UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

DISSERTAÇÃO DE MESTRADO

ASPECTOS DE COMUNICAÇÃO E CIFRAGEM PARA A REDE BLUETOOTH

GUILHERME NUNES MELO

UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE TECNOLOGIA E GEOCIÊNCIAS
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

ASPECTOS DE COMUNICAÇÃO E CIFRAGEM PARA A REDE BLUETOOTH

por

GUILHERME NUNES MELO

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Pernambuco como parte dos requisitos para obtenção do grau de **Mestre em Engenharia Elétrica**.

ORIENTADOR: VALDEMAR CARDOSO DA ROCHA JUNIOR, Doutor

Recife, julho de 2010

Universidade Federal de Pernambuco

Pós-Graduação em Engenharia Elétrica

PARECER DA COMISSÃO EXAMINADORA DE DEFESA DE DISSERTAÇÃO DO MESTRADO ACADÊMICO DE

GUILHERME NUNES MELO

TÍTULO

"ASPECTOS DE COMUNICAÇÃO E CIFRAGEM PARA A REDE BLUETOOTH"

A comissão examinadora composta pelos professores: VALDEMAR CARDOSO DA ROCHA JÚNIOR, DES/UFPE, RICARDO MENEZES CAMPELLO DE SOUZA, DES/UFPE e FRANCISCO MARCOS DE ASSIS, DEE/UFCG sob a presidência do primeiro, consideram o candidato **GUILHERME NUNES MELO APROVADO.**

Recife, 26 de julho de 2010.

RAFAEL DUEIRE LINS
Coordenador do PPGEE

VALDEMAR CARDOSO DA ROCHA JÚNIOR
Orientador e Membro Titular Interno

FRANCISCO MARCOS DE ASSIS
Membro Titular Externo

RICARDO MENEZES CAMPELLO DE SOUZA
Membro Titular Interno

M528a Melo, Guilherme Nunes.
Aspectos de comunicação e cifragem para a rede *Bluetooth* / Guilherme Nunes Melo. - Recife: O Autor, 2010.
121 folhas, il : figs.

Dissertação (Mestrado) – Universidade Federal de Pernambuco. CTG. Programa de Pós-Graduação em Engenharia Elétrica, 2010.

Orientador: Prof. Dr. Valdemar Cardoso da Rocha Jr.
Inclui Bibliografia e Apêndice.

1. Engenharia Elétrica. 2.Tecnologia da Comunicação. 3.*Bluetooth*. 4.Criptografia. I. Título.

UFPE

621.3 CDD (22. ed.) BCTG/2010-202

Todo o esforço direcionado para a confecção e conclusão deste trabalho de pesquisa foi motivado pela existência dos meus filhos: Daniel, Letícia e Lorena, para quem dedico todas as horas de estudo e pesquisa.

AGRADECIMENTOS

Pela oportunidade de retornar às minhas origens de pesquisa, agradeço ao meu orientador: Prof. Dr. Valdemar Cardoso da Rocha Jr., assim como, não poderia deixar de agradecer aos meus pais, que sempre me apoiaram quanto aos estudos, não deixando, nunca, que a educação fosse uma barreira, mas sim uma solução para superar os desafios que encontramos na nossa vida profissional.

Guilherme Nunes Melo

Universidade Federal de Pernambuco
26 de julho de 2010

“O Sucesso compõe-se de noventa e nove por cento de transpiração e um por cento de inspiração. Um gênio é uma pessoa de talento que faz toda a lição de casa”

(Thomas Alva Edson).

Resumo da Dissertação apresentada à UFPE como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica

ASPECTOS DE COMUNICAÇÃO E CIFRAGEM PARA A REDE BLUETOOTH

Guilherme Nunes Melo

Julho/2010

Orientador: Prof. Valdemar Cardoso da Rocha Jr.
Área de Concentração: Comunicações
Palavras-chaves: Bluetooth. Segurança. Criptografia.
Número de páginas: 121

Nesta dissertação são investigados alguns dos algoritmos usados para realizar o processo de criação das chaves de segurança usadas na criptografia da tecnologia Bluetooth, apresentando-se exemplos práticos. Todo o processo foi desenvolvido usando-se a linguagem de programação Visual C++. O *software* desenvolvido nesta dissertação permite o acesso a todo o processo de criação das chaves de segurança, possibilitando analisá-los detalhadamente, inclusive com acesso às saídas intermediárias. Deste modo, além das entradas e saídas que usualmente estão disponíveis, esse desenvolvimento permite acessar vários outros pontos internos do sistema, possibilitando uma melhor percepção de todo o processo de criação das chaves de segurança.

Abstract of Dissertation presented to UFPE as a partial fulfillment of the requirements for the degree of Master in Electrical Engineering

COMMUNICATION AND CIPHERING ASPECTS FOR BLUETOOTH NETWORK

Guilherme Nunes Melo

July/2010

Supervisor: Prof. Valdemar Cardoso da Rocha Jr.
Area of Concentration: Communications
Keywords: Bluetooth. Security. Cryptography.
Number of pages: 121

This dissertation investigates some of the algorithms used to perform the process of creating security keys used for encryption of Bluetooth technology by presenting practical examples. The entire process was developed using the Visual Programming Language C++. The software developed in this dissertation allows access to the entire process of creating security keys, allowing a detailed analysis, including access to the intermediate outputs. Thus, besides the inputs and outputs that are usually available, this development allows access to several other points of the system, enabling a better perception of the whole process of creating security keys.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS E SIGLAS

Termo		Descrição
ACO	*Authenticated Ciphering Offset*	Offset de autenticação da cifra
ADDRESS	*Address*	Endereço
AU_RAND_A	*Autentication Random Number*	Número aleatório usado na Autenticação
BD_ADDR	*Device Address*	Endereço de um dispositivo
BD_ADDR_A	*Device A Address*	Endereço do dispositivo A
BD_ADDR_B	*Device B Address*	Endereço do dispositivo B
$B_p[i]$	*Bias Vector*	Vetor de Bias
CLK	*Master Device Clock*	Clock do dispositivo mestre
COF	*Ciphering Offset*	Offset da cifra
E_0	*E_0 Algorithm*	Algoritmo de cifragem E_0
E_1	*E_1 Algorithm*	Algoritmo de cifragem E_1
E_2	*E_2 Algorithm*	Algoritmo de cifragem E_2
E_{21}	*E_{21} Algorithm*	Algoritmo de cifragem E_{21}
E_{22}	*E_{22} Algorithm*	Algoritmo de cifragem E_{22}
E_3	*E_3 Algorithm*	Algoritmo de cifragem E_3
EDR	*Enhanced Data Rate*	Melhoria da taxa de transmissão de dados
EN_RAND	*Encryption Random Number*	Número aleatório usado na cifragem
HASH	*Hash Function*	Função Hash
IEEE	*Institute of Electrical & Electronics Engineers*	Instituto de Engenheiros Elétricos e Eletrônicos

K	*Link Key*	Chave de conexão
$\tilde{K}$	*K offset*	Offset da chave K
K_A	*A Unit Key*	Chave da unidade A
K_{AB}	*AB combination Key*	Chave combinada das unidades A e B
K_B	*B Unit Key*	Chave da unidade B
K_C	*Encryption Key*	Chave de cifragem
K_{CIFRA}	*Binary Keystream Cipher Key*	Sequência binária usada na criação da chave de cifragem
K_{INIT}	*Initialization Key*	Chave de iniciação
K_{MASTER}	*Temporary Key*	Chave mestra ou chave temporária
$K_p[j]$	*SAFER+ Round Keys*	Chaves das etapas do SAFER+
L	*Number of PIN octects*	Número de octetos do PIN
L’	*Number of PIN’ octects*	Número de octetos do PIN’
$L_{SUG}^{(M)}$	*Sugested L – Master device*	L sugerido pelo dispositivo mestre
$L_{SUG}^{(S)}$	*Sugested L – Slave device*	L sugerido pelo dispositivo escravo
$L_{MAX}^{(M)}$	*Maximum L – Master device*	L máximo suportado pelo dispositivo Mestre
$L_{MIN}^{(S)}$	*Minimal L – Slave device*	L mínimo suportado pelo dispositivo Escravo
LFSR	*Linear Feedback Shift Register*	Registrador de Deslocamento com Realimentação Linear
LK_KA	*Link Key – A device*	Chave de comunicação do dispositivo A
LK_KB	*Link Key – B device*	Chave de comunicação do dispositivo B
LK_RAND_A	*Link Key Random Number A*	Número aleatório usado na criação da chave de ligação do dispositivo A
LK_RAND_B	*Link Key Random Number B*	Número aleatório usado na criação da chave de ligação do dispositivo B
L_{MAX}	*Maximum L Lenght*	Comprimento máximo de L
L_{MIN}	*Minimal L Lenght*	Comprimento mínimo de L
OVL	*Overlay*	Calculado para permitir a autenticação
PHT	*Pseudo Hadamard Transform*	Transformada Pseudo aleatória de Hadamard

PIN	*Personal Identification Number*	Número Pessoal de Identificação
PIN’	*PIN Expansion*	Expansão do PIN
RAND1	*Random Number 1*	Número aleatório 1
RAND2	*Random Number 2*	Número aleatório 2
$RAND_A$	*Random Number A*	Número aleatório A
$RAND_B$	*Random Number B*	Número aleatório B
SAFER+	*Encryption function SAFER+*	Função de cifragem SAFER+
SAFER-SK [128]	*Encryption function SAFER-SK*	Função de cifragem SAFER-SK [128]
SRES	*Signed Response*	Resposta assinada
WiFi	*Wireless Fidelity*	Fidelidade sem fio
XOR	*Exclusive OR*	Função OU exclusivo

1 INTRODUÇÃO

A tecnologia Bluetooth [1] foi desenvolvida, principalmente, para reduzir a quantidade de fios necessários para podermos realizar a comunicação entre dois dispositivos próximos, tais como o *mouse* e o teclado. A idéia inicial foi muito bem explorada, o que nos permite, hoje em dia, conectar praticamente quaisquer equipamentos usando esta tecnologia. Já temos uma grande parte dos equipamentos eletrônicos atuais dispondo desta facilidade, o que expande as suas possibilidades de uso rotineiro, tais como, atender ao telefone sem precisar tocar no aparelho, escutar músicas no equipamento de som da nossa casa, que estejam armazenadas em um aparelho celular, ou mesmo em um tocador de músicas portátil, não havendo conexão de nenhum fio entre estes equipamentos. Podemos então imaginar que muitos outros equipamentos poderão dispor desta tecnologia em um curto espaço de tempo, facilitando muito a nossa vida. No entanto, será necessário termos segurança quanto às ações realizadas por estes equipamentos, ou poderemos ter surpresas desagradáveis, tais como subtração de informações confidenciais, ou mesmo manipulação de equipamentos eletrônicos, por terceiros, não autorizados originalmente a terem acesso a estes. Nesta dissertação é abordado o funcionamento desta tecnologia, relativo aos aspectos de segurança.

1.1 OBJETIVO E ESTRUTURA DA DISSERTAÇÃO

O objetivo inicial desta dissertação foi o de identificar possíveis falhas no processo de criação das chaves de segurança da tecnologia Bluetooth, porém ao iniciarmos as pesquisas, verificamos que seria necessário implementar todos os processos antes de buscar quaisquer possíveis falhas. Deste modo, nesta dissertação são analisados detalhadamente os processos de criação das chaves de segurança da tecnologia Bluetooth.

Esta dissertação contém quatro capítulos e dois anexos. No **Capítulo 2** descrevemos as rotinas de criação das chaves de segurança, tais como os algoritmos E_0, E_1, E_{21}, E_{22} e E_3 [3], os quais são detalhados ao longo deste capítulo.

No **Capítulo 3** são explicadas as funções criadas no *software* desenvolvido nesta dissertação, assim como alguns exemplos das rotinas e da execução do *software*.

No **Anexo 1** é disponibilizado o *software* Bluetooth.cpp, que é bastante simples, uma vez que a complexidade estará no arquivo responsável pela execução visual do *software*.

No **Anexo 2** o arquivo Bluetooth.h é disponibilizado para que o leitor possa ter acesso integral a todas as funções criadas para execução dos processos de criação das chaves de segurança. A complexidade deste arquivo, exige conhecimento na área de programação em visual C++ para ser melhor analisado.

1.2 BLUETOOTH – UMA VISÃO GERAL

Bluetooth é o nome da tecnologia de conexão sem fio, também conhecido como IEEE 802.15 e a sua tecnologia de comunicação permite mobilidade, desde que dentro dos limites de conexão. O nome desta tecnologia foi dado em homenagem ao Rei Harold Bluetooth [2] que unificou tribos norueguesas, suecas e dinamarquesas no século X. Até o presente, o alcance máximo entre dois dispositivos é de cerca de 100 metros em linha de visada.

Tabela 1.1 – Classes, potências e alcances da tecnologia Bluetooth.

Classe	Potência máxima permitida (mW, dBmW)	Alcance (aproximado)
Classe 1	100 mW (20 dBmW)	100m
Classe 2	2.5 mW (4 dBmW)	10m
Classe 3	1 mW (0 dBmW)	1m

Na Tabela 1.1, podemos verificar que existem três classes possíveis para definição de um dispositivo compatível com o Bluetooth, sendo a Classe 1, a qual possui a maior potência, a que proporciona o alcance máximo para esta tecnologia, conforme indicado na terceira coluna.

Tabela 1.2 – Versões e taxas de transmissão de dados da tecnologia Bluetooth.

Versão	Taxa de transmissão
Versão 1.2	1 Mbits/s
Versão 2.0 + EDR (*Enhanced Data Rate*)	3 Mbits/s
Versão 3.0	54 Mbits/s
Versão 4.0	(em desenvolvimento)

Até o momento em que este texto foi escrito, não tínhamos ainda a conclusão da versão 4.0 da tecnologia Bluetooth, que está sendo desenvolvida com o foco na redução do consumo de energia. As versões e suas respectivas taxas máximas de transmissão de dados podem ser vistas na Tabela 1.2. A versão 2.0 + EDR, permite triplicar a taxa da versão 1.2. A versão 3.0 usa a tecnologia "Wi-Fi" para transmitir dados, podendo atingir o máximo de 54 Mbits/s. A versão 4.0 deverá ter uma taxa de transmissão de dados mais baixa do que as versões anteriores, pois um dos grandes "problemas" da tecnologia Bluetooth é o seu consumo de energia, que é relativamente alto para poder ser colocado em equipamentos de tamanho físico reduzido, esta nova versão está sendo desenvolvida com a intenção de solucionar este "problema".

