Escola Politécnica da Universidade de São Paulo

Projeto de Formatura 1999

Codificação e Decodificação Digital de Dados

Pedro Hancon Cavalcanti Yue

Nº USP: 1449893
End.: Rua Anunze, 181 Tel.: 835-3735
E-mail: *pyue@larc.usp.br*

Rogério Luiz Pereira

No USP: 1932777
End.: Rua Napoleão de Barros, 420 ap. 44 Tel.: 549-4477
E-mail: *lpereira@colband.com.br*

Tiago Zanco

No USP: 2046669
End.: R. Martinho da Silva, 107 ap. 42 Tel.: 5061-0461
E-mail: *tzanco@hotmail.com*

Orientador: Prof. Dr. Jorge Kinoshita

V.1

→ anexar documentação Dallas.
- timer 2 do 8051
- não full-duplex → sim
- descrição

Índice

1 Objetivos

O objetivo deste projeto é a implementação de um Sistema de Transmissão de Dados Seguro entre dois pontos. Como segurança entende-se, no contexto deste projeto, a proteção dos dados transmitidos contra leitura por pessoas não autorizadas e a garantia da autenticidade das mensagens.

O objetivo é alcançado através da codificação dos dados transmitidos, de forma a dificultar sua interpretação caso sejam interceptados. Como o sistema utiliza uma chave para codificação, a qual deve ser previamente combinada entre as pessoas que vão se comunicar, a utilização da chave correta garante que qualquer mensagem recebida foi realmente transmitida pela pessoa que se espera.

2 Especificação Funcional

O Sistema de Transmissão de Dados Seguro implementado é constituído por dois microcomputadores pessoais IBM-PC compatíveis, conectados a Módulos Codificadores, que, por sua vez, são conectados entre si, como apresentado na Figura 1. Os módulos são idênticos entre si.

A função básica do sistema é garantir que o canal de comunicação entre os módulos é seguro, ou seja, garantir que por ele só trafegam dados codificados. Além disso, o sistema também oferece aos usuários uma interface para envio e recepção dos dados e configuração e controle da comunicação.

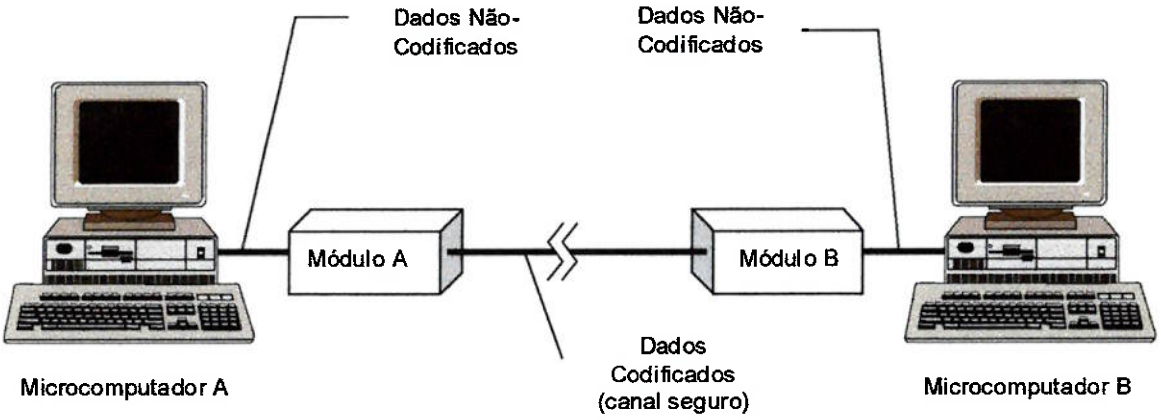


Figura 1 - Sistema de Transmissão Segura de Dados

O projeto constitui-se na implementação do módulo codificador, cuja função é basicamente codificar todo dado transmitido e decodificar todo dado recebido utilizando-se de um algoritmo aberto de codificação, o Triple DES. O protocolo físico de comunicação é o RS-232.

Além dos módulos, o projeto inclui também o desenvolvimento do software dos microcomputadores, que oferece aos usuários a interface para uso do sistema. Para que a comunicação possa ser controlada, um protocolo de comunicação foi implementado nos módulos e nos microcomputadores.

Nos próximos capítulos será apresentado um panorama histórico da segurança em redes de transmissão de dados e da codificação de dados. Será também apresentado o algoritmo de codificação a ser implementado.

3 Segurança na Rede

Nos seus primeiros anos de existência, redes de computadores eram usadas principalmente por pesquisadores universitários para troca de mensagens e por usuários corporativos para compartilhamento de impressoras. Nestas circunstâncias, o fator segurança nunca recebeu muita atenção. Mas agora, com a disseminação do uso de redes de computadores com os mais diversos fins e com o advento da superpovoada rede mundial de computadores, a Internet, o assunto segurança adquiriu mais importância.

Além da necessidade de segurança exigida pelas corporações na transmissão de dados sobre seus negócios, milhões de cidadãos estão conectados à Rede, para fazer transações bancárias, para fazer compras, para entregar a declaração de imposto de renda, entre outras atividades que exigem que os dados transmitidos sejam mantidos em sigilo. Dessa forma, uma garantia de privacidade e segurança nestas transações tornou-se indispensável.

Por segurança podemos compreender uma vasta gama de assuntos. Na sua forma mais simples, por segurança entende-se evitar que pessoas estranhas leiam ou alterem as mensagens trafegando pela rede, ou por evitar que pessoas entrem em áreas de acesso restrito, isto é, para as quais não são autorizadas. Entende-se também por ter garantias de que uma mensagem recebida origina-se realmente da pessoa que a assina.

Considerando estes pontos, podemos dividir o problema de segurança em quatro grandes áreas: securidade, autenticação, validação e controle de integridade.

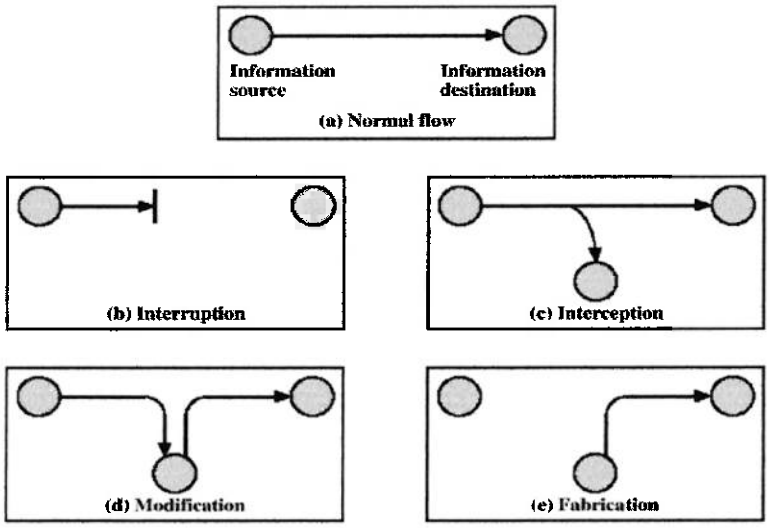


Figura 2 - Ameaças à Segurança

Securidade tem a ver com a garantia de que pessoas estranhas não tenham acesso a determinadas mensagens; *autenticação* se refere a saber se a pessoa com quem estamos nos comunicando é mesmo a pessoa que pensamos ser; *validação* está associada a garantir que uma

mensagem recebida vem integralmente de um remetente conhecido; e, finalmente, por *controle de integridade* entende-se saber se a mensagem recebida foi alterada maliciosamente por um estranho entre o destinatário e o remetente.

Com relação às áreas de securidade e autenticação, para estabelecer uma transmissão segura de dados numa rede podem ser utilizados mecanismos de codificação e decodificação, garantindo, ao menos, que uma mensagem interceptada por um estranho não possa ser entendida ou modificada.

3.1 Criptografia tradicional

A *criptografia* compreende os métodos de codificação e decodificação de mensagens através da aplicação de um algoritmo de transformação. A arte de tentar descobrir a mensagem original a partir da mensagem cifrada é denominada *criptoanálise*.

Historicamente, quatro grupos de interesse têm contribuído significativamente com a arte da criptografia: os militares, os defensores da privacidade, os profissionais da área de segurança de dados e os amantes do assunto. Entre eles, destaca-se a atuação dos militares.

Inicialmente, nas organizações militares, as mensagens a serem transmitidas de forma segura eram passadas a poucos profissionais, que se encarregavam de codificá-las, muitas vezes de forma totalmente manual. Entretanto, com o advento do computador e de novos e mais complexos algoritmos de codificação, era necessário um número cada vez maior de especialistas para lidar com estas máquinas. Sempre que se mudava de um algoritmo de codificação para outro, todos os responsáveis tinham que receber novo treinamento. Além disso, caso uma dessas pessoas fosse capturada pelo inimigo, o algoritmo de codificação deveria ser rapidamente alterado.

Para contornar esses problemas, um novo modelo de criptografia foi proposto, o qual ficou conhecido como *Sistema de Criptografia Tradicional*.

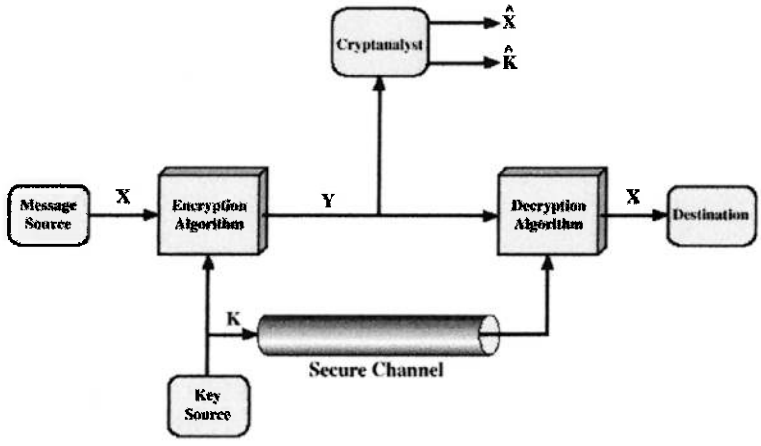


Figura 3 - Modelo de um Sistema de Criptografia Tradicional

Neste modelo, a função que codificará a mensagem pura, gerando a mensagem em cifras (*ciphertext*), é parametrizada por uma chave (*encryption key*). Da mesma forma, a função de decodificação também possui sua própria chave (*decryption key*). O princípio fundamental da cripto-

grafia tradicional admite que o criptoanalista pode ouvir e copiar precisamente todas as mensagens em cifras passadas pelo remetente e que ele conhece o algoritmo de criptografia utilizado. Entretanto, ao contrário do destinatário pretendido, ele não sabe quais são as chaves de codificação e decodificação e, portanto, não pode entender ou alterar de forma não-detectável a mensagem original.

Para mostrar a relação entre mensagens puras, cifradas e chaves adotar-se-á uma notação. Por essa notação, C = Ek (P) significa que a codificação da mensagem pura P, através do uso do algoritmo E com a chave K, resulta numa mensagem em cifras C. Similarmente, P = Dk' (C) representa a decodificação de uma mensagem em cifras através de uma chave K', resultando na mensagem pura P original. Como esta notação sugere, E e D são funções matemáticas com dois parâmetros – a chave e a mensagem a ser codificada ou decodificada.

Antes da utilização do Sistema de Criptografia Tradicional, era um grande problema a quantidade de tempo necessária para elaborar, testar e instalar um novo algoritmo de codificação sempre que o anterior se tornava obsoleto ou era comprometido, já que o algoritmo deveria ser totalmente modificado, o que acarretava a necessidade de treinar novamente as pessoas envolvidas e adaptar os equipamentos de codificação.

Esse problema foi solucionado com o uso da chave de codificação. A chave consiste de uma seqüência arbitrária de bits, que selecionam um dentre vários potenciais códigos de codificação possibilitados pelo algoritmo adotado. A chave pode ser modificada sempre que necessário, bastando apenas informar a mudança às pessoas com quem se faz contato.

Dessa forma, o grau de segurança de um sistema de transmissão de dados não está baseado no sigilo do algoritmo de codificação adotado, e sim no sigilo da chave de codificação utilizada e no comprimento (número de bits) dessa chave. Considere-se, por exemplo, uma chave formada por uma seqüência de bits e uma mensagem a ser codificada. Uma chave de 16 bits de comprimento proporciona uma variedade de $2^{16} = 65536$ possibilidades de mensagens codificadas; uma chave de 24 bits de comprimento oferece mais de 16 milhões de possibilidades. Assim, quanto maior a chave, maior o fator de trabalho (*work factor*) que o criptoanalista tem para quebrar a mensagem. Esse fator de trabalho é exponencialmente proporcional ao comprimento da chave. É necessário também que o algoritmo seja sensível à mudanças na chave, isto é, uma variação de apenas 1 bit na chave (ou na mensagem a ser codificada) deve resultar numa mensagem em cifras totalmente diferente.

