 a técnica RSA, tornando o RC4 uma criptografia híbrida – criptografia híbrida é aquela que é simétrica e assimétrica. Funcionária da seguinte maneira, a chave simétrica do usuário é convertida em uma chave assimétrica, isso tudo aconteceria transparentemente.

A chave híbrida não é uma grande novidade, ela é fortemente usada nos dias atuais, por ela conter a combinação de vantagens da simétrica e da assimétrica.

Adicionando o sistema caótico dimensional na técnica criptográfica RC4. Altera inconvenientes da chave fraca, expande ritmo fundamental do algoritmo original, e em seguida, faz com que a maioria dos métodos de atacar o RC4 não funcione e aumenta a segurança do algoritmo como um todo. É proposta uma leitura e gravação do mecanismo seguro que efetivamente impede variedades de ataques e melhora a segurança do sistema e dados combinados com o RC4. Assim, o

algoritmo RC4 melhorado tem as características de baixo consumo de energia, criptografia rápida.

6 PROJETO

O RC4 usa o modo de fluxo de cifras que funciona pela codificação de um vetor de referência com uma chave para obter um bloco de saída. O bloco de saída é então codificado, usando-se a chave para obter um segundo bloco de saída. Em seguida, esse bloco é codificado para obter um terceiro bloco e assim por diante. A sequência de bloco de saída é tratada como uma chave única e submetida a uma operação XOR com texto simples para obter texto cifrado.

O programa foi baseado no algoritmo do RC4 com algumas melhorias e alterações pequenas, nada que mudasse a funcionalidade ou a lógica, do algoritmo, tragicamente. Ele foi dividido em dois arquivos, o *main.c*, arquivo principal, e a biblioteca *rciv.h* que contém as funções utilizadas no arquivo principal.

6.1 Main.c

Dividimos o programa em dois arquivos, *main* e *rciv*, para facilitar a leitura do mesmo.

O arquivo principal possui apenas a função *main*, porque ela é usada para declarar os apontadores, usado para apontar o endereço de memória de algumas variáveis que são usados continuamente nas funções da biblioteca *rciv.h*, declarar as funções e as chamar para serem usadas.

A imagem abaixo de um trecho do algoritmo demonstra o conteúdo da função *main* contida no arquivo principal.

O programa se inicia chamando a função menu que está definida no segundo arquivo do programa, *rciv.h*, juntamente com as outras funções usadas para criptografar.

A função menu retorna um valor do tipo inteiro, inserido pelo usuário. Caso o valor seja igual a um, ele entra no primeiro bloco que recebe a mensagem simples, uma chave simétrica e com a combinação da mensagem simples e da chave simétrica, cria uma mensagem criptografada. Caso o valor seja igual a dois, ele entra no segundo bloco que recebe a mensagem criptografada e a chave simétrica, usada para criptografar a mensagem, e com a combinação da mensagem criptografada e da chave simétrica o programa decifra a mensagem mostra a original.

Imagem 07 – Trecho Algoritmo main.

```

inicio
  escolha(menu())
  caso 1 :
    escreva ("#####\n");
    escreva ("##### Digite abaixo a Mensagem a ser Criptografada #####\n");
    escreva ("#####\n");
    leia (plain);
    tamanhoPlainText1 <-- strlen(plain);

    escreva ("#####\n");
    escreva ("##### Digite abaixo a Chave Simetrica #####\n");
    escreva ts("#####\n");
    leia (key);
    tamanhoKey <-- strlen(key);

    escreva ("#####\n");
    escreva ("##### Mensagem Criptografada #####\n");
    escreva ("#####\n");
    escreva (chiperText(plain, key, i, j, tamanhoKey, tamanhoPlainText));
    escreva ("#####\n");
    escreva ("##### Aperte qualquer tecla para voltar ao menu #####\n");
    escreva ("#####\n");

  caso 2 :
    escreva ("#####\n");
    escreva ("##### Digite abaixo a Mensagem a ser Descriptografada #####\n");
    escreva ("#####\n");
    leia (plain);
    tamanhoPlainText <-- strlen(plain);

    escreva ("#####\n");
    escreva ("##### Digite abaixo a Chave Simetrica #####\n");
    escreva ("#####\n");

```

Fonte: Própria, 2015.

Caso o usuário digite algum valor diferente de um ou dois, o programa retorna uma mensagem de erro e volta a solicitar outro valor.

Não muda muita coisa do algoritmo para o código em C, porque o que interessa no algoritmo é a lógica que será convertida para linguagem C. Podemos ver, na imagem abaixo, que o que muda de fato é as palavras reservadas, escreva é *printf*, leia é *gets*, caso é *case*. Os demais comandos são palavras reservadas da linguagem C e realizam tarefas específicas, *fflush()* serve para limpar o *buffer* do teclado, assim evitando lixos, *free()* serve para “soltar” memória usada pelo programa, *getch()* é bem parecido com o *gets()* a diferença é que o *getch()* ler apenas um caractere e o *gets()* ler mais que um, e a função *system()* serve para dar algum comando pelo *command*, como limpar o *command* ou abrir algum arquivo.

A imagem abaixo mostra apenas o primeiro caso, mas o segundo é semelhante.

Imagem 08 – Main em C.

```
case 1:
    system("cls");
    fflush(stdin);

    puts("#####");
    puts("##### Digite abaixo a Mensagem a ser Criptografada #####");
    puts("#####");
    gets(plain);
    *tamanhoPlainText = strlen(plain);
    fflush(stdin);

    puts("#####");
    puts("##### Digite abaixo a Chave Simetrica #####");
    puts("#####");
    gets(key);
    fflush(stdin);
    *tamanhoKey = strlen(key);

    puts("#####");
    puts("##### Mensagem Criptografada #####");
    puts("#####");
    puts(chiperText(plain, key, i, j, tamanhoKey, tamanhoPlainText));
    puts("#####");
    puts("##### Aperte qualquer tecla para sair do programa #####");
    puts("#####");

    getch();
    free(plain);
    free(key);
    free(i);
    free(j);
    free(tamanhoKey);
    free(tamanhoPlainText);
    free(texto);
    system("start main");
```

Fonte: Própria, 2015.

6.2 Rcv.h

O arquivo secundário é uma biblioteca desenvolvida especialmente para a técnica de criptografia RC4 e contém as funções necessárias para criptografar uma mensagem simples, usando uma chave simétrica.

A imagem abaixo mostra a função menu, declarada e chamada no arquivo *main*, que serve para visualizar um menu, recebe a opção desejada pelo usuário e retorna a mesma.

Imagem 09 – Função Menu.

```
função menu() : inteiro
    início
        r : inteiro;

        escreva ("#####\n");
        escreva ("##### Criptografia RC4 + MFC + RSA #####\n");
        escreva ("#####\n");
        escreva ("##### Escolha uma das opcoes abaixo #####\n");
        escreva ("#####\n");
        escreva ("##### \t1 - Criptografar\t #####\n");
        escreva ("##### \t2 - Descriptografar\t #####\n");
        escreva ("#####\n");
        escreva ("\t\t\t Opcao: ");
        leia (r);
        escreva ("#####\n");

        retorne (r);
    fim
```

Fonte: Própria, 2015.

Como explicado no capítulo 5.1, as transformações no algoritmo RC4 são lineares. Ele tem uma fase de inicialização da chave chama de *key-scheduling algorithm* (KSA). A imagem abaixo mostra esta inicialização em C, já que no capítulo 5.1 foi mostrado em português.

Imagem 10 – Função KSA.

```
void ksa(unsigned short *i, unsigned short *j, unsigned short *tamanhoKey)
{
    for (*i = 0; *i < 256; (*i)++)
    {
        s[*i] = *i;
    }

    *j = 0;

    for (*i = 0; *i < 256; (*i)++)
    {
        *j = (*j + s[*i] + key[*i % *tamanhoKey]) % 256;
        troca(s, i, j);
    }

    *i = 0;
    *j = 0;
}
```

Fonte: Própria, 2015.

A imagem à cima mostra também o processo de pseudoaleatório, que combina simultaneamente o resultado da geração pseudoaleatório com os valores da chave.

O fluxo principal de cifração utiliza o algoritmo FRGA. A cada iteração, o algoritmo geração pseudoaleatória modifica o seu estado interno e gera um byte de saída. O FRGA permanece ativo até que todos os valores do texto plano sejam processados. A saída é o resultado de uma operação XOR entre o byte de entrada e o módulo de 256 do valor do vetor S, na posição $S[i] + S[j]$. Cada valor da posição $S[i]$ é trocado com a posição $S[j]$, a cada 256 iterações.