Podemos observar, por exemplo, que ao ativarmos a comunicação via Bluetooth nos nossos aparelhos de telefonia celular, a sua bateria é consumida muito mais rapidamente do que com o recurso desativado. Isto é um ponto negativo para o consumidor, que precisará recarregar o seu aparelho com maior frequência.

Já existem no mercado diversos equipamentos com a tecnologia Bluetooth, inclusive os fabricantes de consoles de jogos Sony [25] e Nintendo [26] aderiram ao uso deste recurso, pois puderam eliminar a necessidade de fios de conexão entre o console e o *joystick*. Outros equipamentos estão sendo e serão lançados com a tecnologia Bluetooth, só para citar alguns que facilitarão a nossa vida no futuro, vamos incluir os *mouses* e os teclados para computadores, que a cada dia estão ficando mais desejados devido a sua praticidade.

2 VISÃO SOBRE A SEGURANÇA DO BLUETOOTH

O sistema Bluetooth dispõe de segurança [3] na camada de aplicação e na camada de comunicação. Em cada dispositivo as rotinas de autenticação e de cifragem são implementadas do mesmo modo. Quatro diferentes entidades são usadas para manter a segurança na camada de comunicação:

- *BD_ADDR* (endereço do dispositivo);
- Duas chaves secretas;
- Número pseudo-aleatório [4] que deve ser gerado a cada nova transação.

Tabela 2.1 – Entidades e seus respectivos comprimentos (em bits).

Entidade	Tamanho
BD_ADDR (Endereço do dispositivo)	48 bits
Chave secreta do usuário para autenticação	128 bits
Chave secreta do usuário para cifragem (tamanho configurável – por byte)	8 – 128 bits (1 a 16 bytes)
Número Pseudo aleatório	128 bits

Na Tabela 2.1 podemos identificar as quatro entidades usadas para manter a segurança na camada de comunicação. O endereço do dispositivo é único e identifica os dispositivos capazes de realizar comunicação de dados via rede Bluetooth. O endereço pode ser obtido manualmente via iterações do usuário, ou automaticamente, via rotina própria de um dispositivo. As chaves secretas são provenientes da inicialização e nunca são divulgadas. A chave de cifragem é derivada da chave de autenticação durante o processo de autenticação e pode ser configurada desde um byte até dezesseis bytes, por duas razões:

- Adaptação aos diferentes modos de cifragem usados em cada país;
- Facilidade para realização de futuras atualizações dos algoritmos de cifragem, sem necessidade de alteração de *hardware*.

A chave de cifragem é diferente da chave de autenticação:

- A chave de autenticação é única para cada comunicação estabelecida;
- A chave de cifragem é renovada para cada nova transmissão realizada.

Deste modo, podemos ter várias chaves de cifragem para uma mesma chave de autenticação. O número pseudo-aleatório é gerado por um processo pseudo-aleatório interno do dispositivo, não é um parâmetro estático, está sempre sendo modificado. O processo de obtenção deste número aleatório é realizado via *software*, embora pudesse ser realizado via *hardware*. Aqui, poderíamos utilizar, por exemplo, o ruído térmico gerado por um resistor para obter o número aleatório. Este modo de obtenção iria garantir um número aleatório, porém é muito mais simples implementarmos uma rotina via *software*, que nos fornecerá um número pseudo-aleatório com características aceitáveis, para usarmos nos processos de criação das chaves de segurança.

Embora a chave de cifragem tenha comprimento variável, não pode ser alterada pelo usuário. O comprimento desta chave é próprio de cada dispositivo, sendo definido pelo seu fabricante. Também não será aceita uma chave criada ou alterada por camadas de *software* de nível mais alto (para tentar evitar ataque de hackers).

Para acomodar os diferentes tipos de aplicações, quatro tipos de chaves foram definidos:

- Chave de combinação: K_{AB};
- Chave da unidade ou chave do dispositivo: K_A, K_B;
- Chave temporária ou chave mestra: K_{master};
- Chave de inicialização: K_{init};

Adicionalmente a estas chaves, temos ainda a chave de cifragem K_C. K_A, K_B e K_{AB} tem suas funcionalidades semelhantes, a diferença está apenas no modo como são geradas:

- K_A e K_B dependem apenas de um dispositivo, *A* e *B* respectivamente;
- K_{AB} depende dos dois dispositivos *A* e *B*;

K_{master} deve ser usada apenas durante a sessão atual, ela substitui temporariamente a chave de ligação. Por exemplo, para conectar-se a dois dispositivos distintos ao mesmo tempo.

K_{init} deve ser usada como chave de ligação durante o processo de inicialização quando nenhuma combinação de chaves foi ainda realizada, ou quando a chave de ligação foi perdida. Esta chave é obtida a partir de um número aleatório, de um número PIN (em bytes) e pelo *BD_ADDR* e deve ser usada apenas na inicialização.

O PIN pode ser um número fixo fornecido com o dispositivo, pode ser escolhido pelo usuário, ou ainda pode ser definido por um dos dispositivos, sendo renovado a cada início de conexão. Em aparelhos de som para carro, temos alguns exemplos de PIN fixo, já em aparelhos de telefonia celular, temos condições de escolher o PIN que desejarmos. Quando queremos conectar um computador, por exemplo, a um aparelho de telefonia celular, podemos ter um PIN definido pelo sistema operacional do computador, o qual será renovado a cada início de conexão. Um valor inicial do PIN = 0x00 (zerado) pode ser usado; o comprimento padrão deste PIN zerado é de um byte. O comprimento mais comum de um PIN é de quatro dígitos decimais (cerca de quatorze dígitos binários), apenas por costume popular, mas o PIN pode ter de um a dezesseis bytes (binário), ou cerca de trinta e oito dígitos decimais no máximo. Vale ressaltar que com trinta e nove dígitos decimais, ultrapassamos a barreira dos 128 bits, ou 16 bytes, que é o máximo comprimento permitido.

2.1 GERAÇÃO DA CHAVE DA UNIDADE

Para serem geradas as chaves de cada unidade, primeiro cada dispositivo deve gerar um número aleatório: $RAND_A$ para o dispositivo *A* e $RAND_B$ para o dispositivo *B*. Após este passo, utilizamos o algoritmo E_{21} em cada dispositivo, para encontrar a K_A e para encontrar a K_B respectivamente. Para o cálculo da K_A, usamos o $RAND_A$ e o BD_ADDR_A e para o cálculo da K_B, usamos o $RAND_B$ e o BD_ADDR_B. Nesta etapa, a nossa chave de ligação *K*, usada no cálculo de K_A e K_B será igual à K_{init}. A chave K_A poderá ser enviada para o dispositivo *B* conforme mostrado na Figura 2.1.

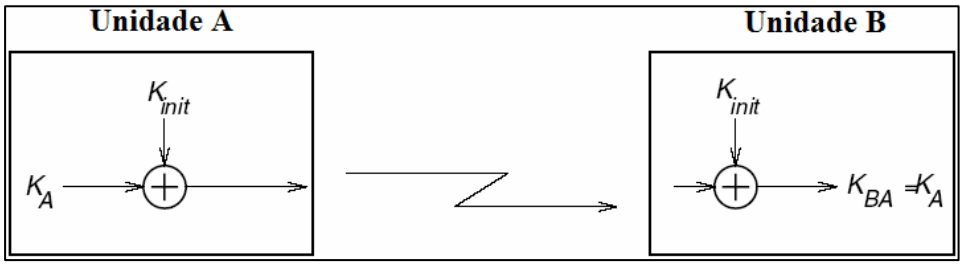


Figura 2.1 – Transmissão da chave da unidade.

2.2 GERAÇÃO DA CHAVE COMBINADA

Para gerarmos a chave combinada, ou seja, a chave comum a dois dispositivos, precisamos dispor das seguintes chaves: LK_K_A e LK_K_B, ver a expressão (2.1) e a expressão (2.2). O LK_RAND_A e o LK_RAND_B são gerados nos dispositivos *A* e *B*, respectivamente. No dispositivo *A*, calculamos C_A e no dispositivo *B*, calculamos C_B, ambos são calculados realizando-se uma operação XOR entre a chave gerada pela unidade e a chave *K*, ver expressão (2.4) e a expressão (2.5). C_A é então enviado para o dispositivo *B* e C_B é enviado para o dispositivo *A*. De posse de C_B, o dispositivo *A* é capaz de encontrar o LK_RAND_B, realizando uma operação XOR entre C_B e *K*. O dispositivo *B*, por sua vez, é capaz de encontrar o LK_RAND_A. A partir do LK_RAND_B e do BD_ADDR_B, o dispositivo *A* calcula LK_K_B e de modo análogo o dispositivo *B* calcula LK_K_A. A chave combinada é então encontrada em ambos os dispositivos realizando-se uma operação XOR entre LK_K_A e LK_K_B. Esta explicação está ilustrada na Figura 2.2.

$$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A), \tag{2.1}$$

$$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B), \tag{2.2}$$

$$K = K_{init}, \tag{2.3}$$

$$C_A = LK\_RAND_A \oplus K, \tag{2.4}$$

$$C_B = LK\_RAND_B \oplus K, \tag{2.5}$$

$$K_{AB} = LK\_K_A \oplus LK\_K_B. \tag{2.6}$$

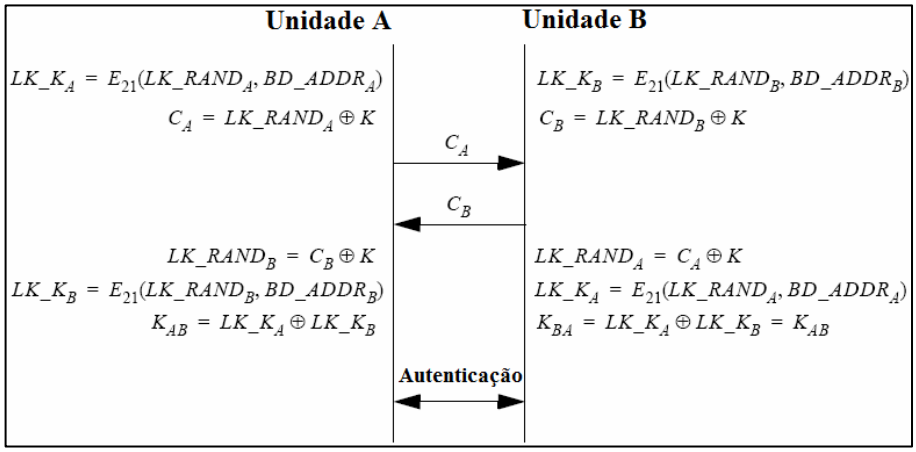


Figura 2.2 – Geração da chave combinada (K_{AB}).

2.3 GERAÇÃO DA CHAVE DE CIFRAGEM

A chave de cifragem K_C é obtida pelo algoritmo E_3, usando-se como parâmetros:

- A chave de ligação *K*;
- o *COF* (96-bit Ciphering Offset);
- e um número aleatório de 128 bits.

O *COF* pode ser determinado de duas maneiras, a depender da chave *K*. Se *K* for igual à K_{master} então o *COF* será o endereço do dispositivo concatenado duas vezes. Se *K* for diferente de K_{master} então o *COF* será obtido pelo cálculo do *ACO* (Authenticated ciphering offset), que é um parâmetro auxiliar gerado por um processo de autenticação bem sucedido. Na expressão (2.7), $\cup$ denota concatenação.

$$COF = \begin{cases} BD\_ADDR \cup BD\_ADDR, se\ K = K_{master}, \\ ACO, nos\ \ outros\ \ casos. \end{cases} \quad (2.7)$$

2.4 GERAÇÃO DA CHAVE TEMPORÁRIA OU CHAVE MESTRE

A rotina que será descrita a seguir é considerada semi-permanente, pois deve ser gerada a cada nova sessão. A chave mestra deverá substituir a chave de ligação durante uma sessão. Antes de tudo, o dispositivo mestre, que no nosso texto será definido como sendo o dispositivo A, deverá gerar dois números aleatórios novos, $RAND_1$ e $RAND_2$, depois a K_{master} será gerada usando-se o algoritmo E_{22}, como mostrado na expressão (2.8),

$$K_{master} = E_{22}(RAND_1, RAND_2, 16). \tag{2.8}$$

O motivo pelo qual o algoritmo E_{22} é usado decorre do fato que os números aleatórios gerados são na verdade números pseudo-aleatórios. Um terceiro número aleatório *RAND*, deve ser gerado pelo mestre e transmitido para o dispositivo escravo. Após esta transmissão, mestre e escravo devem utilizar o algoritmo E_{22} para calcular o overlay (*OVL*), desta vez com os parâmetros *K* e *RAND*, sendo *K* a nossa chave de ligação atual,

$$OVL = E_{22}(K, RAND, 16). \tag{2.9}$$

O mestre deve calcular *C* por meio de:

$$C = OVL \oplus K_{master}, \tag{2.10}$$

e então transmitir o resultado para o escravo; por sua vez, o escravo, que já calculou o *OVL*, poderá encontrar a chave mestre K_{master}, realizando também uma operação XOR entre *OVL* e *C*, como pode ser observado na Figura 2.3.

Para confirmar o sucesso desta operação, os dispositivos devem realizar uma autenticação mútua usando a nova chave de ligação. O valor do *ACO* encontrado deve servir apenas para confirmar o processo de autenticação, mas não deve substituir o valor do *ACO* atual, pois em caso de nova conexão, precisaremos do valor atual do *ACO*.

O mestre pode então ativar a cifragem dos dados, porém deve garantir pela autenticação de todos os escravos, que o número aleatório *EN_RAND*, gerado e transmitido pelo mestre, tenha chegado a todos os escravos. Com tudo confirmado, todos os dispositivos devem calcular a chave de cifragem K_C, como pode ser visto na expressão (2.11),

$$K_C = E_3(K_{master}, EN\_RAND, COF). \quad (2.11)$$

O valor do *COF* é decorrente do *BD_ADDR* do mestre, como especificado na expressão (2.7). Mais detalhes deste processo de geração da chave K_C são vistos na sequência do texto. Podemos ver o processo de distribuição da chave mestre na Figura 2.3.

