UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Diogenes Lemos Carneiro

Um estudo sobre a aplicabilidade de redes neurais em criptografia

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação.

Prof. Dr. Mauro Roisenberg

Florianópolis, Dezembro de 2001

Um estudo sobre a aplicabilidade de redes neurais em criptografia

Diogenes Lemos Carneiro

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.

_____

Prof. Dr. Mauro Roisenberg

Banca Examinadora

_____

Prof. Dr. João Bosco da Mota

_____

Prof. Dr. Jorge Muniz Barreto

“Divulga o beneficio recebido, demonstrando, porém, maior agradecimento, quem se esquece do benefício para lembrar-se, unicamente, do benfeitor.”

Borne

À minha esposa Piedade, e a meus filhos Stephanie e
Dio que com compreensão me ajudaram muito
durante este período.

Agradecimentos

À CESUPA, na pessoa de seu diretor geral Sérgio Mendes, que teve coragem de dar vazão a uma demanda reprimida de conhecimento em nosso estado.

Aos professores Maia, Mauro, Rosvelter, Silvia, Bosco, Elizabeth, Paulo Borges e Paulo Freitas, que com seus conhecimentos e didática transformaram este período de minha vida em um dos mais profícuos.

Também é hora de agradecer o esforço e dedicação com que o professor Gustavo conduziu a difícil missão de coordenar nosso mestrado.

Lembrar também os amigos Eugênio Pessoa, Mauro Cavalcante, Luiz e Lidio que sempre estiveram junto comigo e todos que me ajudaram em sala de aula ou fora dela, é imprescindível.

Há muito tinha um amigo, que filosofava na pessoa de sua avó, e dizia : “a gratidão é a memória do coração”. É com este sentimento que me dirijo a meu orientador, professor e amigo Mauro, que não poupou esforços para tornar a difícil caminhada até aqui menos pesada, sempre com palavras de incentivo, carinho e atenção.

Agradecer a Deus que todo dia nos acompanha nas pequenas e grandes coisa da vida.

Sumário

Figuras

Resumo

Diversos dispositivos eletrônicos conectados uns aos outros e o crescimento exponencial da internet levam à necessidade de prover segurança neste processo de comunicação. Uma das formas de melhorar a confiança dos usuários é a implementação de algoritmos criptográficos eficientes.

Inspirado no perfeito processo da comunicação entre as células do sistema nervoso, implementa-se três propostas de cifradores, utilizando redes neurais artificias. E para completar tal estudo, procura-se realizar a criptoanálise dessas propostas e do RSA, utilizando também as redes neurais.

Palavras Chaves : Redes neurais artificiais, criptografia e criptoanálise.

Abstract

Several connected electronic devices each other and the exponential growth of the internet bring the necessity to provide safety in this communication process, one way of improving the users' trust is the cryptography algorithms.

Inspired in the perfect process of the communication among cells of the nervous system, is implemented three proposed of cipher algorithms, using articial neural network. To complete this study, is tried to accomplish the cryptanalysis of those proposals and of RSA, using also the articial neural network.

Key words: articial neural network, cryptography and cryptanalysis.

Capítulo 1

Introdução

1.1 Motivação

No mundo moderno, a utilização de diversos dispositivos eletrônicos, que se comunicam entre si, é cada vez mais freqüente, e a necessidade de que a comunicação seja feita de uma forma segura cresce na razão direta desta expansão. A internet também é responsável por um crescimento exponencial do anseio coletivo por privacidade e segurança na troca de mensagens.

A grande rede é o meio mais democrático e global, porém as distorções do mundo real passaram a fazer parte do seu cotidiano, pois fraudes, roubos, clonagem de cartões de crédito etc. são noticiados pela imprensa nacional e internacional. Logo, codificar - criptografar - o conteúdo das mensagens trocadas em meio inseguro, para que os intrusos não consigam entender o conteúdo das mesmas, é de vital importância, principalmente para e-commerce – comércio eletrônico entre empresas (B2B – Business to Business) e entre empresas e consumidores (B2C – Business to consumer), e-goverment – os governos disponibilizando diversos serviços para os cidadãos , o e-learn e o e-training – universidades, escolas e empresas voltadas para o treinamento, disponibilizando a oportunidade de aprendizado a distância.

É bastante comum ouvir perguntas do tipo: Você faz compras na internet ? Paga com seu cartão de crédito ? Sua empresa solicita uma certidão da Receita Federal pela internet ? Você acredita num curso de inglês pela internet? As respostas a estas indagações serão mais positivas quanto maior for a confiança da sociedade na segurança que os algoritmos de criptografia e outras medidas de segurança poderem propiciar.

No período de aulas do mestrado, foram ministradas disciplinas de inteligência artificial e de redes de computadores, fato que levou a procura por um tema que permitisse mesclar, de forma balanceada, o conhecimento das duas áreas. O grande interesse por redes neurais surgiu, pois, no início do curso.

Ao observar a estrutura e a funcionalidade de um neurônio, percebe-se que ele possui um perfeito processo de comunicação com as outras células do sistema nervoso e partindo desta análise, visualiza-se a possibilidade de realizar um estudo sobre a aplicabilidade de redes neurais artificiais em criptografia. Antes, porém, necessário se torna uma breve explicação sobre algumas característica dos atuais cifradores que utilizam números primos muito grandes para gerar suas chaves, centrais de distribuição de senhas e muito tempo de processamento. Por outro lado o paralelismo observado na arquitetura das redes neurais pode perfeitamente facilitar sua utilização nas funções de criptografia.

Como principal objetivo desta pesquisa tem-se a criação de algumas propostas de cifradores – algoritmos utilizados para codificar mensagens – partindo da imagem do neurônio biológico e tecendo um paralelo com o neurônio artificial.

Também, faz-se o estudo da utilização das redes neurais artificias para realizar ataques[1] a algoritmos criptograficos de ampla utilização na atualidade. A importância da criptoanálise tem sido muito grande , influenciando inclusive o avanço da computação.

1.2 – Organização do texto

O 2° capítulo é dedicado aos conceitos de redes neurais artificiais, os quais são abordados de forma simples, propiciando uma visão geral desta importante ferramenta da inteligência artificial.

No 3° capítulo apresentam-se os conceitos de criptografia, esteganografia e criptoanálise. Também dissecou-se os algoritmos DES – Data Ecryptation Standard e o RSA – a primeira letra dos nomes dos criadores do algoritmo Ronald **R**ivest, Adi **S**hamir e Len **A**dleman- , muito utilizados nos dias de hoje.

O capítulo 4° é totalmente dedicado ao experimento no que tange à criação de cifradores, que inspirados no biológico, procuram uma nova maneira de criptografar e prover mais segurança na comunicação entre dispositivos eletrônicos.

[1] Também conhecidos como criptoanálise, a ciência para reaver o texto plano sem ter acesso à chave, ou descobrir a chave, ou ainda mostrar o quão fraco o sistema de criptografia é (STALLINGS, 1999)

No capítulo 5° ,procurou-se mostrar os ataques feitos ao algoritmos RSA e aos cifradores propostos neste trabalho, utilizando redes neurais artificiais diretas e recorrentes, descrevendo de forma minuciosa todos os ataques e avaliando os resultados obtidos.

No último capítulo, teceu-se considerações finais sobre essa pesquisa e foram apresentadas sugestões para trabalhos futuros.

Capítulo 2

Redes Neurais Artificiais

2.1 – O Biológico

O cérebro é a base do sistema nervoso e um dos grandes mistérios da ciência. Em 335 a.C. Aristóteles escreveu que “o ser humano tem o maior cérebro de todos os animais”, porém hoje se sabe que esta afirmação não é verdadeira, pois algumas espécies de golfinhos e baleias têm o cérebro maior.

A célula funcional básica do cérebro humano - sistema nervoso - é o neurônio. Assim como as outras células, os neurônios se alimentam, respiram, têm os mesmos genes, os mesmos mecanismos bioquímicos e as mesmas organelas. O que os difere das outras células é que eles são capazes de processar informações sobre o estado interno do organismo e seu ambiente externo, avaliar esta informação e coordenar atividades apropriadas à situação e às necessidades correntes das pessoas (CARDOSO, 2001). FIGURA 2.1.

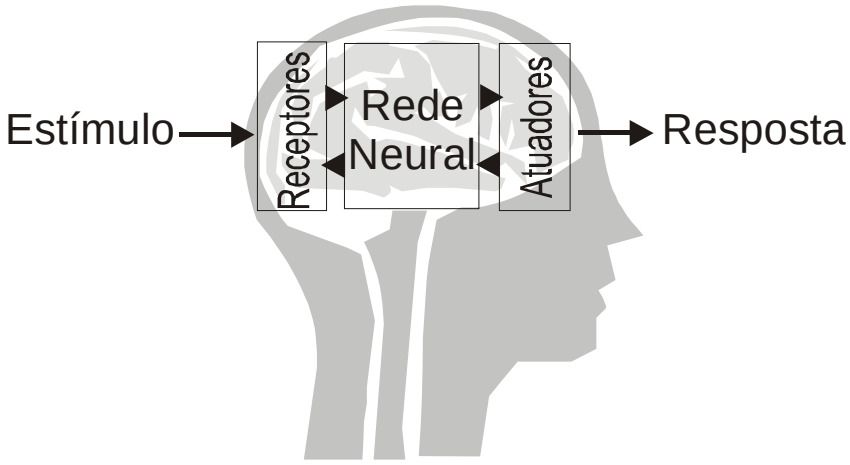

FIGURA 2.1 – Representação em diagrama de bloco
do sistema nervoso (HAYKIN, 2001)

As partes do neurônio são o corpo celular ou soma, responsável pela produção de proteínas para as outras partes do neurônio; os dentritos, estruturas ramificadas como galhos de árvores, que servem como principal aparato para receber os sinais de outras células nervosas; o Axônio, a mais longa estrutura celular e também a principal unidade condutora do neurônio; e o terminal nervoso (terminal pré-sináptico), estruturas em que ocorrem as sinapses que são as junções das células do sistema nervoso, e é nestas junções que os neurônios são excitados, inibidos ou modulados. Existem dois tipos de sinapses: a elétrica e a química (CARDOSO, 2001) . FIGURA 2.2

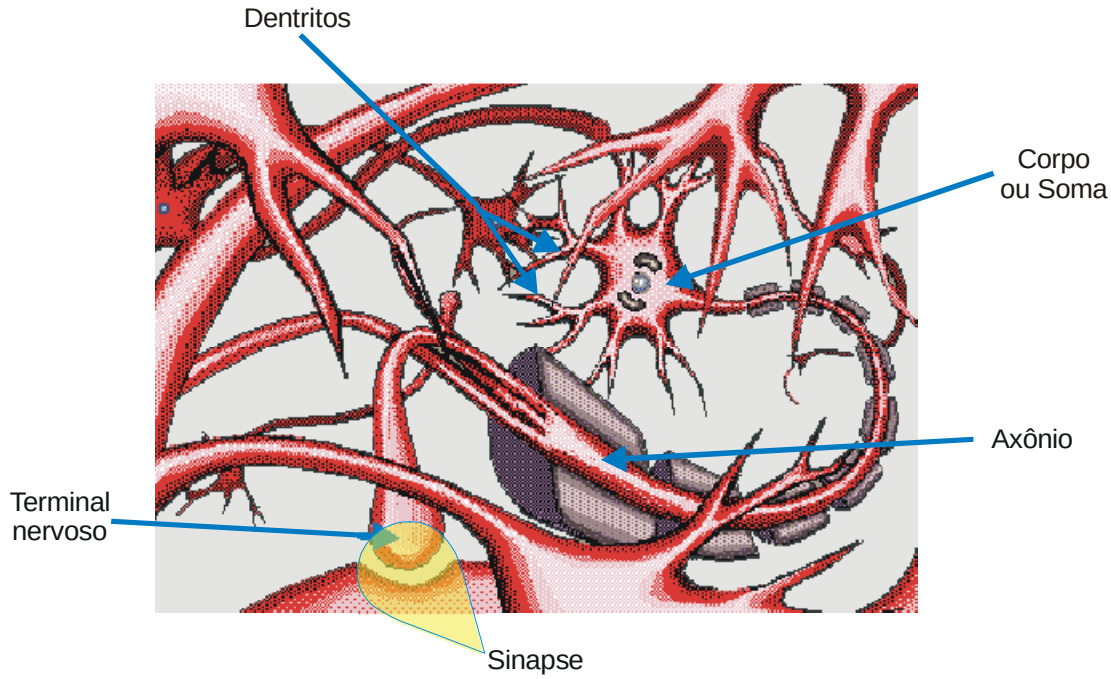

FIGURA 2.2 – Célula Nervosa (MICROSOFT, 1999)

2.2 – O Artificial

Os estudos de McCulloch e Pitts (1943) e o Perceptron de Frank Rosemblat (1958) foram a origem das redes neurais artificiais. As redes neurais artificiais são inspiradas no sistema nervoso biológico, porém com as limitações do conhecimento do natural e com as limitações tecnológicas.

No neurônio artificial, as entradas são os dentritos, os pesos as sinapses e a saída o axônio. FIGURA 2.3.

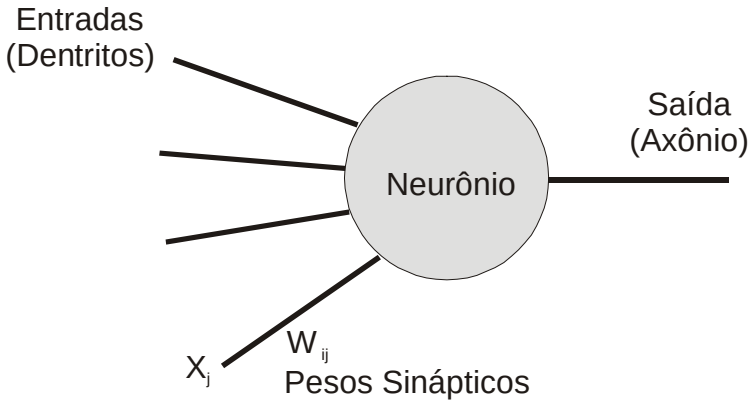

FIGURA 2.3 – Primeiro modelo de neurônio artificial

Este modelo foi criticado por Minky e Papert (1969), porque só seria capaz de resolver problemas linearmente separáveis, porém, no mesmo livro, eles afirmam que a colocação de mais uma camada de neurônios resolveria esta limitação, embora sem saber como. Após um longo período, coube a Paul Werbos (1974) lançar as bases do algoritmo de retro-propagação (backpropagation), que permitiu a utilização de redes neurais artificiais com múltiplas camadas (BARRETO, 1999).