Imagem 11 – Função FRGA.

```
void prga (unsigned char *plain, unsigned short *i, unsigned short *j, unsigned short *tamanhoPlainText)
{
    unsigned int aux;
    unsigned char result[*tamanhoPlainText-1];

    for (aux = 0; aux < *tamanhoPlainText; aux++)
    {
        *i = (*i + 1) % 256;
        *j = (*j + s[*i]) % 256;
        troca(s, i, j);
        result[aux] = (s[(s[*i] + s[*j]) % 256]) ^ *(plain + aux);
    }

    chiper = (unsigned char*) calloc((*tamanhoPlainText - 1), (sizeof(unsigned char)));
    strcpy(chiper, result);
}
```

Fonte: Própria, 2015.

A imagem abaixo mostra a função que chama as funções, citada anteriormente, para que seja realizado a cifração da mensagem simples.

Imagem 12 – Função *ChiperText*.

```
unsigned char *chiperText(unsigned char *plain, unsigned char *key1, unsigned short *i, unsigned short *j, unsigned short *tamanhoKey,
unsigned short *tamanhoPlainText)
{
    int cont = 0;

    while(cont < 256){
        key[cont] = *(key1 + cont);
        cont++;
    }

    ksa(i, j, tamanhoKey);
    prga(plain, i, j, tamanhoPlainText);

    return(chiper);
}
```

Fonte: Própria, 2015.

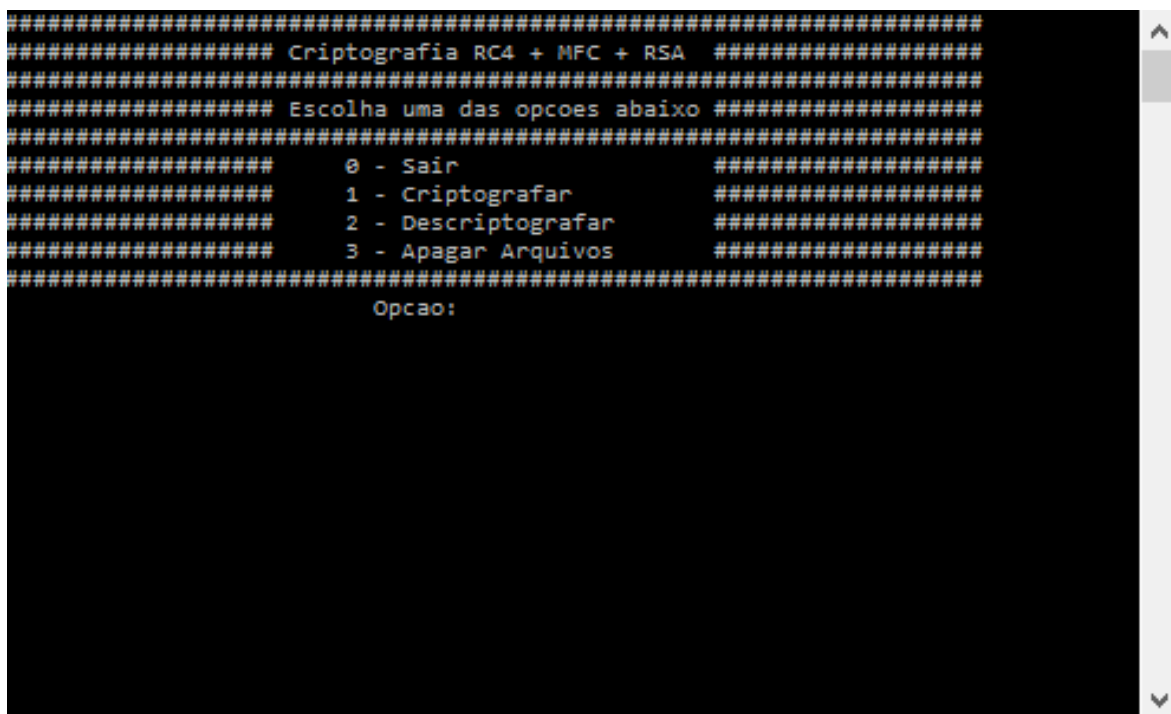
Ao todo, somando os dois arquivos, foram escritos duzentos e cinquenta e quatro linhas de código.

7 APRESENTAÇÃO

Utilizamos o *Windows 8.1* e *Windows 10* para os testes do programa. O *Visual Studio Code 0.9.2* e o *Sublime Text* foram usados para a escrita do programa. O *command* do *Windows 8.1* e *Windows 10*, juntamente com o *MinGW*, foram usados para a compilação e execução do programa.

7.1 Printscreen da Home

Imagem 13 – Printscreen da Home.

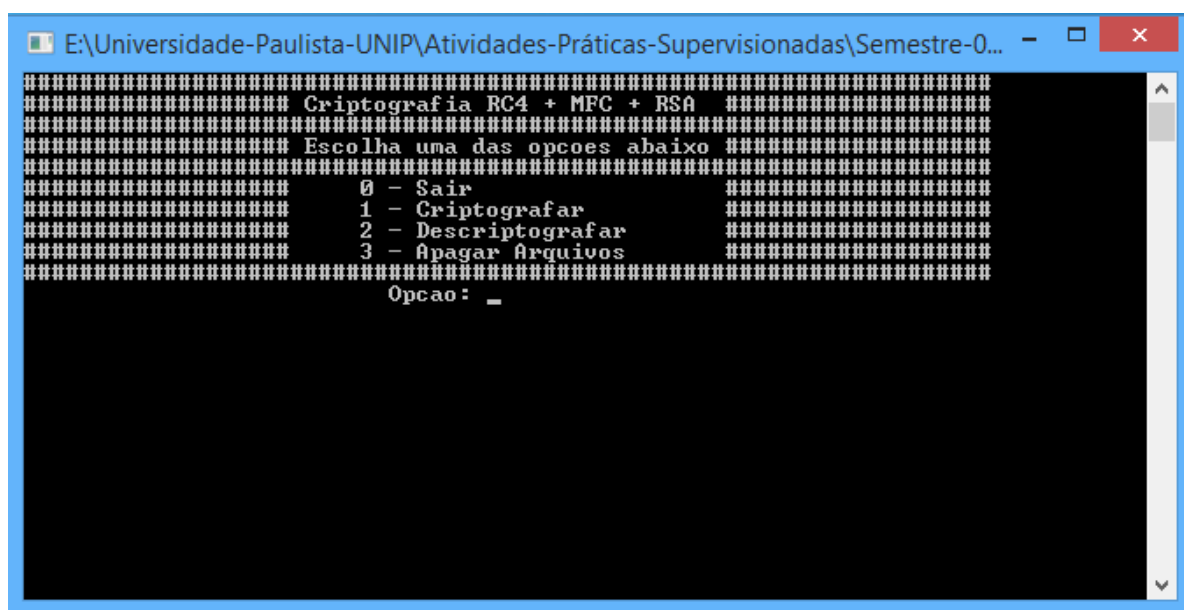
A imagem é uma captura de tela de uma interface de linha de comando (terminal) com fundo preto e texto em amarelo. O texto está alinhado à esquerda e é delimitado por linhas de caracteres de preenchimento (hashes) em ambas as extremidades. O conteúdo da tela é o seguinte:

```
#####  
##### Criptografia RC4 + MFC + RSA #####  
##### Escolha uma das opcoes abaixo #####  
#####  
##### 0 - Sair #####  
##### 1 - Criptografar #####  
##### 2 - Descriptografar #####  
##### 3 - Apagar Arquivos #####  
#####  
##### Opcao: #####
```

À direita da janela do terminal, há uma barra vertical cinza com setas de rolagem no topo e na base.

Fonte: Própria, 2015.

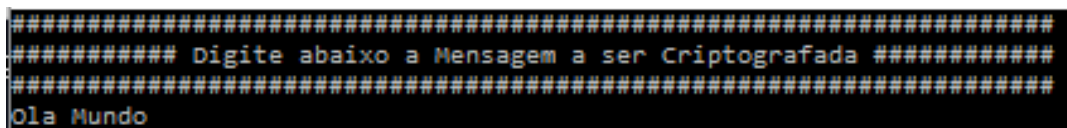
Imagem 14 – Printscreen da Home.



Fonte: Própria, 2015.