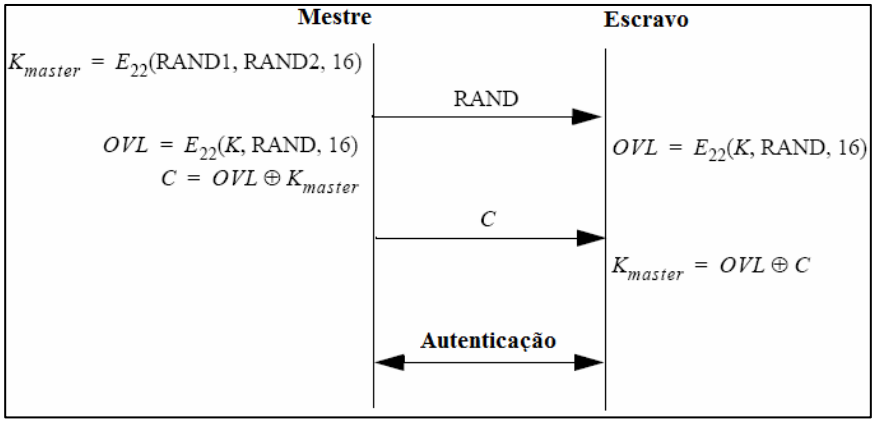


Figura 2.3 – Distribuição da chave mestre.

2.5 CRIPTOGRAFIA DA TECNOLOGIA BLUETOOTH

As informações do usuário podem ser protegidas pela cifragem dos pacotes de dados, porém o código de acesso e o cabeçalho dos pacotes nunca devem ser cifrados. A cifragem dos dados deve ser efetuada com uma cifra sequencial chamada E_0, que deve ser re-sincronizada para cada nova entrada de dados.

O sistema de geração da chave de cifragem é composto por duas partes:

– A primeira parte realiza a inicialização da chave de carga (*payload key generator*), que consiste, resumidamente, em combinar os bits das entradas na

ordem apropriada e inseri-los nos quatro registradores de deslocamento com realimentação linear (LFSRs), usados no gerador da chave de cifragem;

– A segunda parte gera os bits seqüenciais da chave de cifragem. O método usado é derivado de uma combinação não-linear de geradores de cifras sequenciais, atribuído a Massey [5] e .Rueppel [6]. Esta parte é a principal do sistema de cifragem, e também é usada para inicialização. Após estas duas etapas é realizada a cifragem ou a decifragem.

Um resumo do método de cifragem E_0, pode ser visualizado na Figura 2.4. O método de Massey e Rueppel foi bastante analisado e revelou-se bastante confiável para as técnicas de cripto-análise conhecidas até o momento. Porém sabe-se que o gerador com somas pode sofrer um ataque por correlação. Este ataque será minimizado devido à alta frequência de re-sincronização.

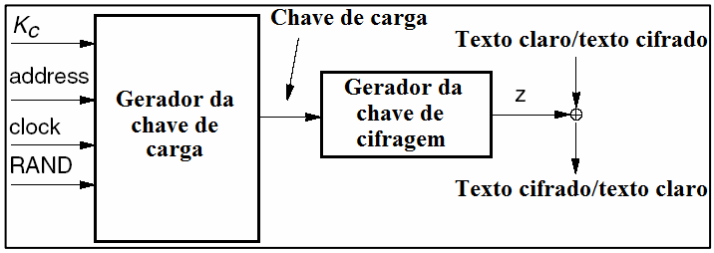


Figura 2.4 – Sistema E_0: Sequência de cifragem para IEEE 802.15.1-2005.

2.5.1 Comprimento da chave de cifragem

Cada dispositivo possui um comprimento máximo de chave de cifragem, L_{max}, $1 \leq L_{\max} \leq 16$ (onde L_{max} representa o número de octetos da chave). Para cada aplicação que use cifragem, um número L_{min} deve ser definido, indicando o comprimento mínimo permitido para a chave. Antes de gerar a chave de cifragem, os dispositivos envolvidos devem negociar entre si e decidir qual o comprimento da chave de cifragem.

O mestre deve inicialmente sugerir um valor para o escravo, $L_{sug}^{(M)}$. O valor inicial deve ser sugerido como $L_{\max}^{(M)}$. Se o $L_{\min}^{(S)} \leq L_{sug}^{(M)}$ e se o escravo aceitar o comprimento sugerido, deverá autorizar este valor, o qual será o comprimento da chave de cifragem. Porém, caso as condições não sejam atendidas, o escravo deverá enviar uma nova sugestão, $L_{sug}^{(S)} < L_{sug}^{(M)}$, para o mestre. Esta sugestão do escravo deverá ser a de maior comprimento possível para si, porém menor que a sugestão anterior do mestre. O mestre, por sua vez, deverá analisar a sugestão do escravo e concordar. Caso o mestre não concorde, o processo deverá ser repetido até que ambos concordem em usar um comprimento de chave de cifragem.

Existe a possibilidade de um processo de decisão de comprimento de chave terminar sem sucesso, porém é um risco necessário, para evitar que um dispositivo fraudulento tente realizar uma comunicação com uma chave muito curta. Caso não haja sucesso, um novo processo de decisão poderá ser iniciado pelo mestre, se várias tentativas falharem, o dispositivo mestre pode considerar o processo como sendo uma tentativa de invasão por parte do dispositivo requisitante, não permitindo que a conexão seja estabelecida.

2.5.2 Tipos possíveis de cifragem

O processo de cifragem pode ser desabilitado, ou seja, nenhuma mensagem será cifrada, este é o modo de operação definido como padrão. Caso o modo de cifragem seja habilitado, teremos dois modos de funcionamento:

- Cifragem apenas ponto-a-ponto, onde as mensagens de rádio difusão (*broadcast*) não serão cifradas, apenas as mensagens ponto-a-ponto serão cifradas;
- Ponto-a-ponto e rádio difusão com cifragem, onde todas as mensagens serão cifradas;

2.5.3 Conceito de cifragem para a tecnologia Bluetooth

Para realizar uma rotina de cifragem, usamos o algoritmo E_0, que gera a chave, que será adicionada módulo dois, bit a bit, com os dados a serem transmitidos via interface aérea. Cada um dos pacotes que contém os dados será cifrado separadamente. O algoritmo E_0 usa o endereço do dispositivo mestre (*BD_ADDR*), 26 bits do clock do mestre (CLK_{26-1}) e a chave K_C, como entradas. Podemos verificar na Figura 2.5 um resumo deste processo, no qual o dispositivo *A* é definido como sendo o mestre.

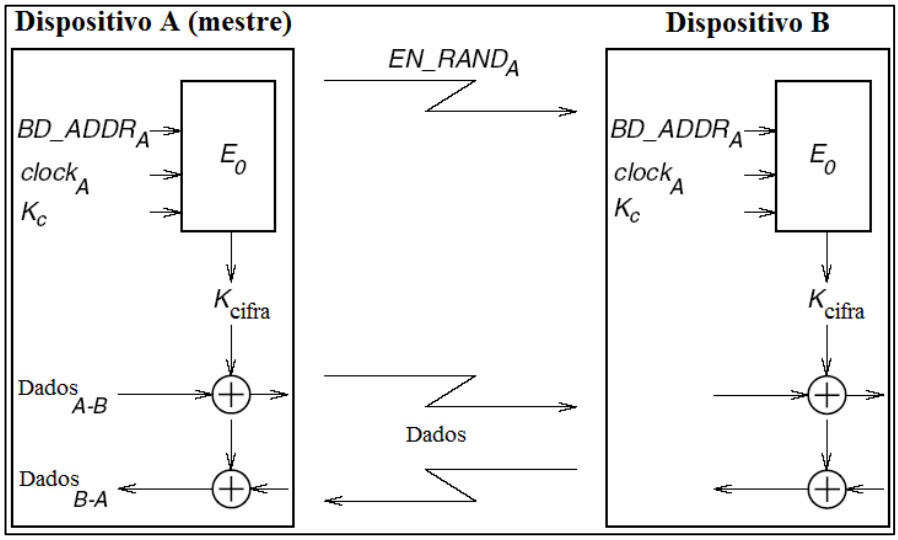


Figura 2.5 – Descrição funcional do processo de cifragem.

A chave de cifragem K_C é derivada da chave de ligação atual, do *COF* e do número aleatório EN_RAND_A. Este número aleatório deverá ser transmitido pelo mestre antes de entrar no modo de cifragem. Vale lembrar que EN_RAND_A é conhecido publicamente, pois é transmitido pelo mestre como texto claro.

Através do algoritmo E_0, a chave K_C é transformada em outra chave, denominada K'_C. O comprimento máximo efetivo desta chave é determinado pelo fabricante do dispositivo, sendo obrigatoriamente um múltiplo de oito (entre oito e 128 bits). O clock deve ser incrementado a cada *slot*, ou seja, na subida e na descida e o algoritmo E_0 deve ser reiniciado a cada início de um novo pacote de dados, ou seja, para uma transmissão mestre-escravo ou para uma

transmissão escravo-mestre. Deste modo, podemos perceber que a cada novo pacote de dados, uma nova chave será gerada.

O algoritmo de cifragem E_0 gera uma chave binária, K_{cifra}, a qual deverá ser adicionada módulo dois aos dados a serem cifrados, como pode ser observado na Figura 2.5. Esta cifra é simétrica e a decifragem é realizada do mesmo modo da cifragem, usando-se a mesma chave em ambos os processos.

2.5.4 Algoritmo de cifragem para a tecnologia Bluetooth

O sistema usa quatro registradores de deslocamento com realimentação linear, que denotaremos como 4 LFSRs [7], da sigla em inglês significando *linear feedback shift register*, cujas saídas são combinadas através de uma máquina de estados finitos (*summation combiner*), que iremos interpretar como lógica combinacional aditiva. A saída desta máquina poderá ser a sequência de cifragem Z_t composta por 200 bits, dos quais os 128 últimos bits formarão a chave de cifragem, ou, durante a inicialização, cria um valor aleatório inicial. O algoritmo usa uma chave de cifragem K_C, um endereço de 48 bits, os bits do clock do mestre $CLK_{26\text{-}1}$ e um valor aleatório de 128 bits. Podemos ver na Figura 2.6 um resumo deste algoritmo.

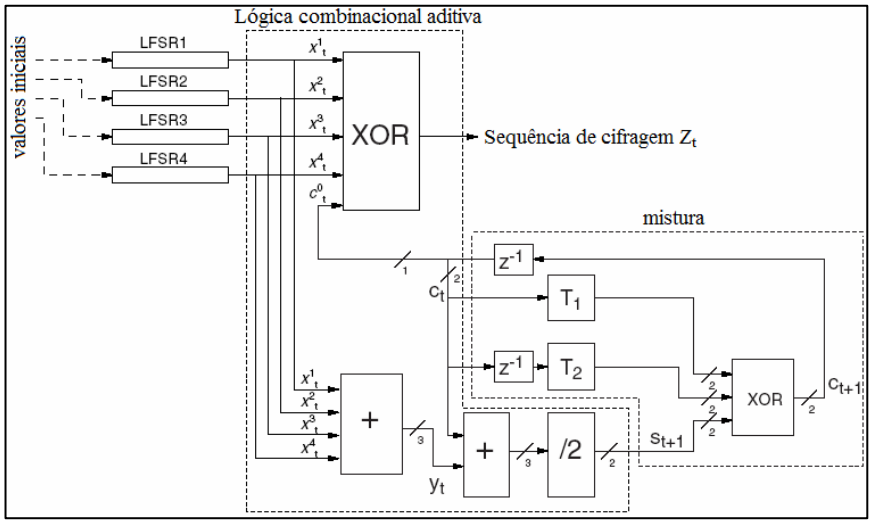


Figura 2.6 – Estrutura da máquina de cifragem.

Podemos observar na Tabela 2.2 que L_i é o comprimento dos registradores de deslocamento, com os polinômios de realimentação representados por $f_i(t)$. O comprimento total somado dos registradores de deslocamento é de 128 bits e o peso de Hamming = 5 [8], foi escolhido para diminuir o número de portas XOR necessárias para implementação do *hardware* e também para obter boas propriedades estatísticas das sequências geradas.

Tabela 2.2 – Os quatro polinômios primitivos de realimentação.

i	L_i	Realimentação $f_i(t)$	Peso
1	25	$t^{25}+t^{20}+t^{12}+t^{8}+1$	5
2	31	$t^{31}+t^{24}+t^{16}+t^{12}+1$	5
3	33	$t^{33}+t^{28}+t^{24}+t^{4}+1$	5
4	39	$t^{39}+t^{36}+t^{28}+t^{4}+1$	5

Seja x_t^i o t-ésimo símbolo do $LFSR_i$, o valor y_t é obtido através da 4t-upla $x_t^1,...,x_t^4$ usando a equação (2.12):

$$y_t = \sum_{i-1}^{4} x_t^i \,, \tag{2.12}$$

onde a soma é sobre os decimais, podendo, portanto, assumir os valores 0, 1, 2, 3 ou 4. A saída da lógica combinacional aditiva é obtida pela equação (2.13), equação (2.14) e equação (2.15).

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0,1\}, \tag{2.13}$$

$$s_{t+1} = (s_{t+1}^1, s_{t+1}^0) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0,1,2,3\}, \tag{2.14}$$

$$c_{t+1} = (c_{t+1}^1, c_{t+1}^0) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \tag{2.15}$$

onde T_1[.] e T_2[.] são duas bijeções lineares diferentes sobre GF(4). Suponha que GF(4) é gerado pelo polinômio primitivo $x^2 + x + 1$, e seja α um zero deste polinômio em GF(4). O mapeamento de T_1 e T_2 é definido por:

$$T_1 : GF(4) \to GF(4)$$
$$x \mapsto x$$

$$T_2 : GF(4) \to GF(4)$$
$$x \mapsto (\alpha + 1)x.$$

GF(4) corresponde ao *Galois Field 4*, ou seja estamos trabalhando sobre os corpos finitos de ordem 4. Os elementos de GF(4) podem ser vistos na Tabela 2.3.

Tabela 2.3 – Mapeamentos de T_1 e T_2

x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

Uma vez que os mapeamentos são lineares, eles podem ser implementados usando portas XOR:

$$T_1: \quad (x_1, x_0) \mapsto (x_1, x_0)$$

$$T_2: \quad (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0)$$

2.5.5 Inicialização dos LFSRs

O gerador da chave de cifragem é carregado com os valores dos quatro LFSRs (128 bits) e com os 4 bits referentes aos valores de c_0 e c_{-1}. Os parâmetros de entrada são a chave K_C, um número aleatório (*RAND*) de 128 bits, o endereço do dispositivo com 48 bits e 26 bits do clock do mestre (CLK_{26-1}). A chave K_C tem comprimento de 128 bits, pois é gerada pelo algoritmo E_3, porém, podemos reduzir o comprimento efetivo da chave para até oito bits, o que é possível com o uso do algoritmo E_0. O algoritmo E_3 pode ser observado em 2.7.4.