Logo, a securidade de uma rede reside na existência de um sólido algoritmo de codificação, que pode ser público, e de uma chave de comprimento longo, mantida em sigilo.

Os algoritmos de codificação têm historicamente sido divididos em 2 categorias principais: substituição de cifras (*substitution cypher*) e transposição de cifras (*transposition cypher*). Eles também podem ser divididos em algoritmos simétricos, que utilizam uma mesma chave para a codificação e para a decodificação e em algoritmos de chave pública e privada, que utilizam uma chave diferente para a codificação (chave pública) e para a decodificação (chave privada). Os algoritmos de chave pública e privada são muito usados na Internet.

Nos algoritmos de substituição de cifras cada caractere ou grupo de caracteres é trocado por um outro caractere ou grupo de caracteres. Um dos métodos de substituição de cifras mais utilizado é atribuído a *Julius Caesar*, e é denominado *Caesar Cypher*. Neste método, o caractere *a* é substituído por *D*, *b* por *E*, *c* por *F*... e *z* por *C*. Por exemplo, temos a seguinte conversão para a palavra "*attach*":

attach → DWWDFN

Uma melhoria neste método pode ser alcançada através da utilização de uma técnica chamada de substituição monoalfabética, com a chave a ser utilizada sendo um string de caracteres de comprimento 26, correspondendo ao alfabeto completo.

Assim, temos:

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

Já na técnica de transposição de cifras temos também a reordenação dos caracteres e não somente sua substituição como no caso anterior. O algoritmo DES, que se pretende implementar neste projeto, é desse tipo e será descrito adiante.

4 Descrição do algoritmo DES

O algoritmo a ser utilizado no projeto será baseado no algoritmo DES – *Data Encryption Standard* – que é o algoritmo de criptografia convencional mais utilizado no mundo.

Embora muitos outros algoritmos baseados na modelo da criptografia convencional tenham sido criados, o DES foi o mais bem sucedido, e hoje é utilizado em transações bancárias, comércio eletrônico através da Internet e em aplicações que necessitam de alto grau de segurança.

Para facilitar a descrição do DES, optou-se pela descrição de uma versão didática, simplificada, do algoritmo, chamada S-DES (Symplified DES). Essa versão permite maior entendimento de como é feita tanto a codificação como a decodificação, assim como dos detalhes do funcionamento do algoritmo. Feita esta descrição, pode-se partir para a versão completa do DES, e, finalmente, para o Triple DES.

4.1 Descrição do algoritmo S-DES

S-DES é considerado como um algoritmo didático, não como um algoritmo de criptografia seguro. Entretanto, o seu funcionamento global é idêntico ao do DES. O S-DES foi desenvolvido pelo professor Edward Schaefer, da Universidade de Santa Clara.

A Figura 4 descreve a estrutura geral do S-DES. O algoritmo recebe como entrada uma mensagem pura (*plaintext*) de 8 bits e uma chave de 10 bits e produz na saída um bloco de 8 bits de mensagem codificada (*cyphertext*). O algoritmo de decodificação, igualmente, recebe como entrada um bloco de 8 bits (*cyphertext*), a mesma chave de 10 bits e produz na saída outro bloco de 8 bits, contendo a mensagem original. O S-DES é, portanto, um algoritmo simétrico.

O algoritmo envolve 5 etapas de transformação:

- uma permutação inicial (IP);
- uma função complexa chamada f_k, a qual envolve tanto operações de permutação como substituição e utiliza a chave k_1;
- uma função simples de permutação que permuta as duas metades do bloco de bits;
- a função f_k novamente, agora com a chave k_2;
- e uma função de permutação, que é a inversa da função de permutação inicial.

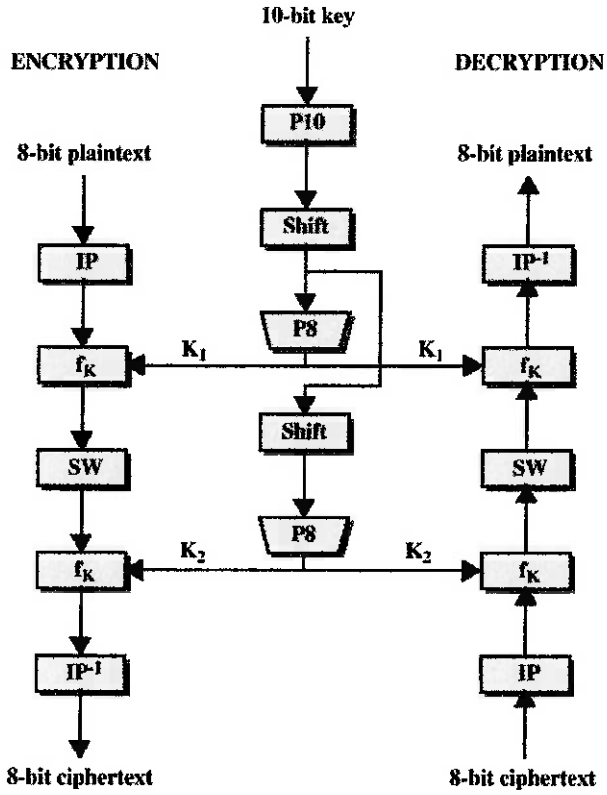


Figura 4 - Esquema do DES Simplificado

As várias permutações e substituições em que o algoritmo incorre têm como objetivo torna-lo complexo e, portanto, difícil de ser criptoanalisado.

A função f_k tem como entrada os bits da mensagem codificada e uma chave de 8 bits, gerada a partir de transformações aplicadas na chave de 10 bits fornecida como entrada para o algoritmo. A chave de 10 bits é submetida a uma função de permutação (P10), seguida de uma operação de deslocamento (*shift*); a saída desta operação é passada então a outra função de permutação que produz uma saída de 8 bits (P8), a qual constitui a primeira chave (k_1). A saída da operação de deslocamento também serve como entrada para outra função de permutação (P8) para produzir a segunda chave (k_2).

Pode-se sucintamente expressar o algoritmo como uma composição de funções:

$$IP^{-1} \times f_{k2} \times SW \times f_{k1} \times IP$$

Que também pode ser escrita como

Mensagem codificada = IP-1 (fk2(SW(fk1(IP(mensagem)))))

Onde

K1 = P8(Shift(P10(key)))

K2 = P8(Shift(Shift(P10(key))))

O algoritmo de decodificação também é mostrado na Figura 4 e é essencialmente o inverso da codificação:

Mensagem = IP-1 (fk1(SW(fk2(IP(mensagem codificada)))))

Passa-se agora a análise de cada elemento do S-DES em maiores detalhes.

4.1.1 Geração da chave S-DES

O algoritmo S-DES depende da utilização de uma chave de 10 bits tanto pelo emissor como pelo receptor da mensagem. A partir desta chave, duas novas chaves de 8 bits são geradas para uso particular de outros estágios do algoritmo.

Inicialmente, permuta-se a chave. Supondo-se que ela seja da forma $(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_{10})$, a permutação é definida como:

$$P10(k_1,k_2,k_3,k_4,k_5,k_6,k_7,k_8,k_9,k_{10}) = (k_3,k_5,k_2,k_7,k_4,k_{10},k_1,k_9,k_8,k_6)$$

Desta forma, a permutação bit a bit é feita da seguinte maneira:

P10									
3	5	2	7	4	10	1	9	8	6

Para ilustrar o processo, seja a chave dada por (1010000010). A chave permutada fica sendo (1000001100).

A seguir é feito um deslocamento (shift) à esquerda (LS-1) separadamente para os 5 primeiros bits e para os 5 bits finais. No nosso exemplo teríamos como resultado (00001 11000). Segue-se a função P8, que permuta 8 dos 10 bits de acordo com a seguinte regra de permutação:

P8							
6	3	7	4	8	5	10	9

O resultado é a chave k_1 (10100100).

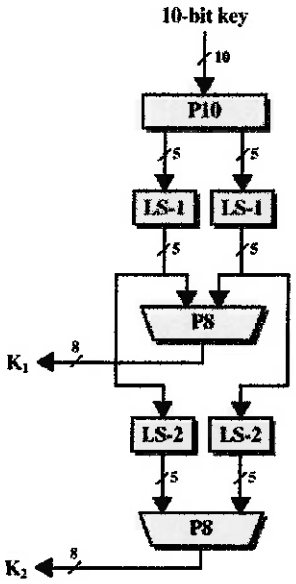


Figura 5 - Geração da Chave para o DES Simplificado

Para obter a chave k_2, voltamos aos pares de 5 bits produzidos pelas funções LS-1 e fazemos um deslocamento à esquerda (*left shift*) de 2 posições para cada par. No nosso exemplo, o bloco (00001 11000) torna-se (00100 00011). Finalmente, a função P8 é aplicada novamente para produzir k_2. No nosso exemplo a chave k_2 é dada por (01000011).

4.1.2 Codificação S-DES

A Figura 6 mostra o algoritmo de codificação S-DES em detalhes. A codificação envolve 3 transformações básicas, aplicadas em 5 etapas.

Transformação 1: Permutações inicial e final

A entrada do algoritmo é um bloco de 8 bits, o qual é permutado na etapa inicial usando a função IP:

IP							
2	6	3	1	4	8	5	7

Esta função mantém os 8 bits originais, mas faz a permutação entre eles. Na etapa final do algoritmo a permutação inversa é utilizada:

IP^{-1}							
4	1	3	5	7	2	8	6

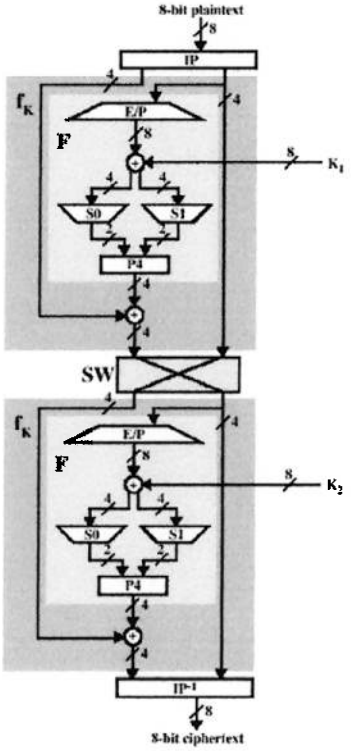


Figura 6 - Detalhes da Codificação do DES Simplificado

Transformação 2: Função f_k

Esta é a função mais complexa do algoritmo S-DES. Ela consiste de uma combinação de funções de permutação e substituição. Fazendo L e R conter os 4 bits mais e menos significativos da entrada de 8 bits, e sendo F uma função de mapeamento (não necessariamente um-para-um) de 4 em 4 bits, temos:

$$F_k (L,R) = (L \oplus F(R,SK),R)$$

Onde SK é uma sub-chave e ⊕ representa a função ou-exclusivo. Por exemplo, supondo que a saída do estágio IP é (10111101) e F(1101,SK) = (1110) para alguma chave SK. Então f_k(10111101) = (01011101) pois (1011) ⊕ (1110) = (0101).

A função mapeadora F recebe um número de 4 bits (n1 n2 n3 n4). A primeira operação é uma expansão/permutação:

E/P							
4	1	2	3	2	3	4	1

O resultado pode ser colocado na seguinte forma:

n4 n1 n2 n3

n2 n3 n4 n1

A chave $k_1 = (k_{11},k_{12},k_{13},k_{14},k_{15},k_{16},k_{17},k_{18})$ é adicionada a este valor através da função ou-exclusivo (XOR):

n4 + k11	n1 + k12	n2 + k13	n3 + k14
n2 + k15	n3 + k16	n4 + k17	n1 + k18

Renomeando os 8 bits:

p0,0	p0,1	p0,2	p0,3
p1,0	p1,1	p1,2	p1,3

Os primeiros 4 bits (primeira linha da matriz anterior) são alimentados numa caixa S (*S-box*) S0 para produzir uma saída de 2 bits, e os 4 bits remanescentes (segunda linha) são alimentados em S1 para produzir outros 2 bits de saída. Estas 2 caixas S0 e S1 são definidas como segue:

	0	1	2	3
0	1	0	3	2
$S0 = 1$	3	2	1	0
2	0	2	1	3
3	3	1	3	2

	0	1	2	3
0	0	1	2	3
$S1 = 1$	2	0	1	3
2	3	0	1	0
3	2	1	0	3

As *S-boxes* operam da seguinte maneira: o primeiro e o quarto bits de entrada são tratados como números de 2 bits que especificam uma linha da *S-box*, e os segundo e terceiro bits especificam uma coluna da *S-box*. A entrada naquela linha e coluna, na base 2, é a saída de 2 bits. Por exemplo, se (p0,0; p0,3) = (00) e (p0,1;p0,2) = (10), então a saída vem da linha 0, coluna 2 de S0, a qual é 3, ou (11) em binário. Igualmente, (p1,0; p1,3) e (p1,1;p1,2) são usados para a produção de 2 bits adicionais.