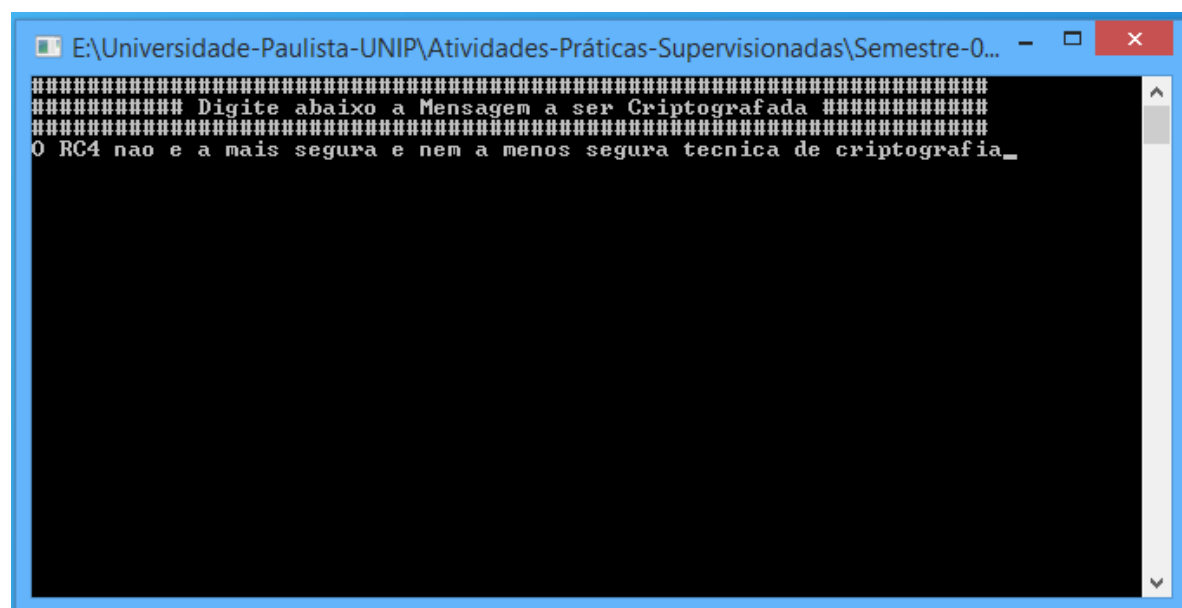
7.2 Printscreen da Mensagem Simples

Imagem 15 – Printscreen da Mensagem Simples.



Fonte: Própria, 2015.

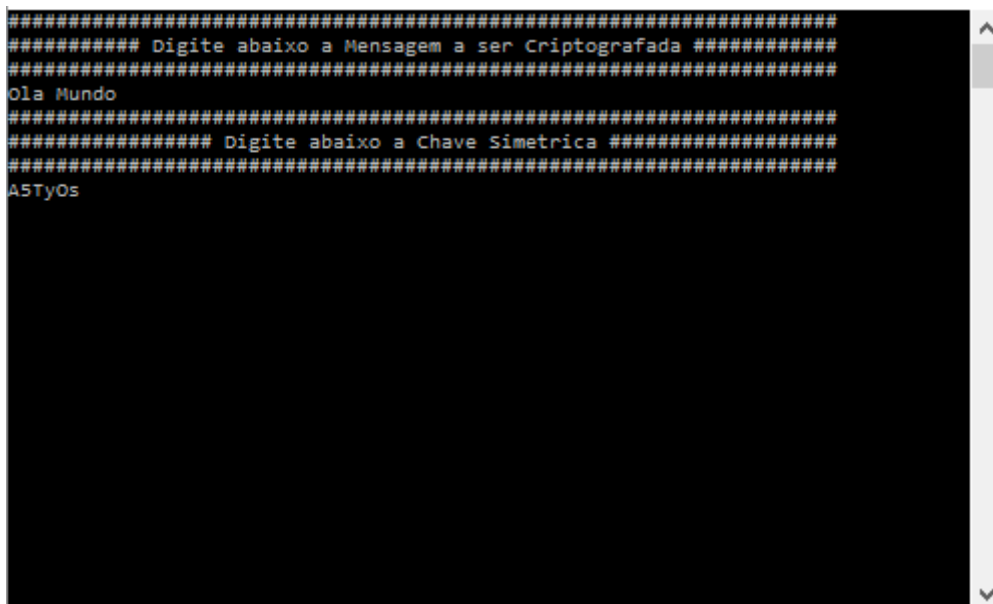
Imagem 16 – Printscreen da Mensagem Simples.



Fonte: Própria, 2015.

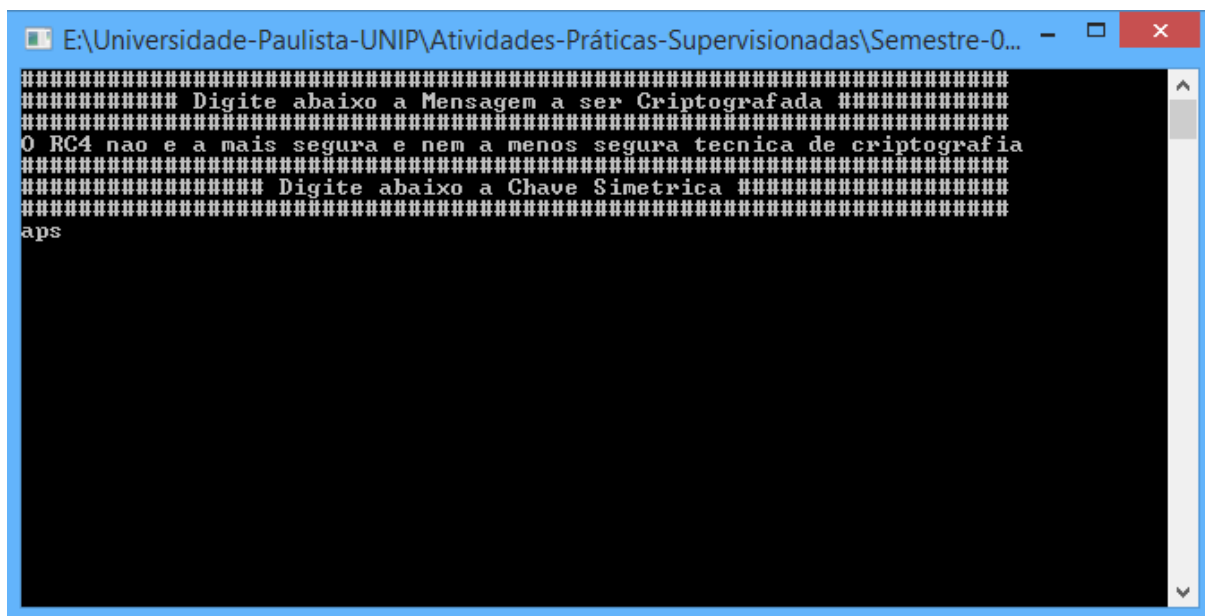
7.3 Printscreen da Chave Simétrica

Imagem 17 – Printscreen da Chave Simétrica.



Fonte: Própria, 2015.

Imagem 18 – Printscreen da Chave Simétrica.



Fonte: Própria, 2015.

Imagem 19 – Printscreen da Chave Simétrica.

```
#####
##### Digite abaixo a Mensagem a ser Descriptografada #####
#####
XhBîú^Yu-■)
##### Digite abaixo a Chave Simétrica #####
#####
A5TyOs
```

Fonte: Própria, 2015.

Imagem 20 – Printscreen da Chave Simétrica.

```
E:\Universidade-Paulista-UNIP\Atividades-Práticas-Supervisionadas\Semestre-0...
#####
##### Digite abaixo a Mensagem a ser Descriptografada #####
#####
ΔN=Iãx6 |Sye2ü *û||Uua>β-pôg*2ôuS`14$^ôQJi. +ã;B ì9HΓ+ÿ||\▽qr=>I♦-p-g>^H;ùvá<
##### Digite abaixo a Chave Simétrica #####
#####
aps_
```

Fonte: Própria, 2015.

7.4 Printscreen da Mensagem Criptografada

Imagem 21 – Printscreen da Mensagem Criptografada.

```
##### Digite abaixo a Mensagem a ser Criptografada #####
#####
Ola Mundo
##### Digite abaixo a Chave Simetrica #####
#####
A5TyOs
##### Mensagem Criptografada #####
#####
Kh8i0ú0u-■)
#####
##### Aperte qualquer tecla para voltar ao menu #####
#####
-

```

Fonte: Própria, 2015.