2.2.1 – O neurônio artificial básico

É possível descrever o funcionamento de diversos neurônios artificiais a partir da definição formal do mesmo, bastando particularizar os parâmetros que o definem. Também é importante a escolha da função de transição de estado Φ e a função de saída λ e a forma de combinar os valores de entradas dos neurônios.

As partes de um neurônio artificial são as entradas, que podem ser as saídas de outros neurônios, entradas externas, bias ou qualquer combinação destes elementos; o net que é o somatório de todas as entradas, multiplicadas por suas respectivas forças de conexões sinápticas (pesos); função de transferência (saída) que atualiza o valores após a determinação do net, produzindo ou não uma valor em determinada escala. FIGURA 2.4.

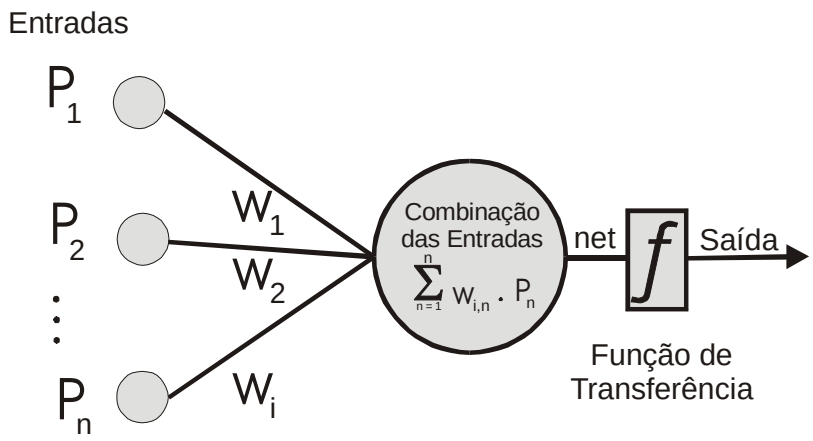

FIGURA 2.4 – Entradas, pesos, net e função de transferência

2.2.2 – Funções de transferência

As funções contínuas tal que x e y(x) ∈ ℜ, e as monotônicas crescentes tal que x ∈ ℜ e y(x) ∈ [-1,1], podem ser usadas como funções de transferência na modelagem neural. Existem algumas funções que são mais utilizadas: função linear, função sigmóide ou logística e a função tangente hiperbólica.

2.2.2.1 – Função linear

São todas as funções onde $y(x) = ax \in \Re$, sendo "a" um número real não-nulo qualquer. FIGURA 2.5.

$x \in \Re$

$y(x) = ax \in \Re$

$a \neq 0$

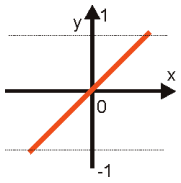

Figura 2.5 – Função linear

2.2.2.2 – Função sigmóide

É a função unipolar mais utilizada na modelagem neural, cujo gráfico tem a forma de "S" , onde $x \in \Re$ e $y(x) \in [0,1]$. FIGURA 2.6.

$x \in \Re$

$y(x) \in [0,1]$

$$y(x) = \frac{1}{1 + e^{-kx}}$$

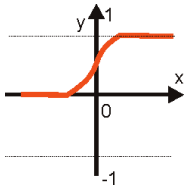

Figura 2.6 – Função Sigmóide

2.2.2.3 – Função tangente hiperbólica

É a função bipolar mais utilizada na modelagem neural, cujo o gráfico também tem a forma de "S" , onde $x \in \Re$ e $y(x) \in [-1,1]$. FIGURA 2.7.

$$x \in \Re$$

$$y(x) \in [-1,1]$$

$$y(x) = \frac{1 - e^{-kx}}{1 + e^{-kx}}$$

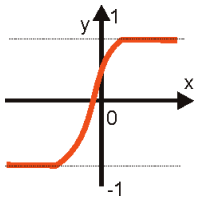

Figura 2.7 – Função tangente hiperbólica

2.2.3 – Arquiteturas das redes neurais

O tipo de estrutura das redes neurais está diretamente ligado ao algoritmo de aprendizagem. Pode-se identificar dois tipos de arquiteturas de redes neurais: redes diretas e redes com realimentação , com uma única camada ou com múltiplas camadas.

2.2.3.1 – Redes diretas

As redes neurais diretas - feedforward – são aquelas cujos grafos não possuem nenhum ciclo, e a informação se desloca em um único sentido entre as camadas adjacentes. Nesta arquitetura, os neurônios que recebem os sinais de excitação externos são chamados de camada de entrada; não processam, somente propagam os sinais para todos os neurônios da próxima camada, conhecida como camada intermediária ou escondida , a qual processa e propaga o sinais para uma próxima camada escondida ou para a camada de saída. Observe que na camada de saída também ocorre processamento. FIGURA 2.8 e FIGURA 2.9

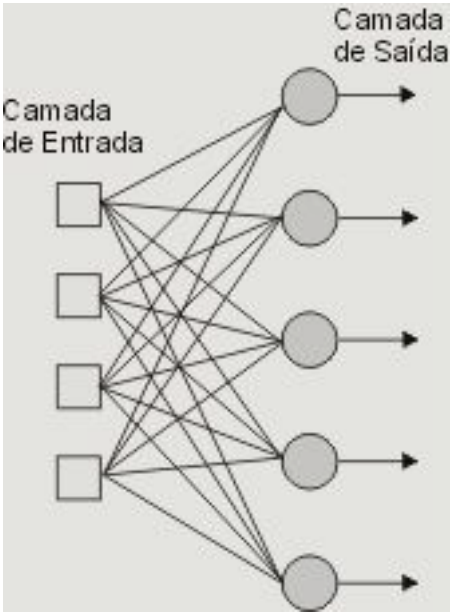

Figura 2.8 – Rede direta com uma única camada (HAYKIN, 2001)

As redes com uma única camada são capazes de resolver problemas linearmente separáveis; as redes com duas camadas são capazes de resolver problemas não linearmente separáveis; as redes com mais de duas camadas são capazes de resolver problemas com um elevado grau de complexidade(BARRETO, 1999).

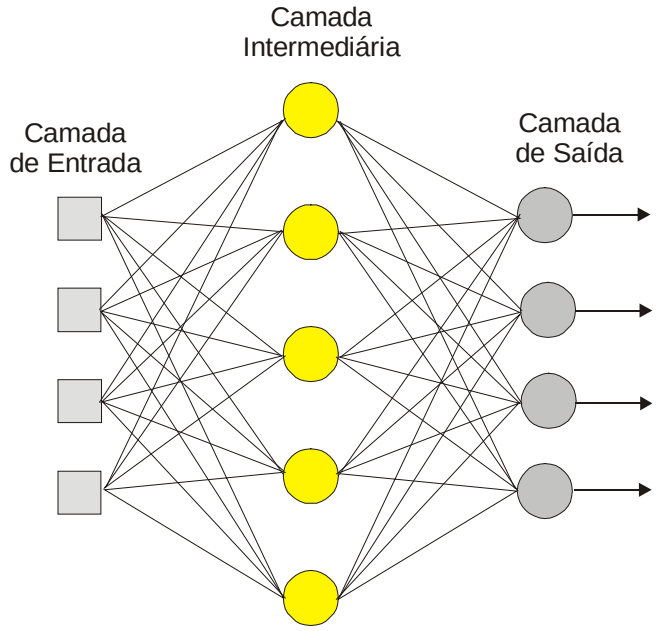

Figura 2.9 – Rede direta com múltiplas camadas (HAYKIN, 2001)

2.2.3.2 – Redes com realimentação

As redes neurais com realimentação - feedback – são aquelas cujos grafos de conectividade contêm pelo menos um ciclo, também conhecido como laço de realimentação, não existindo direção privilegiada para propagação da informação. FIGURA 2.10. A recorrência deste tipo de rede resulta em um comportamento dinâmico (HAYKIN, 2001). Hopfield, Jordan e Elman são as principais redes que apresentam este tipo de estrutura.

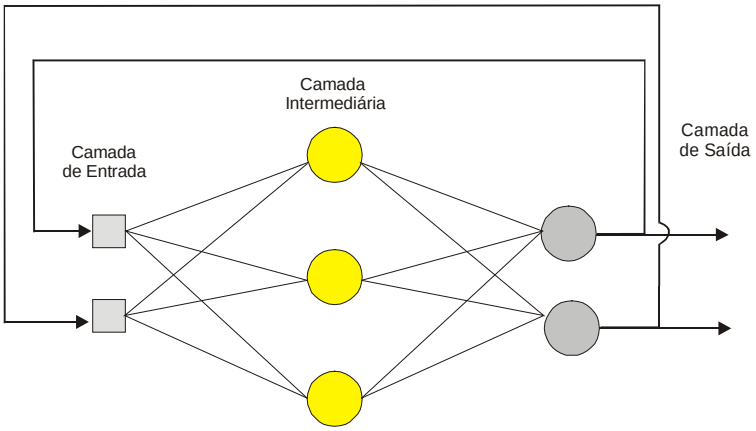

Figura 2.10 – Rede com realimentação

2.2.4 – Treinamento

Treinar uma rede neural artificial significa ajustar a matriz de pesos de tal forma que a saída obtida seja igual ou muito próxima à saída informada para um determinado valor de entrada. Existem alguns algoritmos de treinamento que, além de ajustar os pesos, provocam também mudanças na rede, alterando o número de neurônios.

O treinamento pode ser de dois tipos: supervisionado que exige a disponibilidade de um conjunto de treinamento formado por valores de entradas e saídas; e não

supervisionado no qual o conjunto de treinamento são exclusivamente valores de entrada.

A maioria dos algoritmos de treinamento das redes neurais artificiais são inspirados na lei de Hebb - a intensidade de uma ligação sináptica entre dois neurônios aumenta se ambos são excitados simultaneamente - podendo ser formalizada da seguinte maneira:

$$\Delta w_{ij} = \eta x_i x_j$$

A variação no peso de conexão entre os neurônios i e j, em que η é a taxa de aprendizado e x_i e x_j são a atividade dos neurônios. A lei de Hebb pode ser modificada para levar em conta o valor desejado para a atividade do neurônio, dando origem a regra delta.

$$\Delta w_{ij} = \eta(d_i - x_i)x_j$$

O algoritmo mais conhecido para treinamento das redes neurais artificias é o de retropropagação – backpropagation – que é uma generalização da regra delta.

2.2.4 – Backpropagation

A retropropagação é um algoritmo de treinamento, supervisionado seu funcionamento pode ser descrito da seguinte forma (BITTENCOURT, 1998):

- Apresenta-se um exemplo à rede e obtém-se a saída correspondente.
- Calcula-se o vetor de erro, que consiste na diferença entre a saída obtida e a esperada.
- Calcula-se o gradiente do vetor de erro e atualizam-se os pesos da camada de saída.
- Propagam-se, para as camadas anteriores, os valores, atualizando os pesos das mesmas.

2.3 – Aplicações

As redes neurais são aproximadores universais de funções e a cada novo treinamento geram um novo conjunto de pesos. A atual facilidade de implementar as mesmas em hardware utilizando dispositivos de memória PAM – Programmable Active Memories, levou a utilização das redes neurais para criptografia, estendo assim o gigantesco espectro de utilização destas na atualidade.

Existem diversas aplicações para redes neurais artificias, tanto que o volume de investimentos nesta área tem alcançado somas expressivas tanto no Brasil quanto no exterior. Pode-se destacar as aplicações nas áreas médicas, financeiras, robótica, aeroespacial etc.

Neste trabalho, procurou-se ampliar a utilização das redes neurais artificias para criptoanalisar mensagens criptografadas .

Capítulo 3

Criptografia

3.1 – Definição

“Cripto” significa escondido, oculto ou obscuro e **“grafia”** ação de escrever (HOLANDA, 1999), então criptografia é a arte ou ciência de escrever em cifra ou em código, utilizando um conjunto de técnicas que tornam uma mensagem incompreensível.

3.2 - Terminologia

Um **remetente** deseja enviar uma mensagem a um **destinatário**, porém também quer que nenhum **curioso – intruso ou inimigo -** possa ler esta mensagem no canal de comunicação até o destinatário (SCHNEIER, 1996).

“M” ou “P” são as letras empregadas para representar o **texto plano** ou **texto limpo** . Com o avanço tecnológico, uma mensagem ***M*** poderá ter como conteúdo som, imagem, programas, texto ou a combinação dos mesmos.

Ao processo de tornar secreta uma mensagem chama-se de **criptografar** e sua função é representada pela letra “E” maiúscula . À mensagem criptografada chama-se de **texto cifrado,** representada pela letra “C”. Ao processo de transformar o texto cifrado em um texto plano chama-se de **decriptografar**, representado pela letra “D” maiúscula .FIGURA 3.1. Seguindo o padrão ISO 7498-2 , denomina-se o algoritmo que criptografa a mensagem de **cifrador** e o que decriptografa a mensagem de **decifrador**.