A seguir, os 4 bits produzidos por S0 e S1 são permutados pela função P4 como segue:

P4			
2	4	3	1

A saída da função P4 é a saída da função mapeadora F.

Transformação 3: A função SW

A função f_k apenas altera os 4 bits mais significativos da entrada. A função de troca SW, aplicada na etapa intermediária do algoritmo, inverte os 4 bits mais e menos significativos de forma que a segunda instância de f_k opera em 4 bits diferentes. Nesta segunda instância, as funções E/P, S0, S1 e P4 são idênticas, mas a chave de entrada é a sub-chave k_2.

4.2 Análise do algoritmo S-DES

Em termos de probabilidade de eventuais quebras de privacidade da mensagem, o S-DES é bastante vulnerável. Com uma chave de 10 bits, há apenas $2^{10} = 1024$ variações da chave. Dada uma mensagem codificada, um intruso pode tentar cada possibilidade e analisar o resultado para determinar uma possível mensagem decodificada válida. Na prática, o S-DES corresponde a um algoritmo de substituição de cifras, por isso tem apenas utilidade didática.

4.3 Relação do algoritmo S-DES com o DES

O algoritmo DES utiliza as mesmas transformações básicas do S-DES, aplicando-as, porém, em blocos de entrada de 64 bits. O esquema de codificação da mensagem pura pode ser definido como:

$$\mathbf{IP^{-1} \times f_{k16} \times SW \times f_{k15} \times SW \times \ldots \times SW \times f_{k1} \times IP}$$

A chave utilizada é de 56 bits e dela são calculadas outras 16 chaves de 48 bits cada para uso nas funções f_k. Na geração dessas outras chaves, aplica-se uma permutação inicial de 56 bits seguida por uma seqüência de deslocamentos e permutações.

Internamente ao algoritmo, ao invés de F atuar em 4 bits (n1 n2 n3 n4) como no S-DES, atua em 32 bits (n1...n32). Após a expansão/permutação inicial, a saída de 48 bits pode ser esquematizada como:

n32	n1	n2	n3	n4	n5
n4	n5	n6	n7	n8	n9
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
n28	n29	n30	n31	n32	n1

Esta matriz é adicionada (ou-exclusivo) a uma chave de 48 bits. Existem 8 linhas, correspondendo a 8 *S-boxes*. Cada *S-box* contém 4 linhas e 16 colunas. O primeiro e último bit de uma linha da matriz anterior selecionam uma linha numa *S-box*, e os 4 bits centrais selecionam uma coluna.

4.4 O Algoritmo DES (Data Encryption Standard)

O método de criptografia mais utilizado mundialmente é baseado no algoritmo DES (*Data Encryption Standard*), adotado em 1977 pelo *National Bureau of Standards*.

No DES, conforme descrito anteriormente, os dados são codificados em blocos de 64 bits usando uma chave de 56 bits. O algoritmo transforma os 64 bits de entrada em 64 bits de mensagem cifrada na saída através de várias etapas sucessivas de transformação. As mesmas etapas, numa ordem diferente, com a mesma chave, são usadas na decodificação.

Embora tenha sido alvo de críticas ferrenhas, como o uso de uma chave de apenas 56 bits e o conhecimento público das *S-Boxes*, o algoritmo é utilizado mundialmente em aplicações financeiras e comerciais desde 1944 e até hoje seu uso tem sido intenso.

4.5 Codificação DES

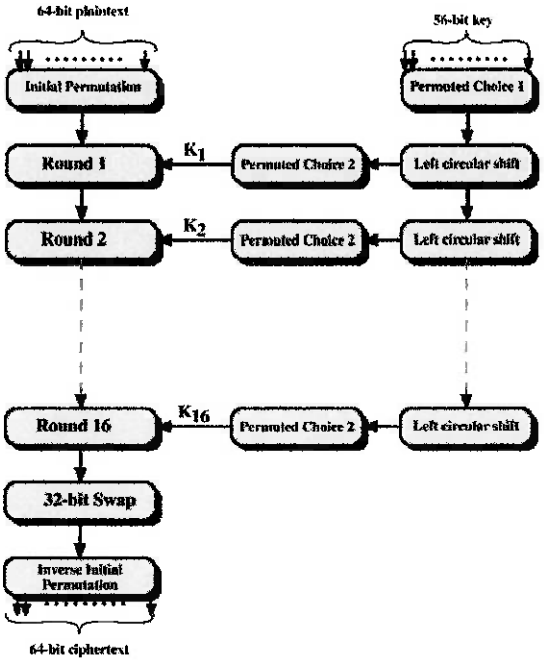


Figura 7 - Esquema Geral do Algoritmo de Codificação DES

O esquema geral da codificação DES é mostrado na Figura 7. Assim como qualquer método, ele tem como entradas a mensagem original e a chave de codificação. Neste caso, a mensagem contém 64 bits e a chave K possui 56 bits.

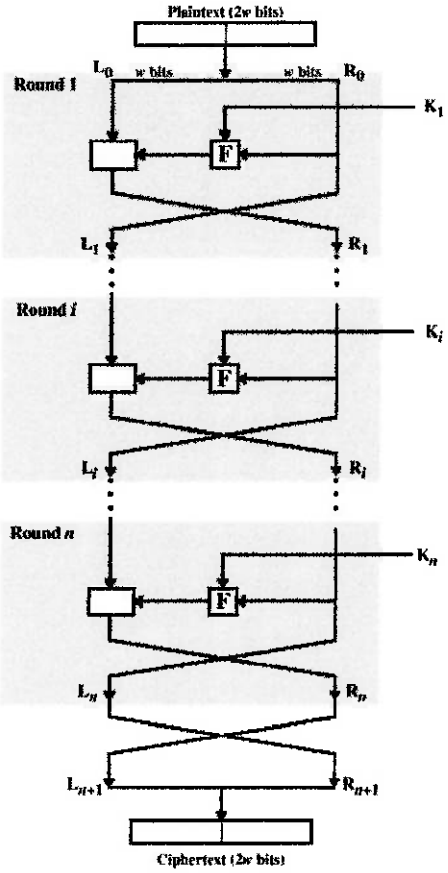


Figura 8 - Rede de Feistel Clássica

Observando a parte esquerda da figura, vemos que o processamento da mensagem é feito em 3 fases. Inicialmente, os 64 bits da mensagem sofrem uma permutação inicial (IP) que rearranja os bits, de forma a produzir uma nova seqüência de bits. A seguir temos uma seqüência de 16 rodadas de uma mesma função, contendo permutações e substituições. A saída do último estágio (décimo sexto estágio) consiste de um bloco de 64 bits que é uma função da entrada e da chave. A seguir temos uma troca entre os 32 bits mais e menos significativos (32-bit swap). Finalmente, os bits são passados para uma função de permutação, para produzir o bloco de 64 bits finais, que constituem a mensagem codificada. Com a exceção das fases inicial e final, todas as outras funções têm a mesma estrutura da codificação Feistel, esquematizada na Figura 8.

A parte direita mostra como a chave de 56 bits é tratada. Inicialmente, a chave é permutada. Logo após, para cada uma das 16 rodadas de codificação, uma sub-chave K_i é produzida por uma combinação de deslocamento de bits à esquerda com permutação. A função de permutação é a mesma para cada rodada, mas uma chave diferente é produzida em cada iteração.

Prova-se que a função de permutação IP final é a inversa da função de permutação IP inicial, podendo-se desta forma se recuperar os bits originais que foram permutados por IP. Essas funções são definidas através das tabelas apresentadas a seguir.

Função IP

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

Função IP inversa

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

4.5.1 Detalhes de um estágio

A Figura 9 mostra os detalhes da estrutura interna de um dos 16 estágios do algoritmo. Novamente, observando o lado esquerdo, vemos que os 32 bits mais e menos significativos são tratados separadamente. Cada bloco de 32 bits é denominado L (*left*) e R (*right*).

Cada etapa pode ser resumida através das seguintes fórmulas:

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

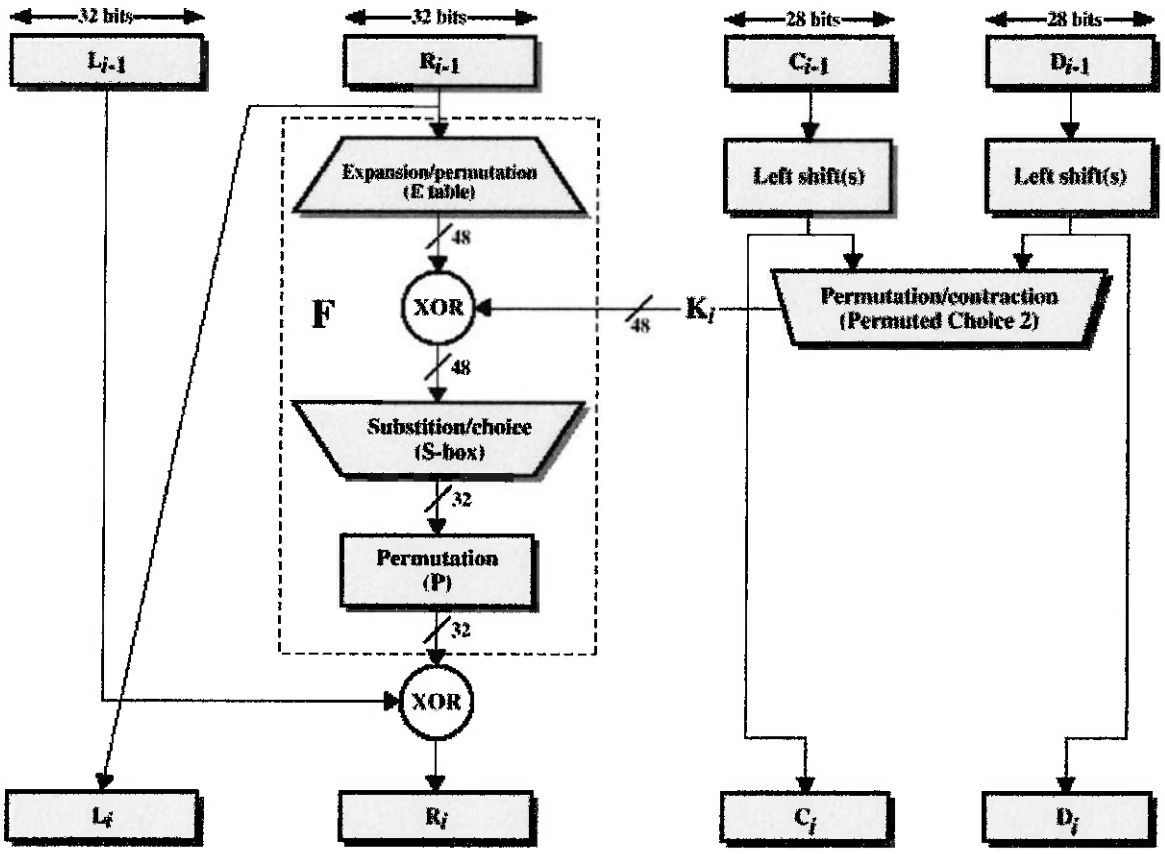


Figura 9 - Estágio Único do Algoritmo DES

A chave K_i é de 48 bits. A entrada R é de 32 bits. A entrada R é inicialmente expandida para 48 bits usando uma tabela que define uma permutação e mais uma expansão que envolve duplicação de 16 bits de R. Ver tabela abaixo:

Expansão/Permutação

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

Os 48 bits resultantes sofrem a operação ou-exclusivo (XOR) com a chave K_i. Estes 48 bits passam por uma função de substituição que produz uma saída de 32 bits, a qual é definida pela tabela abaixo:

Função de permutação P

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

O papel das *S-boxes* na função F é ilustrado na Figura 10. A substituição consiste de um conjunto de 8 *S-boxes*, cada uma aceitando 6 bits de entrada e produzindo 4 bits de saída. As transformações são feitas através de tabelas, que podem ser assim interpretadas: o primeiro e o último bit na entrada da *S-box* formam um número binário de 2 bits que selecionam uma de quatro substituições definidas pelas 4 linhas na tabela para S_i. Os 4 bits do meio selecionam uma coluna. O valor decimal da célula especificada pela linha e coluna anterior, é convertido a um número binário de 4 bits para produzir a saída. Por exemplo, em S1, se a entrada é 011001, a linha é 01 (linha 1) e a coluna é 1100 (coluna 12). O valor na linha 1, coluna 12 é 9, então a saída é 1001.

Cada linha numa *S-box* define uma substituição de reversão geral.