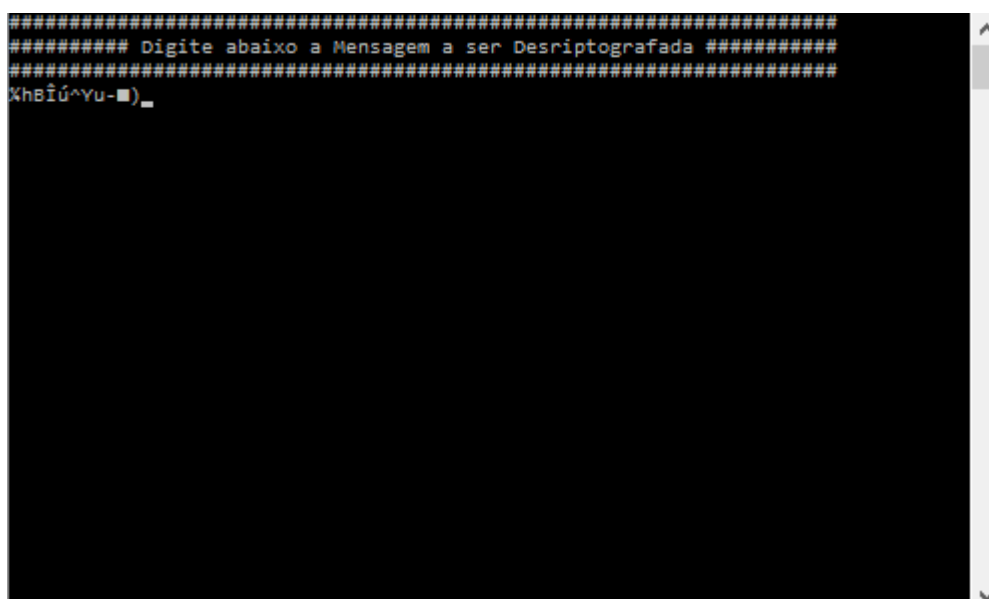
Imagem 22 – Printscreen da Mensagem Criptografada.

```
E:\Universidade-Paulista-UNIP\Atividades-Práticas-Supervisionadas\Semestre-0...
##### Digite abaixo a Mensagem a ser Criptografada #####
#####
O RC4 nao e a mais segura e nem a menos segura tecnica de criptografia
##### Digite abaixo a Chave Simetrica #####
#####
aps
##### Mensagem Criptografada #####
#####
ΔN=Iãx6 |Sye2ü *ü||Uua>β-p0g*20uS`1#5^0QJi. *ã;B i9H|7+ÿ||\vqr=>I♦p-g>^H;ùvá■<
#####
##### Aperte qualquer tecla para voltar ao menu #####
#####
-

```

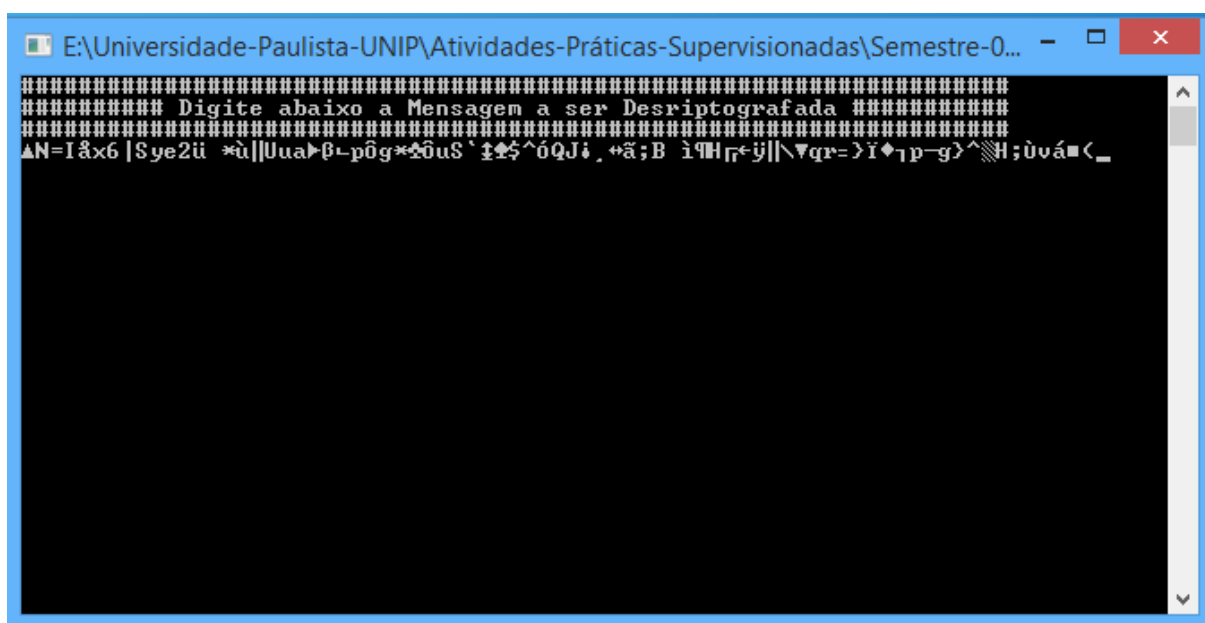
Fonte: Própria, 2015.

Imagem 23 – Printscreen da Mensagem Criptografada.



Fonte: Própria, 2015.

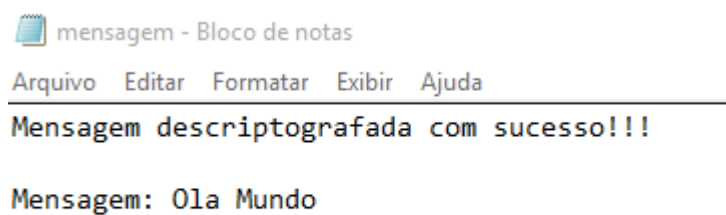
Imagem 24 – Printscreen da Mensagem Criptografada.



Fonte: Própria, 2015.

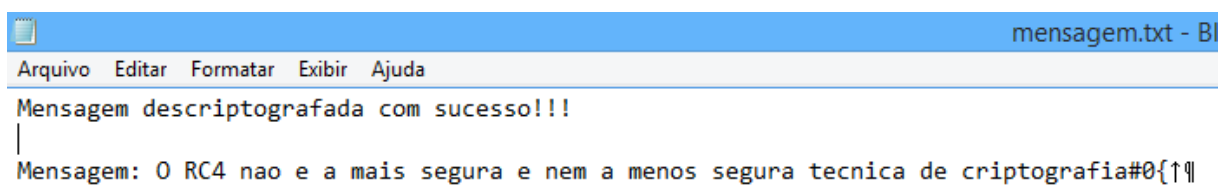
7.5 Printscreen da Mensagem Descriptografada

Imagem 25 – Mensagem Descriptografada.



Fonte: Própria, 2015.

Imagem 26 – Mensagem Descriptografada.



Fonte: Própria, 2015.

8 CONCLUSÃO

As redes de computadores, nunca tiveram tantos usuários ativos como hoje. A troca e armazenamento de dados via meios eletrônicos vêm aumentando. Muitos desses dados são informações sigilosas, as quais necessitam de proteção contra o acesso de terceiro não autorizados. A criptografia é uma das ferramentas utilizadas para prover a proteção desses dados. Ela nada mais é do que escrever secretamente, comunicar-se de maneira que nenhuma outra pessoa, a não ser o remetente e o destinatário, possa decifrar a mensagem. O objetivo da criptografia na transmissão de dados é codificar os dados a serem transmitidos de forma que fiquem ininteligíveis em eventuais espionagens. Pode ser combinada com outras tecnologias, ou utilizada isoladamente. Um dos instrumentos necessários para o uso de criptografia é o uso de algoritmos criptográficos. O exemplo de outros ramos da criptografia, eles também estão em constante evolução para atender o aumento das trocas seguras de informações via meios eletrônicos. Prova disso foi o esforço do NIST que, após décadas utilizando e recomendando o uso do cifrador DES, promoveu um concurso para selecionar um novo cifrador simétrico de blocos. Esse cifrador, denominado AES, veio para atender as novas exigências do mundo atual em termos de cifração simétrica de dados. É quase tão antiga quanto à escrita, pois, desde que as pessoas começaram a escrever, passaram a sentir a necessidade de esconder ou de não permitir que qualquer um entendesse o que foi escrito.

Um processo criptográfico envolve a aplicação de três conceitos elementares: a mensagem / texto, o algoritmo e a chave. A mensagem consiste, pura e simplesmente, na informação que se deseja transmitir de forma segura; o algoritmo é a forma que será utilizada para cifrar e decifrar uma mensagem; e a chave, em alguns modelos computacionais, pode ser entendida como o segredo compartilhado que deve ser conhecido apenas pelas duas partes envolvidas no processo de comunicação. É justamente em função da chave de cifração que os métodos de criptografia são classificados em duas categorias: chave simétrica e chave assimétrica.