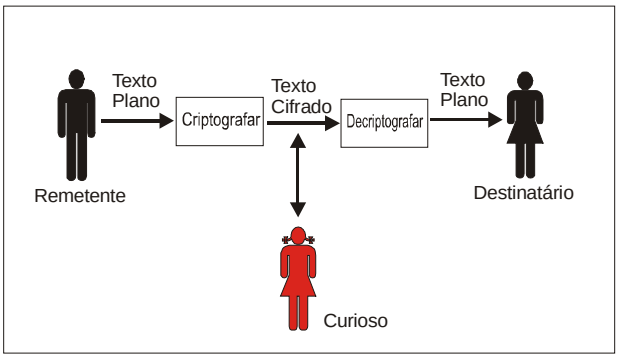

FIGURA 3.1 – Criptografar e Decriptografar

A função de criptografar “E” opera a mensagem “M” para produzir o texto cifrado “C”:

E(M) = C

No processo reverso, a função para decriptografar “D” opera no texto cifrado “C” para produzir o texto plano “M”:

D(C) = M

Logo:

D(E(M)) = M

3.3 – Confiança na comunicação

A confiança no processo de comunicação está relacionada a princípios básicos, análogos aos de uma interação face a face :

Autenticação - A certeza de quem enviou a mensagem, evitando, assim, que um intruso possa enviar uma mensagem utilizando a identidade do remetente.

Confidencial - A certeza de que a mensagem não está sendo ouvida - entendida - por um intruso.

Integridade - A certeza de que a mensagem original não foi alterado por nenhum intruso ao longo do caminho.

Não repúdio – Tanto o remetente quanto o destinatário têm que ter certeza de que o envio ou o recebimento, bem como o perfeito entendimento da mesma, obtiveram sucesso, evitando o repúdio.

3.4 – Chave

O próprio nome define a função de uma chave em qualquer sistema, por exemplo, a chave de um cofre permite acesso ao valores nele contidos ; em um sistema de criptografia, a chave permite o acesso à informação contida na mensagem.

A chave representada pela letra “K” é um número ou um conjunto de letras que, aplicado ao algoritmo, possibilita a transformação do texto plano em texto cifrado e vice-versa. A chave deve ser sempre mantida em segredo. Quanto maior for a chave, mais difícil será decifrar a mensagem.

$$E_K(M) = C$$

ou

$$D_K(E_K(M)) = M$$

ou

$$D_K(C) = M$$

3.5 – Algoritmo Criptográfico

Os algoritmos criptográficos são funções matemáticas utilizadas para criptografar e decriptografar. Geralmente, existem duas funções relacionadas: uma para cifrar e outra para decifrar. Os algoritmos criptográficos têm um interesse histórico muito grande, que remonta quase ao início da escrita, porém os algoritmos utilizados

nos primórdios não poderiam ser utilizados hoje, a exemplo do utilizado pelo imperador romano Júlio César(CARVALHO, 2000), analisado a seguir.

O imperador Júlio César utilizava um algoritmo de substituição. Cada letra do texto plano era substituída por outra letra algumas vezes, à frente no alfabeto, gerando, assim, o texto cifrado, por exemplo:

Texto Plano - Vamos atacar no fim da tarde
Texto Cifrado - YDPRV DWDFDU QR ILP GD WDUGH

Para decifrar, os oficiais romanos saberiam o número de vezes (K) que a letra estaria à frente e poderiam montar uma tabela assim:

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

C = E(M) = (M+3)

K = 3 (chave)

Observa-se que a chave neste algoritmo é 3. Além do mais com um pouco de força bruta para analisar as 25 possíveis chaves, esta mensagem facilmente seria decifrada.

Os algoritmos podem ser classificados quanto à chave: simétricos ou assimétricos; quanto ao número de bits: de blocos ou de fluxo.

3.5.1 – Algoritmo de Chaves Simétrica

Também é conhecido como algoritmo convencional, de chave secreta ou de única chave .Figura 3.2 .O remetente e o destinatário devem compartilhar a mesma chave para cifrar e decifrar a mensagem. Antes de transmitir a mensagem, o remetente e

o destinatário devem comunicar-se por um meio seguro para trocar a informação da chave – para este tipo de algoritmo é vital que a chave permaneça secreta .

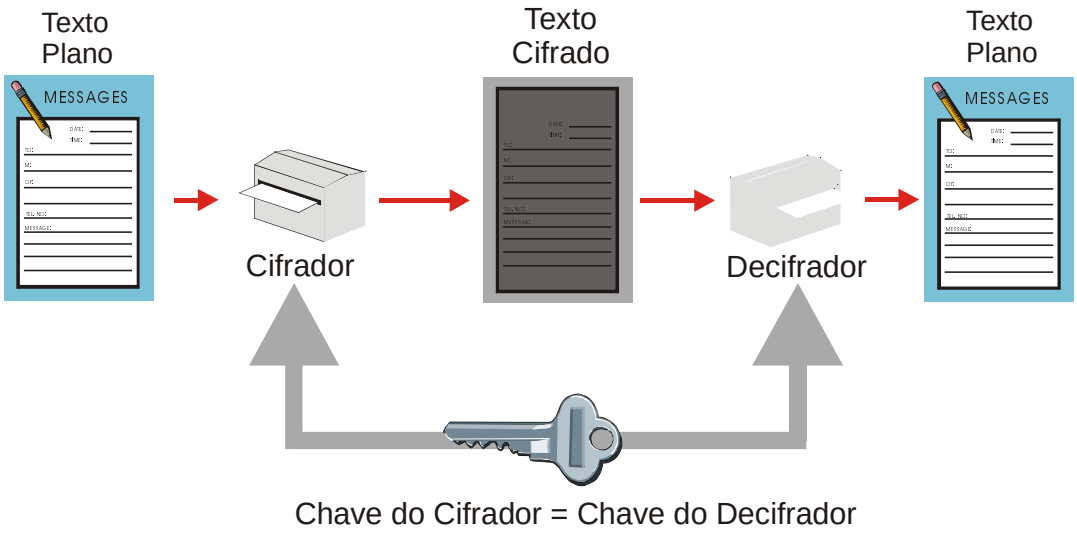

FIGURA 3.2 – Mesma chave para cifrar e decifrar

3.5.1.1 – Principais algoritmos de chave simétrica

DES – (Data Encription Standard). Baseado num algoritmo desenvolvido pela IBM, é o padrão utilizado pelo governo americano para a criptografia de seus dados desde 1978. Em 1981, foi adotado como padrão pela ANSI com o nome de DEA, como tentativa de padronizar procedimentos de cifragem do segmento privado, especialmente instituições financeiras. O DES utiliza cifras de blocos de 64 bits, usando uma chave de 56 bits, fazendo diversas permutações.

Triple-DES – Baseia-se na utilização três vezes seguidas do DES com chaves diferentes.

RC2, RC4 – Algoritmos criados pelo Professor Ronald Rivest, que é um dos proprietários da RSA Data Security. Estes algoritmos usam chaves que variam de 1 a 1024 bits de extensão. Com chaves pequenas (menores que 48 bits), são códigos fáceis de serem quebrados, e como são proprietários, não se têm muitas informações

sobre sua segurança com chaves extensas. RC2 é uma cifra de bloco, similar ao DES. RC4 é um algoritmo de fluxo, em que o algoritmo produz uma corrente de pseudo-números que são cifrados através de uma operação lógica XOR com a própria mensagem.

IDEA – Sigla que designa International Data Encryption Algorithm - é um algoritmo de cifragem de bloco desenvolvido na Suíça e publicado em 1990. IDEA utiliza uma chave de 128 bits. É um algoritmo que ainda não pode ser conceituado como forte, devido a seu pouco tempo de vida, porém aparenta ser robusto. Sua chave com 128 bits dificulta a possibilidade de alguém usar computadores atuais para ataques por força bruta.

Skipjack – Algoritmo secreto desenvolvido pela National Security Agency para uso por civis. É o coração do chip Clipper, desenvolvido pela NSA.

3.5.2 – Algoritmo de Chaves Assimétricas

Também conhecido como algoritmo chave pública, é projetado de uma forma tal que a chave que cifra a mensagem, conhecida como chave pública, seja diferente da chave que decifra, conhecida como chave privada (FIGURA 3.3).

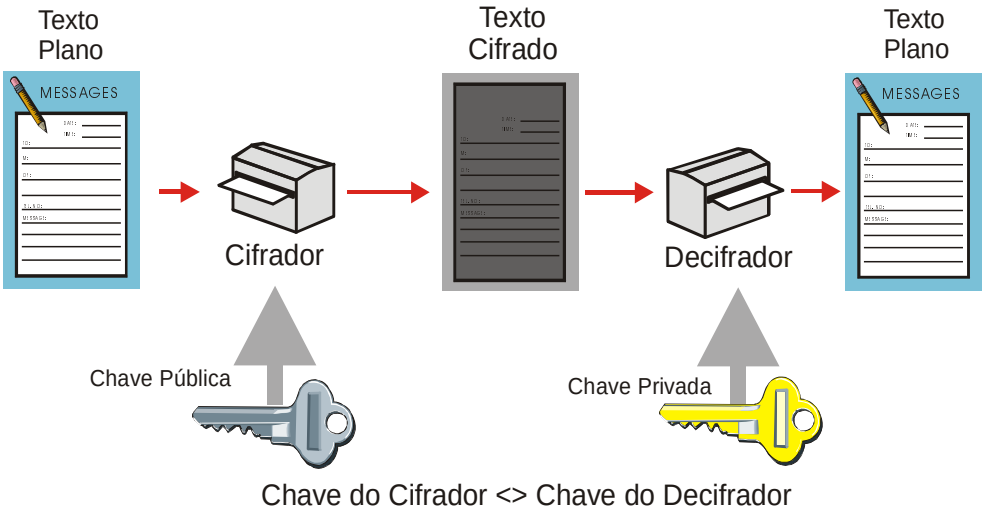

FIGURA 3.3 – Chaves diferentes

3.5.2.1 – Principais algoritmos de chave assimétrica

Diffie-Hellman - Foi o ponto de partida para a criptografia por chave pública, através do artigo chamado "Multi-User Cryptographic Techniques", de Whitfield Diffie e Martin Hellman. A técnica baseia-se na troca de uma chave de cifragem de tal forma que uma terceira parte não autorizada, não tenha como deduzi-la. Cada participante inicia com sua chave secreta e através da troca de informações é derivada uma outra chave, chamada chave de sessão, que será usada para futuras comunicações. O algoritmo baseia-se na exponenciação discreta, pois sua função inversa, os logaritmos discretos, são de alta complexidade.

RSA - Desenvolvido por Ronald Rivest, Adi Shamir e Len Adleman, o algoritmo tomou por base o estudo feito por Diffie e Hellman, porém usando outro fundamento matemático para a criação das chaves públicas. Eles utilizaram o fato de que é fácil obter-se o resultado da multiplicação de dois números primos extensos, - mas é muito difícil obter-se os fatores primos de um número muito extenso.

Merkle-Hellman - Baseava-se em um jogo matemático chamado Knapsack (Mochila), em que dada uma coleção de itens, verificam-se as possíveis maneiras de armazená-la dentro de um repositório de tamanho fixo, de forma a não sobrar espaço. Foi usado durante muitos anos, porém, após a descoberta de uma falha crucial, foi inutilizado para fins práticos.

3.5.3 – Chave simétrica X chave Assimétrica

Analisando os dois tipos de algoritmos, pode-se observar que a criptografia por chave pública tem a vantagem sobre a chave privada no sentido de viabilizar a comunicação segura entre pessoas comuns. Com a chave pública também acaba o problema da distribuição de chaves existentes na criptografia por chave secreta, pois não há necessidade do compartilhamento de uma mesma chave, nem de um pré-acordo

entre as partes interessadas. Com isto o nível de segurança é maior - Tabela 3.1. A principal vantagem da criptografia por chave secreta está na velocidade dos cifradores e decifradores, pois estes tendem a ser mais rápidos que os de chave pública.

TABELA 3.1 – Simétrico X Assimétrico (STALLINGS, 1999)

Algoritmo Simétrico		Algoritmo Assimétrico	
Como funciona	1 - O mesmo algoritmo é usado para cifrar e decifrar. 2 - O remetente e o destinatário devem compartilhar a chave e o algoritmo.	Como funciona	1 - O algoritmo utiliza um par de chaves, uma para cifrar, outra para decifrar. 2 - O remetente e o destinatário devem ter um par de chaves que combine.
Segurança	1 - A chave deverá ficar sempre secreta. 2- O conhecimento do algoritmo deve ser insuficiente para determinar a chave. 3 - Tem que ser impossível ou impraticável decifrar a mensagem na ausência de uma das informações.	Segurança	1 - Somente uma das chaves deverá permanecer secreta. 2 - O conhecimento do algoritmo e de uma das chaves deve ser insuficiente para determinar a outra chave. 3 - Tem que ser impossível ou impraticável decifrar a mensagem na ausência de uma das informações.

3.5.4 – Algoritmo de fluxo

Os algoritmos de fluxo cifram a mensagem bit a bit, e geralmente são utilizados em situação cuja transmissão dos dados é feita da mesma forma, não permitindo acúmulo de bits. Como exemplo deste tipo de algoritmo temos o ONE-TIME PAD, conhecido em português como FOLHA DESCARTÁVEL, a chave deste é do mesmo tamanho da mensagem e só é utilizada uma única vez, devido a isto este tipo de algoritmo é muito pouco utilizado, porém é considerado o algoritmo mais seguro que existe.

3.5.5 – Algoritmo de Bloco

Os algoritmos de blocos processam conjuntos de bits, denominados blocos. Com ampla utilização, estes algoritmos são muito rápidos e eficientes, o bloco padrão na

atualidade é de 64 bits, grande o bastante para evitar uma analise e pequeno para capacidade dos computadores atuais. DES, IDEA, RC5, RSA e muitos outros são exemplo deste tipo de algoritmo.

3.6 – Criptoanálise

Em criptografia, o ponto é conseguir que o texto plano, a chave ou ambos sejam sempre secretos para o curioso - inimigo - , lembrando que este tem total acesso aos meios de comunicação entre o remetente e o destinatário.