Os detalhes da inicialização são descritos abaixo:

a) Devemos criar uma chave de cifragem K_C com 128 bits e um número aleatório *EN_RAND* de 128 bits que é conhecido publicamente. Seja L, $1 \le L \le 16$, o comprimento efetivo da chave em números de octetos; definimos a chave K'_C por:

$$K'_C(x) = g_2^{(L)}(x)(K_C(x) \bmod g_1^{(L)}(x)), \qquad (2.16)$$

onde o grau do polinômio $g_1^{(L)}(x)$ é igual a 8L e o grau do polinômio $g_2^{(L)}(x)$ é menor ou igual a 128-8L. Ambos os polinômios são definidos pela Tabela 2.4, observe que a notação está em hexadecimal.

Tabela 2.4 – Polinômios usados na criação de K'_C em notação hexadecimal.

L	Grau	$g_1^{(L)}$	Grau	$g_2^{(L)}$
1	[8]	00000000 00000000 00000000 0000011d	[119]	00e275a0 abd218d4 cf928b9b bf6cb08f
2	[16]	00000000 00000000 00000000 0001003f	[112]	0001e3f6 3d7659b3 7f18c258 cff6efef
3	[24]	00000000 00000000 00000000 010100db	[104]	000001be f66c6c3a b1030a5a 1919808b
4	[32]	00000000 00000000 00000001 000000af	[96]	00000001 6ab89969 de17467f d3736ad9
5	[40]	00000000 00000000 00000100 00000039	[88]	00000000 01630632 91da50ec 55715247
6	[48]	00000000 00000000 00010000 00000291	[77]	00000000 00002c93 52aa6cc0 54468311
7	[56]	00000000 00000000 01000000 00000095	[71]	00000000 000000b3 f7fffce2 79f3a073
8	[64]	00000000 00000001 00000000 0000001b	[63]	00000000 00000000 a1ab815b c7ec8025
9	[72]	00000000 00000100 00000000 00000609	[49]	00000000 00000000 0002c980 11d8b04d
10	[80]	00000000 00010000 00000000 00000215	[42]	00000000 00000000 0000058e 24f9a4bb
11	[88]	00000000 01000000 00000000 0000013b	[35]	00000000 00000000 0000000c a76024d7
12	[96]	00000001 00000000 00000000 000000dd	[28]	00000000 00000000 00000000 1c9c26d9
13	[104]	00000100 00000000 00000000 0000049d	[21]	00000000 00000000 00000000 0026d9e3
14	[112]	00010000 00000000 00000000 0000014f	[14]	00000000 00000000 00000000 00004377
15	[120]	01000000 00000000 00000000 000000e7	[7]	00000000 00000000 00000000 00000089
16	[128]	1 00000000 00000000 00000000 00000000	[0]	00000000 00000000 00000000 00000001

b) Devemos carregar os LFSRs com K_C, com o endereço do dispositivo, com o clock e com a constante 111001 de acordo com a Figura 2.7, num total de 208 bits, seguindo-se as etapas:

1) Abra todas as chaves mostradas na Figura 2.7;
2) Organize os bits como é mostrado na Figura 2.7, zere todos os bits dos LFSRs e faça $t = 0$;
3) Comece a deslocar os bits dentro dos LFSRs. Os bits mais a direita de cada nível serão os primeiros a entrar nos seus respectivos LFSRs;
4) Quando o primeiro bit a ter sido deslocado em cada um dos níveis alcançar a posição mais a direita do seu LFSR, feche a chave deste registrador;
5) Quando $t = 39$ (a chave do $LFSR_4$ será fechada), zere os dois registradores $c_{39} = c_{39-1} = 0$. Até este ponto, o conteúdo destes dois registradores não possuía nenhuma utilidade, porém a partir de agora, os seus conteúdos serão usados para calcular a sequência de saída.
6) De agora em diante os símbolos serão gerados. Os bits restantes serão continuamente deslocados dentro dos seus respectivos registradores. Após o

último bit ter sido carregado no registrador, a entrada deverá ser colocada para zero;

Obs: Quando o processo tiver sido concluído, o $LFSR_1$ terá efetivamente sido deslocado 30 vezes, o $LFSR_2$ 24 vezes, o $LFSR_3$ 22 vezes e o $LFSR_4$ 16 vezes, todos com a chave de realimentação fechada.

c) Para misturar os dados iniciais, continue com o clock ativo até que sejam produzidos 200 símbolos com todas as chaves fechadas (t = 239);

d) Mantenha os valores dos registradores c_t e c_{t-1}; faça o carregamento paralelo dos últimos 128 bits gerados nos LFSRs, de acordo com a Figura 2.8. (t = 240).

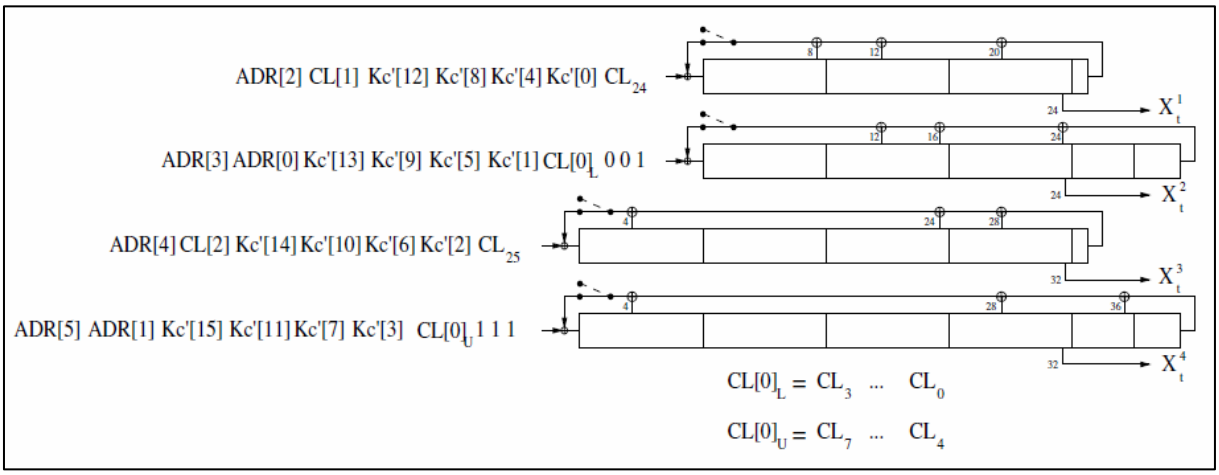


Figura 2.7 – Organização das entradas dos LFSRs

A Figura 2.7 mostra a sequência dos dados que serão carregados nos LFSRs, iniciando-se pelo bit menos significativo de cada octeto. Por exemplo: para o terceiro octeto do endereço *BD_ADDR[2]*, o primeiro bit será o BD_ADDR_{16}, seguido pelo BD_ADDR_{17}. CL_0 corresponde a CLK_1 e CL_{25} corresponde a CLK_{26}. Podemos observar também que os símbolos de saída $x_t^i, i = 1, ...,4$, são obtidos das posições 24, 24, 32 e 32 dos respectivos $LFSR_1$, $LFSR_2$, $LFSR_3$ e $LFSR_4$, onde a posição um é definida como sendo a primeira do lado esquerdo.

Na Figura 2.8, os símbolos de saída Z_0, ... , Z_{127} são organizados em octetos, representados por *Z[0]*, ... , *Z[15]*. O bit menos significativo de *Z[0]* corresponde ao primeiro destes símbolos, enquanto que o bit mais significativo de *Z[15]* corresponde à última saída dos LFSRs. Os octetos devem ser carregados com uma entrada paralela dos dados, não havendo nenhuma modificação nos dados de saída gerados e o bit menos significativo será o

localizado mais a esquerda, o oposto do que foi feito anteriormente, por exemplo, Z_{24} será carregado na posição um do $LFSR_4$.

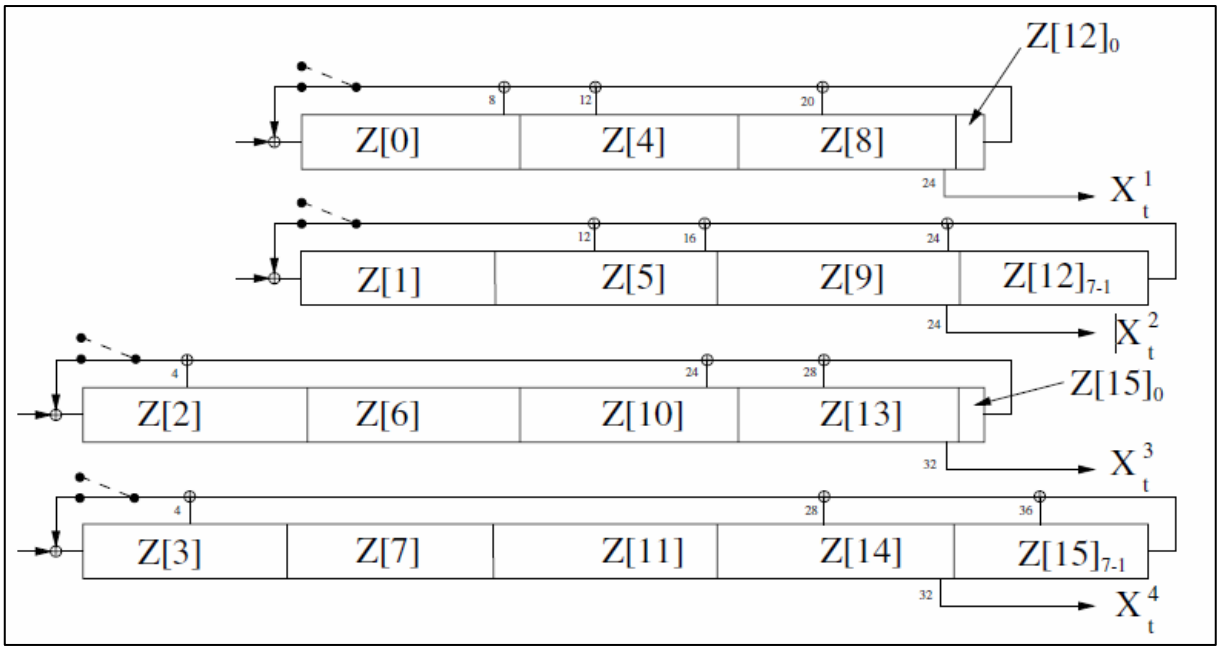


Figura 2.8 – Distribuição dos últimos 128 símbolos gerados na saída dos LFSRs

2.6 AUTENTICAÇÃO

A autenticação usa um esquema de desafio/resposta, no qual o conhecimento da chave secreta pelo requerente é verificado através de um protocolo de dois movimentos, usando chaves secretas simétricas, ou seja, este processo verifica se os dois dispositivos possuem a mesma chave secreta, *K*. Neste esquema de desafio/resposta o verificador desafia o requerente a autenticar um número aleatório AU_RAND_A, usando-se o algoritmo E_1 e retornando o valor do *SRES* (*signed response*), resposta assinada, para o dispositivo verificador. A chave secreta *K*, compartilhada pelos dois dispositivos *A* e *B* é a chave de ligação atual. Podemos ver o esquema de desafio/resposta na Figura 2.9.

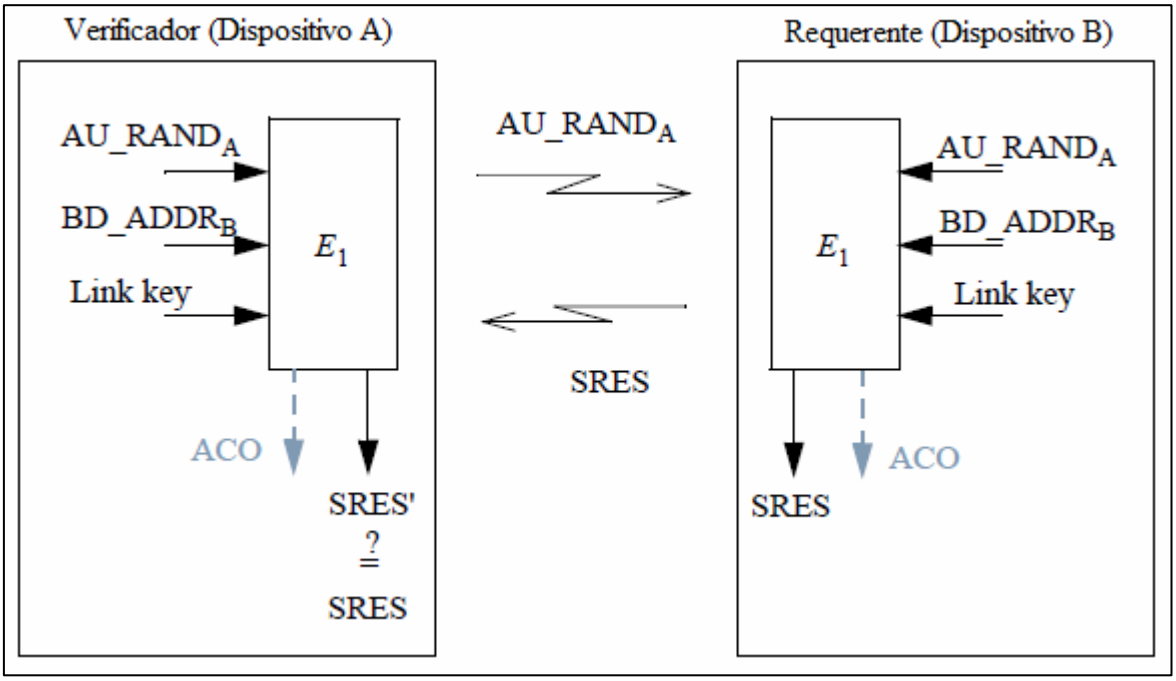


Figura 2.9 – Esquema de pergunta/resposta

O verificador não precisa ser necessariamente o mestre, na verdade, a aplicação é quem define a necessidade de autenticação, podendo ser unilateral ou bilateral. Na unilateral, apenas o verificador precisa autenticar o requerente, já na bilateral, após o primeiro processo de autenticação, o dispositivo requerente irá se tornar o verificador, precisando autenticar o outro dispositivo, agora fazendo o papel de requerente.