Os profissionais diferenciam cifras e códigos. Cifra é uma transformação de caractere por caractere ou de bit por bit, sem levar em conta a estrutura linguística da mensagem e código substitui uma palavra por outra palavra ou símbolo. A cifra opera na sintaxe (símbolos) da mensagem, enquanto um código geralmente opera

na semântica (significado). Um código é armazenado como um mapeamento num livro de códigos, enquanto cifras transformam símbolos individuais conforme um algoritmo.

Este trabalho levantou os conceitos, aplicações e usos das técnicas criptográficas, mas especificamente, estudou o cifrador nomeado por RC4. Para tanto, foi feita uma revisão sobre criptografia, ataques criptográficos, modos de cifras e técnicas mais conhecidas.

A partir do algoritmo da criptografia RC4 que utiliza uma chave simétrica conseguimos criptografar e decifrar uma mensagem por inteiro preservando a segurança do programa e das informações inseridas nele, o programa se torna ainda mais potente dependendo do tamanho da chave utilizada, já que quanto maior a chave é mais difícil de descobri-la.

Este estudo envolveu análise de algumas obras publicadas por estudiosos especialistas, comparou também, o desempenho da RC4 com outros cifradores modernamente usados: RSA, Sosemanuk e AES. Como a RC4 não é tão segura como suas predecessoras há vários métodos que podem ser utilizados para torná-la mais eficiente este trabalho também implementou e propôs a inclusão de melhorias no RC4. Essas melhorias visam tornar o RC4 mais competitivo.

A primeira melhoria implementada foi a otimização. No algoritmo original, o texto simples ocupa muita memória, desnecessária, durante a execução do programa. A modificação implementada reduziu o tamanho de dados ocupados pelo texto simples e ainda após receber o texto simples e passar para a função de criptografia, o programa devolve a memória. Em comparação ao RSA, isso é fundamental, já que os resultados realizados nos cálculos ultrapassaram, no teste, a capacidade de armazenamento da variável da linguagem C, dando um erro de *overflow*.

Outra importante melhoria foi a criação de uma biblioteca. O uso dessa biblioteca deixou o programa mais modularizado. As vantagens que isso trouxe foram a redução de sequência de linhas de programação idênticas, relacionadas com uma única tarefa, aumentou a legibilidade escondendo pormenores poucos relevantes para a compreensão do problema e reduziu o tamanho e a complexidade do mesmo, dividindo-o em conjunto de subprogramas mais simples.

Outra melhoria implementada foi a capacidade de o programa criar, editar, apagar e abrir arquivos *.txt*. Deixando ele mais interativo com o usuário.

A primeira melhoria proposta foi na geração da chave. No algoritmo original, ele pede apenas uma chave. A modificação proposta iria pegar essa chave e criptografar ela, gerando duas chaves, uma pública e outra privada, com isso deixando o RC4 híbrida juntamente com a técnica RSA.

Outra melhora proposta foi a implementação do sistema caótico dimensional. Alterando inconvenientes da chave fraca, expandindo o ritmo fundamental do algoritmo original, e em seguida, faz com que a maioria dos métodos de atacar o RC4 não funcione, consequentemente aumentando a sua segurança.

Para avaliar a aplicação prática do RC4, foram realizados diversos testes de desempenho e confiabilidade. Todos os testes foram realizados em ambiente Windows. Os testes de desempenho foram aplicados também ao cifrador RSA, com o objetivo de comparar o RC4 com um cifrador utilizado atualmente.

Embora não tenha a melhor segurança, comparada ao RSA, 3DES, AES, Sosemanuk, Serpent, Twofish e One-Time-Pad, o RC4 demonstrou resultados significativos, superando as técnicas DES, César e RC2. Enquanto que no “custo x benefícios”, ele superou as técnicas citadas no texto, exceto o DES e a de César. Isso leva a crer que, se o sistema não é possuidor de uma grande capacidade de processamento o RC4 é a melhor alternativa para a implantação, por causa de sua flexibilidade, simplicidade e a facilidade de instalação em hardware e software.

Após a conclusão do programa, foi possível perceber que é muito importante escolher uma linguagem de programação eficiente, já que há um processamento considerado “robusto” que pode facilmente apresentar erros futuros. Temos como exemplo a técnica RSA, o cálculo para criptografar a mensagem resulta em um grande valor, que a variável da linguagem C não foi capaz de armazenar o mesmo, causando um overflow. A escolha da linguagem também afeta a interação máquina e humano, por haver um processo significativo para o homem conseguir copiar a mensagem criptografada, para mais tarde usar a cópia para descriptografar a mesma. Tudo isso ficaria mais interativo e prático com uma interface gráfica.

Diante de tudo que foi apresentado, conclui-se que a criptografia é um conceito totalmente necessário e importante nos meios de comunicação e informação que utilizamos hoje, já que há um crescente aumento no uso da internet e uma frequente troca de dados entre servidores e usuários que transmitem dados pessoais e necessitam de sigilo e segurança. Dependendo do ambiente, algumas técnicas criptográficas se sobressaem em relação a outras, mas se o proprietário

desejar ter um nível de segurança elevado terá que investir no ambiente em que a técnica de cifragem irá atuar. Acredita-se que com o contínuo avanço da tecnologia, as criptografias atuais tendem a ser tornarem obsoletas, como DES e César.

Por fim, trabalhos sendo realizados apontam os primeiros passos em direção à computação quântica. A fatoração de números primos grandes (algoritmo RSA) poderá ser feita em minutos, inviabilizando os atuais algoritmos com base matemática.

REFERÊNCIA BIBLIOGRÁFICA

CARUSO, C. A. A; STEFFEN, F. D. **Segurança em Informática e de Informações**. 2. ed. São Paulo: Senac, 1999. 367 p.

ABE, J. M; SCALZITTI, A; FILHO, J. I. S. **Introdução à Lógica para a Ciência da Computação**. 2. ed. São Paulo: Arte & Ciência, 2002. 247 p.

SOUSA, L. B. **Redes de Computadores Dados, Voz e Imagem**. 7. ed. São Paulo: Érica, 2002. 484 p.

TÂMEGA, F. **Hacker Inside Top Secret File 2**. 1. ed. Goiás: Terra Ltda., 2003. 88 p.

CARVALHO, L. G. **Segurança de Redes**. 1. ed. Rio de Janeiro: Ciência Moderna, 2005. 96 p.

STALLINGS, W. **Redes e Sistemas de Comunicação de Dados**. 5. ed. Rio de Janeiro: Campus, 2005. 439 p.

STALLINGS, W. **Criptografia e Segurança de Redes: Princípios e Práticas**. 4. ed. São Paulo: Pearson Education, 2007. 476 p.

NAKAMURA, E. T; GEUS, P. L. **Segurança de Redes em Ambientes Cooperativos**. 3. ed. São Paulo: Novatec, 2007. 483 p.

CARISSIMI, A. S; ROCHOL, J; GRANVILLE, L. Z. **Redes de Computadores – Vol. 20**. Rio Grande do Sul: Bookman, 2009. 391 p.

TENENBAUM, A. S; WETHERALL, D. **Redes de Computadores**. 5. ed. São Paulo: Pearson Education, 2011. 582 p.

WHITE, C. M. **Redes de Computadores e Comunicação de Dados**. 6. ed. São Paulo: Cengage Learning, 2012. 424 p.

PEIXINHO, I. C; FONSECA, F. M; LIMA, F. M. **Segurança de Redes e Sistemas**. 2. ed. Rio de Janeiro: Escola Superior de Redes RNP, 2013. 251 p.

MANZANO, J. A. N. G; OLIVEIRA, J. F. O. **Algoritmos: Lógica para Desenvolvimento de Programação de Computadores**. 27. ed. São Paulo: Érica, 2014. 328 p.

FORMOLO, D. **Criptografia com Caos**. São Leopoldo: Universidade do Vale do Rio dos Sinos, 2009. Disponível em: <<http://livros01.livrosgratis.com.br/cp106771.pdf>>. Acesso em 09 nov. 2015.

SEMENTE, R. S. **Otimização de Algoritmos Criptográficos para Redes de Sensores e Atuadores Sem-fio para Poços do Tipo Plunger Lift**. Rio Grande do Norte: Universidade Federal do Rio Grande do Norte, 2011. Disponível em: <<http://repositorio.ufrn.br/jspui/bitstream/123456789/15359/1/DISSERTACAO%20RODRIGO%20SOARES%20SEMENTE.pdf>>. Acesso em: 09 nov. 2015.