Criptoanálise é a ciência para reaver o texto plano sem ter acesso à chave, ou descobrir a chave, ou ainda mostrar o quão fraco o sistema de criptografia é (STALLINGS, 1999). A criptoanálise tem sido alvo de uma ampla pesquisa há vários anos, sendo que a literatura especializada, hoje disponível sobre criptografia, contém poucos exemplos de métodos universais, que possam ser aplicados com sucesso a uma grande variedade de algoritmos criptográficos.

À tentativa de criptoanalisar um texto cifrado chama-se de ataque. Existem sete tipos de ataques, pressupondo que para realizar o ataque o criptonalista tenha perfeito conhecimento do algoritmo usado para criptografar a mensagem.

Texto cifrado – O criptoanalista possui vários textos cifrados criptografados pelo mesmo algoritmo, devendo deduzir o maior número de textos planos possível, ou melhor, deduzir a chave usada para criptografar as mensagens, a fim de inferir o texto plano que deseja decifrar.

$C_1=E_K(M_1), C_2=E_K(M_2), \ldots C_i=E_K(M_i)$

Deduzir $P_1, P_2 \ldots\ldots P_i; K$

Inferir P_{i+1} de $C_{i+1}=E_K(M_{i+1})$

Textos Planos Conhecidos – O criptoanalista possui vários pares de textos planos e textos cifrados pelo mesmo algoritmo, devendo deduzir a chave usada para criptografar as mensagens, podendo então decifrar a mensagem desejada.

$P_1, C_1=E_K(M_1); P_2, C_2=E_K(M_2); \ldots ;P_i, C_i=E_K(M_i)$
Deduzir K
Inferir P_{i+1} de $C_{i+1}=E_K(M_{i+1})$

Texto Plano Escolhido – O criptoanalista, além de possuir vários pares de textos planos e textos cifrados, escolhe textos planos. Este tipo de ataque é mais poderoso que o anterior, porque o criptoanalista pode escolher um bloco grande de texto plano para criptografar. Com um maior número de informação fica mais simples deduzir a chave usada para criptografar as mensagens.

$P_1, C_1=E_K(M_1); P_2, C_2=E_K(M_2); \ldots ;P_i, C_i=E_K(M_i)$
Escolhe $P_1, P_2 \ldots\ldots P_i$
Deduzir K
Inferir P_{i+1} de $C_{i+1}=E_K(M_{i+1})$

Texto Plano Escolhido Adaptado – Este é um caso especial do ataque "Texto Plano Escolhido". O criptoanalista pode modificar o texto escolhido, baseado nos resultados preliminares obtidos. No ataque anterior, escolhe-se um bloco grande de texto, nele pode-se escolher um bloco pequeno de texto.

Texto Cifrado Escolhido – O criptoanalista escolhe diferentes textos cifrados e tem acesso a textos planos, tendo como trabalho principal deduzir a chave.

Nota-se que a escolha do ataque está totalmente ligada ao número de informações em poder de quem decidirá o tipo de ataque, conforme TABELA 3.2.

TABELA 3.2 – Tipos de ataques a mensagens criptografadas (STALLINGS, 1999)

Tipo de Ataque	Informações em poder do criptoanalista
Texto cifrado	• O Algoritmo • Texto cifrado a ser decifrado
Texto plano conhecido	• O algoritmo • Texto cifrado a ser decifrado • Um ou mais pares texto plano – texto cifrado com a chave secreta
Texto plano escolhido	• O algoritmo • Texto cifrado a ser decifrado • O texto plano escolhido pelo criptoanalista em conjunto com o texto cifrado gerado com a chave secreta
Texto cifrado escolhido	• O algoritmo • Texto cifrado a ser decifrado • O texto cifrado escolhido pelo criptoanalista em conjunto com o texto plano decifrado com a chave secreta
Texto plano escolhido Adaptado	• O algoritmo • Texto cifrado a ser decifrado • O texto plano escolhido pelo criptoanalista em conjunto com o texto cifrado gerado com a chave secreta • O texto cifrado escolhido pelo criptoanalista em conjunto com o texto plano decifrado com a chave secreta

Seria muito fácil para o criptanalista pesquisar todas as possíveis chaves no algoritmo utilizado pelo imperador Júlio César, pois como foi visto só existiriam 25. A este tipo de ataque, chama-se de Força Bruta, porém na moderna criptografia a pesquisa exaustiva da chave pode ser impossível, TABELA 3.3.

TABELA 3.3 – Tempo médio gasto em relação ao tamanho da chave [3]

chave (bits)	Número de Chaves	Tempo Médio
32	$2^{32}=4.3 \times 10^{9}$	2^{31} μs = 35.8 min
56	$2^{56}=7.2 \times 10^{16}$	2^{55} μs = 1142 anos
128	$2^{128}=3.4 \times 10^{38}$	2^{127} μs = 5.4×10^{24} anos

3 .7 – Esteganografia

Um bom sistema criptográfico não necessita de “Esteganografia” , técnicas para esconder as mensagens de maneira que as mesmas passem despercebidas. Como exemplo, pode-se citar uma carta escrita com tinta invisível.

Existem hoje diversos programas que substituem o bit menos significativo de uma imagem pelos bits da mensagem que se quer esconder. Como o olho humano não percebe tais variações, a mensagem intercalada não altera a visualização da imagem (SCHNEIER, 1996).

3 .8 – DES

Data Encryption Standard é o padrão adotado a partir de janeiro de 1977 pelo "National Bureau of Standard" (NBS, 1977), para departamentos e agências do governo federal americano, bem como para qualquer outra entidade que deseje usá-lo para obter proteção criptográfica. O DES está sendo substituído pelo novo padrão, que se chama **AES** - Advanced Ecryption System (NIST, 1998).

Apresentar-se-á abaixo o funcionamento do algoritmo de criptografia DES, FIGURA 3.4, que é a modificação do sistema de Meyer, desenvolvido pela IBM (SCHNEIER, 1996) , mostrando, com detalhes, como um bloco de 64 bits de texto plano percorre até se transformar em texto cifrado.

A permutação inicial ***IP*** é o primeiro passo do algoritmo, e nada mais é que uma função que muda as posições dos bits seguindo uma matriz 64 posições, conforme exemplo abaixo (CARVALHO, 2000):

$$\boldsymbol{IP} = \begin{Bmatrix} 58 & 50 & 42 & 34 & 26 & 18 & 10 & 2 & 60 & 52 & 44 & 36 & 28 & 20 & 12 & 4 \\ 62 & 54 & 46 & 38 & 30 & 22 & 14 & 6 & 64 & 56 & 48 & 40 & 32 & 24 & 16 & 8 \\ 57 & 49 & 41 & 33 & 25 & 17 & 9 & 1 & 59 & 51 & 43 & 35 & 27 & 19 & 11 & 3 \\ 61 & 53 & 45 & 37 & 29 & 21 & 13 & 5 & 63 & 55 & 47 & 39 & 31 & 23 & 15 & 7 \end{Bmatrix}$$

O primeiro bit vai para posição 58, o segundo para posição 50, o terceiro para posição 42 e assim por diante .Tanto a permutação inicial quanto a permutação inicial invertida não têm muita importância quanto ao aspecto de segurança do algoritmo.

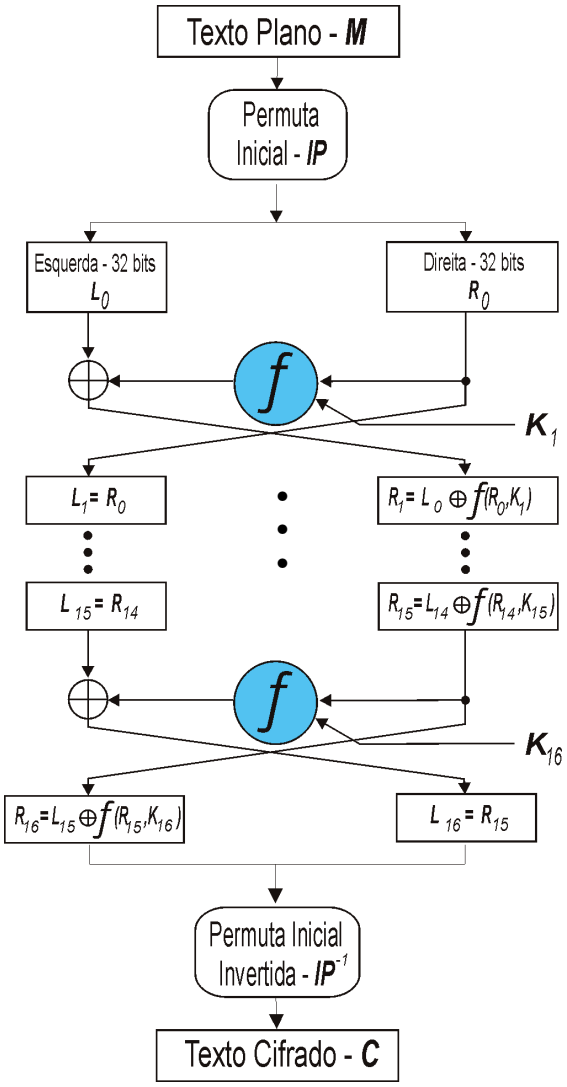

FIGURA 3.4 – Transformação do Texto plano em texto cifrado **– *DES***

No segundo passo, dividiu-se o bloco permutado em dois, esquerda (***L*$_n$**) e direita (***R*$_n$**) e aplicou-se as 15 interações, pois a 16ª é diferente das anteriores. Depois, juntou-se direita (***R*$_{16}$**) e esquerda (***L*$_{16}$**) e aplicou-se a permutação inicial invertida ou permutação final, conforme exemplo abaixo:

$$IP^{-1} = \begin{Bmatrix} 40 & 8 & 48 & 16 & 56 & 24 & 64 & 32 & 39 & 7 & 47 & 15 & 55 & 23 & 63 & 31 \\ 38 & 6 & 46 & 14 & 54 & 22 & 62 & 30 & 37 & 5 & 45 & 13 & 53 & 21 & 61 & 29 \\ 36 & 4 & 44 & 12 & 52 & 20 & 60 & 28 & 35 & 3 & 43 & 11 & 51 & 19 & 59 & 27 \\ 34 & 2 & 42 & 10 & 50 & 18 & 58 & 26 & 33 & 1 & 41 & 9 & 49 & 17 & 57 & 25 \end{Bmatrix}$$

Na figura 3.5, mostra-se como funciona uma interação no algoritmo DES. O bloco da direita (R_i) passa por uma função de permutação, que além de trocar, expande de 32 bits para 48 bits. Existe a necessidade de expandir o bloco para poder realizar o passo seguinte que é a operação de **XOR** (ou - exclusivo) com a chave (K_i) para que esta operação possa ser realizada o algoritmo transforma a chave de 56 bits em uma chave de 48 bits.

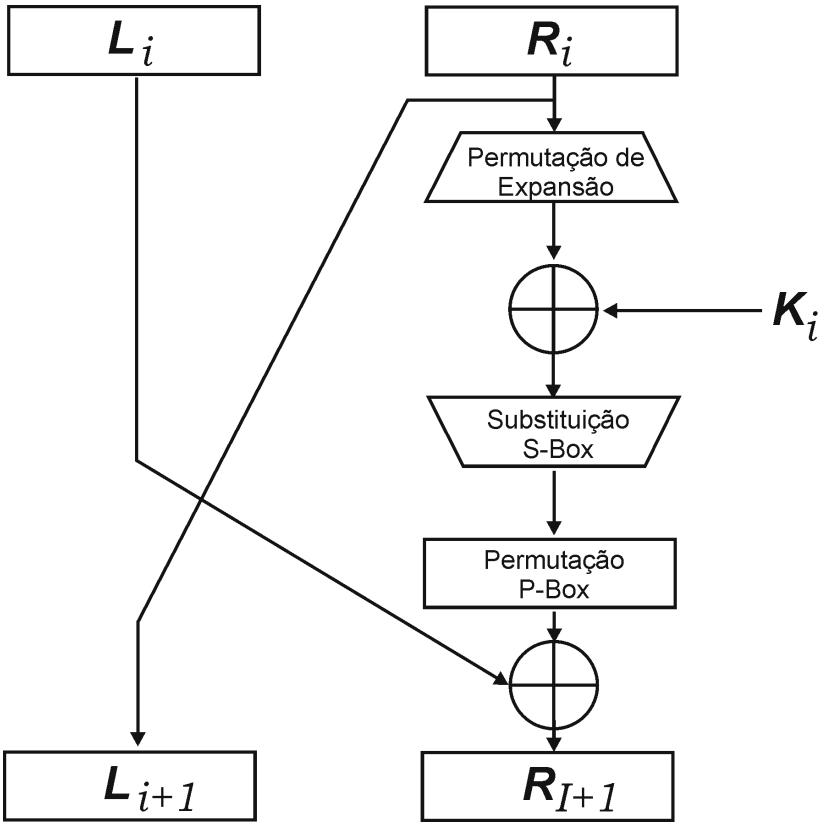

FIGURA 3.5 – Uma interação do DES

A matriz de permutação e expansão, também é conhecida como tabela de expansão ***(ET)***:

$$ET = \begin{Bmatrix} 32 & 1 & 2 & 3 & 4 & 5 & 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 9 & 10 & 11 & 12 & 13 & 12 & 13 & 14 & 15 & 16 & 17 \\ 16 & 17 & 18 & 19 & 20 & 21 & 20 & 21 & 22 & 23 & 24 & 25 \\ 24 & 25 & 26 & 27 & 28 & 29 & 28 & 29 & 30 & 31 & 32 & 1 \end{Bmatrix}$$

O passo seguinte é uma função de substituição e compressão denominada S-Box, a qual tem 8 diferente caixas (matrizes), cada S-Box tem 6 bits de entrada e 4 bits de saída, necessitando de 256 bytes de memória, conforme FIGURA 3.6.