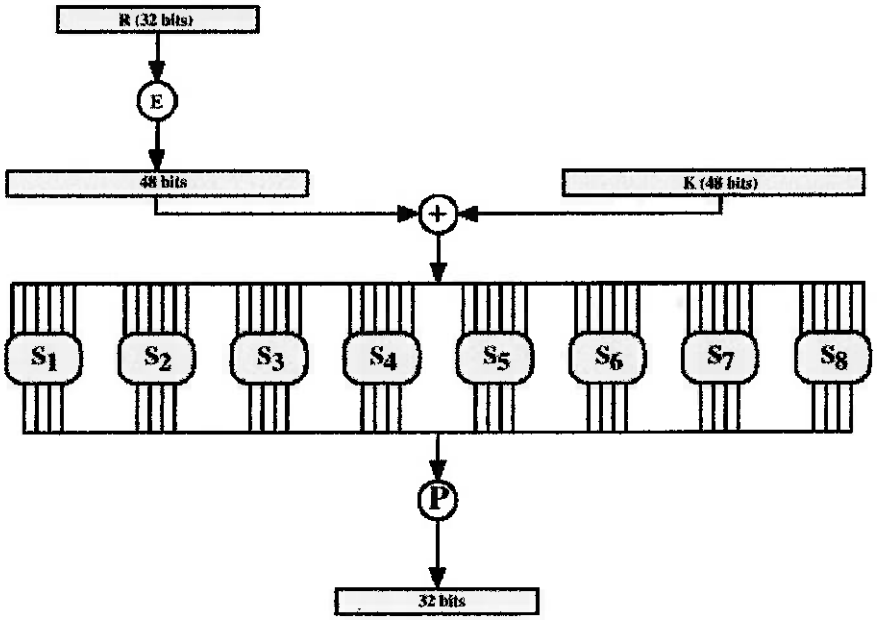


Figura 10 - Cálculo de F(R,K)

Geração da Chave

Retornando às Figuras 9 e 10, vemos que uma chave de 56 bits é usada como entrada. Inicialmente é feita uma permutação definida pela tabela "Permuted Choice One". Os 56 bits resultantes são então tratados como 2 blocos de 28 bits cada, chamados C_0 e D_0. Em cada rodada, C_{i-1} e D_{i-1} são separadamente submetidos a um deslocamento de bits à esquerda (*left shift*), de 1 ou 2 bits, de acordo com a tabela abaixo. Estes bits deslocados servem como entrada para a próxima rodada. Servem também como entrada para a tabela "Permuted Choice Two", a qual produz uma saída de 48 bits que serve como entrada para a função $f(T_{i-1},K_i)$.

"Permuted Choice One (PC-1)"

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

"Permuted Choice Two (PC-2)"

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Número de bits deslocados (em função da rodada)

Número da rodada	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Número de bits	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

4.6 Decodificação DES

A decodificação usa o mesmo algoritmo da codificação, com exceção que a aplicação das sub-chaves é invertida.

4.7 Análise do algoritmo DES

Desde sua adoção pelo Governo Americano, surgiram vários questionamentos quanto à força do algoritmo.

O poder do algoritmo está baseado em duas características principais:

- uso de uma chave de 56 bits;
- caráter público do algoritmo de codificação

O questionamento de sua força advém principalmente pela chave de 56 bits, considerada pequena para um algoritmo tão amplamente utilizado atualmente. Com o passar do tempo, máquinas com capacidade de processamento cada vez maior surgem, o que tornam o ataque de bruta força, isto é, varrer todas as possibilidades para se descobrir o valor da chave, amplamente viável.

Assim, uma evolução do algoritmo DES que vem sendo estudado é a adoção de um maior número de bits para a chave, o que tornaria o ataque por bruta força muito mais difícil, em face ao grande leque de possibilidades para o valor da chave.

Outra força considerada é o caráter aberto da estrutura do algoritmo, podendo dessa forma ser implementado por qualquer pessoa interessada. Por um lado, isto é uma vantagem, pois permite o estudo e a discussão em domínio público de suas fraquezas e possíveis melhorias.

Mas o principal questionamento sobre o DES se refere à implementação das funções de substituição, as chamadas *S-Boxes*, cujos critérios de implementação e projeto não foram abertos para o público. Assim, suspeita-se que essas funções tenham sido construídas de forma que através de criptoanálise o oponente (ou concorrente) possa fazer a decodificação dos dados. Entretanto muito estudo tem sido feito e nenhuma conclusão quanto às fraquezas das *S-boxes* foi revelada.

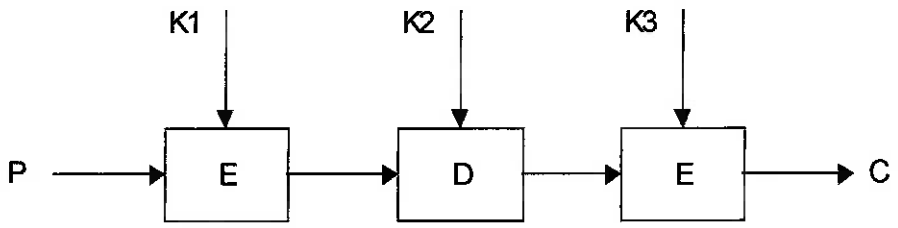
4.8 Triple DES: Reforçando o poder do DES

Dada a vulnerabilidade potencial do DES a um ataque por força bruta, têm-se estudado alternativas ao seu uso. Para isso pode-se criar novos algoritmos a partir do zero, sem utilizar nenhum conceito embutido em outro algoritmo, ou reaproveitar os algoritmos existentes para implementar algoritmos mais poderosos.

Seguindo esta última filosofia, o Triple DES surge como um forte concorrente para a substituição do DES, já que ele se utiliza totalmente da estrutura de software e hardware hoje existentes usados para implementar o DES.

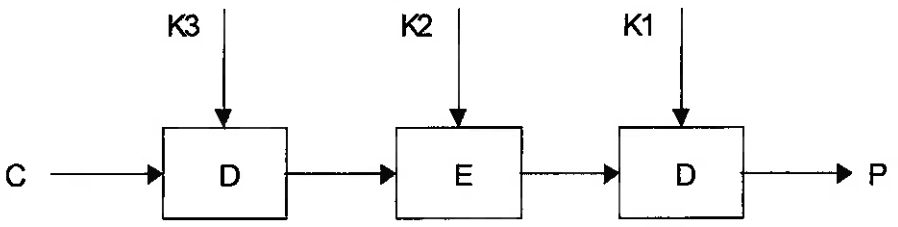
O Triple DES é um algoritmo baseado no DES, mas utilizando uma chave efetiva de 168 bits, aumentando o poder do algoritmo significativamente, de forma que ataques de força bruta sejam totalmente inviáveis.

A codificação do Triple DES pode ser esquematizada conforme figura a seguir:



A codificação é feita através de 3 chaves (k1, k2, k3), por um processo de codificação-decodificação-codificação, cada etapa usando uma chave específica. A entrada é a mensagem pura P e a saída é a mensagem codificada C.

Já a decodificação é esquematizada abaixo:



Na decodificação temos um processo de decodificação-codificação-decodificação, usando as chaves k1,k2,k3, mas na ordem inversa do uso das chaves para a codificação.

Os blocos de codificação e decodificação são os mesmos usados no algoritmo DES, com a mesma estrutura interna de funcionamento. Desta forma não há necessidade de se construir novos blocos de software para implementá-lo.

Comparado ao DES, o Triple DES é uma alternativa bastante viável e bastante segura, principalmente pelo fato de nunca ter sido quebrado, ao contrário do DES.

5 Arquitetura do Projeto

O Sistema de Transmissão de Dados Seguro, incluindo os microcomputadores e os módulos, está representado na Figura 1.

Os microcomputadores são os pontos terminais da comunicação. Eles recebem e enviam as mensagens na forma não-codificada, fornecendo ao usuário uma interface que lhe permite controlar a comunicação fim-a-fim e a comunicação entre o microcomputador e seu módulo.

Os módulos são responsáveis pela codificação e decodificação das mensagens. Eles cuidam da comunicação ponto-a-ponto com o microcomputador local e com o módulo remoto. A interface com o usuário nos módulos resume-se a um botão de *reset*.

Não se pretende abordar a forma de transmissão da chave por um canal seguro, apenas assumindo-se que ambos os módulos têm à disposição as chaves necessárias, configurada pelo usuário através do programa de interface rodando no microcomputador. A chave deve ser independentemente configurada em cada módulo e deve ser igual nos dois módulos para que a comunicação possa se efetuar.

6 Descrição do Hardware

6.1 Descrição Funcional do Hardware

Cada módulo recebe do microcomputador dados na forma de seqüências de bytes (seqüências de caracteres), codifica-os e transmite-os ao outro módulo. Da mesma forma, cada módulo recebe do outro módulo os dados codificados, decodifica-os e transmite-os ao microcomputador local.

Essas funções são implementadas em software, portanto, o hardware contido nos módulos é relativamente simples. A única exigência especial é a presença de duas portas seriais com tensão no nível RS-232.

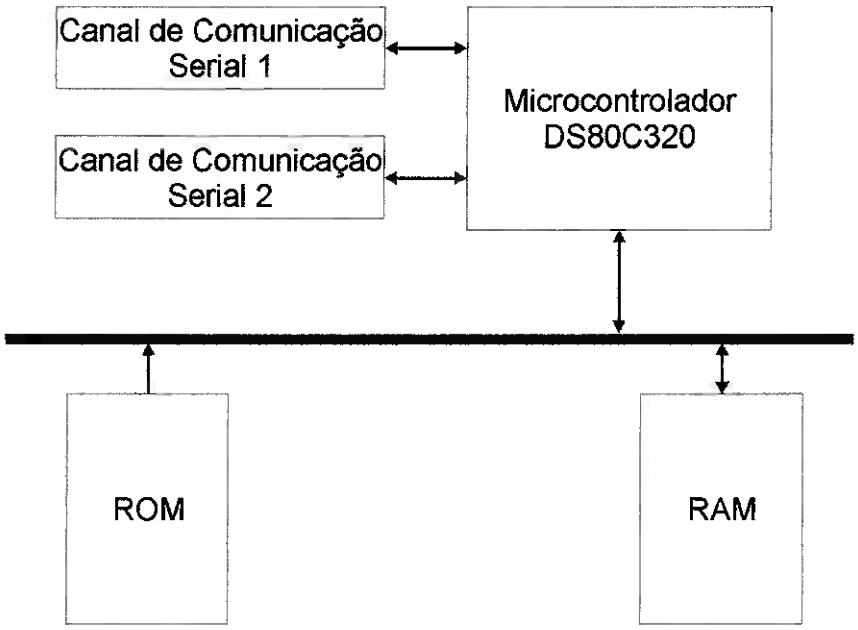


Figura 11 - Diagrama de Blocos do Sistema

O circuito eletrônico de cada módulo conta com um microcontrolador compatível com o 8051, memórias RAM e ROM e duas interfaces seriais, no padrão RS-232, para permitir a comunicação simultânea com uma porta serial do computador e com o outro módulo, de forma *full-duplex*. É necessário um driver para compatibilizar os níveis de tensão das portas do microprocessador com os níveis RS-232. O microcontrolador é responsável principalmente por realizar o processo de codificação e decodificação dos dados, além de implementar as camadas inferiores do protocolo de comunicação.

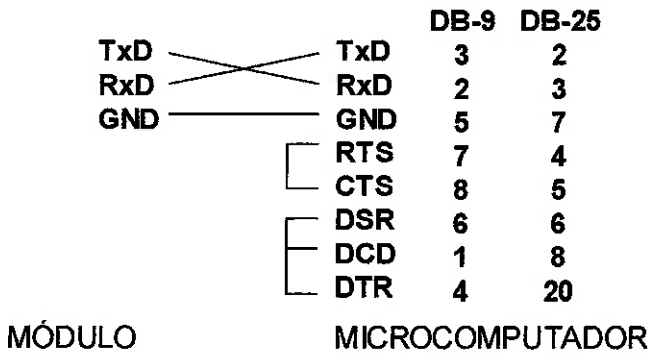


Figura 12 - Ligação dos pinos do cabo serial

O circuito foi implementado numa placa de *wire-wrap*. São necessários cabos para comunicação serial. Optou-se por utilizar cabos seriais na configuração *null-modem* sem *handshake*. A ligação dos pinos para essa configuração está indicada na Figura 12. Tais cabos são idênticos aos cabos presentes no kit8051 utilizado em laboratório.

6.2 *Esquema elétrico*

Ligada da forma indicada no esquema elétrico em anexo, a ROM ocupa os endereços de 0000h a 7FFFh (*32kbytes*) e, a RAM, os endereços de 8000h a FFFFh (*32kbytes*). Nos endereços de 8000h a FFFFh da ROM está mapeada a RAM para facilitar os testes e desenvolvimento através da utilização do programa monitor do kit8051 que, durante o desenvolvimento, rodava nos *32kbytes* mais baixos de ROM. Esse programa monitor permite o carregamento do programa a ser testado na área de RAM.