PALMA, S. M; PERREIRA, A. A. S. **Análise Crítica da Implementação da Cifra RC4 no Protocolo WEP**. São Paulo: Universidade Estadual de Campinas. Disponível em <<http://www.lasca.ic.unicamp.br/home/media/publications/1WEP.pdf>>. Acesso em 09 nov. 2015.

LINHARES, A. G; GONÇALVES, P. A. S. **Uma Análise dos Mecanismos de Segurança de Redes IEEE 802.11: WEP, WPA, WPA2 e IEEE 802.11w***. Pernambuco: Universidade Federal de Pernambuco. Disponível em: <<http://www.cin.ufpe.br/~pasg/gpublications/LiGo06.pdf>>. Acesso em 09 nov. 2015.

TRINTA, F. A. M; MACÊDO, R. C. **Um Estudo sobre Criptografia e Assinatura Digital**. Disponível em: <<http://www.di.ufpe.br/~flash/ais98/cripto/criptografia.htm>>. Acesso em: 10 nov. 2015.

INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO. **Criptografia**. Disponível em: <<http://www.dei.isep.ipp.pt/~andre/documentos/criptografia.html>>. Acesso em: 10 nov. 2015.

CANALTECH. **Google, Microsoft e Mozilla irão abandonar criptografia RC4 em seus navegadores**. Disponível em: <<http://canaltech.com.br/noticia/seguranca/google-microsoft-e-mozilla-irao-abandonar-criptografia-rc4-em-seus-navegadores-48481/>>. Acesso em: 10 nov. 2015.

DEVMEDIA. **Utilizando Criptografia Simétrica em Java**. Disponível em: <<http://www.devmedia.com.br/utilizando-criptografia-simetrica-em-java/31170>>. Acesso em: 10 nov. 2015.

9 FICHAS DE ATIVIDADES PRÁTICAS SUPERVISIONADAS

Ficha de cada membro da APS.

Imagem 27: Gabriel de Almeida Batista.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratório, atividades em biblioteca, avaliação científica, trabalhos individuais e em grupo, práticas de ensino e pesquisa)

NOME: Gabriel de Almeida Batista

RA: C4493F-1

CAMPUS: Tatuapé

CURSO: Ciências da Computação

SEMESTRE: 2º Semestre

TURNO: Manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
19/10	Caligrafia de textos	4	Gabriel Almeida	
20/10	Caligrafia de textos	4	Gabriel Almeida	
21/10	Caligrafia de textos	4	Gabriel Almeida	
22/10	Caligrafia de textos	4	Gabriel Almeida	
23/10	Caligrafia de textos	4	Gabriel Almeida	
24/10	Caligrafia de textos	4	Gabriel Almeida	
25/10	Caligrafia de textos	4	Gabriel Almeida	
26/10	Caligrafia de textos	4	Gabriel Almeida	
27/10	Dissertação	4	Gabriel Almeida	
28/10	Dissertação	4	Gabriel Almeida	
29/10	Dissertação	4	Gabriel Almeida	
30/10	Dissertação	4	Gabriel Almeida	
31/10	Dissertação	4	Gabriel Almeida	
01/11	Dissertação	4	Gabriel Almeida	
02/11	Dissertação	4	Gabriel Almeida	
03/11	Dissertação	4	Gabriel Almeida	
04/11	Dissertação	4	Gabriel Almeida	
05/11	Desenvolvimento de software	4	Gabriel Almeida	
06/11	Desenvolvimento de software	4	Gabriel Almeida	
07/11	Desenvolvimento de software	4	Gabriel Almeida	
08/11	Desenvolvimento de software	4	Gabriel Almeida	
09/11	Desenvolvimento de software	4	Gabriel Almeida	
10/11	Revisão Geral	4	Gabriel Almeida	
11/11	Revisão Geral	4	Gabriel Almeida	
12/11	Revisão Geral	4	Gabriel Almeida	

TOTAL DE HORAS: 100

Fonte: Própria, 2015.

Imagem 28: Felipe Ramos da Silva.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos individuais e em grupo, práticas de ensino e outras)

NOME: Felipe Ramos da Silva

RA: C55 JHD-8CURSO: Ciências da Computação

CAMPUS: ItaquapeSEMESTRE: 2º SemestreTURNO: Manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
19/10	Calisto de Jhon	4	Felipe Ramos	
20/10	Calisto de Jhon	4	Felipe Ramos	
21/10	Calisto de Jhon	4	Felipe Ramos	
22/10	Calisto de Jhon	4	Felipe Ramos	
23/10	Calisto de Jhon	4	Felipe Ramos	
24/10	Calisto de Jhon	4	Felipe Ramos	
25/10	Calisto de Jhon	4	Felipe Ramos	
26/10	Calisto de Jhon	4	Felipe Ramos	
27/10	Disertação	4	Felipe Ramos	
28/10	Disertação	4	Felipe Ramos	
29/10	Disertação	4	Felipe Ramos	
30/10	Disertação	4	Felipe Ramos	
31/10	Disertação	4	Felipe Ramos	
01/11	Disertação	4	Felipe Ramos	
02/11	Disertação	4	Felipe Ramos	
03/11	Disertação	4	Felipe Ramos	
04/11	Desenvolvimento de Software	4	Felipe Ramos	
05/11	Desenvolvimento de Software	4	Felipe Ramos	
06/11	Desenvolvimento de Software	4	Felipe Ramos	
07/11	Desenvolvimento de Software	4	Felipe Ramos	
08/11	Desenvolvimento de Software	4	Felipe Ramos	
09/11	Desenvolvimento de Software	4	Felipe Ramos	
10/11	Revisão Geral	4	Felipe Ramos	
11/11	Revisão Geral	4	Felipe Ramos	
12/11	Revisão Geral	4	Felipe Ramos	

TOTAL DE HORAS: 100

Fonte: Própria, 2015.

Imagem 29: Felipe da Silva Borges Neves.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos individuais e em grupo, práticas de ensino e outras)

NOME: Felipe da Silva Borges Neves

RA: 049770-3CURSO: Ciências da computação

CAMPUS: ItatupeSEMESTRE: 2º semestreTURNO: manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
19/10	Coleta de dados	4	Felipe Borges	
20/10	Coleta de dados	4	Felipe Borges	
21/10	Coleta de dados	4	Felipe Borges	
22/10	Coleta de dados	4	Felipe Borges	
23/10	Coleta de dados	4	Felipe Borges	
24/10	Coleta de dados	4	Felipe Borges	
25/10	Coleta de dados	4	Felipe Borges	
26/10	Coleta de dados	4	Felipe Borges	
27/10	Desenvolvimento	4	Felipe Borges	
28/10	Desenvolvimento	4	Felipe Borges	
29/10	Desenvolvimento	4	Felipe Borges	
30/10	Desenvolvimento	4	Felipe Borges	
31/10	Desenvolvimento	4	Felipe Borges	
01/11	Desenvolvimento	4	Felipe Borges	
02/11	Desenvolvimento	4	Felipe Borges	
03/11	Desenvolvimento	4	Felipe Borges	
04/11	Desenvolvimento de software	4	Felipe Borges	
05/11	Desenvolvimento de software	4	Felipe Borges	
06/11	Desenvolvimento de software	4	Felipe Borges	
07/11	Desenvolvimento de software	4	Felipe Borges	
08/11	Desenvolvimento de software	4	Felipe Borges	
09/11	Desenvolvimento de software	4	Felipe Borges	
10/11	Revisão Geral	4	Felipe Borges	
11/11	Revisão Geral	4	Felipe Borges	
12/11	Revisão Geral	4	Felipe Borges	

TOTAL DE HORAS: 100

Fonte: Própria, 2015.