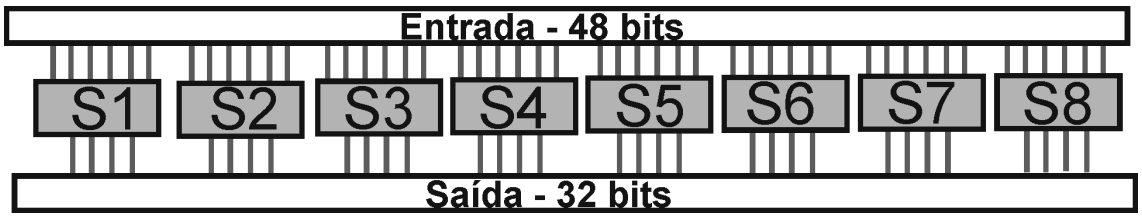

FIGURA 3.6 – Estrutura do S-BOX

Cada matriz S-Box tem 4 linhas e 16 colunas. A entrada 110011,- que são as posições de 1 a 6 de uma palavra de 48 bits, o primeiro e o último bits combinados formam 11 que é 3 em decimal,- corresponde à linha 3 da S1,- os 4 bits do meio formam o 1001 que é 9 em decimal, coluna 9 da mesma S-Box,- então, ao se localizar, na matriz S1 o valor correspondente à linha 3, coluna 9 que é 11 em decimal e 1011 binário, lembrando que se deve começar a contar da linha 0 e coluna 0. No exemplo acima, tem-se uma entrada na função S-Box de 110011 e uma saída 1011.

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
5	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Após este passo, o algoritmo passa por uma nova função de permutação denominada de Wire-Crossing ***(WC)***, P-Box ou permutação direta.

$$WC = \left\{ \begin{matrix} 16 & 7 & 20 & 21 & 29 & 12 & 28 & 17 & 1 & 15 & 23 & 26 & 5 & 18 & 31 & 10 \\ 2 & 8 & 24 & 14 & 32 & 27 & 3 & 9 & 19 & 13 & 30 & 6 & 22 & 11 & 4 & 25 \end{matrix} \right\}$$

E depois realiza a operação XOR com o bloco esquerdo (L_n), gerando, assim, o novo bloco direito (R_{i+1}). O novo bloco esquerdo (L_{n+1}) recebe o valor do bloco direito (R_i) antes da interação.

Procurando entender melhor o que ocorre com a chave ***K*** de 56 bits durante cada interação do algoritmo DES, pode-se dizer que ela sofre uma divisão similar à ocorrida com o bloco, gerando duas palavras de 28 bits – esquerda e direita - as quais passam por uma função que aplica uma rotação circular à esquerda, que a cada interação muda o número de bits a ser rotacionado TABELA 3.4:

TABELA 3.4 – Interações X Número de bits

Interações	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Nº de bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Após a rotação, as duas palavras de 28 bits são unidas novamente e passam por uma função de permutação e compressão, gerando K_i que é aplicado na primeira operação XOR de cada interação do DES. Observa-se que K_i é de 48 bits e para a próxima interação, a chave será a união da esquerda e direita sem passar pela função de permutação e compressão. A FIGURA 3.7 demonstra exatamente como ocorre a transformação da chave em cada interação.

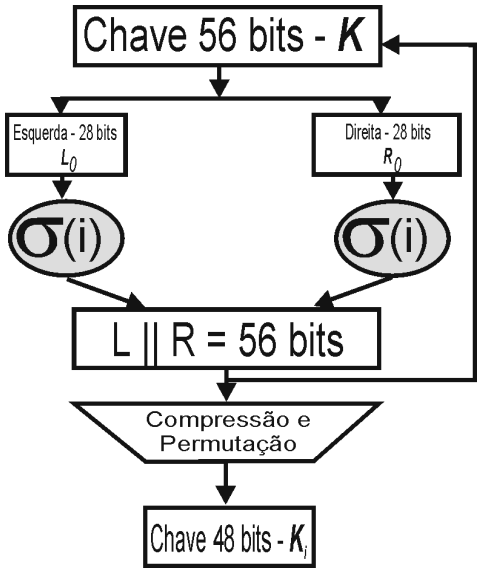

FIGURA 3.7 – A chave no DES

Também será apresentada a matriz de compressão e permutação ***PC:***

$$PC = \begin{Bmatrix} 14 & 17 & 11 & 24 & 01 & 05 & 03 & 28 & 15 & 06 & 21 & 10 \\ 23 & 19 & 12 & 04 & 26 & 08 & 16 & 07 & 27 & 20 & 13 & 02 \\ 41 & 52 & 31 & 37 & 47 & 55 & 30 & 40 & 51 & 45 & 33 & 48 \\ 44 & 49 & 39 & 56 & 34 & 53 & 46 & 42 & 50 & 36 & 29 & 32 \end{Bmatrix}$$

Apresenta-se o algoritmo DES, que, por muito tempo, foi alvo de fortes suspeitas por parte da comunidade internacional, inclusive o governo americano seria capaz de decriptografar as mensagens cifradas com o DES. As críticas também recaíam quanto ao tamanho da chave, números das interações e como foram desenhadas as S-Boxs, porém não existem explicações de como foram criadas. A IBM alega que foram gastos 17 anos de muita criptoanálise para a concepção das mesmas.

3 .9 – RSA

O artigo de Diffie e Hellman (1976) introduziu um novo conceito em criptografia: o de chave pública ou chave assimétrica. Em 1978 três pesquisadores do MIT – Ron **R**ivest, Adi **S**hamir e Len **A**dleman criaram um método para obter

assinaturas digitais e um sistema de criptografia de chave pública, batizado com as iniciais de seus sobrenomes (STALLINGS, 1999).

Todos conhecem a chave pública que cifrou a mensagem e somente o destinatário possui a chave correspondente a privada para poder decifrá-la. Já a assinatura digital funciona de forma inversa, existe uma chave pública, que garante a autenticidade de quem assinou a mensagem com uma chave privada correspondente.

"O RSA é o algoritmo de chave pública mais usado e testado, e mostrou-se extremamente forte, quando usado adequadamente. Ele baseia-se no fato de que é extremamente difícil fatorar números muito grandes." (CARVALHO, 2000).

3.9.1 – Descrição do algoritmo

Para criptografar uma mensagem ***M,*** dividi-se o resultado da exponenciação de ***M*** (texto plano) por ***e*** (chave pública) pelo módulo ***n*** e o resto desta divisão será ***C*** (texto cifrado).

$$C = M^e \bmod n$$

E para decifrar, dividi-se o resultado da exponenciação de ***C*** (texto cifrado) por ***d*** (chave privada) pelo mesmo módulo ***n*** e o resto desta divisão será ***M*** (texto plano).

$$M = C^d \bmod n$$

Para o algoritmo RSA funcionar são necessários alguns passos para gerar as chaves bem como o módulo.

Primeiro gera-se o módulo ***n***, que é oriundo da multiplicação de dois números primos ***p*** e ***q*** , os quais devem ser muito grandes e também mantidos em segredo, "causando, assim, uma enorme dificuldade para se conseguir fatorar ***n*** , evitando-se saber o jeito que ***d*** (chave privada) deriva de ***e*** (chave pública) " . (RIVEST;SHAMIR;ADLEMAN, 1976).

$$n = p \times q$$

Escolhe-se a chave privada ***d*** que deverá ser um número primo menor que **n** e também deverá ser primo relativo a ***n***. Utilizando a função de Euler:

$$\phi(n) = (p-1) \times (q-1)$$

$mdc(d, \phi(n)) = 1$ (condição para ser primo relativo)

então :

$$e \equiv d^{-1} \ mod \ \phi(n)$$

Por exemplo, cria-se uma aplicação com a finalidade de calcular as chaves e o módulo do algoritmo RSA. FIGURA 3.8.

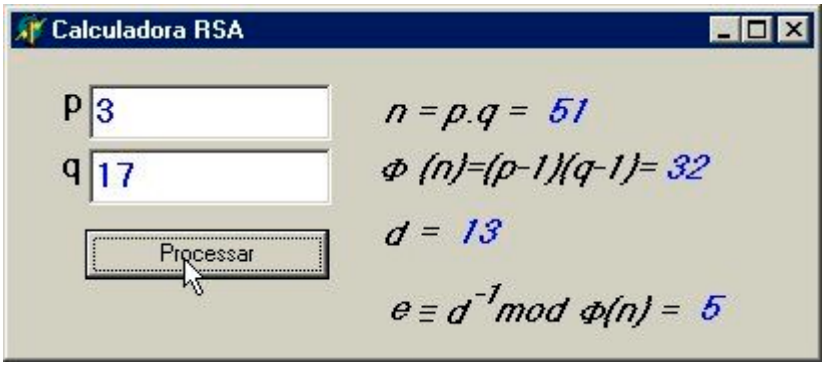

FIGURA 3.8 – Calculadora RSA

Estes valores encontram-se registrados na TABELA 3.5.

TABELA 3.5 – Chaves RSA

Exemplo

Chave Pública	Módulo	Chave Privada
13	51	5

Considerando estes valores, criptografando:

M = 011001 = 25

Obtêm-se:

$$E(M) = C = 25^{13} \bmod 51 = 43 = 101011$$

Por outro lado, decifrando:

C = 101011 = 43

$$D(C) = M = 43^{5} \bmod 51 = 25 = 011001$$

Apresenta-se o algoritmo RSA, “este considerado de execução lenta” (CARVALHO, 2000), porém, com o considerável aumento da capacidade computacional, o RSA passou a ser mais utilizado.

3 .10 – Aplicações

Ao longo da história, sempre foi muito necessária a utilização da criptografia, e, na atualidade, esta necessidade cresceu de maneira exponencial em função da internet, e-commerce, telefonia celular, transações financeiras e muitas outras aplicações.

Acredita-se que esta necessidade do mundo real migrou para os dispositivos eletrônicos utilizados na atualidade. Os intrusos (inimigos) tentam de toda forma conseguir tirar alguma informação para fraudar os remetentes, tais como, clonar cartões de crédito, falsificar documentos diversos, clonar telefones celulares, interceptar mensagens importantes, desviar transferências bancárias etc.

Como se verifica ao longo deste capítulo, os estudos dos algoritmos para cifrar e decifrar mensagens sempre foram da maior importância para a atividade militar e comercial, obtendo, assim, significativos recursos para pesquisa e implementação de

novos aplicativos. No que tange à segurança de uma nação, a criptoánalise se faz muito necessária no processo de espionagem.

Capítulo 4

Análise para criação de um cifrador usando redes neurais artificiais.

4.1 – Princípios

As propostas para construção de algoritmos de criptografia estão baseadas no mais complexo processo de comunicação existente na natureza, a comunicação entre uma célula (neurônio) e outra, ou seja, na troca de informação entre as mesmas. Ao se analisar o modelo de neurônio artificial, serão utilizados como entradas o texto plano, os pesos (sinapes) como chave de nosso algoritmo e o "net" como texto cifrado.

Também procurou-se enquadrar esses cifradores nos critérios estabelecido em 1973 pelo ainda chamado NBS (National Bureau of Standard), hoje NIST (National Institute of Standards and Technology), o qual editou uma proposta de como deveriam ser os algoritmos de criptografia, utilizando os critérios abaixo (SCHNEIER, 1996):

1 – O algoritmo deve prover o mais alto nível de segurança.

2 – O algoritmo deve estar totalmente especificado e de ser fácil entendimento.

3 – A segurança deve estar na chave e não em segredos no próprio algoritmo.

4 – O algoritmo deve estar disponível a todos.

5 – O algoritmo deve adaptar-se a diversas aplicações.

6 – O algoritmo deve ter uma implementação econômica em dispositivos eletrônicos.

7 – O algoritmo deve ser eficiente.

8 – O algoritmo deve ser fácil de validar.

9 – O algoritmo deve poder ser exportado.

4.2 – Primeira proposta

Sabe-se que as redes neurais artificiais são capazes de apreender diversas funções (MINSKY;PARPERT, 1969). Partindo deste pressuposto e admitindo que a cada treinamento as redes são inicializadas com pesos diferentes, pensa-se, então, em utilizar a função $f(M) = M$, em que M é a mensagem, e $C = M \times iw$ (peso da entrada) + b_1 (bias) o valor da saída do neurônio da camada escondida, texto cifrado. O destinatário deve saber o valor de lw e de b_2 , obtendo, assim, o texto plano $M = C \times lw$ (peso da saída) + b_2 (bias). FIGURA 4.1

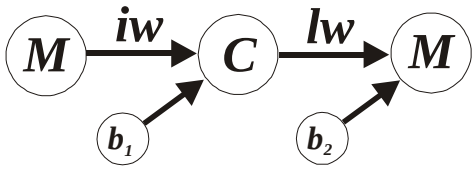

FIGURA 4.1 – Rede direta

Na primeira proposta, utiliza-se uma rede neural artificial do tipo feedforward com o algoritmo de treinamento backpropagation, contendo um neurônio na camada de entrada, um neurônio na camada escondida e um neurônio na saída. Nota-se que não seriam necessários mais neurônios na camada intermediária em razão da natureza da função matemática que essa rede deveria apreender . Após a escolha da arquitetura da rede, passa-se ao processo de treinamento. Escolhe-se, para conjunto de exemplos, 50 números entre 0 e 1000 randomicamente que são as entradas, e, como saída desejada, utiliza-se os mesmos 50 números, já que a função que a rede deve apreender é $f(M) = M$, ou seja, saída desejada é igual à entrada. No que tange à função de saída dos neurônios tanto da camada escondida quanto da camada de saída de nossa rede, a escolha recaiu em uma função de linha explicada no 2^{o} capítulo, a qual retorna a valores absolutos e não valores dispostos em um intervalo como nas funções tangentes hiperbólicas ou sigmoidais.

Utilizando o algoritmo otimizado de Levenberg-Marquardt[2], realizou-se o processo de treinamento de nossa rede, cujos dados obtidos estão na TABELA 4.1.