Quando o processo de autenticação falha, um intervalo de tempo deve ser respeitado antes que o verificador inicie um novo processo de tentativa de autenticação com o mesmo requerente, identificado pelo seu *BD_ADDR*. Para cada processo sem sucesso o tempo de espera deve ser aumentado exponencialmente. Um parâmetro de acréscimo que poderá ser escolhido é o de dobrar o intervalo de tempo de espera anterior. O objetivo desta restrição é evitar que um invasor tente realizar o processo de autenticação com um grande número de chaves diferentes em um curto espaço de tempo, usando o método da tentativa e erro.

Só para exemplificar, caso seja definido um tempo inicial de 1ms (um mili segundo) e dobremos este tempo a cada falha, na tentativa de número 11, o tempo de espera seria de cerca de um segundo. O tempo de espera máximo, entre tentativas falhas, é definido pelo dispositivo verificador, não havendo nenhum limite teórico máximo definido. Mais uma vez, tomando-se o nosso exemplo, caso o tempo de espera máximo seja de um segundo, para testar

todas as possibilidades de chaves secretas, que possuem 128 bits, seriam necessários aproximadamente $1{,}079 \times 10^{31}$ anos.

Quando o modo de operação com segurança do Bluetooth está habilitado, o comprimento do PIN define diretamente o intervalo de tempo no qual a nossa transmissão estará segura. Caso o usuário use o PIN padrão de quatro dígitos decimais, o criptoanalista poderá quebrar a chave de segurança em cerca de 17 segundos sem a restrição de dobrar o tempo a cada erro, ou cerca de quatro horas e meia usando-se a restrição, supondo ainda que o tempo entre tentativas falhas é de um mili segundo.

Se estipularmos um prazo relativamente seguro, de no mínimo 100 anos, podemos chegar a conclusão que um PIN de 14 dígitos decimais, ou cerca de 44 dígitos binários, faria o criptoanalista trabalhar por cerca de 557 anos sem a restrição de dobrar o tempo a cada erro ou 557000 anos com a restrição habilitada, onde o tempo entre tentativas falhas continua sendo um mili segundo.

Deste modo, é possível definir um tempo de segurança mínimo da sua informação e restringir a quantidade de dígitos decimais do seu PIN, de modo a trabalhar com o mínimo de segurança necessário às suas informações.

2.7 FUNÇÕES DE GERAÇÃO DA CHAVE E DA AUTENTICAÇÃO

2.7.1 A função de autenticação E_1

A função de autenticação E_1 é um código de autenticação computacionalmente seguro. E_1 usa a função de cifragem SAFER+ [9]. Este algoritmo é uma versão aperfeiçoada da cifra de bloco de 64 bits SAFER-SK 128 [10] e está disponível gratuitamente. Vamos representar a cifra de bloco SAFER+ como sendo A_r, que faz um mapeamento de uma chave de 128 bits, uma entrada de 128 bits e produz uma saída de 128 bits.

$$A_r : \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128} \\ (k \times x) \rightarrow t. \tag{2.17}$$

A função E_1 é construída usando-se A_r do seguinte modo:

$$E_1 : \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32} \times \{0,1\}^{96} \\ (K, RAND, address) \rightarrow (SRES, ACO), \tag{2.18}$$

onde $SRES = Hash(K, RAND, address, 6)[0, ..., 3]$, onde *Hash* [11] é uma função definida como:

$$Hash : \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{128} \\ (K, I_1, I_2, L) \rightarrow A_r([\tilde{K}], [E(I_2, L)] +_{16} (A_r(K, I_1) \oplus_{16} I_1)]), \tag{2.19}$$

e onde

$$E : \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{8 \times 16} \\ (X[0, ..., L-1], L) \rightarrow (X[i (\mathrm{mod}\, L)] \ \ Para\ i = 0...15), \tag{2.20}$$

é uma expansão das palavras *X*, de *L* octetos, em uma palavra de 128 bits. A função A_r é calculada duas vezes para cada cálculo de E_1. A chave *K* para o segundo cálculo usando-se A_r (na verdade A'_r) é o offset de *K*, definido por $\tilde{K}$, como pode ser observado na equação (2.21),

$$\begin{aligned}
&K[0] = (K[0] + 233) \mod 256, && K[1] = (K[1] \oplus 229), \\
&K[2] = (K[2] + 223) \mod 256, && K[3] = (K[3] \oplus 193), \\
&K[4] = (K[4] + 179) \mod 256, && K[5] = (K[5] \oplus 167), \\
&K[6] = (K[6] + 149) \mod 256, && K[7] = (K[7] \oplus 131), \\
&K[8] = (K[8] \oplus 233), && K[9] = (K[9] + 229) \mod 256, \\
&K[10] = (K[10] \oplus 223), && K[11] = (K[11] + 193) \mod 256, \\
&K[12] = (K[12] \oplus 179), && K[13] = (K[13] + 167) \mod 256, \\
&K[14] = (K[14] \oplus 149), && K[15] = (K[15] + 131) \mod 256.
\end{aligned} \tag{2.21}$$

Podemos ver um resumo do processo de cálculo de E_1 na Figura 2.10. E_1 também é usado para calcular o parâmetro *ACO* que é usado na geração da chave de cifragem por E_3, veja a expressão (2.7) e a expressão (2.29). O valor do *ACO* é formado pelos octetos quatro até quinze da saída da função *Hash*, definida pela expressão (2.19),

$$ACO = Hash(K, RAND, address, 6)[4, ..., 15]. \quad (2.22)$$

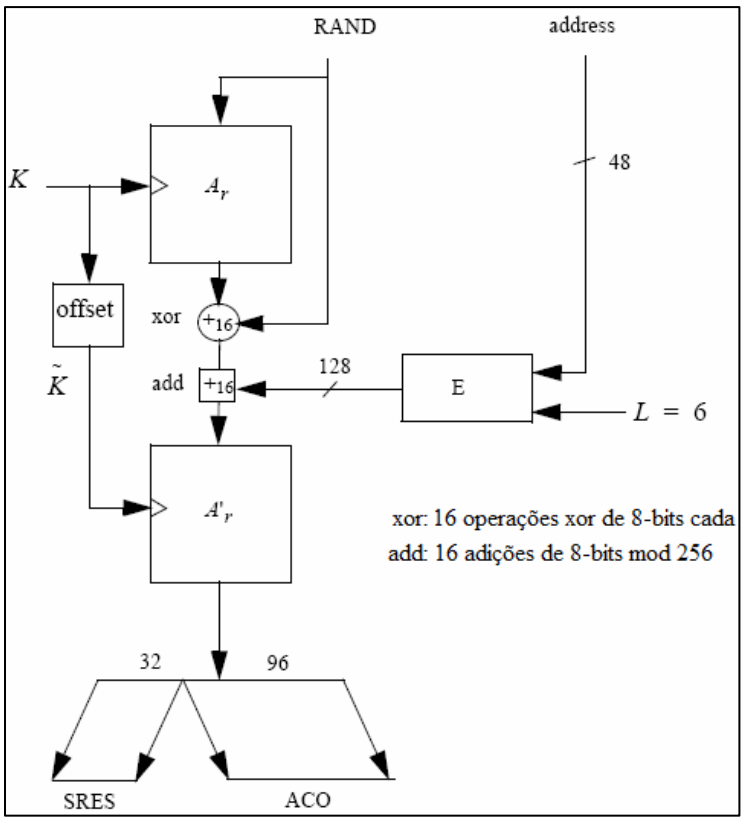


Figura 2.10 – Fluxo dos dados para o cálculo de E_1

2.7.2 As funções A_r e A'_r

A função A_r é idêntica ao SAFER+, que consiste em um conjunto de oito camadas, cada camada é chamada de *round* (etapa), e um mecanismo paralelo de geração das subchaves $K_p[j]$, p = 1, 2, ..., 17, que são as chaves, a serem usadas em cada etapa. Esta função irá produzir uma saída de 128 bits a partir de uma sequência de entrada aleatória e de uma chave de 128 bits. Além da função A_r, iremos usar uma versão ligeiramente modificada, a qual iremos nos referir como A'_r. A modificação consiste em adicionar a entrada da primeira etapa

à entrada da terceira etapa. Esta modificação torna o processo não inversível e impede o uso de A'_r, especialmente em E_{2x}, como uma função de cifragem. Veja a Figura 2.11 para mais detalhes.

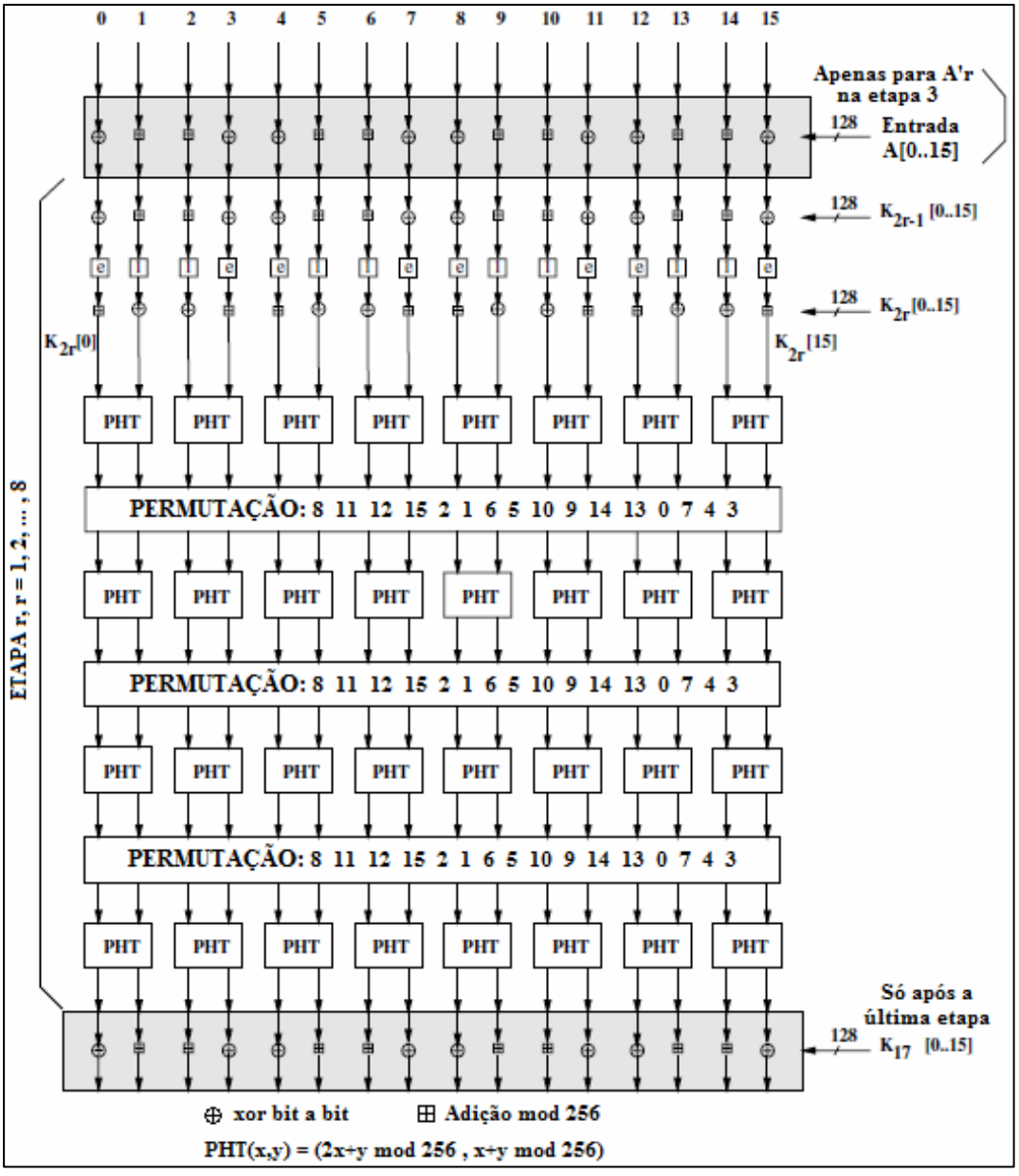


Figura 2.11 – Uma etapa em A_r ou A'_r.

Os cálculos realizados nas etapas são uma composição da cifragem com as chaves intermediárias de cada etapa (subchaves), substituições, pseudo transformada de Hadamard (PHT) [12] e permutações. As sub-chaves para cada etapa r, r = 1, 2, ..., 8 são representadas por $K_{2r-1}[j]$, $K_{2r}[j]$, j = 0, 1, ..., 15. Após a última etapa, $K_{17}[j]$ é aplicada da mesma forma que todas as chaves anteriores. O processo de geração das subchaves pode ser visto na Figura

2.12. Nesta mesma figura, são mostradas duas caixas, marcadas com *e* e *l*. Estas caixas implementam as mesmas substituições usadas no SAFER+:

$$\begin{aligned} e, l \quad &: \quad \{0, ..., 255\} \rightarrow \{0, ..., 255\}, \\ e \quad &: \quad i \mid\rightarrow (45^i \,(\mathrm{mod}\, 257))(\mathrm{mod}\, 256), \\ l \quad &: \quad i \mid\rightarrow j \ \text{tal que: i = e(j).} \end{aligned}$$

A intenção destas substituições é introduzir não linearidade ao processo. As caixas de permutações mostradas na Figura 2.11 mostram como os bytes de entrada serão mapeados nos índices de saída, ou seja, a posição oito é mapeada na posição zero, mais à esquerda, a posição onze é mapeada na posição um e assim por diante.