Imagem 30: Felipe Corniani de Genaro.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação Científica, trabalhos individuais e em grupo, práticas de ensino e outras)

NOME: Felipe corniani de Genaro

RA: C444 IF - G

CAMPUS: Itaquape

CURSO: Ciência da computação

SEMESTRE: 2º

TURNO: manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
19/10	Coleta de dados	4	Felipe Corniani	
20/10	Coleta de dados	4	Felipe Corniani	
21/10	Coleta de dados	4	Felipe Corniani	
22/10	Coleta de dados	4	Felipe Corniani	
23/10	Coleta de dados	4	Felipe Corniani	
24/10	Coleta de dados	4	Felipe Corniani	
25/10	Coleta de dados	4	Felipe Corniani	
26/10	Coleta de dados	4	Felipe Corniani	
27/10	Coleta de dados	4	Felipe Corniani	
28/10	Dinâmica	4	Felipe Corniani	
29/10	Dinâmica	4	Felipe Corniani	
30/10	Dinâmica	4	Felipe Corniani	
31/10	Dinâmica	4	Felipe Corniani	
01/11	Dinâmica	4	Felipe Corniani	
02/11	Dinâmica	4	Felipe Corniani	
03/11	Dinâmica	4	Felipe Corniani	
04/11	Desenvolvimento de software	4	Felipe Corniani	
05/11	Desenvolvimento de software	4	Felipe Corniani	
06/11	Desenvolvimento de software	4	Felipe Corniani	
07/11	Desenvolvimento de software	4	Felipe Corniani	
08/11	Desenvolvimento de software	4	Felipe Corniani	
09/11	Desenvolvimento de software	4	Felipe Corniani	
10/11	Revisão Geral	4	Felipe Corniani	
11/11	Revisão Geral	4	Felipe Corniani	
12/11	Revisão Geral	4	Felipe Corniani	

TOTAL DE HORAS: 100

Imagem 31: Luís Henrique Giusepin Alonso.

FICHA DE ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

Atividades Práticas Supervisionadas (laboratórios, atividades em biblioteca, iniciação científica, trabalhos individuais e em grupo, práticas de ensino e outras)

NOME: Luís Henrique Giusepin Alonso

RA: C43CF9-8CURSO: Ciência da Computação

CAMPUS: TatuapéSEMESTRE: 2ºTURNO: Manhã

DATA	ATIVIDADE	TOTAL DE HORAS	ASSINATURA	
			ALUNO	PROFESSOR
19/10	Coleta de dados	4	Luís Henrique	
20/10	Coleta de dados	4	Luís Henrique	
21/10	Coleta de dados	4	Luís Henrique	
22/10	Coleta de dados	4	Luís Henrique	
23/10	Coleta de dados	4	Luís Henrique	
24/10	Coleta de dados	4	Luís Henrique	
25/10	Coleta de dados	4	Luís Henrique	
26/10	Coleta de dados	4	Luís Henrique	
27/10	Desenvolvimento	4	Luís Henrique	
28/10	Desenvolvimento	4	Luís Henrique	
29/10	Desenvolvimento	4	Luís Henrique	
30/10	Desenvolvimento	4	Luís Henrique	
31/10	Desenvolvimento	4	Luís Henrique	
01/11	Desenvolvimento	4	Luís Henrique	
02/11	Desenvolvimento	4	Luís Henrique	
03/11	Desenvolvimento	4	Luís Henrique	
04/11	Desenvolvimento de software	4	Luís Henrique	
05/11	Desenvolvimento de software	4	Luís Henrique	
06/11	Desenvolvimento de software	4	Luís Henrique	
07/11	Desenvolvimento de software	4	Luís Henrique	
08/11	Desenvolvimento de software	4	Luís Henrique	
09/11	Revisão Geral	4	Luís Henrique	
10/11	Revisão Geral	4	Luís Henrique	
11/11	Revisão Geral	4	Luís Henrique	
12/11	Revisão Geral	4	Luís Henrique	

TOTAL DE HORAS: 100

Fonte: Própria, 2015.

ANEXOS

ANEXO A – ALGORITMO DO PROGRAMA

programa Criptografia

var

key : conjunto [0...256] de caracteres;
 s : conjunto [0...256] de caracteres;
 chiper : conjunto [0...256] de caracteres;
 i, j, tamanhoKey, tamanhoPlainText : inteiro;

função menu() : inteiro;

procedimento troca (s : conjunto [0...256] de caracteres, i, j : inteiro);

procedimento ksa(i, j, tamanhoKey : inteiro);

procedimento prga(plain : conjunto [0...256] de caracteres, i, j, tamanhoPlainText : inteiro);

função chiperText(plain : conjunto [0...256] de caracteres, key1 : conjunto [0...256] de caracteres, i, j, tamanhoKey, tamanhoPlainText : inteiro) : conjunto [0...256] de caracteres;

função plainText(plain : conjunto [0...256] de caracteres, key1 : conjunto [0...256] de caracteres, i, j, tamanhoKey, tamanhoPlainText : inteiro) : conjunto [0...256] de caracteres;

var

plain1 : conjunto [0...256] de caracteres;
 key1 : conjunto [0...256] de caracteres;
 i1, j1, tamanhoPlainText1, tamanhoKey1 : inteiro;

inicio

escolha(menu())

caso 1 :

```

        escreva

("#####
###\n");

        escreva ("##### Digite abaixo a Mensagem a ser
Criptografada #####\n");

        escreva

("#####
###\n");

        leia (plain);
        tamanhoPlainText1 <-- strlen(plain);

        escreva

("#####
###\n");

        escreva ("##### Digite abaixo a Chave
Simetrica #####\n");

        escreva

ts("#####
###\n");

        leia (key);
        tamanhoKey <-- strlen(key);

        escreva

("#####
###\n");

        escreva      ("#####      Mensagem
Criptografada #####\n");

        escreva

("#####
###\n");

        escreva (chiperText(plain, key, i, j, tamanhoKey,
tamanhoPlainText));

```

escreva

```
("#####  
###\n");
```

escreva ("##### Aperte qualquer tecla para voltar
ao menu #####\n");

escreva

```
("#####  
###\n");
```

caso 2 :

escreva

```
("#####  
###\n");
```

escreva ("##### Digite abaixo a Mensagem a ser
Descriptografada #####\n");

escreva

```
("#####  
###\n");
```

leia (plain);

tamanhoPlainText <-- strlen(plain);

escreva

```
("#####  
###\n");
```

escreva ("##### Digite abaixo a Chave
Simetrica #####\n");

escreva

```
("#####  
###\n");
```

leia (key);

tamanhoKey <-- strlen(key);

```

        escreva
("#####
###\n");

        escreva      ("#####      Mensagem
Descriptografada #####\n");

        escreva
("#####
###\n");

        escreva (plainText(plain, key, i, j, tamanhoKey,
tamanhoPlainText), texto);

        escreva ("##### Aperte qualquer tecla para voltar
ao menu #####\n");

        escreva
("#####
###\n");

        senão
        escreva ("\\t\\t Opcao Invalida!!!");
        escreva
("#####
###\n");

        fim-escolha

    fim

    função menu() : inteiro
        inicio
            r : inteiro;

            escreva
("#####
###\n");

            escreva ("##### Criptografia RC4 + MFC +
RSA #####\n");

```



```

        escreva
("#####
###\n");

        escreva ("##### Escolha uma das opcoes
abaixo #####\n");

        escreva
("#####
###\n");

        escreva ("##### \t1 - Criptografar\t
#####\n");

        escreva ("##### \t2 - Descriptografar\t
#####\n");

        escreva
("#####
###\n");

        escreva ("\t\t Opcao: ");
        leia (r);
        escreva
("#####
###\n");

        retorne (r);

    fim

procedimento troca (s : conjunto [0...256] de caracteres, i, j : inteiro)
    inicio
        s [i] <-- s [i] + s [j];
        s [j] <-- s [i] - s [j];
        s [i] <-- s [i] - s [j];
    fim

procedimento ksa(i, j, tamanhoKey : inteiro)
    inicio
        para i de 0 até 256 passo 1 faça

```

```

        s [i] <-- i;
    fim-para

    j <-- 0;

    para i de 0 até 256 passo 1 faça
        j <-- (j + s [i] + key [i mod tamanhoKey]) mod 256;
        troca(s, i, j);
    fim-para

    i <-- 0;
    j <-- 0;

fim

```

procedimento prga(plain : conjunto [0...256] de caracteres, i, j, tamanhoPlainText : inteiro)

```

    var
        aux : inteiro;
        result : conjunto [0...tamanhoPlainText - 1];
    inicio
        para aux de 0 até tamanhoPlainText passo 1 faça
            i <-- (i + 1) mod 256;
            j <-- (j + s [i]) mod 256;
            troca(s, i, j);
            result [aux] <-- (s [(s [i] + s [j]) mod 256]) xor plain [aux];
        fim-para

        strcpy(chiper, result);

    fim

```

função chiperText(plain : conjunto [0...256] de caracteres, key1 : conjunto [0...256] de caracteres, i, j, tamanhoKey, tamanhoPlainText : inteiro) : conjunto [0...256] de caracteres

```

    var

```

```

        cont : inteiro;
inicio
    cont <-- 0;
    enquanto cont < 256 faça
        key [cont] <-- key [cont]
        cont++;
    fim-enquanto

    ksa(i, j, tamanhoKey);
    prga(plain, i, j, tamanhoPlainText);

    retorne (chiper);
fim