TABELA 4.1 – Pesos e bias obtidos após o treinamento de nossa rede.

Chaves	Valor
iw	-1.1847859
$\boldsymbol{b_1}$	0.1533674
lw	-0.8440343422147
$\boldsymbol{b_2}$	0.12944738285753

A mensagem a ser cifrada foi divida em blocos iguais de 24 bits, os quais foram considerados individualmente com ***M*** **.** No exemplo abaixo, observa-se como funciona a primeira proposta deste trabalho de cifrador, utilizando uma rede neural artificial.

Exemplo:

$\boldsymbol{M = 111111111111111111111111 = 16777215}$

$\boldsymbol{C = 16777215 \times -1.1847859 + 0.1533674 = -19877407.6}$

Para representar o sinal e o número de casas decimais assim se resolveu: quanto ao sinal, usa-se a notação clássica: se for positivo 0 (zero), negativo 1 (um); e quanto ao número de casas decimais, resolveu-se usar 3 (três) bits, o que permite chegar a 8 casas decimais, garantindo, então, a precisão necessária para um perfeito funcionamento de tal algoritmo.

Sendo assim, o texto cifrado será 10001011110110010000110100111100 (2346257724), o primeiro bit é 1 que representa o sinal negativo, seguido de 000 que representam uma casa decimal - em função de utilizar-se somente três bits para representação de no máximo 8 casas decimais -, e o próprio número sem a vírgula e o sinal 10111101100100000110100111100 (198774076).

[2] - No apêndice A, apresenta-se tabela contendo todas as funções de treinamento de redes neurais.

Ao receber o texto cifrado - 10001011110110010000110100111100 - o destinatário deve verificar o sinal - que é negativo - e o número de casas decimais - que é uma -, então decifrar a mensagem usando suas chaves ***lw*** e b_2 .

C = 10111101100100000110100111100 ∴

C = 198774076 : - 10 ∴

C = -19877407.6

então:

M = -19877407.6 * -0.8440343422147 + 0.12944738285753 = 16777214.78

arredondando:

M = 16777215

Ao analisar essa primeira proposta sob a luz das recomendações do NIST, verificou-se que ela atende perfeitamente as mesmas, porém apresenta uma fragilidade no que tange à linearidade da substituição do texto plano pelo texto cifrado, como será visto na TABELA 4.2, facilitando, assim, qualquer tipo de ataque de criptoánalise.

TABELA 4.2 – Texto Plano e Texto cifrado

1ª proposta.

Texto Plano	**Texto Cifrado**
0	0.1533674
1	-1.0314185
2	-2.2162044
3	-3.4009903
•	•
•	•
•	•
1000000	-1184785.747

Outra fragilidade observada nesta proposta é exatamente quanto à repetição de valores. Supondo que se desejasse trafegar vários blocos de 24 bits iguais, fato muito

comum na criptografia atual – arquivos com imagem, som etc. –, logo o criptoanalista reconheceria um padrão , o que também o levaria a decifrar a mensagem.

4.3 – Segunda Proposta

Visando a eliminar a linearidade da primeira proposta e permanecendo com a função $f(M) = M$, alterou-se a arquitetura dessa rede e se passou a utilizar uma rede de Elman, a qual tem uma recorrência na camada intermediaria. Neste caso, o valor da saída do neurônio da camada escondida , $C_k = (M_k \times iw) + (rw \times C_{k-1}) + b_1$, será o texto cifrado. O destinatário deve saber o valor de lw e de b_2 , conseguindo, assim, o texto plano $M_k = (C_k - C_{k-1} \times rw) \times lw + b_2$. FIGURA 4.2.

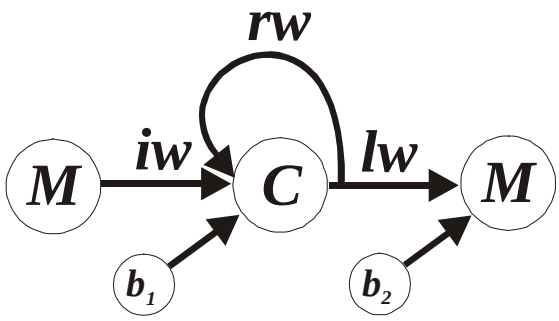

FIGURA 4.2 – Rede de Elman

Utilizou-se o algoritmo de treinamento backpropagation, um neurônio na camada de entrada, um neurônio na camada escondida e um neurônio na saída, como pode ser visto na FIGURA 4.2. Nesta proposta também não são necessários mais neurônios na camada intermediária em razão da natureza da função matemática que essa rede deveria apreender . Após a escolha da arquitetura da rede, passou-se ao processo de treinamento. Utilizou-se o mesmo conjunto de treinamento da primeira proposta e também foi mantida a escolha das funções de saída dos neurônios da camada intermediária e da camada saída.

Utilizando o algoritmo otimizado de Levenberg-Marquardt, foi realizado o processo de treinamento dessa rede cujos dados obtidos estão na TABELA 4.3.

TABELA 4.3 – Pesos e bias 2ª proposta.

Chaves	Valor
Iw	0.7139376
b_1	-1.4890783
Rw	-0.3515081
Lw	1.40068261107608
b_2	2.08572613640992

Nesta proposta, a mensagem a ser cifrada foi também divida em blocos iguais de 24 bits, os quais foram considerados individualmente com ***M.*** No exemplo abaixo, observa-se como funciona esta proposta de cifrador, utilizando uma rede neural artificial de Elman.

Exemplo:

$M_0 = 111111111111111111111111 = 16777215$

$C_0 = 16777215 \times 0.7139376 + 0 \times -0.3515081 + -1.4890783 = 11977883$

$M_1 = 111111111111111111111111 = 16777215$

$C_1 = 16777215 \times 0.7139376 + 11977883 \times -0.3515081 + -1.4890783 = 7767560$

$M_2 = 111111111111111111111111 = 16777215$

$C_2 = 16777215 \times 0.7139376 + 7767560 \times -0.3515081 + -1.4890783 = 9247523$

Manteve-se nesta proposta as mesma decisões quanto ao sinal e ao número de casas decimais da proposta anterior, bem como quanto ao tamanho do bloco de texto cifrado em 32 bits.

Ao analisar a proposta da rede Elman sob a luz das recomendações do NIST, verificou-se que ela também atende perfeitamente às mesmas, porém nesta a fragilidade

apresentada na anterior não existe em função da recorrência na camada escondida, fato este que elimina a linearidade da substituição do texto plano pelo texto cifrado.

Na proposta anterior, caso o criptoanalista repetisse ***M (texto plano),*** logo descobriria seu equivalente em ***C (texto cifrado)***. Já na proposta atual, o mesmo terá pela frente um grau de dificuldade bastante elevado, porque C_1 ***(texto cifrado)*** agora depende do C_0 ***(texto cifrado anterior)*** , ver TABELA 4.4

TABELA 4.4 – Texto Plano e Texto cifrado

2ª proposta.

Texto Plano	Texto Cifrado
16777215	11977883
16777215	7767560
16777215	9247523
12349870	5566456
134678	-1860504
12349870	9471017
12349870	5487896

Em razão do tamanho do bloco a trafegar no meio inseguro, após a vigésima repetição - fato constatado nos testes realizados - do valor de ***M*** (texto plano), o texto cifrado (***C***) também passa a ter valores iguais, o que não descarta o alto grau de segurança deste algoritmo, pois todos os algoritmos têm repetição após atingir o tamanho do bloco ou da chave.

4.4 – Terceira Proposta

Visando também a eliminar a linearidade da primeira proposta e a prover maior segurança ao algoritmo, a terceira proposta altera a arquitetura da rede neural artificial. Enquanto nas anteriores, tinha-se somente um neurônio na camada escondida, na atual tem-se quatro, decisão que levaria o texto cifrado ***(C)*** a ter o tamanho quatro vezes maior que o texto plano ***(M)***, o que pode inviabilizar a proposta deste trabalho. Para solucionar este problema, resolveu-se que o texto cifrado ***(C)*** seria a saída de um só neurônio da camada escondida e que se utilizaria outras redes do tipo feedforward com o algoritmo de treinamento backpropagation, com o objetivo de apreenderem os valores das saídas dos outros três neurônios da camada escondida. FIGURA 4.3.

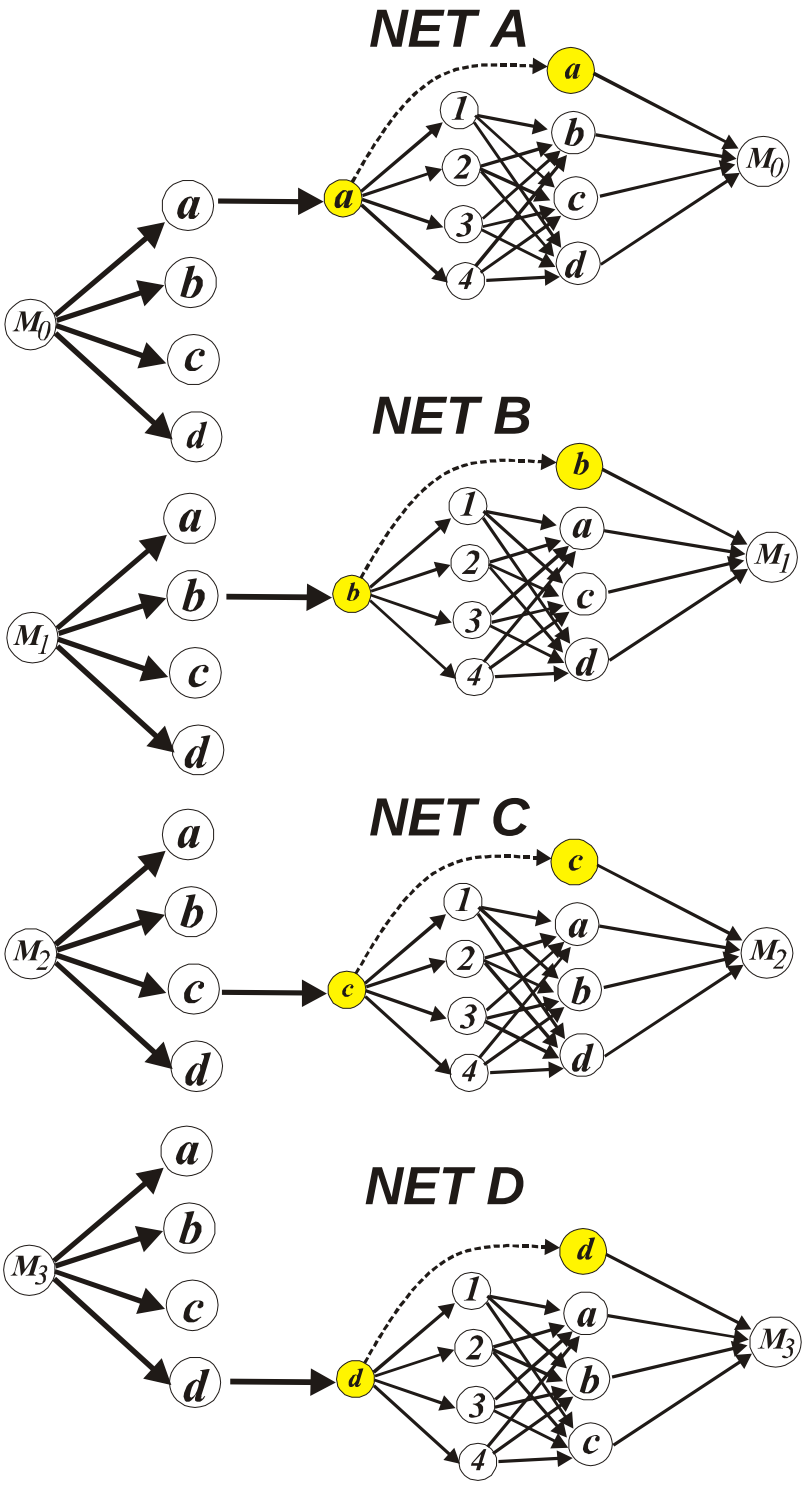

FIGURA 4.3 – Rede Principal e as rede secundárias

Com o mesmo conjunto de treinamento, funções de transferência e algoritmo de treinamento das propostas anteriores, realizou-se o treinamento da rede principal no qual foram obtidos os valores da TABELA 4.5.

TABELA 4.5 – Pesos e bias da rede principal
3ª proposta.

Chaves	Valor
$iw_{1,1}$	0.2668579
$iw_{1,2}$	0.1503838
$iw_{1,3}$	0.9468040
$iw_{1,4}$	0.4572495
$b_{1,1}$	0.5746335
$b_{1,2}$	-0.6685121
$b_{1,3}$	-0.0950285
$b_{1,4}$	0.7685239
$lw_{1,1}$	0.85079061265215
$lw_{1,2}$	-0.15360837609812
$lw_{1,3}$	1.03954485396513
$lw_{1,4}$	-0.41155930154797
$b_{2,1}$	-0.17650225940755

Após o treinamento da rede principal, criou-se quatro redes secundárias as quais se denominou de Net-A, Net-B, Net-C e Net-D. Para iniciar o treinamento das redes secundárias, utilizou-se como conjunto de treinamento as saídas dos neurônios das camadas escondidas da rede principal. Para entrada de Net-A tem-se a saída do neurônio A e para saída desejada de A as saídas dos neurônios B,C e D; para entrada de Net-B tem-se a saída do neurônio B e para saída desejada de B as saídas dos neurônios A,C e D; para entrada de Net-C tem-se a saída do neurônio C e para saída desejada de C as saídas dos neurônios A,B e D; para entrada de Net-D tem-se a saída do neurônio D e para saída desejada de D as saídas dos neurônios A,B e C. Treinou-se uma a uma e foram obtidos os resultados apresentados na TABELA 4.6.

TABELA 4.6 – Pesos e bias das redes secundárias 3ª proposta.