6.3 *Lista de Componentes*

Os componentes são:

- DS80C320-MCG – Microcontrolador da Dallas Semicondutores
- 1 soquete DIP 40 pinos para o microcontrolador
- 1 27C256 – EPROM
- 1 UM62256 – RAM
- 2 soquetes DIP 28 pinos para as memórias
- MAX232A – Driver para conversão RS-232, com empacotamento DIP 16 pinos
- 1 soquete DIP 16 pinos para o MAX232A
- 74HCT373 – Registrador
- 1 soquete DIP 20 pinos para o 74HCT373
- 7404 – Inversor
- 1 soquete DIP 14 pinos para o 7404
- 7408 – porta lógica AND
- 1 soquete DIP 14 pinos para o 7408
- Cristal de clock 11,0592*MHz*
- 1 resistor de 8,2$k\Omega$ – para o circuito de *reset*
- 2 capacitores de 33pF – para o circuito de *clock*
- 1 capacitor de 10μF – para o circuito de *reset*
- 1 botão para o circuito de *reset*
- 1 placa *wire-wrap*
- 2 cabos de comunicação serial na configuração *null-modem* com conectores DB-25 fêmea
- 4 capacitores 1μF – para o DS232A
- 6 capacitores 22nF – para desacoplamento na alimentação dos CIs
- 1 fonte de alimentação DC +5V
- 1 LED 5mA

- 1 resistor $1k\Omega$ para o circuito de acionamento do LED
- soquetes diversos para conexão dos resistores, capacitores, LED e cristal

6.4 *Justificativas e Comentários*

Escolheu-se o DS80C320 da *Dallas Semicondutores*, que é um microcontrolador de alta velocidade, compatível com 8051 em termos de instruções e pinagem. Ele possui duas portas de comunicação serial, simplificando o circuito necessário. Além disso, sua alta velocidade de processamento é necessária para permitir a codificação e decodificação em tempo real.

É necessário um driver para compatibilizar os níveis de tensão das portas do microcontrolador com o padrão RS-232. Para isso foi escolhido o MAX232A, que dispensa a alimentação de 12V.

São também usados um registrador 74373 para demultiplexação das vias de dados e endereço da memória, um inversor para ligação do sinal A15 ao 62256 e um AND para ligar os sinais do PSEN e do RD à RAM.

Implementou-se *reset* automático, no momento do ligamento, através do uso de um circuito com capacitor e resistor. Também se implementou o *reset* forçado com um botão.

São utilizados capacitores de desacoplamento de $22nF$ na alimentação de cada componente para eliminar os transitórios.

6.5 *Implementação do Hardware*

Durante a implementação do hardware foi feita uma alteração no projeto original que reduziu a quantidade de memória ROM, inicialmente estimada como *64kbytes*, para *32kbytes*. Isso foi possível porque os testes do programa de criptografia no 8051 mostraram que seu tamanho é menor que o esperado. Esta modificação facilitou o desenvolvimento do software do módulo, pois permitiu que se utilizasse o programa monitor e o kit8051 para os testes antes do módulo ficar pronto.

Durante a montagem notou-se a necessidade de pequenas correções no projeto original, relativas à lógica de endereçamento de memória.

Também foi corrigido o circuito do DS232A, onde havia sido colocado um capacitor a mais que o necessário e se havia especificado capacitores com valores errados.

Para facilitar a montagem, decidiu-se utilizar portas seriais semelhantes às implementadas no kit8051 do laboratório, ao invés das portas DB-9 macho originalmente especificadas.

A montagem do módulo ocorreu dentro do cronograma inicial, terminando em outubro. Os testes e montagem dos cabos seriais foram feitos em novembro.

Todo o grupo participou da implementação do módulo codificador, ficando as tarefas de ligação da maior parte dos fios do tipo wire-wrap com Tiago Zanco e a parte de solda, montagem da placa da fonte de alimentação e dos cabos seriais com Rogério Luiz Pereira e Pedro Hancon Cavalcanti Yue. Todo o grupo participou dos testes.

Observação: apesar de se ter especificado que o sistema completo é constituído por dois módulos, decidiu-se implementar apenas um. A razão disso foi a necessidade de demonstrar o funcionamento do sistema. A demonstração será feita simulando-se o segundo módulo no softwa-

re de um dos PCs, o que permite que se visualize os dados que estão sendo transmitidos no canal seguro antes da decodificação. Além disso, o segundo módulo seria redundante, por ser idêntico ao primeiro.

7 Descrição do Software

O software utilizado no projeto tem três funções fundamentais:

- Prover a interface com o usuário, permitindo a ele controlar a comunicação, enviar dados e visualizar os dados recebidos;
- Gerenciar a comunicação entre os computadores, entre os módulos e entre cada computador e seu módulo;
- Codificar e decodificar os dados transmitidos.

A implementação do software foi feita em duas partes: uma residente no microcomputador PC e outra residente no módulo.

7.1 Software de Interface

O Software de Interface é um programa instalado no microcomputador, sendo utilizado pelo usuário para controlar todo o processo de comunicação. A partir dele, o usuário é capaz de:

- Configurar a chave de criptografia;
- Enviar os dados desejados;
- Visualizar e salvar os dados recebidos;
- Configurar a porta serial a ser usada (baud rate, start bit, stop bits e paridade são fixos de acordo com o padrão do kit8051);
- Inicializar, reinicializar e finalizar a comunicação.

Este software foi implementado com o uso da ferramenta visual Borland C++ Builder, e utiliza os conhecimentos adquiridos no curso de Laboratório de Microprocessadores II. Foi utilizada a estrutura do programa desenvolvido nas experiências de Laboratório, que implementa a comunicação serial entre o PC e o Kit.

O software residente no microcomputador também implementa uma versão do Protocolo de Comunicação do Módulo. Uma extensão possível do projeto é implementar no Programa de Interface um protocolo de nível superior (não presente nos módulos) onde se pode colocar mecanismos de detecção e correção de erros, controle de fluxo, etc.

Em um dos microcomputadores será executado o Programa de Interface com uma configuração que inclui funções de criptografia para simular o segundo módulo, que não será implementado e também para permitir a visualização dos dados antes da decodificação, como forma de demonstrar o funcionamento do processo de codificação.

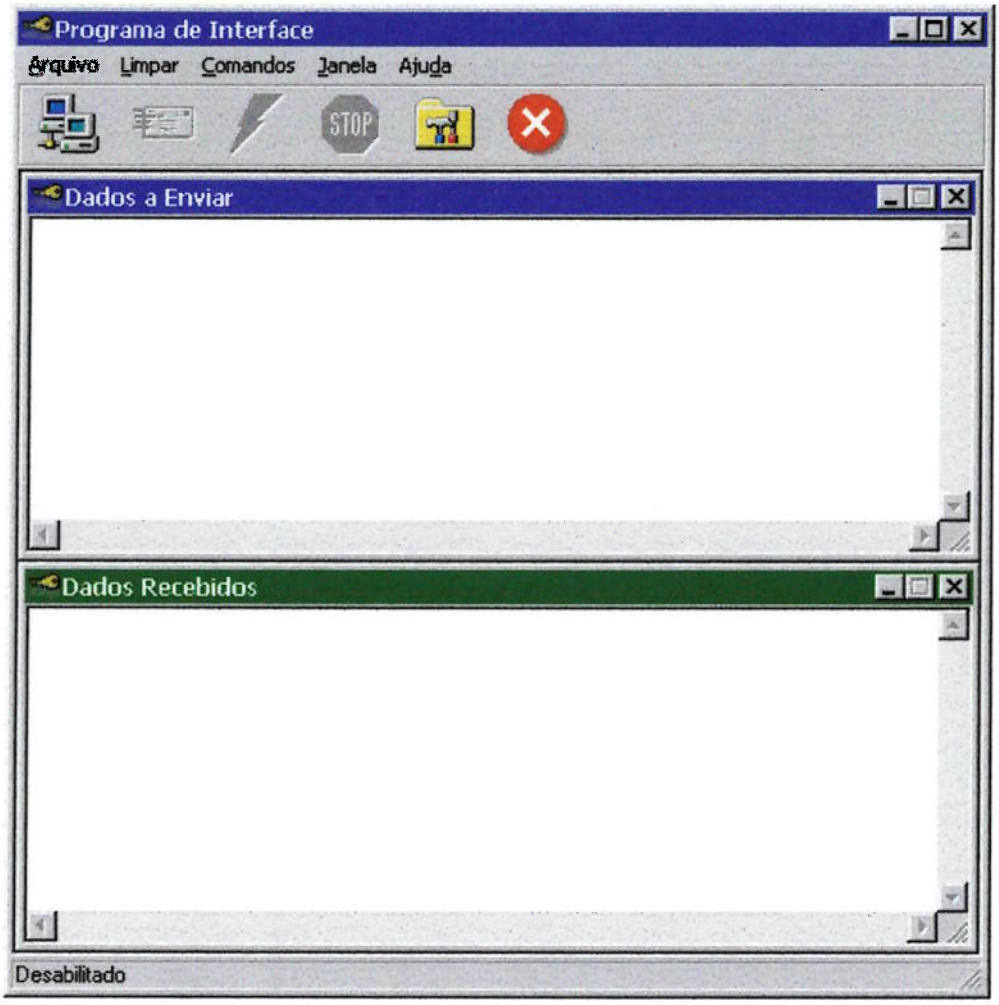


Figura 13 – Tela principal do Programa de Interface

A Figura 13 mostra a tela principal do Programa de Interface no estado Desabilitado. O modo de usar o programa é descrito no Manual do Usuário, anexo.

7.1.1 Implementação do Software de Interface

O Programa de Interface foi implementado entre Agosto e Novembro, sendo testado inicialmente sem o módulo, na comunicação computador a computador e, em Novembro, feita sua integração com o módulo.

A implementação do algoritmo de criptografia foi feita por Rogério Luiz Pereira, a elaboração do módulo do protocolo de comunicação foi feita por Pedro Hancon Cavalcanti Yue, a parte de comunicação serial foi aproveitada de uma experiência executada no Laboratório de Microprocessadores II e a interface com o usuário e a integração desses módulos foi feita por Tiago Zanco.

Foi decidido que a chave de criptografia configurada será armazenada de forma legível num arquivo de configuração localizado no computador local. Isso foi considerado aceitável porque o objetivo do projeto é tornar seguro o canal de comunicação entre os módulos. Assume-se

que a segurança dos microcomputadores não é um problema e não está no escopo do projeto. Devido a algumas restrições das funções de biblioteca que lidam com o arquivo de configurações, a chave não pode conter espaços.

7.2 Software de Criptografia

Esta é a parte principal do software, responsável por codificar e decodificar os dados transmitidos. Foi implementado o algoritmo Triple DES que, conforme discutido anteriormente, é preferível em relação ao DES, por possuir uma chave de codificação de 192 bits e, portanto, apresentar muito mais segurança. Além disso, o código necessário para implementá-lo tem complexidade e tamanho semelhantes aos do DES.

Sua implementação foi feita através de um compilador específico para o 8051, o AvCase51, da Avocet Systems, já que o programa roda diretamente no processador do módulo de criptografia. O AvCase fornece opções de otimização de código que foram úteis para reduzir o tamanho do binário gerado e aumentar o desempenho do algoritmo de criptografia.

Para análise e testes do algoritmo de codificação, o mesmo foi implementado com o auxílio de um compilador para computadores PC, no caso o Borland C++, versão 5.0. Esta versão foi adaptada para o módulo e para o Programa de Interface.

7.2.1 Implementação do Software de Criptografia

O Software de Criptografia, que implementa o algoritmo Triple DES, foi desenvolvido, testado e validado com sucesso, tanto na plataforma do 8051 (o módulo) como em ambiente Windows.

Primeiramente, implementou-se o algoritmo de criptografia no ambiente Windows, utilizando-se o compilador Borland C++. Nesta fase completaram-se com sucesso o desenvolvimento e os testes do algoritmo, de forma que se pôde selecionar definitivamente o algoritmo Triple DES como aquele a ser utilizado no projeto.

O algoritmo foi implementado em C ANSI ao invés de C++ para que pudesse ser mais facilmente portado para o 8051.

A próxima etapa foi o porte do algoritmo para o microcontrolador 8051. Para isso usou-se o compilador AvCase51 e basicamente o mesmo código escrito para Windows.

Nesta fase pôde-se avaliar a viabilidade da implementação do algoritmo no 8051, principalmente quanto à velocidade e tamanho do código. Chegou-se à conclusão de que é necessário compilação com parâmetros de otimização para obter resultados a uma velocidade considerada aceitável pelo grupo.