Em cada etapa são necessárias duas chaves de 16 octetos cada. Estas chaves das etapas são definidas como especificado pelo agendamento de chave do SAFER+. Na Figura 2.12, podemos verificar como as chaves das etapas $K_p[j]$ são determinadas. Os vetores de *bias* B_2, B_3, ..., B_{17} devem ser calculados de acordo com a equação (2.23).

$$B_p[i] = ((45^{(45^{17\,p+i+1} \bmod 257)} \bmod 257) \bmod 256), \ \textit{para} \quad i = 0, ...,15. \tag{2.23}$$

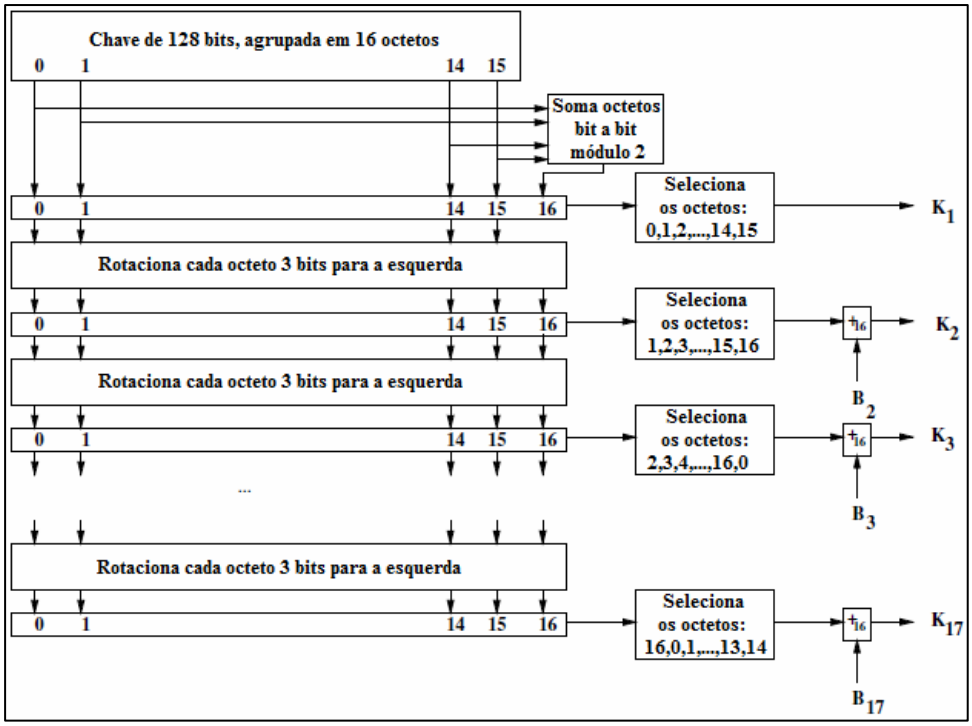


Figura 2.12 – Geração das chaves das etapas para o cálculo de A_r.

2.7.3 Função E_2 de geração da chave para autenticação

A chave usada para autenticação é calculada através do procedimento mostrado na Figura 2.13. Nela são mostrados os dois modos de operação para o algoritmo. No primeiro modo, definido por E_{21}, é produzida uma chave K de 128 bits, usando-se o número aleatório *RAND* de 128 bits e o endereço do dispositivo de 48 bits. Este modo deve ser utilizado quando criamos a chave da unidade e a chave de combinação. No segundo modo, definido por E_{22}, também é produzida uma chave K de 128 bits, desta vez usando-se o número aleatório *RAND* e L octetos do PIN do usuário. O segundo modo deve ser usado para criar a chave de inicialização e também quando a chave mestre é gerada.

Quando a chave de inicialização é gerada, o PIN pode ser concatenado com o endereço do dispositivo *BD_ADDR*, a depender da quantidade de octetos do PIN. O comprimento máximo do PIN usado no algoritmo é de 16 octetos. Existe, portanto, a possibilidade de que em algum caso o *BD_ADDR* não seja usado integralmente. Este algoritmo de geração da chave explora a função de cifragem E_2, que para o modo 1 (definida como E_{21}) é calculada de acordo com a expressão (2.24) e com a expressão (2.25):

$$\begin{aligned} E_{21}: \quad & \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{128} \\ & (RAND, address) \mapsto A'_r(X, Y), \end{aligned} \tag{2.24}$$

onde para o modo 1:

$$\begin{cases} X = \mathrm{RAND}[0...14] \cup (\mathrm{RAND}[15] \oplus 6) \\ Y = \overset{15}{\underset{i=0}{\mathrm{Y}}} \mathrm{address}[i(\mathrm{mod}\,6)]. \end{cases} \tag{2.25}$$

Seja L o número de octetos do PIN do usuário, a concatenação é definida por:

$$\mathrm{PIN}' = \begin{cases} \mathrm{PIN}[0...L-1] \cup \mathrm{BD\_ADDR}[0..\min\{5, 15-L\}]\}, & L < 16, \\ \mathrm{PIN} = [0...L-1], & L = 16. \end{cases} \tag{2.26}$$

Então, no modo 2, E_2 (definido por E_{22}) é:

$$\begin{aligned} E_{22}: \quad & \{0,1\}^{8L'} \times \{0,1\}^{128} \times \{1,\ 2,\ ...,\ 16\} \rightarrow \{0,1\}^{128} \\ & (\text{PIN}', \text{RAND}, \text{L}') \mid\rightarrow A'_r\,(X, Y), \end{aligned} \tag{2.27}$$

onde:

$$\begin{cases} X = \overset{15}{\underset{i=0}{\text{Y}}}\, \text{PIN}'[i(\text{mod}\, L')] \\ Y = \text{RAND}[0...14] \cup (\text{RAND}[15] \oplus L'), \end{cases} \tag{2.28}$$

e $L' = \min\{16, L+6\}$ é o número de octetos de PIN'.

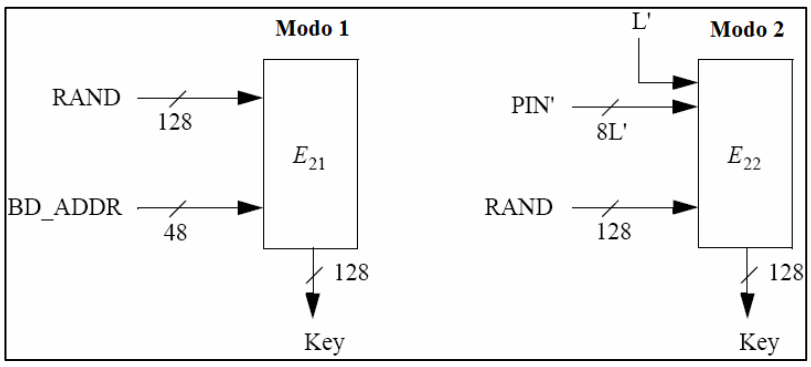


Figura 2.13 – Algoritmo E_2 de geração da chave e seus dois modos.

2.7.4 Função E_3 de geração da chave para cifragem

A chave de cifragem K_C usada por E_0 deve ser gerada por E_3. A função E_3 é construída usando-se A'_r, do seguinte modo:

$$\begin{aligned} E_3: \quad & \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{96} \rightarrow \{0,1\}^{128} \\ & (K, \text{RAND}, \text{COF}) \mid\rightarrow \textit{Hash}(K, \text{RAND}, \text{COF}, 12), \end{aligned} \tag{2.29}$$

onde *Hash* é a função definida pela expressão (2.19). O comprimento da chave gerada é de 128 bits, embora que, antes de usar E_0, a chave de cifragem K_C poderá ter seu comprimento

efetivo reduzido, como pode ser visto em 2.5.5. Um diagrama de blocos de E_3 pode ser visto na Figura 2.14.

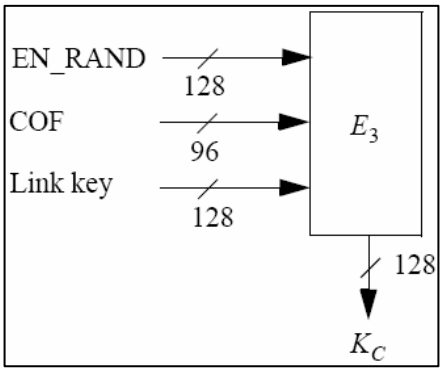


Figura 2.14 – Algoritmo E_3 de geração da chave.

3 REALIZAÇÃO DOS TESTES PRÁTICOS

Para podermos realizar os nossos testes práticos, optamos por desenvolver um *software* adequado às nossas necessidades, no qual pudéssemos trabalhar de modo prático e eficiente, acrescentando-se, caso possível, facilidade de acesso ao utilizador. Devido à experiência prévia na linguagem C [13], optamos pela linguagem visual C++ [14], na qual poderíamos ter um bom resultado visual final.

3.1 DESCRIÇÃO DAS FUNÇÕES CRIADAS NO *SOFTWARE*

A primeira dificuldade a ser superada foi a grande quantidade de cálculos a serem realizados com números binários de até 128 bits. Como não encontramos nenhuma rotina interna dentro da linguagem de programação escolhida, só nos restou criar as rotinas que fossem necessárias. Optamos por criar funções para cada uma das nossas necessidades iniciais, pois ao precisarmos destas rotinas mais adiante, bastaria acessá-las de modo adequado.

Criamos a nossa primeira função que foi denominada “converts”, com o objetivo de converter um número decimal em um número binário de até 64 bits [15]. Ao serem estudadas de modo mais detalhado, verificamos que as rotinas usadas na criptografia para a rede Bluetooth, na sua grande maioria trabalham com comprimento dado em bytes, ou como preferido pelos autores do projeto original: “octetos”. Isto fez com que a nossa rotina “converts” estivesse bem além da nossa necessidade inicial, uma vez que permite converter números binários de até 64 bits.

Em várias rotinas usadas na criptografia, precisávamos realizar cálculos usando-se números decimais, porém, a grande maioria dos dados encontrava-se expresso em binário, o que nos fez criar a nossa segunda função: “binary_to_decimal”. Esta função produz um número decimal a partir de um número binário de até 16 bits. Esta limitação aparente, não nos causou nenhum problema, pois só iríamos precisar converter os octetos, que possuem apenas oito bits.

Nossa próxima função foi definida como sendo “xor”, a qual recebe como parâmetros, dois números decimais e retorna um número decimal com o resultado da operação xor bit a bit entre os dois números. Como exemplo de cálculo usando-se esta função, teremos:

Exemplo 3.1:

$n1 = 231; n2 = 123;$
$n1b = 11100111;$
$n2b = 01111011;$
$resultado : n1b = 10011100;$
$number = 156;$

Os números $n1$ e $n2$ são os dois parâmetros de entrada, enquanto que $n1b$ e $n2b$ são estes números convertidos para binário, respectivamente. O resultado da operação XOR é atribuído a $n1b$. Como último passo, teremos uma conversão para decimal do número $n1b$, sendo este valor atribuído à variável “number”, que é uma variável global, podendo ser utilizada em qualquer parte do *software*.

Após este início, verificamos que seria obrigatório realizarmos a implementação do algoritmo SAFER+, pois este é usado em praticamente todos os algoritmos de criação das chaves de segurança. Posso dizer que este foi um dos passos mais difíceis de serem realizados, pois não se tratava da dificuldade apenas de implementá-lo, mais também e principalmente, de testá-lo, o que comprovou ser demorado e trabalhoso.

O algoritmo do SAFER+ pode ser visto na Figura 2.11, porém o seu detalhamento exigiu entender e implementar todo o algoritmo visto na Figura 2.12. O processo mostrado nesta figura tem como objetivo principal, calcular as chaves intermediárias K_i, que serão usadas

pelo SAFER+. Uma análise mais detalhada da Figura 2.12, nos mostrou a necessidade de encontrar os vetores de *Bias* B_i, que podem ser encontrados por intermédio da equação (2.23). Neste ponto, encontramos uma grande dificuldade, pois as definições disponíveis para inteiros não suportavam a realização simples dos cálculos e trabalhar com números reais iria nos levar à aproximações, que poderiam comprometer os resultados. Para ilustrar o nosso problema, o maior inteiro, por definição, será obtido usando-se o tipo de dados "int128" [16], mais conhecido como "unsigned long long". Usando esta definição, o maior inteiro disponível será "apenas": 18.446.744.073.709.551.615.

Exemplo 3.2:

$$para\ i = 0;\ p = 0;$$
$$B_0[0] = ((45^{(45^{18} \bmod 257)} \bmod 257) \bmod 256);$$
$$45^{18} = 572.565.594.852.444.156.646.728.515.625.$$

Como pode ser observado pelo cálculo feito no exemplo 3.2, o nosso primeiro expoente (18), que também é o menor deles, já dificulta bastante o nosso trabalho. O vetor de *Bias* B_0 não é usado na criação das chaves intermediárias do SAFER+, serve apenas como exemplo da dificuldade na realização dos cálculos.

A solução encontrada usa as operações modulares [17] mostradas no exemplo 3.3, que nos permite realizar os cálculos de modo a não precisarmos de nenhuma variável além da definição de inteiros "int" usada no visual C++, que permite números de valor máximo igual a 2.147.483.647. Vejamos um exemplo de cálculo da exponenciação:

Exemplo 3.3:

$$para\ i = 0;\ p = 0;$$
$$B_0[0] = ((45^{(45^{18} \bmod 257)} \bmod 257) \bmod 256);$$
$$45^{18} \bmod 257 = ((45^2 \bmod 257)(45^{16} \bmod 257)) \bmod 257;$$
$$45^{18} \bmod 257 = (((226 \cdot 45) \bmod 257)(45^{15} \bmod 257)) \bmod 257;$$
$$\text{M}$$
$$45^{18} \bmod 257 = (((8 \cdot 45) \bmod 257)(45 \bmod 257)) \bmod 257;$$
$$45^{18} \bmod 257 = 9.$$

Usando-se esta técnica, podemos continuar realizando nossos cálculos até encontrarmos todos os valores dos vetores de *Bias*. Observe que no exemplo 3.3, para encontrarmos o valor do *Bias*, teríamos que substituir o valor nove no início deste exemplo, como sendo o expoente do valor 45. Este cálculo é feito usando-se o mesmo princípio do exemplo 3.3. No nosso *software*, as funções que realizam estes cálculos são “Calculate_AuxBpi” e “Calculate_Bpi”.

Além de precisarmos calcular os vetores de *Bias* para obtermos as chaves intermediárias K_i, temos que manipular os octetos iniciais conforme orientação observada na Figura 2.12. Devemos gerar o octeto 16, que nada mais é do que a soma dos octetos de zero a quinze, bit a bit módulo dois. Após este passo, devemos rotacionar cada octeto três bits para a esquerda e em seguida, selecionar os octetos apropriados para obtenção das chaves intermediárias.

Podemos observar que as funções criadas no nosso *software* que realizam os procedimentos para obtenção das chaves intermediárias são:

- “Octet16”, que gera o octeto dezesseis;
- “rotate3bitsleft”, que rotaciona cada octeto três bits para a esquerda;
- “rotate1Octetleft”, que rotaciona ciclicamente os octetos de zero a dezesseis;
- “CopyOctetauxtoOctet”, que copia o conteúdo temporário dos octetos auxiliares para os octetos selecionados que irão gerar cada uma das chaves intermediárias K_i;
- “CalculateBpiBinary”, que converte os vetores *Bias* de decimal para binário;
- “Print_K”, que efetivamente gera as chaves intermediárias K_i;
- “kbin_to_kdec”, que converte a chave K_i de binário para decimal.