```

função plainText(plain : conjunto [0...256] de caracteres, key1 : conjunto [0...256] de caracteres, i, j, tamanhoKey, tamanhoPlainText : inteiro) : conjunto [0...256] de caracteres

```

    var
        cont : inteiro;
    inicio
        cont <-- 0;
        enquanto (cont < 256) faça
            key [cont] <-- key [cont]
            cont++;
        fim-enquanto

        ksa(i, j, tamanhoKey);
        prga(plain, i, j, tamanhoPlainText);

        retorne (chiper);
    fim

```

ANEXO B – CÓDIGO DO ARQUIVO MAIN.C

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "rciv.h"

short *menu();

unsigned char *chiperText(unsigned char *, unsigned char *, unsigned short *,
unsigned short *, unsigned short *, unsigned short *);

unsigned char *plainText(unsigned char *, unsigned char *, unsigned short *,
unsigned short *, unsigned short *, unsigned short *);

int main(int argc, char *argv[])
{
    unsigned char *plain = (unsigned char *) calloc(256, sizeof(unsigned
char));

    unsigned char *key = (unsigned char *) calloc(256, sizeof(unsigned
char));

    unsigned short *i = (unsigned short *) calloc(1, sizeof(unsigned short));
    unsigned short *j = (unsigned short *) calloc(1, sizeof(unsigned short));
    unsigned short *tamanhoPlainText = (unsigned short *) calloc(1,
sizeof(unsigned short));

    unsigned short *tamanhoKey = (unsigned short *) calloc(1,
sizeof(unsigned short));

    FILE *texto = fopen("mensagem.txt", "w");
    fputs("Mensagem descriptografada com sucesso!!!\n\nMensagem: ",
texto);

    fclose(texto);

    switch((int) menu())
    {
        case 0:
            free(plain);

```

```

        free(key);
        free(i);
        free(j);
        free(tamanhoKey);
        free(tamanhoPlainText);
        exit(0);
    break;

    case 1:
        system("cls");
        fflush(stdin);

        puts("#####
#####");

        puts("##### Digite abaixo a Mensagem a ser
Criptografada #####");

        puts("#####
#####");

        gets(plain);
        *tamanhoPlainText = strlen(plain);
        fflush(stdin);

        puts("#####
#####");

        puts("##### Digite abaixo a Chave
Simetrica #####");

        puts("#####
#####");

        gets(key);
        fflush(stdin);

```

```
*tamanhoKey = strlen(key);
```

```
puts("#####  
#####");
```

```
puts("##### Mensagem Criptografada  
#####");
```

```
puts("#####  
#####");
```

```
puts(chiperText(plain, key, i, j, tamanhoKey,  
tamanhoPlainText));
```

```
puts("#####  
#####");
```

```
puts("##### Aperte qualquer tecla para voltar ao  
menu #####");
```

```
puts("#####  
#####");
```

```
getch();
```

```
free(plain);
```

```
free(key);
```

```
free(i);
```

```
free(j);
```

```
free(tamanhoKey);
```

```
free(tamanhoPlainText);
```

```
free(texto);
```

```
system("start rc4");
```

```
exit(0);
```

```
break;
```

```
case 2:
```

```
system("cls");
```

```
puts("#####  
#####");
```

```
puts("##### Digite abaixo a Mensagem a ser  
Descriptografada #####");
```

```
puts("#####  
#####");
```

```
gets(plain);
```

```
*tamanhoPlainText = strlen(plain);
```

```
fflush(stdin);
```

```
puts("#####  
#####");
```

```
puts("##### Digite abaixo a Chave  
Simetrica #####");
```

```
puts("#####  
#####");
```

```
gets(key);
```

```
fflush(stdin);
```

```
*tamanhoKey = strlen(key);
```

```
puts("#####  
#####");
```

```
puts("##### Mensagem  
Descriptografada #####");
```

```
puts("#####  
#####");
```

```

        FILE *texto = fopen("mensagem.txt", "a+");
        fputs(plainText(plain, key, i, j, tamanhoKey,
tamanhoPlainText), texto);
        fclose(texto);
        system("start mensagem.txt");

        puts("##### Aperte qualquer tecla para voltar ao
menu #####");

        puts("#####
#####");

        getch();
        free(plain);
        free(key);
        free(i);
        free(j);
        free(tamanhoKey);
        free(tamanhoPlainText);
        free(texto);
        system("start rc4");
        exit(0);
    break;

    case 3:
        system("del mensagem.txt");

        puts("##### Arquivos apagados com
sucesso !!! #####");

        puts("#####
#####");

        puts("##### Aperte qualquer tecla para voltar ao
menu #####");

```



```
puts("#####  
#####");
```

```
    fflush(stdin);  
    free(plain);  
    free(key);  
    free(i);  
    free(j);  
    free(tamanhoKey);  
    free(tamanhoPlainText);  
    free(texto);  
    getch();  
    system("cls");  
    system("start rc4");  
    exit(0);  
break;  
  
default:  
    puts("\t\t\t Opcao Invalida!!!");
```

```
puts("#####  
#####\n");
```

```
    fflush(stdin);  
    free(plain);  
    free(key);  
    free(i);  
    free(j);  
    free(tamanhoKey);  
    free(tamanhoPlainText);  
    free(texto);  
    getch();  
    system("cls");
```

```

        main(0, NULL);
        break;
    }

    return 0;
}

```

ANEXO C – CÓDIGO DO ARQUIVO RCIV.H

```

unsigned char key[256];
unsigned char s[256];
unsigned char *chiper;
unsigned int i, j, tamanhoKey, tamanhoPlainText;

void troca(unsigned char *s, unsigned short *i, unsigned short *j)
{
    s[*i] = s[*i] + s[*j];
    s[*j] = s[*i] - s[*j];
    s[*i] = s[*i] - s[*j];
}

void ksa(unsigned short *i, unsigned short *j, unsigned short *tamanhoKey)
{
    for (*i = 0; *i < 256; (*i)++)
    {
        s[*i] = *i;
    }

    *j = 0;

    for (*i = 0; *i < 256; (*i)++)
    {
        *j = (*j + s[*i] + key[*i % *tamanhoKey]) % 256;
        troca(s, i, j);
    }
}

```

```

    }

    *i = 0;
    *j = 0;
}

void prga (unsigned char *plain, unsigned short *i, unsigned short *j, unsigned
short *tamanhoPlainText)
{
    unsigned int aux;
    unsigned char result[*tamanhoPlainText-1];

    for (aux = 0; aux < *tamanhoPlainText; aux++)
    {
        *i = (*i + 1) % 256;
        *j = (*j + s[*i]) % 256;
        troca(s, i, j);
        result[aux] = (s[(s[*i] + s[*j]) % 256]) ^ *(plain + aux);
    }

    chiper = (unsigned char*) calloc((*tamanhoPlainText - 1), (sizeof(unsigned
char)));
    strcpy(chipper, result);
}

unsigned char *chipertext(unsigned char *plain, unsigned char *key1,
unsigned short *i, unsigned short *j, unsigned short *tamanhoKey, unsigned short
*tamanhoPlainText)
{
    int cont = 0;

    while(cont < 256){
        key[cont] = *(key1 + cont);
        cont++;
    }

```

```

    }

    ksa(i, j, tamanhoKey);
    prga(plain, i, j, tamanhoPlainText);

    return(chiper);
}

unsigned char *plainText(unsigned char *plain, unsigned char *key1, unsigned
short *i, unsigned short *j, unsigned short *tamanhoKey, unsigned short
*tamanhoPlainText)
{
    int cont = 0;

    while(cont < 256){
        key[cont] = *(key1 + cont);
        cont++;
    }

    ksa(i, j, tamanhoKey);
    prga(plain, i, j, tamanhoPlainText);

    return(chiper);
}

short *menu()
{
    short *r = (short *) calloc(1, sizeof(short*));

    puts("#####");
    puts("#####");

```

```

        puts("##### Criptografia RC4 + MFC + RSA
#####");

puts("#####");

        puts("##### Escolha uma das opcoes abaixo
#####");

puts("#####");

        puts("##### \t0 - Sair\t\t #####");
        puts("##### \t1 - Criptografar\t #####");
        puts("##### \t2 - Descriptografar\t
#####");
        puts("##### \t3 - Apagar Arquivos\t
#####");

puts("#####");

        printf("\t\t Opcao: ");
        scanf("%d", &r);

puts("#####");

        fflush(stdin);

        return r;
        free(r);
    }

```