Quanto ao tamanho do código, principal preocupação durante o projeto do sistema, foi inferior ao esperado, em torno de 25 kbytes, já se utilizando a opção de compilação otimizada. Isso permitiu que se reduzisse o tamanho projetado da memória ROM do sistema, que era inicialmente de 64 kbytes.

O desenvolvimento do algoritmo foi feito por Rogério Luiz Pereira, que, na adaptação para o 8051, foi auxiliado por Tiago Zanco. Todo o grupo participou dos testes de validação de ambas as versões.

7.3 Os Protocolos de Comunicação

O sistema de comunicação é subdividido em camadas funcionais independentes, as quais têm a configuração mostrada na Figura 14.

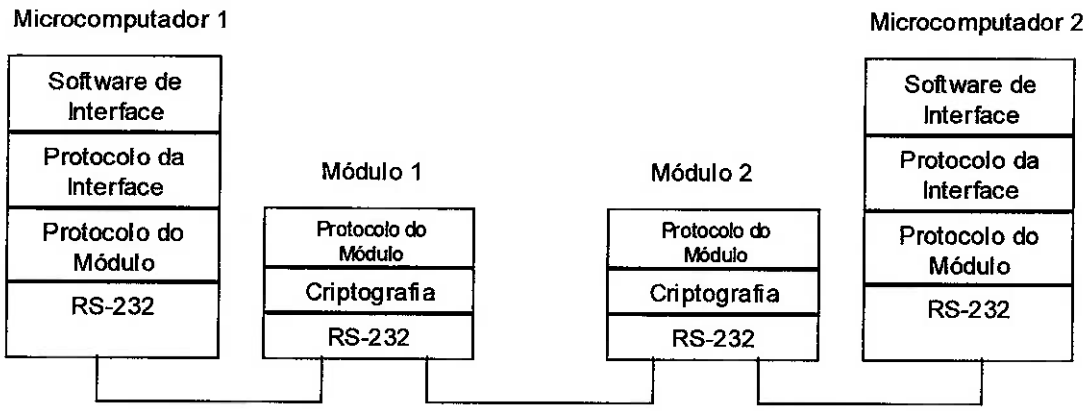


Figura 14 - Modelo do Sistema em Camadas

A versão do Protocolo do Módulo implementada no microcomputador é ligeiramente diferente da implementada nos módulos. Devido à necessidade de enviar a chave do microcomputador para o módulo.

7.3.1 Protocolo da Interface

Este protocolo, não implementado na versão final do projeto, poderia adicionar diversos serviços de alto nível como, por exemplo:

- Detecção e recuperação de erros;
- Controle de seqüência;
- Controle de fluxo.

Esses serviços tornariam a comunicação confiável.

7.3.2 Protocolo de Comunicação do Módulo

Este protocolo é bastante simples, sendo responsável por negociar a chave, iniciar e terminar uma sessão de comunicação e transmitir os dados. Ele não implementa nenhum método de tratamento de erros ou perdas nem controle de fluxo.

Tabela 1 – Mudanças de Estado do Protocolo do Módulo

	Desabilitado	Recebendo Chave	ModAck-Chave Sent	Habilitado	Habilitado Recepção	Habilitado Transmis-são
PCInitChave	*Recebendo Chave*	*Desabilitado* **PCErro**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**
PCChave	*Desabilitado*	*ModAck-Chave Sent* **PCAck-Chave ModAckChave**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModErro PCErro**
ModAck-Chave	*Desabilitado*	*Recebendo Chave*	*Habilitado* **PCAckCon ModAck-Chave**	*Habilitado*	*Habilitado Recepção*	*Habilitado Transmis-são*
ModEnvio	*Desabilitado*	*Recebendo Chave*	*Desabilitado* **ModErro PCErro**	*Habilitado* **PCEnvio**	*Habilitado Recepção* **PCEnvio**	*Desabilitado* **ModErro PCErro**
PCEnvio	*Desabilitado* **PCErro**	*Desabilitado* **PCErro**	*Desabilitado* **ModErro PCErro**	*Habilitado* **ModEnvio**	*Desabilitado* **ModErro PCErro**	*Habilitado Transmis-são* **ModEnvio**
ModFim	*Desabilitado*	*Recebendo Chave*	*Desabilitado* **ModErro PCErro**	*Habilitado Transmis-são* **PCFim**	*Desabilitado* **PCFim**	*Desabilitado* **ModErro PCErro**
PCFim	*Desabilitado* **PCErro**	*Desabilitado* **PCErro**	*Desabilitado* **ModErro PCErro**	*Habilitado Recepção* **ModFim**	*Desabilitado* **ModErro PCErro**	*Desabilitado* **ModFim**
PCErro	*Desabilitado*	*Desabilitado*	*Desabilitado* **ModErro**	*Desabilitado* **ModErro**	*Desabilitado* **ModErro**	*Desabilitado* **ModErro**
ModErro	*Desabilitado*	*Recebendo Chave*	*Desabilitado* **PCErro**	*Desabilitado* **PCErro**	*Desabilitado* **PCErro**	*Desabilitado* **PCErro**
PCReinit	*Desabilitado* **ModReinit**	*Desabilitado* **ModReinit**	*Desabilitado* **ModReinit**	*Desabilitado* **ModReinit**	*Desabilitado* **ModReinit**	*Desabilitado* **ModReinit**
ModReinit	*Desabilitado* **PCReinit**	*Desabilitado* **PCReinit**	*Desabilitado* **PCReinit**	*Desabilitado* **PCReinit**	*Desabilitado* **PCReinit**	*Desabilitado* **PCReinit**

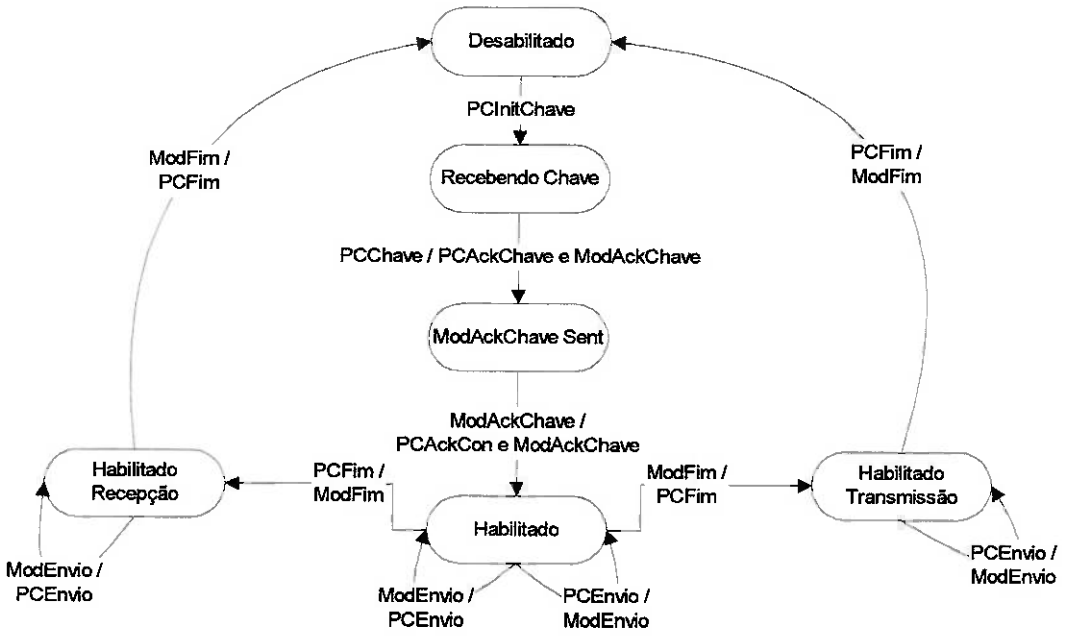


Figura 15 - Máquina de Estados Simplificada do Protocolo do Módulo

Os pacotes têm tamanho fixo de 8 bytes para facilitar a codificação. Os tipos de pacotes são:

- PCChave – 1 byte de primitiva e 7 bytes com parte da chave;
- PCInitChave – 1 byte de primitiva e 7 bytes com a parte inicial da chave;
- ModAckChave – 1 byte de primitiva e 7 bytes não utilizados;
- PCErro – 1 byte de primitiva e 7 bytes não utilizados;
- ModErro – 1 byte de primitiva e 7 bytes não utilizados;
- PCEnvio – 1 byte de primitiva e 7 bytes de dados;
- ModEnvio – 1 byte de primitiva e 7 bytes de dados;
- PCReinit – 1 byte de primitiva e 7 bytes não utilizados;
- ModReinit – 1 byte de primitiva e 7 bytes não utilizados;
- PCFim – 1 byte de primitiva e 7 bytes não utilizados;
- ModFim – 1 byte de primitiva e 7 bytes não utilizados.

O byte de primitiva é um código que indica qual o tipo de pacote.

Este tamanho de pacote permite que todos os pacotes que trafegam entre os módulos sejam codificados. Isso implica que, no estado *Desabilitado* o módulo ignora todos os dados que chegam do outro módulo, pois ele ainda não tem a chave para decodificá-los.

Para simplificação, assume-se que as chaves fornecidas pelos usuários dos microcomputadores são idênticas.

Para facilitar a criação e depuração do programa, o tratamento de erro consiste basicamente em avisar os componentes diretamente conectados e voltar ao estado *Desabilitado*.

A versão deste protocolo presente no microcomputador é ligeiramente diferente, não possuindo as primitivas com prefixo "Mod" e tratando do envio da chave, não de sua recepção.

Tabela 2 - Mudanças de Estado do Protocolo do Módulo (versão do microcomputador)

	Desabilitado	PCChave Sent	Aguardando Conexão	Habilitado	Habilitado Transmissão	Habilitado Recepção
INICIO	*PCChave Sent* **PCInitChave PCChave**	–	–	–	–	–
TIME-OUT	–	*PCChave Sent* **PCInitChave PCChave**	–	–	–	–
PCAck-Chave	*Desabilitado* **PCError ERRO**	*Aguardando Conexão* **CHAVE**	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**
PCAck-Con	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**	*Habilitado* **CONEXAO**	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**	*Desabilitado* **PCError ERRO**
ENVIO	–	–	–	*Habilitado* **PCEnvio**	*Habilitado Transmissão* **PCEnvio**	–
PCEnvio	*Desabilitado* **PCErro ERRO**	*Desabilitado* **PCErro ERRO**	*Desabilitado* **PCErro ERRO**	*Habilitado* **RECEPCAO**	*Desabilitado* **PCErro ERRO**	*Habilitado Recepção* **RECEPCAO**
FIM	–	–	–	*Habilitado Recepção* **PCFIm**	*Desabilitado* **PCFim**	–
PCFim	*Desabilitado* **PCErro ERRO**	*Desabilitado* **PCErro ERRO**	*Desabilitado* **PCErro ERRO**	*Habilitado Transmissão* **FIM**	*Desabilitado* **PCErro ERRO**	*Desabilitado* **FIM**
PCErro	*Desabilitado* **ERRO**	*Desabilitado* **ERRO**	*Desabilitado* **ERRO**	*Desabilitado* **ERRO**	*Desabilitado* **ERRO**	*Desabilitado* **ERRO**
PCReinit	*PCChave Sent* **PCInitChave PCChave**	*PCChave Sent* **PCInitChave PCChave**	*PCChave Sent* **PCInitChave PCChave**	*PCChave Sent* **PCInitChave PCChave**	*PCChave Sent* **PCInitChave PCChave**	*PCChave Sent* **PCInitChave PCChave**

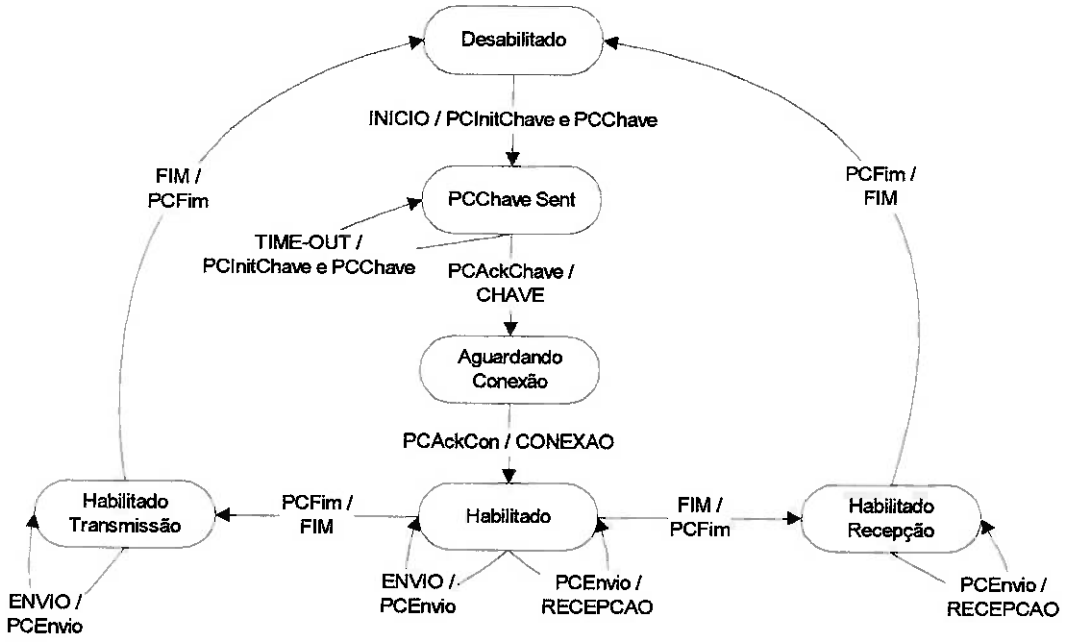


Figura 16 - Máquina de Estados Simplificada do Protocolo do Módulo (versão do microcomputador)

7.3.3 Implementação dos Protocolos de Comunicação

Implementou-se apenas o protocolo de comunicação do módulo. Isto foi feito na última fase do projeto e notaram-se poucas necessidades de modificação em relação ao projeto original, sendo que as principais foram a criação do estado "Recebendo Chave" e a introdução dos pacotes de reinicialização.