Neste ponto, já conseguimos criar todas as chaves intermediárias, vamos agora voltar a analisar o algoritmo do SAFER+ mostrado na Figura 2.11. Podemos verificar que existem vários cálculos representados por $\oplus$ que corresponde à operação XOR bit a bit em um octeto e $\boxed{+}$ que corresponde à operação de soma módulo 256 em um octeto. Ambas as operações são realizadas ao longo do *software*, não tendo sido criada nenhuma função específica para estes cálculos.

Temos duas caixas marcadas com $\boxed{e}$ e com $\boxed{l}$, as quais são responsáveis por realizar as operações matemáticas definidas por:

$$
\begin{aligned}
e,l \quad &: \quad \{0, ..., 255\} \rightarrow \{0, ..., 255\}, \\
e \quad &: \quad i \mapsto (45^{i} (\mathrm{mod}\, 257))(\mathrm{mod}\, 256), \\
l \quad &: \quad i \mapsto j \ \text{tal que} : \mathrm{i} = \mathrm{e(j)}.
\end{aligned}
$$

Após entender o processo de cálculo, podemos executá-lo do seguinte modo:

$$e = (45^{i} (\mathrm{mod}\, 257))(\mathrm{mod}\, 256), \tag{3.1}$$

$$\begin{cases} l = (\log_{45} x)(\mathrm{mod}\, 257)(\mathrm{mod}\, 256), \\ \text{onde} : \begin{cases} x \in \mathrm{N}^*, \\ j = x(\mathrm{mod}\, 257)(\mathrm{mod}\, 256), \end{cases} \end{cases} \tag{3.2}$$

onde *e* é a saída da caixa exponencial $\boxed{e}$ e *l* é a saída da caixa logarítmica $\boxed{l}$. Para realizar estes cálculos, preferimos incluí-los na função “Start_variables”, que realiza todos os cálculos possíveis de *e* e *l*, nos apresentando a Tabela 3.1 e a Tabela 3.2 respectivamente.

A Tabela 3.1 mostra o cálculo da caixa *e*, para todos os possíveis valores de *i*. Na primeira coluna podemos observar as dezenas e na primeira linha podemos observar as unidades referentes aos valores de *i*. Como exemplo, o valor da caixa *e* para *i* =125, pode ser encontrado na célula interceptada pela linha que se inicia com o número 12 e pela coluna que se inicia com o número 5, nos dando como resultado o valor: *e* (125) = 250.

Podemos realizar o mesmo procedimento para encontrar o valor de *l* para *j* = 13, basta verificar, na Tabela 3.2, o conteúdo da célula interceptada pela linha que se inicia com o número 1 e pela coluna que se inicia com o número 3, nos dando o seguinte resultado: *l* (13) = 186.

A Tabela 3.1 e a Tabela 3.2, após serem geradas, são armazenadas e quando necessário, serão acessadas, fornecendo imediatamente o resultado para a caixa *e*, ou para a caixa *l*, respectivamente.

Tabela 3.1 – $e = (45^i \pmod{257}) \pmod{256}$.

i	0	1	2	3	4	5	6	7	8	9
0	1	45	226	147	190	69	21	174	120	3
1	135	164	184	56	207	63	8	103	9	148
2	235	38	168	107	189	24	52	27	187	191
3	114	247	64	53	72	156	81	47	59	85
4	227	192	159	216	211	243	141	177	255	167
5	62	220	134	119	215	166	17	251	244	186
6	146	145	100	131	241	51	239	218	44	181
7	178	43	136	209	153	203	140	132	29	20
8	129	151	113	202	95	163	139	87	60	130
9	196	82	92	28	232	160	4	180	133	74
10	246	19	84	182	223	12	26	142	222	224
11	57	252	32	155	36	78	169	152	158	171
12	242	96	208	108	234	250	199	217	0	212
13	31	110	67	188	236	83	137	254	122	93
14	73	201	50	194	249	154	248	109	22	219
15	89	150	68	233	205	230	70	66	143	10
16	193	204	185	101	176	210	198	172	30	65
17	98	41	46	14	116	80	2	90	195	37
18	123	138	42	91	240	6	13	71	111	112
19	157	126	16	206	18	39	213	76	79	214
20	121	48	104	54	117	125	228	237	128	106
21	144	55	162	94	118	170	197	127	61	175
22	165	229	25	97	253	77	124	183	11	238
23	173	75	34	245	231	115	35	33	200	5
24	225	102	221	179	88	105	99	86	15	161
25	49	149	23	7	58	40	-	-	-	-

Tabela 3.2 – $\begin{cases} l = (\log_{45} x)(\mathrm{mod}\,257)(\mathrm{mod}\,256), \\ \text{onde}: \begin{cases} x \in \mathrm{N}^*, \\ j = x(\mathrm{mod}\,257)(\mathrm{mod}\,256). \end{cases} \end{cases}$

j	0	1	2	3	4	5	6	7	8	9
0	128	0	176	9	96	239	185	253	16	18
1	159	228	105	186	173	248	192	56	194	101
2	79	6	148	252	25	222	106	27	93	78
3	168	130	112	237	232	236	114	179	21	195
4	255	171	182	71	68	1	172	37	201	250
5	142	65	26	33	203	211	13	110	254	38
6	88	218	50	15	32	169	157	132	152	5
7	156	187	34	140	99	231	197	225	115	198
8	175	36	91	135	102	39	247	87	244	150
9	177	183	92	139	213	84	121	223	170	246
10	62	163	241	17	202	245	209	23	123	147
11	131	188	189	82	30	235	174	204	214	53
12	8	200	138	180	226	205	191	217	208	80
13	89	63	77	98	52	10	72	136	181	86
14	76	46	107	158	210	61	60	3	19	251
15	151	81	117	74	145	113	35	190	118	42
16	95	249	212	85	11	220	55	49	22	116
17	215	119	167	230	7	219	164	47	70	243
18	97	69	103	227	12	162	59	28	133	24
19	4	29	41	160	143	178	90	216	166	126
20	238	141	83	75	161	154	193	14	122	73
21	165	44	129	196	199	54	43	127	67	149
22	51	242	108	104	109	240	2	40	206	221
23	155	234	94	153	124	20	134	207	229	66
24	184	64	120	45	58	233	100	31	146	144
25	125	57	111	224	137	48	-	-	-	-

Criamos duas funções chamadas “Calculate_Fase_x_y” e “Calculate_Fase2_x_y”, que realizam os cálculos sequenciais desde as somas com as chaves intermediárias K_{2r-1}, os blocos *e* e *l*, as somas com as chaves intermediárias K_{2r} e o cálculo da primeira caixa PHT.

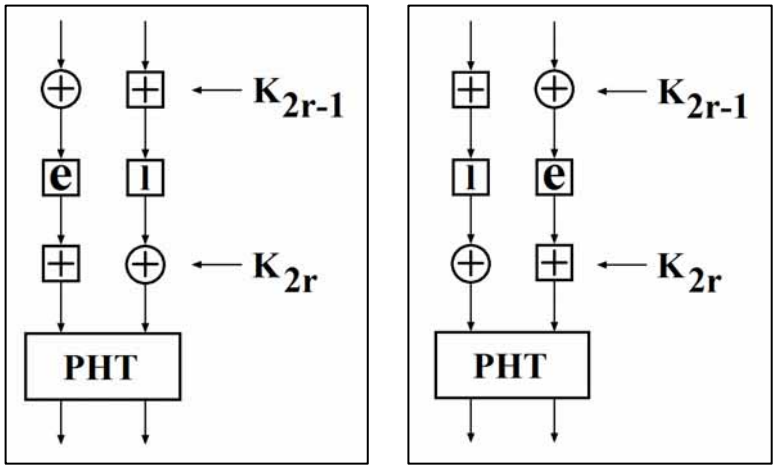


Figura 3.1 – “Calculate_Fase_x_y” e “Calculate_Fase2_x_y”, respectivamente.

Na Figura 3.1 podemos observar as operações matemáticas executadas pela função “Calculate_Fase_x_y”. De modo análogo, na Figura 3.2 temos as operações matemáticas executadas pela função “Calculate_Fase2_x_y”. Em cada etapa do cálculo do SAFER+, executamos quatro vezes cada uma destas funções. Podemos verificar que até a primeira caixa PHT de cada etapa, ambas as funções são independentes, não havendo necessidade de execução sequencial. Na verdade a execução deveria ser em paralelo, o que o *software* não poderia realizar. Posto isto, executamos primeiro a função “Calculate_Fase_x_y” por quatro vezes, depois executamos a função “Calculate_Fase2_x_y” também por quatro vezes. O que precisa mudar em cada execução é o octeto inicial em que cada função irá realizar os cálculos, o que não é problema, pois ambas as funções são parametrizadas por x e por y, onde x é o octeto inicial e y é o octeto final de cada procedimento de cálculo.

Após a realização destes cálculos, fazemos a nossa primeira permutação, para a qual criamos a função “permutation”, que será executada três vezes em cada etapa. Entre cada uma das rotinas de permutação, realizamos oito rotinas de PHT. A função criada para a realização da pseudo transformada de Hadamard foi nomeada como “PHT”. Esta função é parametrizada, precisando-se informar qual o octeto inicial usado na realização dos cálculos.

Ao término da última etapa precisamos realizar operações de XOR e de adição módulo 256, as quais são realizadas pela função “input”. Observando-se mais uma vez a Figura 2.11, podemos identificar que a diferença entre a rotina que realiza A_r e a rotina que realiza A'_r, consiste em adicionar a entrada inicial para A'_r durante a terceira etapa. Deste modo, a nossa

função “input” foi escrita para permitir as operações de XOR e de adição módulo 256 em três casos: na entrada dos dados iniciais de cada rotina A_r ou A'_r, na terceira etapa para A'_r e na última etapa tanto para A_r quanto A'_r. A função criada para realizar A_r e A'_r foi definida como “saferplus” e é a função mais usada no processo de criação das chaves de segurança. A diferença entre o procedimento A_r e o procedimento A'_r é contornada pela parametrização da função “saferplus”, assim como vários testes internos a esta função.

Para conseguirmos gerar todas as possíveis chaves de segurança da tecnologia Bluetooth, precisamos utilizar os sistemas: E_0, E_1, E_2 e E_3. O sistema E_0 pode ser visualizado na Figura 2.4, o E_1 na Figura 2.10, o E_2 na Figura 2.13 e o E_3 na Figura 2.14. Nosso objetivo, a partir de agora é detalhar cada um destes sistemas, explicando como o nosso *software* consegue executar cada um destes sistemas.

Resumo das rotinas de criação das chaves de segurança:

$$
\begin{aligned}
&K_A = E_{21}(RAND_A, BD\_ADDR_A),\\
&K_B = E_{21}(RAND_B, BD\_ADDR_B),\\
&K_{AB} = K_A \oplus K_B,\\
&K_{init} = E_{22}(PIN', RAND, L'),\\
&K_{master} = E_{22}(RAND_1, RAND_2, 16),\\
&K_C = E_3(K_{master}, EN\_RAND, COF).
\end{aligned}
$$

Parâmetros auxiliares à criação das chaves de segurança:

$E_1(K, RAND, address, L) = A'_r\ ([\tilde{K}], [E(address, L)] +_{16}\ (A_r(K, RAND) \oplus_{16} RAND)])$,
onde $E(address, L) = \cup_{i=0}^{15} BD\_ADDR[i(\bmod L)]$,

e $K\ (\mathrm{K}_{\mathrm{offset}})$ é definido por:

$K[0] = (K[0] + 233) \bmod 256,\ \ K[1] = (K[1] \oplus 229),$

$K[2] = (K[2] + 223) \bmod 256,\ \ K[3] = (K[3] \oplus 193),$

$K[4] = (K[4] + 179) \bmod 256,\ \ K[5] = (K[5] \oplus 167),$

$K[6] = (K[6] + 149) \bmod 256,\ \ K[7] = (K[7] \oplus 131),$

$K[8] = (K[8] \oplus 233),\qquad K[9] = (K[9] + 229) \bmod 256,$

$K[10] = (K[10] \oplus 223),\qquad K[11] = (K[11] + 193) \bmod 256,$

$K[12] = (K[12] \oplus 179),\qquad K[13] = (K[13] + 167) \bmod 256,$

$K[14] = (K[14] \oplus 149),\qquad K[15] = (K[15] + 131) \bmod 256.$

Detalhes do sistema E_{21}:

$$E_{21} = A'_r(X, Y),$$
$$\text{onde} \begin{cases} X = \text{RAND}[0...14] \cup (\text{RAND}[15] \oplus 6), \\ Y = \overset{15}{\underset{i=0}{\text{Y}}} \text{address}[i(\text{mod}\, 6)]. \end{cases}$$

Detalhes do sistema E_{22}:

$$E_{22} = A'_r(X, Y),$$
$$\text{onde} \begin{cases} X = \overset{15}{\underset{i=0}{\text{Y}}} \text{PIN}'[i(\text{mod}\, L')], \\ Y = \text{RAND}[0...14] \cup (\text{RAND}[15] \oplus L'), \end{cases}$$
$$\text{e} \begin{cases} \text{PIN}' = \begin{cases} \text{PIN}[0...L-1] \cup \text{BD\_ADDR}[0..\min\{5, 15-L\}]\}, & L < 16, \\ \text{PIN} = [0...L-1], & L = 16, \end{cases} \\ \text{L}' = \min\{16, \text{L} + 6\}. \end{cases}$$

Detalhes do sistema E_3:

$$E_3 = Hash(K, \text{RAND}, \text{COF}, 12),$$
$$\text{onde } Hash = E_1; \text{substituindo} - \text{se}: \begin{cases} address \text{ por } COF, \\ \text{e } 6 \text{ por } 12 \text{ (número de octetos)}, \end{cases}$$
$$\text{onde } COF = \begin{cases} BD\_ADDR \cup BD\_ADDR, \text{se } K = K_{master}, \\ ACO, \text{ nos outros casos}, \end{cases}$$
$$\text{e } ACO = Hash(K, RAND, address, 6)[4, ..., 15].$$

Começando pelo mais simples, que é o sistema E_2, podemos observar que temos duas variações, denominadas E_{21} e E_{22}. O cálculo de E_{21} é definido pela expressão (2.24) e pela expressão (2.25). Resumidamente temos que executar a função "saferplus" usando-se a rotina A'_r, com parâmetros de entrada X e Y, conforme expressão (2.25). O modo E_{21} é usado para gerar as chaves das unidades (K_A ou K_B), e a chave combinada entre unidades (K_{AB}), enquanto que o modo E_{22} é usado para gerar a chave inicial (K_{init}) e a chave mestre (K_{master}). O modo E_{22} é muito semelhante ao modo E_{21}, a mudança encontra-se apenas na definição dos parâmetros de entrada X e Y, como pode ser verificado na expressão (2.25) e na expressão (2.28).