Chave	Net-A	Net-B	Net-C	Net-D
iw_{11}	1.54551062015900	0.25902658634663	-0.19315668357620	0.33029277693395
iw_{12}	0.86378619896213	-0.24050634656418	0.38503518163849	0.86589175640465
iw_{13}	0.81652280893915	-1.34734004924228	0.25290171309188	0.44895545134545
iw_{14}	0.13882939521477	-1.54998829144672	-0.35667862673513	1.04133733105042
b_{11}	0.43398286002001	0.71153721332213	-0.86846504086335	0.07328594220423
b_{12}	0.05484647676547	-0.48361911170240	0.44446046799954	-0.32627355009277
b_{13}	-0.26116675586408	-0.85165991730397	-0.09871269408809	-0.03801852573852
b_{14}	-0.98919572412899	-0.92400208126825	0.18689875543684	-0.14607025126306
lw_1	0.41033525520790	-0.16333605823241	-0.62754592822840	-0.10786364981355
	1.53484287807244	-1.91841717561257	0.48514985169971	0.40647889542090
	0.58361739688375	-0.15712607684112	-0.29060031624009	0.24866145625698
lw_2	0.56254174137130	-0.96087645315096	0.65640500539179	0.74141822351953
	0.39016120007357	-1.62215586199499	0.65973221687626	1.01135787657722
	-0.05891815687074	-0.59446616691731	0.59839528634476	0.50618688875304
lw_3	-0.80104738785000	-0.77741832381187	-0.14872834618233	0.61256695383118
	0.87150409609577	-1.96674506483073	0.84641175469156	-0.74465004615953
	0.91699049521632	-0.89773919958698	0.21831826697662	0.50536496335317
lw_4	0.70241068557880	-0.34727816703311	0.15276717112739	-0.28594111214442
	0.91648961733835	-2.42119613848440	0.60428650970973	-0.33301505699199
	0.81840847049932	-1.11531018763646	-0.39585310388171	1.27079853091819
b_{21}	-0.71565547672133	0.42945630659217	-0.27856365693884	0.35744254552177
b_{22}	-1.68711047036515	0.78220011435019	-0.55469644215418	-0.69803420270915
b_{23}	0.58291915471560	0.83035738281943	0.39161303609192	-1.33460195824514

Nesta proposta, a mensagem a ser cifrada foi também divida em blocos iguais de 24 bits, os quais foram considerados individualmente com ***M***. No exemplo abaixo, observa-se como funciona esta proposta de cifrador, utilizando uma rede neural artificial principal e redes neurais secundárias.

Exemplo:

$$M_0 = 111111111111111111111111 = 16777215$$

$$a = C_0 = 16777215 \times 0.2668579 + 0.5746335 = 4477132.93$$

$$M_1 = 111111111111111111111111 = 16777215$$

$$b = C_1 = 16777215 \times 0.1503838 + - 0.6685121 = 2523020.67$$

$$M_2 = 111111111111111111111111 = 16777215$$

$$c = C_2 = 16777215 \times 0.9468040 + - 0.0950285 = 15884734.2$$

$$M_3 = 111111111111111111111111 = 16777215$$

$$d = C_3 = 16777215 \times 0.4572495 + 0.7685239 = 7671373.93$$

Manteve-se nesta proposta as mesma decisões quanto ao sinal e ao número de casas decimais da proposta anterior, bem como quanto ao tamanho do bloco de texto cifrado em 32 bits.

Ao analisar a 3ª proposta sob a luz das recomendações do NIST, verificou-se que ela também atende perfeitamente às mesmas, porém nesta também a fragilidade apresentada na primeira não existe em função da arquitetura proposta e do algoritmo de substituição do neurônio na camada escondida que envia o texto cifrado, o que elimina a linearidade da substituição do texto plano pelo texto cifrado.

Nesta proposta, utilizou-se quatro neurônios na camada escondida para facilitar a visualização do modelo, porém pode-se estender ao número de neurônio, elevando, assim, a sua segurança.

4.5 – Características e funcionalidades das três propostas

Notou-se como características das três propostas que elas são algoritmos de bloco com chave assimétrica, em que os pesos e bias da camada de entrada (chave para cifrar) têm um valor diferente dos pesos e bias da camada de saída (chave para decifrar); que não são necessários algoritmos especiais para gerar a chave, pois o próprio treinamento das rede neurais já gera as chaves diferentes sempre que se submete a rede a um novo processo de treinamento - TABELA 4.7-, bem como que não são necessários números primos muito grandes, fato que dificulta bastante o processo de escolha das

chaves. Tanto remetente quanto destinatário podem realizar o treinamento das redes neurais artificiais e distribuir as chaves (pesos e bias), as quais devem ser mantidas secretas.

TABELA 4.7 – Chaves geradas nos diferentes treinamentos 1ª proposta

Chaves	1º Treinamento	2º Treinamento	3º Treinamento
iw	-1.1847859	1.0548574	0.7577838
b_1	0.1533674	-0.4912096	0.5014847
lw	-0.8440343422147	0.94799543285452	-0.84403434221470
b_2	0.12944738285753	0.46566453049123	-0.66177811055938

O tamanho do bloco está diretamente ligado ao tamanho do texto cifrado em função da precisão necessária para recuperar o valor do texto plano. Caso se desejasse criptografar um bloco de 64 bits, o texto cifrado teria 80 bits. Essa é uma característica marcante nas propostas apresentadas , devido o acréscimo de 25% no tamanho do texto cifrado em relação ao texto plano.

A primeira proposta apresenta uma enorme fragilidade como já foi comentado, porém as outras duas são altamente seguras e com a implementação em computadores e outros dispositivos eletrônicos bastante simples, o algoritmo de treinamento atingia o erro desejado de 1e-25 em apenas 6 épocas, em poucos segundos.

Acredita-se que o cifrador neural vem para ampliar o leque de aplicações das redes neurais.

Capítulo 5

Criptoanálise usando redes neurais artificiais.

A necessidade de se avaliar nossas propostas e o alto grau de segurança do RSA e a simples implementação do mesmo, levou a avaliação da utilização das redes neurais artificiais na criptoanálise.

5.1 – Ataques ao RSA

Resolveu-se atacar o RSA em função de ser um algoritmo de chave pública, o que facilitaria em muito a obtenção de pares de texto plano e texto cifrado possibilitando um ataque do tipo texto plano escolhido. Também se acha muito interessante e conhecida a forma como são calculadas as chaves utilizadas neste famoso algoritmo de criptografia, o que também incentivou a realização a um ataque direto a elas. O desafio de realizar ataques a um dos mais importantes algoritmos criptográficos da atualidade despertou um interesse extra que fez utilização neste treabalho um enorme número de configurações de redes neurais artificiais e a todo momento pensar em ataques diferentes os quais se passará a relatar.

5.1.1 – Ataque direto às chaves do RSA

A primeira proposta de ataque recai na seguinte questão: sabendo o valor da chave pública ***(e)*** , uma rede neural artificial seria capaz de descobrir o valor da chave privada ***(d)*** ?

Para facilitar a construção dos diversos conjuntos de treinamentos necessários para o aprendizado das redes utilizadas neste trabalho, construíu-se uma aplicação em DELPHI - Figura 5.1- , que gera, de forma automática, chaves públicas ***(e)***, chaves privadas ***(p)*** e módulos ***(n)*** . Vale ressaltar que a aplicação escolhe de forma aleatória pares de números primos ***(p, q)*** menores que 2000, o que de certa forma reduz o número de possíveis chaves.

FIGURA 5.1 – Aplicação para gerar o conjunto de treinamento

O primeiro conjunto de treinamento deste trabalho tem 29 pares, sendo os valores das chaves públicas as entradas e os valores das chaves privadas as saídas, os quais foram transferidos para o MATLAB, software escolhido para realizar o treinamento e a avaliação dos resultados de nossas propostas.

Realizou-se muitos experimentos, e abaixo foram relacionados os mais significativos em forma de ficha.

1. Rede:

Tipo da rede:	Direta
Entradas:	1
Camadas escondidas:	1
Neurônios:	8
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	373872
Dados:	Decimal
Conclusão:	A rede não conseguiu apreender. Na próxima tentativa aumentar o número de neurônios da camada intermediária

2. Rede:

Tipo da rede:	Direta
Entradas:	1
Camadas escondidas:	1
Neurônios:	16
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	373872
Dados:	Decimal
Conclusão:	Tentou-se diminuir o erro mínimo, dobrando o número de neurônios, o que não deu certo.

3. Rede:

Tipo da rede:	Direta
Entradas:	1
Camadas escondidas:	2
Neurônios:	16
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	373872
Dados:	Decimal
Conclusão:	Mudou-se para duas camadas intermediárias, porém nada mudou em relação ao erro.

4. Rede:

Tipo da rede:	Direta
Entradas:	1
Camadas escondidas:	1
Neurônios:	64
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	373872
Dados:	Decimal
Conclusão:	Incrementou-se o número de neurônios da camada escondida, mas não se obteve a resposta desejada. Passou-se a usar uma rede recorrente.

5. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	1
Camadas escondidas:	1
Neurônios:	8
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	373872
Dados:	Decimal
Conclusão:	Na primeira tentativa, usando este tipo de rede, não se obteve sucesso, porém se acreditou que aumentando o número de neurônios nosso erro poderá diminuir.

6. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	1
Camadas escondidas:	1
Neurônios:	16
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	273903
Dados:	Decimal
Conclusão:	O erro diminui, porém está muito longe do desejado. Observando o ocorrido, procurou-se aumentar novamente o número de neurônios.

7. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	1
Camadas escondidas:	1
Neurônios:	32
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	398351
Dados:	Decimal
Conclusão:	O erro cresceu, o que prova que a decisão de aumentar o número de neurônios nem sempre é benéfica.

8. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	1
Camadas escondidas:	2
Neurônios:	8
Saídas:	1
Algoritmo de treinamento:	BFG
Erro mínimo obtido:	247895
Dados:	Decimal
Conclusão:	Mudou-se o número de camadas intermediárias, o erro diminui, porém muito longe do erro mínimo desejado.

As diversas redes testadas acima tinham como conjunto de treinamento entradas e saídas em decimal, e sofreram alterações nos números de neuônios, número de camadas e arquitetura, porém nenhuma delas conseguiu apreender os pares apresentados nos períodos de treinamento, o que levou a repensar o modelo e decidir pela mudança do conjunto de treinamento para binário, usando o comando do MATLAB - de2bi(p , 'left-msb') - então expusou-se o mesmo a novos treinamentos.

9. Rede:

Tipo da rede:	Direta
Entradas:	16
Camadas escondidas:	1
Neurônios:	16
Saídas:	16
Algoritmo de treinamento:	LM
Erro mínimo obtido:	2.86178e-031
Dados:	Binário

Os resultados obtidos na nona rede foram bastante animadores,pois, ao se dar entrada em qualquer um dos valores desse conjunto de treinamento, notou-se que a rede realmente apreendeu, então demos entrada em um valor que não constava do conjunto de treinamento – o par 137,47 - a rede devolveu como resposta 1075, logo notou-se que a mesma não conseguia extrapolar e interpolar. Também neste momento, observou-se que o conjunto de treinamento não poderia ser avaliado de forma correta, porque não se expôs à rede o valor do módulo, valor de extrema relevância para o RSA.

Fez-se as alterações necessárias para corrigir os problemas anteriores e abandonou-se, definitivamente, conjuntos com valores decimais; logo a seguir, reinicia-se a fase de treinamento de nossas redes.

10. Rede:

Tipo da rede:	Direta
Entradas:	16
Camadas escondidas:	1
Neurônios:	32
Saídas:	16
Algoritmo de treinamento:	GDX
Erro mínimo obtido:	3.64704e-011
Dados:	Binário

A décima rede, com a inclusão do módulo no conjunto de treinamento, teve o comportamento um pouco diferente da rede anterior, porém, ao trocar o algoritmo de treinamento pelo GDX, obteve-se o resultado bastante parecido com o anterior. Expôs-se os valores – 137,466 – e se obteve 1983, quando o valor esperado era 47.

11. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	16
Camadas escondidas:	1
Neurônios:	48
Saídas:	16
Algoritmo de treinamento:	GDX
Erro mínimo obtido:	4.82952e-011
Dados:	Binário

Na décima primeira rede, alterou-se a arquitetura da rede, porém se obteve o resultado bastante parecido com o anterior. Expôs-se os valores – 137,466 – e se obteve 80, quando o valor correto é 47.

Realizou-se muitas outras tentativas, no entanto os resultados sempre foram bastante desanimadores, o que leva à conclusão de que um ataque direto à chave do RSA, utilizando redes neurais artificiais, é bastante difícil ou quase impossível em função de a rede não conseguir interpolar ou extrapolar pontos da função observada na Figura 5.2 – que mostra o gráfico de nosso conjunto de treinamento.