Para simplificar a montagem dos pacotes contendo 7 bytes de dados foram usados espaços para preencher a string a ser enviada, tornando seu comprimento múltiplo de 7.

O protocolo foi implementado por Tiago Zanco e Pedro Yue, integrado ao software do módulo por Rogério Luiz Pereira e integrado ao Programa de Interface por Tiago Zanco.

8 Sugestão de Empacotamento

Como sugestão, apresenta-se este esquema de empacotamento para um produto final. Ele possui um LED que indica se o circuito está alimentado, um botão de reset, duas portas seriais, uma para conectar ao PC e a outra para conectar ao outro módulo (a outra está do lado oposto à que está aparente) e um cabo de alimentação. A grade é para ventilação da fonte de alimentação.

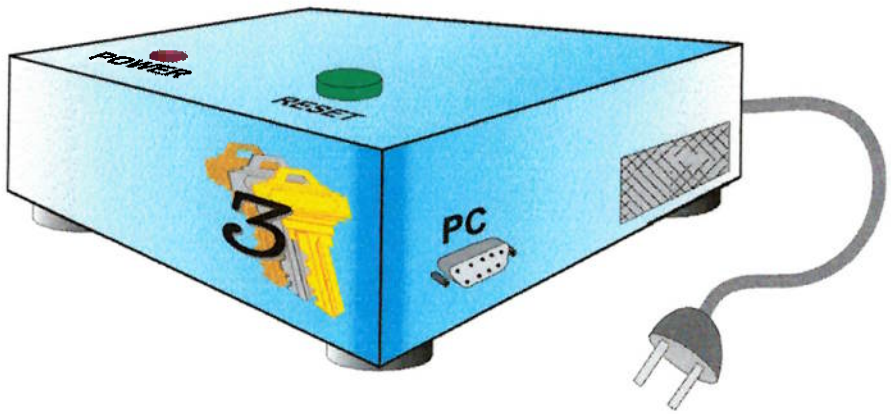


Figura 17 - Sugestão de Empacotamento

O produto conteria dois desses módulos e 3 cabos seriais para as conexões.

Ou então, uma alternativa melhor seria montar o módulo na forma de uma placa que possa ser conectada a um slot ISA ou PCI de um microcomputador PC.

9 Cronograma

As atividades foram desenvolvidas de acordo com o seguinte cronograma:

ATIVIDADES	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
Estudo da comunicação serial	X	X									
Pesquisa do algoritmo de criptografia	X	X	X								
Testes do algoritmo de criptografia em PC			X	X	X	X					
Implementação do algoritmo de criptografia para 8051							X	X			
Implementação do Módulo Codificador							X	X	X		
Testes do circuito do módulo									X	X	
Protocolos de comunicação										X	
Integração entre Hardware e Software										X	
Comunicação do módulo com o PC								X	X	X	
Teste do Sistema Completo										X	
Revisão e documentação final									X	X	X

10 Divisão de Tarefas

De forma geral, a divisão de tarefas no grupo foi a seguinte:

- Pedro Hancon Cavalcanti Yue: responsável pela implementação de parte dos protocolos de comunicação e montagem de parte do hardware (placa de alimentação e cabos);
- Rogério Luiz Pereira: responsável pelo software do módulo e pelo software de criptografia;
- Tiago Zanco: responsável pela implementação de parte do hardware, pela interface com o usuário e pela integração do Programa de Interface e pela implementação de parte dos protocolos de comunicação;
- Todo o grupo: projeto, testes e documentação.

11 Bibliografia

[1] Stallings, William **Network and Internetwork Security** Prentice-Hall, 1995

[2] Féris, Edson Benedicto Ramos **Criptografia Básica** LCS/PEE/EPUSP, 1998

[3] Tanenbaum, Andrew S. **Computer Networks** Prentice Hall, 1996

[4] Stallings, William **Cryptography and Network Security** Prentice Hall, 2nd edition

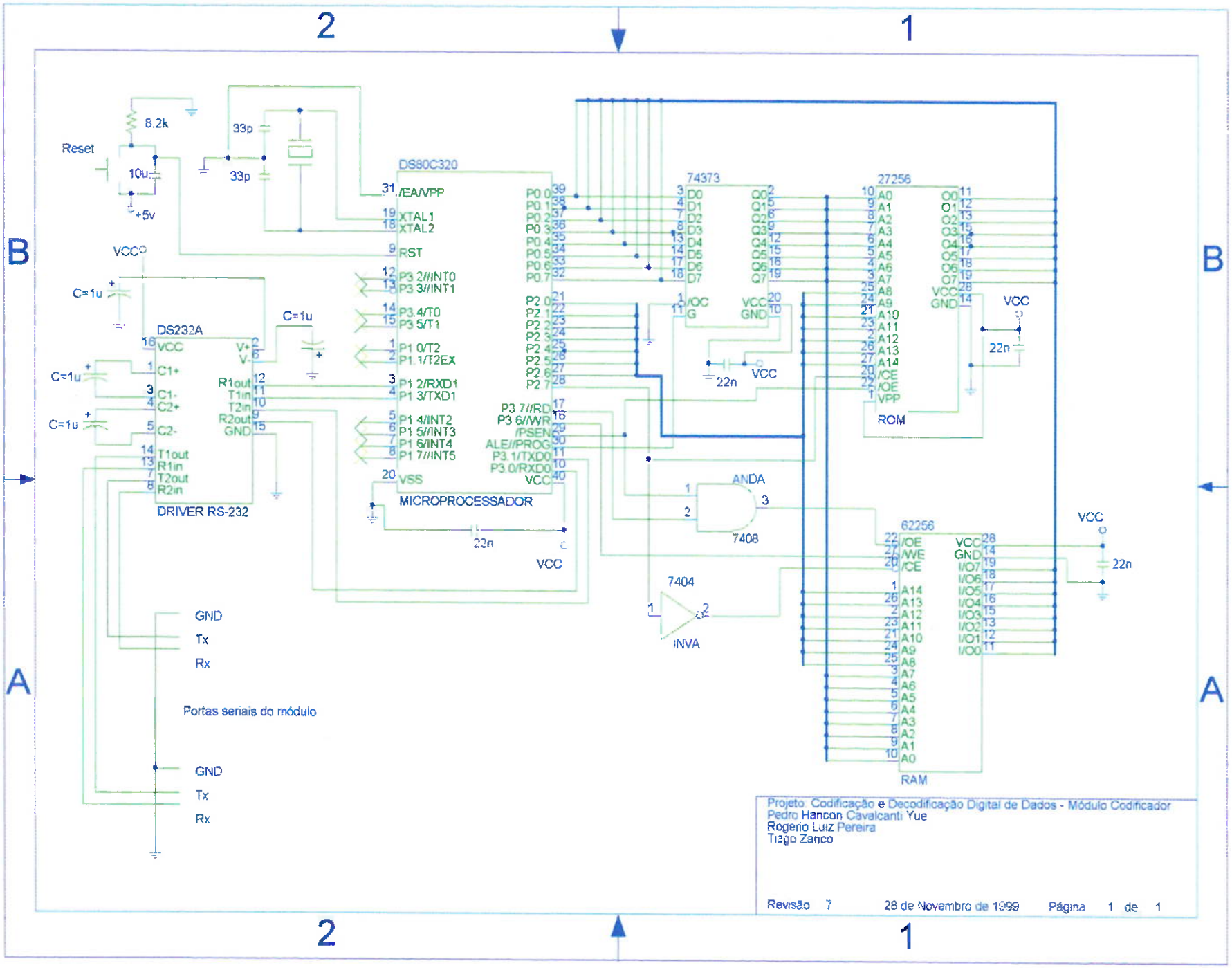
[5] Silva, Luiz Veiga da; Moreira, Marcelo Tocci; Nogueira, Rubens de Jesus **Criptografia em Transmissão Digital** Projeto de Formatura, PCS/EPUSP, 1998

[6] Cugnasca, Prof. Dr. Carlos E. **Apostila 5 – Evolução dos Microprocessadores e Microcontroladores** PCS-550, EPUSP, 1998

[7] **DS80C320/DS80C323 High-Speed/Low-Power Micro Datasheet** Dallas Semiconductor, 1998

[8] **DS232A Dual RS-232 Transmitter/Receiver Datasheet** Dallas Semiconductor, 1998

[9] Peacock, Craig **Interfacing the Serial / RS232 Port V5.0** 1998, http://www.senet.com.au/~cpeacock/


2
1
B
A
8.2k
Reset
10u
+5v
33p
33p
DS80C320
/EA/VPP
XTAL1
XTAL2
RST
P3.2//INT0
P3.3//INT1
P3.4/T0
P3.5/T1
P1.0/T2
P1.1/T2EX
P1.2/RXD1
P1.3/TXD1
P1.4/INT2
P1.5/INT3
P1.6/INT4
P1.7//INT5
VSS
MICROPROCESSADOR
22n
VCC
VCC
C=1u
DS232A
VCC
V+
V-
C1+
C1-
C2+
C2-
T1out
R1in
T2out
R2in
R1out
T1in
T2in
R2out
GND
DRIVER RS-232
GND
Tx
Rx
Portas seriais do módulo
GND
Tx
Rx
74373
/OC
G
VCC
GND
22n
VCC
27256
ROM
22n
VCC
ANDA
7408
7404
INVA
62256
RAM
22n
VCC
Projeto: Codificação e Decodificação Digital de Dados - Módulo Codificador
Pedro Hancon Cavalcanti Yue
Rogerio Luiz Pereira
Tiago Zanco
Revisão 7
28 de Novembro de 1999
Página 1 de 1

Sistema de Transmissão Segura de Dados

Manual do Usuário

1 Componentes do Sistema

O *Sistema de Transmissão Segura de Dados* é formado pelos seguintes componentes:

- Dois microcomputadores pessoais IBM-PC compatíveis com uma porta serial externa livre em cada um;
- Dois Módulos de Criptografia;
- Três cabos seriais;
- Programa de Interface.

2 Conexão dos Módulos de Criptografia

Os Módulos de Criptografia são responsáveis pela codificação e decodificação das informações transmitidas através do *Sistema de Transmissão Segura de Dados*. Na Figura 1 vê-se o esquema de um módulo com as partes mais importantes indicadas.

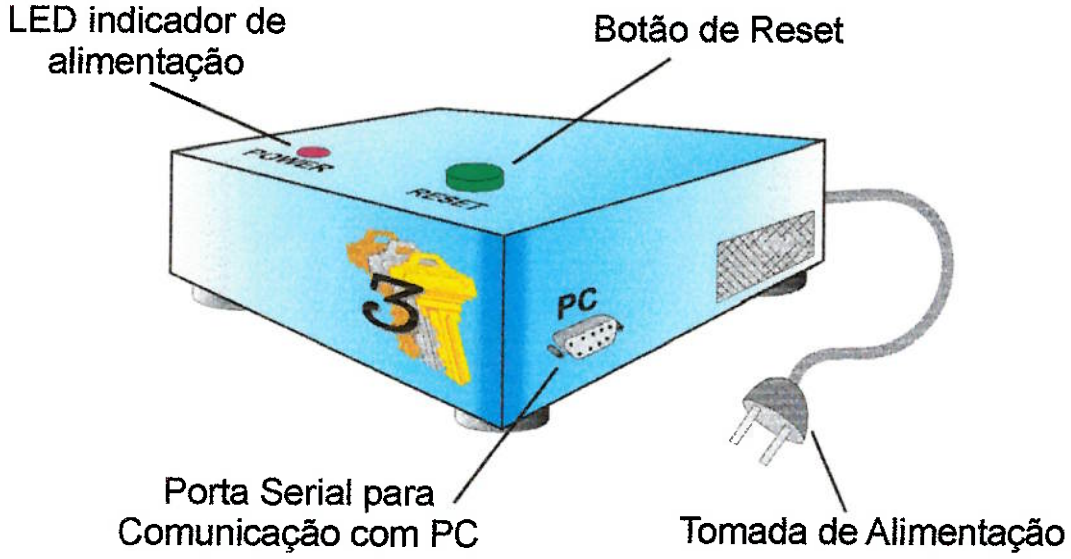


Figura 1 - Módulo de Criptografia

- **LED indicador de alimentação** – acende quando a tomada de força está conectada;
- **Botão de Reset** – para reinicializar o software do módulo;
- **Porta Serial para Comunicação com o PC**;
- **Porta Serial para Comunicação com o outro módulo** – esta não aparece no esquema da Figura 1. Ela está localizada no lado oposto ao da porta serial para comunicação com o PC e está identificada com a legenda "MOD".