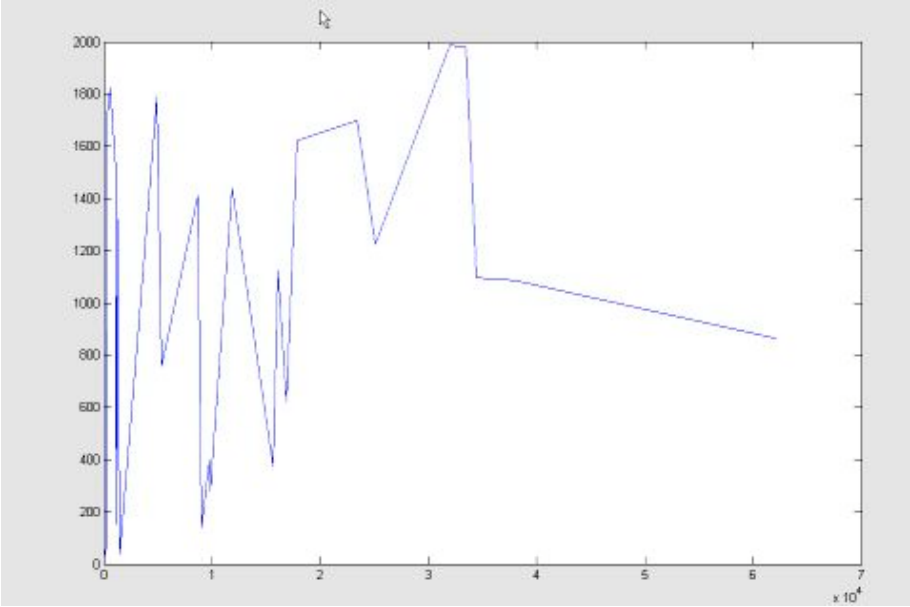

FIGURA 5.2 – Gráfico do conjunto de treinamento

5.1.1 – Ataque texto plano escolhido ao RSA

Como se sabe no RSA, o remetente possui a chave pública ***(e)*** e o módulo ***(n)*** , e por ser de conhecimento também do intruso, este pode gerar um grande conjunto de pares para realizar a criptoánalise.

Construí-se um cifrador RSA para gerar um conjunto de treinamento que será composto de pares de textos cifrados na entrada e textos planos na saída. Figura 5.3.

FIGURA 5.3 – Cifrador e decifrador RSA

O conjunto de treinamento tem 12 pares, sendo os texto cifrado as entradas (P) e texto plano as saídas (T), todos eles foram transferidos para o MATLAB e neste transformados em binário.

12. Rede:

Tipo da rede:	Direta
Entradas:	11
Camadas escondidas:	1
Neurônios:	16
Saídas:	11
Algoritmo de treinamento:	GDX
Erro mínimo obtido:	8.84698e-12
Dados:	Binário

A décima segunda rede também conseguiu apreender os pontos do conjunto de treinamento, porém não conseguiu interpolar e extrapolar. Notou-se que o problema desta rede poderia ser o diminuto conjunto de treinamento, então se resolveu ampliar o número de pares para 170.

13. Rede:

Tipo da rede:	Direta
Entradas:	11
Camadas escondidas:	1
Neurônios:	16
Saídas:	11
Algoritmo de treinamento:	GDX
Erro mínimo obtido:	6.26803e-5
Dados:	Binário

Apesar de ter ampliado o conjunto de treinamento, a rede não conseguiu extrapolar nem interpolar. Passou-se, então, à última tentativa que foi a troca da arquitetura direta por uma rede recorrente.

14. Rede:

Tipo da rede:	Recorrente – Elman
Entradas:	11
Camadas escondidas:	1
Neurônios:	128
Saídas:	11

Algoritmo de treinamento:	GDX
Erro mínimo obtido:	7.44623-6
Dados:	Binário

Após longo período tentando um ataque que realmente conseguisse decifrar o RSA , acredita-se que, para este tipo de algoritmo de criptografia as redes neurais diretas e de Elman não são eficientes.

5.2 – Ataques aos cifradores neurais

Achou-se que seria de suma importância para este trabalho realizar a criptoanálise dessas propostas, utilizando o tipo de ataque texto plano escolhido e a mesma metodologia dos ataques feito ao RSA.

Nas próximas seções, serão descritos os resultados e os detalhes dos ataques realizados a cada uma dessas propostas.

5.2.1 – Ataques à primeira proposta

Desenvolveu-se um cifrador com os resultados obtidos na primeira proposta do capítulo 4°, Figura 5.4 , o qual gerou um conjunto de treinamento, que tem 40 pares, sendo o texto cifrado a entrada e o texto plano a saída, para isto foram transferidos os valores para o MATLAB.

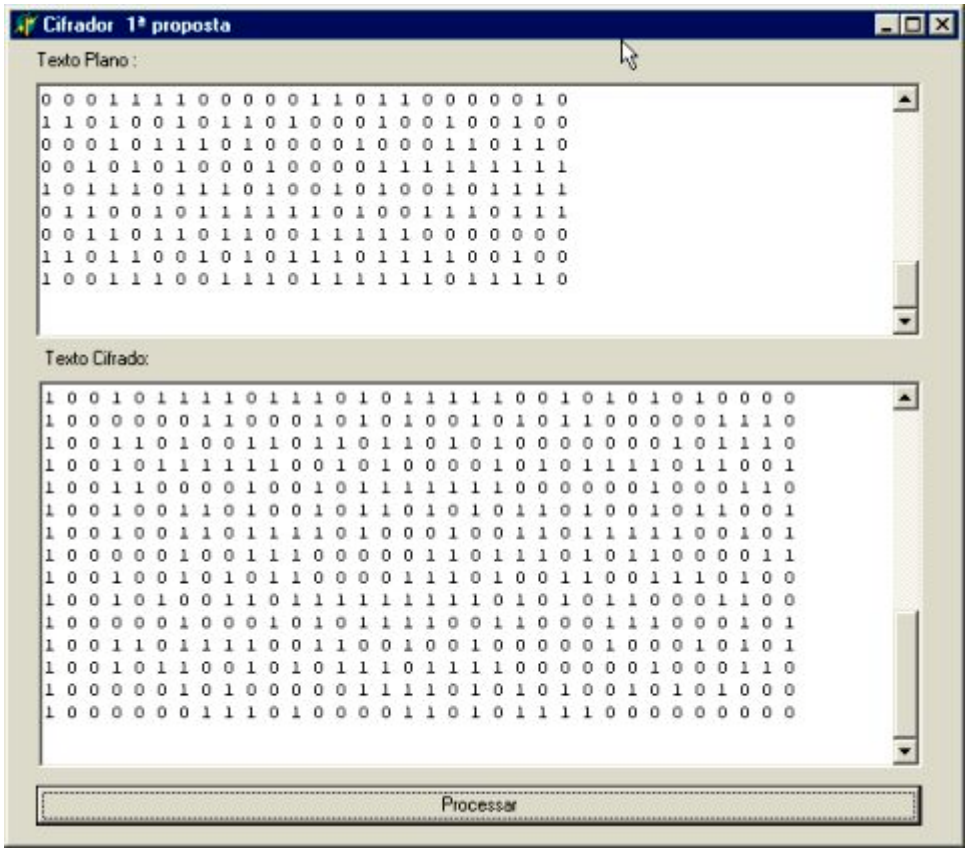

FIGURA 5.4 – Cifrador neural

Para o primeiro ataque, utilizou-se um rede direta com 24 neurônios na camada de entrada, 48 em uma única camada intermediaria e 34 neurônios na camada de saída. Como funções de transferência utilizou-se a tangente hiperbólica e linear respectivamente. Como algoritmo de treinamento utilizou-se o GDX. Abaixo estão apresentados os resultados obtidos:

Após 10000 épocas foi obtido o erro 2.02195e-005, erro bastante satisfatório, porém a rede voltou a decepcionar, mesmo sendo esta primeira proposta de cifrador bastante simples e de fácil criptoanálise, ela somente apreendeu os pontos não conseguindo interpolar e extrapolar.

Utilizando os mesmos parâmetros, só alterando o tipo da rede por uma rede de Elman, voltou-se aos treinamentos. Foi obtido o erro de 3.31217e-006 após 10000 épocas, a rede conseguiu novamente apreender, porém não resolveu o problema proposto.

Com os resultados decepcionantes na utilização de redes neurais diretas e recorrentes para realizar ataques (no que concerne ao algoritmo consagrado como o

RSA, quanto à nossa proposta mais simples de cifrador) não foram atacadas as duas outras propostas, por entender que elas são mais fortes que a primeira.

Capítulo 6

Conclusão

A incontestável necessidade de pesquisa para prover mais segurança nas comunicações em um mundo que cada vez mais está conectado levou a abordagem sobre este interessante tema neste trabalho.

Ao longo da pesquisa procurou-se exaustivamente e sem sucesso por bibliografia, que levasse a um caminho para construção de cifradores utilizando redes neurais, dificuldade esta muito em razão da natureza da aplicação dos mesmo, que geralmente se tornam públicos, quando obsoletos, ou quando falham.

Os resultados obtidos na criação de um cifrador foram bastante expressivos, por apresentarem uma perspectiva nova em relação aos algoritmos tradicionais que dependem de geradores de chaves e da busca de números primos muito grandes. Como principal desvantagem, pode-se destacar o acréscimo de 25% em relação ao tamanho do texto plano e ao texto cifrado, mas também vale ressaltar que a primeira proposta tem um grau de segurança bastante limitado em função da linearidade da substituição da mensagem pelo texto cifrado. A segunda e a terceira propostas apresentam uma substituição não linear e alternada, ou seja ao criptografar mensagens iguais do mesmo tamanho do bloco, obtêm-se textos cifrados diferentes, o que eleva bastante o grau de segurança em relação inclusive a algoritmos tradicionais.

Vale destacar que a terceira proposta é a mais complexa e a mais segura de todas, e que foi usado um modelo com quatro neurônios para simplificar o estudo. Caso se faça necessário, pode-se aumentar este número elevando, assim, ainda mais o grau de segurança.

A proposta para realizar criptoanálise utilizando redes neurais, foi um verdadeiro fracasso: a rede não apreendia e quando apreendia não interpolava ou extrapolava. Após centenas de experimentos, ao longo deste ano, pode-se afirmar, que as redes neurais diretas e a rede recorrente de Elman apresentam fortes indícios de não

serem instrumentos apropriados para realizar ataques a algoritmos de criptografia, em razão da superfície da função ser descontínua.

Sendo este um primeiro trabalho, acredita-se que existam muitas pesquisas a serem realizadas em breve, e que terão como objetivo estudos para estabelecer o limite de crescimento do número de neurônios na camada intermediária da terceira proposta; utilização de redes neurais para identificar mensagens escondidas em fotografias digitais – Esteganografia; novos modelos neuronais para ataques ao algoritmo RSA e criptoanálise do **AES** - Advanced Encryption System (NIST, 1998) – novo algoritmo criptográfico recomendado pelo NIST.

Referencias bibliográficas

1. BARRETO, Jorge Muniz – Inteligênica Artificial No limiar do Século XXI . Florianopolis - Duplic , 1999

2. BITTENCOURT, Guilherme – Inteligência Artificial: ferramentas e teorias – Florianópolis : Editora da USFC, 1998

3. CARDOSO, Silvia Helena – Comunicação entre as células nervosas. Disponível em:< http://www.epub.org.br/cm/n12/fundamentos/neurotransmissores> Acesso em: 16 nov. 2001.

4. CARVALHO, Daniel Balparda de – *Criptografia: metodos e algoritmos*. Book Express, 2000

5. DIFFIE, W. The first ten years of public-key cryptography. In: *Proceedings of the IEEE*, v.76, Maio de 1988.

6. HAYKIN, Simon – *Redes Neurais princípios e prática – 2ª Edição*. Bookman, 2001

7. HOLANDA, Aurélio B. – *Dicionáio Aurélio Eletrônico – Século XXI*. Lexikon Informática Ltda, 1999 – 1 CD-ROM

8. MICROSOFT Enciclopédia Encarta 99, versão 1.0 : Software educacional [S.E.]: Microsoft Corporation, 1999. 1 CD-ROM

9. MINSKY, M. L. and PAPERT, S. A. – *Perceptrons: An Introduction to Computacional Geometry . MIT Press, 1969*

10. NBS - National Bureau of Standard – *Data Encryption Standard*. 1977

11. NIST - National Institute of Standards and Technology - *Advanced Ecryption System* . 1998

12. RIVEST, R.L. , SHAMIR, A. and ADLEMAN, L. *A method for obtaining digital signature and public-key cryptosystems* , 1976

13. RUSSELL, Stuart J. and NORVIG, Peter - *Artificial intelligence: a modern approach*. Prentice Hall, 1995

14. SCHNEIER, B. *Applied Cryptography*. New York: John Wiley & Sons, 2. ed., 1996.

15. SKUBISZEWSKI, Marcin – A hardware emulator for binary neural networks. Digital, 2001

16. SMITH, R. E. *Internet Cryptography*. [S.l.]: Addison Wesley Longman, 1997.

17. STALLINGS, W. *Cryptography and Network Security*. Upper Saddle River: Prentice-Hall, 2. ed., 1999.

18. STINSON, D. R. *Cryptography – Theory and Practice*. Boca Raton: CRC Press, 1995.

Apêndice A

MatLab - NNET

Funções para treinamento

Função	Descrição
Trainbfg	BFGS quasi-Newton retropropagação (backpropagation)
Trainbr	Regularização Bayesiana
Traincgb	Powell-Beale retropropagação (backpropagation) com gradiente conjugado
Traincgf	Fletcher-Powell retropropagação (backpropagation) com gradiente conjugado
Traincgp	Polak-Ribiere retropropagação (backpropagation) com gradiente conjugado
Traingd	Retropropagação (backpropagation) com gradiente descendente
Traingda	Retropropagação (backpropagation) com gradiente conjugado e taxa de aprendizado adaptativa
Traingdm	Retropropagação (backpropagation) com gradiente conjugado com momentum
Traingdx	Retropropagação (backpropagation) com gradiente conjugado , taxa de aprendizado adaptativa e momentum
Trainlm	Levenberg-Marquard retropropagação (backpropagation)
Trainscg	Retropropagação (backpropagation) com gradiente escalar
Trainwb	Função de treinamento por pesos e bias
Trainwb1	Função de treinamento por pesos e bias um vetor por vez

Funções de transferência

Função	Descrição
Compet	Função de transferência por competição
Hardlim	Função de transferência por limite rígido
Hardlims	Função de transferência por limite rígido e simétrico
Logsig	Função de transferência sigmoidal
Poslin	Função de transferência linear positiva
Purelin	Função de transferência linear
Radbas	Função de transferência base radial
Satlin	Função de transferência por saturação linear
Satlins	Função de transferência por saturação linear simétrica
Tansig	Função de transferência tangente hiperbólica
Tribas	Função de transferência base triangular