Antes do uso, os módulos devem ser ligados à rede elétrica – alimentação 110V – e conectados aos computadores e entre si.

Primeiro, faça o seguinte procedimento para conectar cada módulo a um computador:

1. Encontre uma porta serial externa livre na parte traseira do microcomputador;
2. Conecte um dos cabos seriais a essa porta – talvez seja necessário um adaptador para compatibilizar a pinagem da porta com a do cabo;
3. Conecte a outra ponta do cabo à porta serial do módulo identificada com a legenda "PC".

Este procedimento está ilustrado na Figura 2.

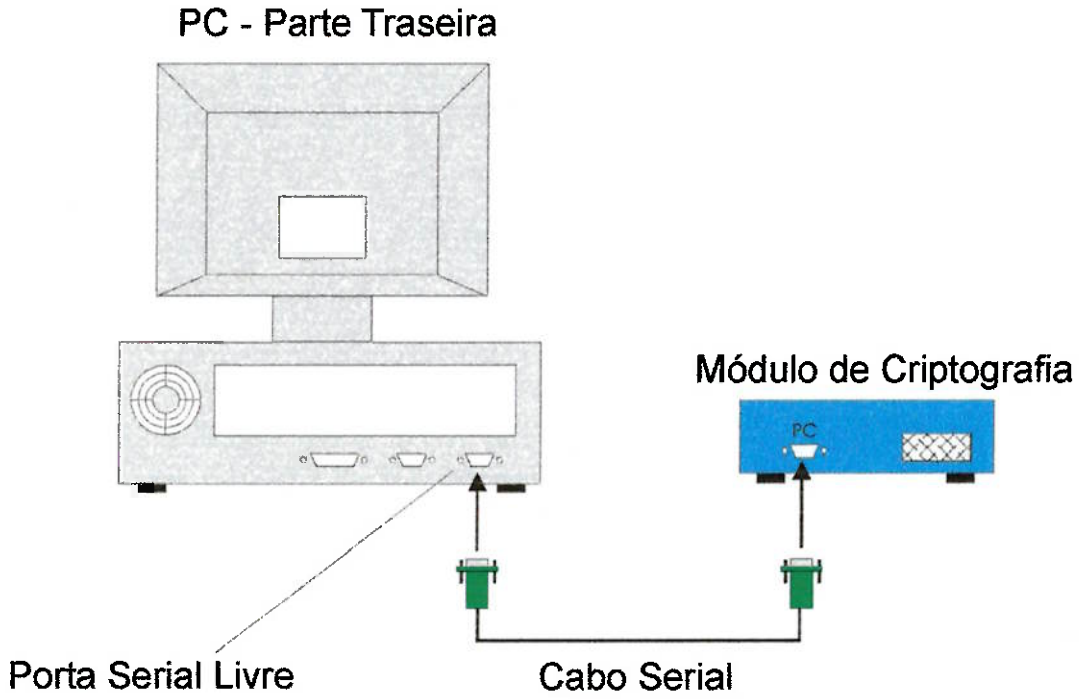


Figura 2 - Conexão dos Módulos aos PCs

Depois, é necessário conectar os módulos entre si utilizando o terceiro cabo serial. Ele deve ser conectado às portas dos módulos identificadas com a legenda "MOD".

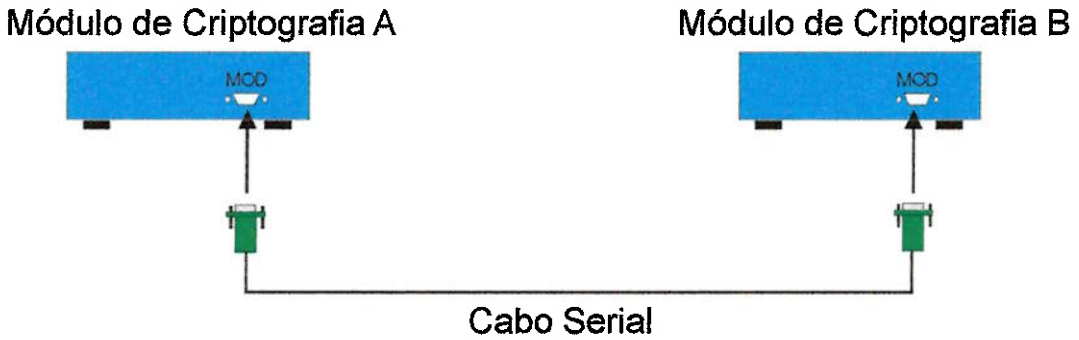


Figura 3 - Conexão dos Módulos entre si

3 Programa de Interface

O Programa de Interface oferece para o usuário uma interface para uso do *Sistema de Transmissão Segura de Dados*. Ele deve ser instalado em ambos os microcomputadores. Para isso, siga os seguintes passos (Obs.: o Programa de Interface é compatível apenas com Windows 9x):

1. Crie um diretório;
2. Copie os arquivos do disco de instalação do Programa de Interface para esse diretório;
3. Copie o arquivo *interf.ini* para o diretório do Windows;
4. Crie um atalho para o arquivo *interface.exe*.

Agora, execute o Programa de Interface através do atalho. Antes de utilizá-lo pela primeira vez, é necessário configurá-lo. Para isso:

1. Aperte o botão Configuração na barra de ferramentas ou selecione a opção Configuração do menu Comandos;

Botão Configuração

2. Na tab de Comunicação Serial, selecione a porta serial onde você conectou o módulo;
3. Não altere a configuração do delay por enquanto;
4. Na tab de Criptografia, digite a chave de codificação a ser utilizada. Atenção: a chave pode conter qualquer caractere válido, exceto espaços e deve ter 24 caracteres. Para que a comunicação funcione, a chave deve ser a mesma nos dois computadores. A chave é armazenada de forma legível no arquivo *interf.ini*, por isso não permita o acesso de pessoas estranhas ao computador;
5. Selecione a opção Módulo Presente. Esta opção informa ao Programa de Interface que o módulo está conectado. Se ela não for selecionada, o Programa de Interface passará a simular a presença no módulo. Nesse caso, a opção Decodificar Dados Recebidos indica se os dados exibidos devem ser decodificados ou não. Essa opção é útil para verificar o resultado da codificação.

A tela principal do Programa de Interface está indicada na Figura 4.

3.1 Estabelecendo a Conexão

Antes de estabelecer a conexão, é aconselhável acionar o botão de Reset no módulo. Depois, basta apertar o botão Iniciar da barra de ferramentas, ou selecionar a opção Iniciar do menu Comandos. Verifique o estado da conexão na barra de status. Quando aparecer a indicação Habilitado, o processo de conexão foi completado.

Botão Iniciar

Caso ao se apertar o botão Iniciar o estado continue como Desabilitado, o programa não pôde abrir a porta serial. Nesse caso, verifique se não há outro programa utilizando a mesma porta. Se não houver, verifique se a porta selecionada no Programa de Interface está corretamente configurada no Painel de Controle do Windows e no computador. Se necessário, mude a porta serial em que o módulo está conectado.

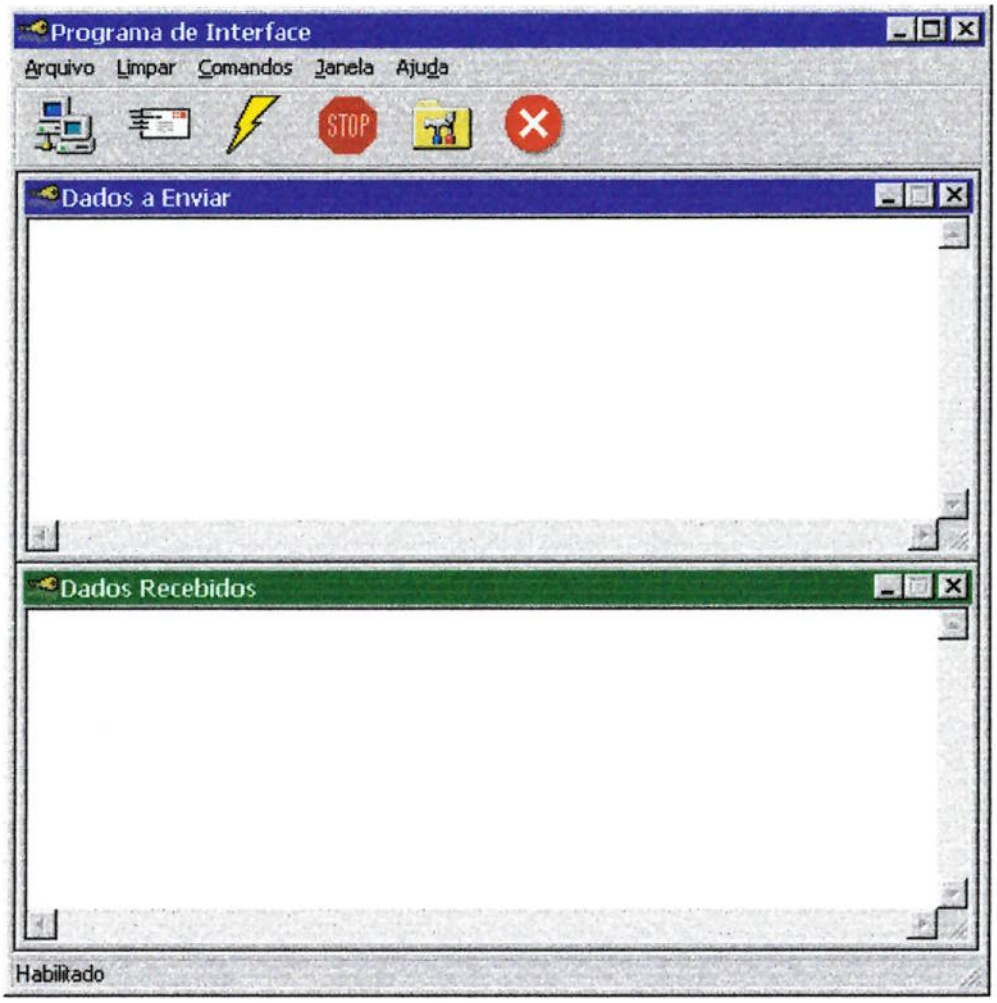


Figura 4 - Tela Principal do Programa de Interface

3.2 Enviando Dados

Você pode escrever o texto a ser enviado diretamente na janela Dados a Enviar ou pode selecionar a opção Ler Arquivo no menu Arquivo. Essa opção permite que você selecione um arquivo texto e carregue seu conteúdo na janela Dados a Enviar.

Botão Enviar

Depois de ter as informações que se deseja na tela, basta acionar o botão Enviar na barra de ferramentas ou a opção Enviar no menu Comandos. O botão volta à posição inicial ao final da transmissão.

3.3 Dados Recebidos

Os dados recebidos são automaticamente exibidos na janela Dados Recebidos. É possível salvar o conteúdo dessa janela num arquivo texto selecionando a opção Salvar Como no menu Arquivo.

3.4 Limpando o Conteúdo das Janelas

É possível limpar o conteúdo das janelas Dados a Enviar e Dados Recebidos selecionando as opções correspondentes no menu Limpar.

3.5 Reinicializando a Conexão

Em caso de erro na conexão, acione o botão Reinicializar na barra de ferramentas ou a opção Reinicializar no menu Comandos. Se isso falhar, aperte o botão Reset no módulo e depois o botão Reinicializar.

Botão Reinicializar

3.6 Fechando a Conexão

Para finalizar a conexão, aperte o botão Terminar na barra de ferramentas ou selecione a opção Terminar no menu Comandos e aguarde o estado mudar para Desabilitado.

Botão Terminar

3.7 Saindo do Programa

Para sair do programa, aperte o botão Sair da barra de ferramentas ou selecione a opção Sair no menu Arquivo. Todos os dados nas janelas são perdidos mas as configurações são salvas.

Botão Sair

3.8 Ajuda

Para obter ajuda on-line, selecione a opção Conteúdo no menu Ajuda.