Para o nosso *software*, preferimos definir duas funções distintas: "E_{21}" e "E_{22}", facilitando a escolha entre o modo 1 ou modo 2, respectivamente. Para o modo 1, podemos

escolher de forma parametrizada entre a variável booleana “RA” ou “RB” que definem, respectivamente, o cálculo da chave da unidade A (K_A) ou o cálculo da chave da unidade B (K_B). Para encontrarmos a chave combinada das unidades A e B (K_{AB}), podemos utilizar a expressão (2.6), na qual a chave da unidade A transmitida, é definida como LK_K_A e para a unidade B é definida como LK_K_B. Como já estamos com estas duas chaves calculadas, basta uma operação utilizando-se nossa função “xor” e teremos calculado a chave K_{AB}.

A função “E_{22}” pode calcular a chave inicial ou a chave mestra, para definirmos qual das duas é a chave a ser gerada, antes de chamar a função “E_{22}” devemos definir, no nosso *software*, como verdadeira, a variável booleana “Kinit_bool” ou “Kmaster_bool”, que determinam, respectivamente, a escolha de qual chave queremos gerar.

Para o sistema E_1, devemos realizar uma vez a rotina A_r e uma vez a rotina A'_r. Podemos entender melhor a execução da função “E_1”, observando a Figura 2.11. Para iniciarmos a rotina, precisamos de quatro entradas: *K*, *RAND*, *address* e *L*. Nossa chave *K* usada nesta rotina é a chave K_{AB}, o número aleatório *RAND* é gerado pelo *software*, o endereço é o MAC address do dispositivo mestre e *L* é o número de octetos que possui o endereço do mestre. O *RAND* e a chave *K* são informados para a função “saferplus”, que é executada, produzindo a saída definida pela variável “SAIDA_AR”. A seguir é realizada uma operação de XOR entre a saída da rotina A_r e o número *RAND*. Iremos precisar expandir o MAC address do dispositivo A, que possui seis octetos, para 16 octetos, o que pode ser realizado usando-se a função “E_box”. De posse deste valor, podemos adicioná-lo módulo 256 ao resultado da operação XOR já realizada, produzindo a variável “ENTRADA_ARL”. Neste momento, precisaremos executar a rotina A'_r, a qual precisa de uma chave denominada K (K offset). Para encontrarmos o valor de *K* offset, usamos a função “Calculate_K_offset”, que realiza os cálculos descritos pela equação (2.21). Executamos novamente a função “saferplus”, desta vez para realizar a rotina A'_r. A saída desta rotina já é o nosso resultado, que compreende o valor do *SRES*, correspondendo aos octetos de zero a três, e o valor do *ACO*, correspondendo aos octetos de quatro a dezesseis.

O sistema E_3 é executado pela função “E_3”, que é muito semelhante à função “E_1”. Como diferença, temos apenas a troca do valor do MAC address com 48 bits, pelo *ACO* com 96 bits, o uso da chave K_{master} como chave *K* e o uso de outro número aleatório *EN_RAND* em

substituição ao *RAND*. Como resultado de saída do sistema E_3, temos a geração da chave K_C, este sistema pode ser visto na Figura 3.2. Para executar o procedimento E_3, é necessário que haja uma chave K_{master} já gerada, por este motivo, não existe no nosso *software* uma execução isolada deste procedimento. A função "E_3" é utilizada por outra função, chamada de "E_0", que é a responsável pela geração da sequência de cifragem Z_t.

O sistema E_0 é o mais complexo de todo o processo de criptografia para a tecnologia Bluetooth e é ele que de fato realiza a cifragem/decifragem dos dados. No nosso *software* a função "E_0" executa o sistema E_0. A complexidade na implementação desta rotina encontra-se principalmente nas execuções das funções "find_KCl" e "LFSRs".

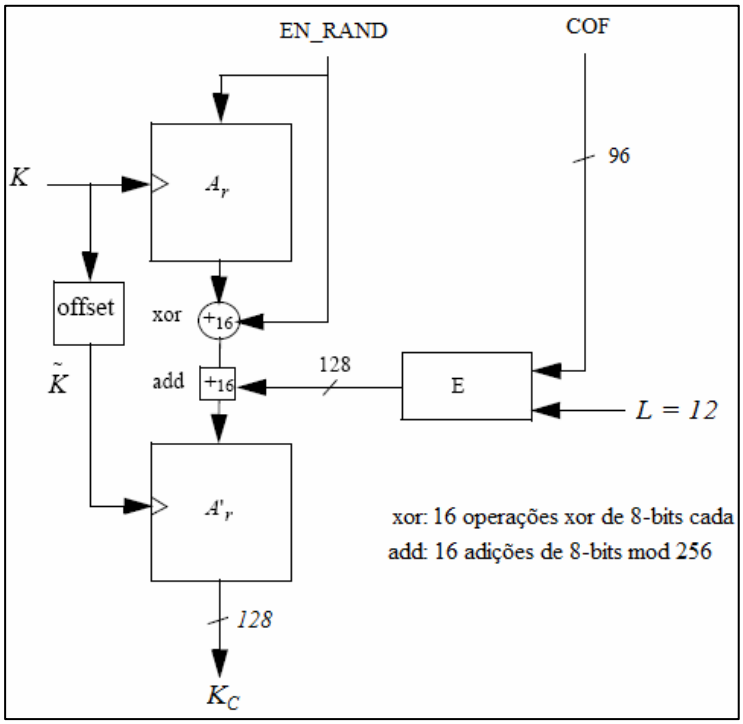


Figura 3.2 – Fluxo dos dados para o cálculo de E_3

A função "find_KCl" tem o objetivo de encontrar a chave K'_C. Para encontrarmos esta chave, devemos executar os cálculos da equação (2.16), que consiste em reduzir o comprimento efetivo da chave de cifragem K_C. Na equação (2.16), devemos realizar inicialmente a seguinte operação: $K_C(x) \bmod g_1^{(L)}(x)$, onde $K_C(x)$ é a chave de cifragem K_C, que deve ser interpretada como sendo um polinômio de grau máximo igual a 127, $g_1^{(L)}(x)$ é um polinômio definido pela Tabela 2.4 e L é o comprimento efetivo da chave, dado em

octetos. Após o cálculo desta operação matemática modular, precisaremos multiplicar o resultado pelo polinômio $g_2^{(L)}(x)$, também definido pela Tabela 2.4, e só então obtermos a nossa chave K'_C.

Inicialmente, na nossa função "find_KCl", definimos os polinômios $g_1^{(L)}(x)$ e $g_2^{(L)}(x)$, que serão usados por duas outras funções: "Divides_polynomials" e "Multiply_polynomials" respectivamente. A função "Divides_polynomials" realiza o cálculo de divisão entre dois polinômios através de subtrações sucessivas [18], obtendo-se o resto desta divisão e atribuindo o seu valor na variável "POL3". O resto da divisão obtido é o valor que deve ser multiplicado pelo polinômio $g_2^{(L)}(x)$. A função "Multiply_polynomials" multiplica dois polinômios usando-se somas sucessivas [19], armazenando o resultado na variável "Pmul_res". O valor armazenado nesta variável é a chave K'_C que queríamos calcular, bastando realizar um deslocamento no momento de copiar a variável "Pmul_res" para K'_C, pois o comprimento destas duas variáveis é diferente.

A função "LFSRs" executa os procedimentos de iniciação dos quatro registradores de deslocamento com realimentação linear, o bloco da lógica de combinação aditiva e o bloco de mistura, que podem ser observados na Figura 2.6. Inicialmente, a nossa função "LFSRs" ordena as entradas de cada um dos registradores, de acordo com a organização sequencial observada na Figura 2.7. Em seguida os dados são deslocados para cada um dos registradores com as chaves abertas, ou seja, sem realimentação. Após cada um dos registradores estar totalmente preenchido, a sua chave é fechada e os dados de entrada continuam sendo deslocados. Quando é feito o deslocamento de número 39, todas as chaves estão fechadas e começamos a gerar os símbolos de saída z_t.

No total são gerados 200 símbolos, dos quais os últimos 128 formam a sequência de saída Z_t. Podemos observar na Figura 2.6 as seguintes caixas de cálculo: [+], [/2], [z^{-1}], [T_1], [T_2] e [XOR]. As duas caixas [+] realizam a soma dos bits de entrada; [/2] retorna a divisão inteira por 2, desprezando o resto; [z^{-1}] realiza um retardo; [T_1] e [T_2] realizam o mapeamento definido na Tabela 2.3 e XOR realiza a operação ou exclusivo entre as suas três entradas. Para conseguirmos realizar a operação de mistura, observada na Figura 2.6, é necessário que sejam

previamente definidos os valores de c_t e de c_{t-1}, sem os quais não podemos realizar os cálculos. Vamos mostrar os cálculos realizados pelo misturador no exemplo 3.4.

Exemplo 3.4:
São dados:
$t = 39;$
$c_{39} = c_{38} = 00;$
$x_{39}^1 = 0; x_{39}^2 = 1; x_{39}^3 = 1; x_{39}^4 = 1.$

A variável c_t é composta por dois dígitos binários, e a variável x_t^i é composta por um dígito binário. Vamos calcular o valor da variável y_t, usando a equação (2.12):

$$y_{39} = x_{39}^1 + x_{39}^2 + x_{39}^3 + x_{39}^4 = 0 + 1 + 1 + 1 = 3_d = 011_b.$$

A variável y_t é composta por três dígitos binários. Os índices "d" e "b", representam, respectivamente as representações decimal e binária do valor de y_t. Vamos calcular s_{t+1}, usando a equação (2.14):

$$s_{40} = \left\lfloor \frac{3+0}{2} \right\rfloor = 1_d = 01_b.$$

A variável s_{t+1} é composta por dois dígitos binários. Vamos calcular c_{t+1}, usando a equação (2.15):

$$c_{40} = 01 \oplus T_1[00] \oplus T_2[00] = 01 \oplus 00 \oplus 00 = 01.$$

A variável c_{t+1} é composta por dois dígitos binários. Resta-nos calcular z_t, que pode ser obtido pela equação (2.13), definida na página 27:

$$z_{39} = 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 1.$$

Para o cálculo de z_t, usamos além do conteúdo dos quatro registradores de deslocamento o bit menos significativo de c_t. Podemos verificar que a partir do exemplo 3.4, dispomos de c_{39} e de c_{40}, portanto podemos continuar realizando os cálculos para todas as próximas saídas dos registradores de deslocamento sem maiores dificuldades.

No nosso *software*, temos uma variável definida por “zt”, que é responsável por guardar toda a sequência de cifragem Z_t, composta por 200 dígitos binários, porém, para realizar a cifragem do texto claro, são usados apenas os últimos 128 dígitos binários gerados, que formam a chave K_C, armazenados na variável “zt_dec”. A cifragem/decifragem dos dados de fato é realizada usando-se a operação XOR entre o texto claro/cifrado e a variável “zt_dec”.

3.2 ORIENTAÇÕES PARA USO DO *SOFTWARE*

Podemos observar, na Figura 3.3, a tela de acesso às funcionalidades do *software*, que possui botões, caixas de texto, dados obrigatórios, entre outras opções. Vamos descrever a funcionalidade de cada um dos itens presentes na Figura 3.3, de modo que o usuário possa ter acesso aos dados de entrada, dados de saída, até chegarmos a cifragem do texto claro e a decifragem do texto cifrado.

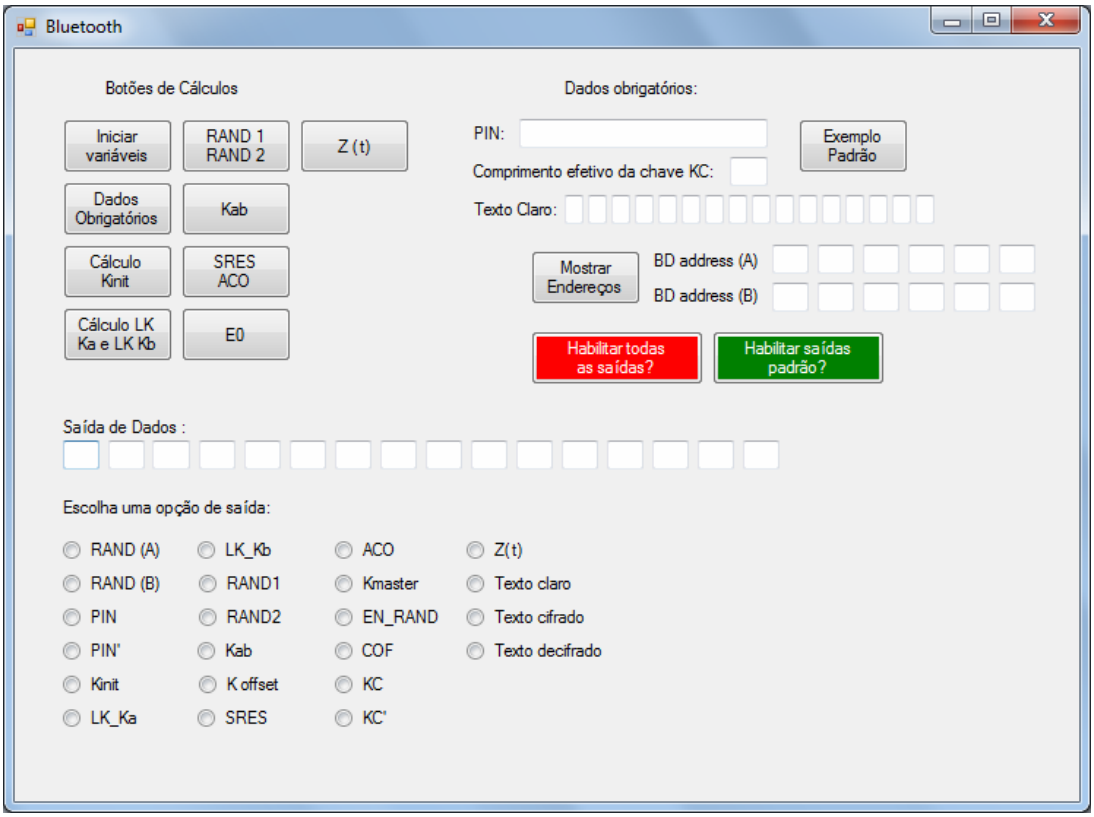


Figura 3.3 – Tela de acesso às funcionalidades do *software*

Preferimos deixar a tela do *prompt* de comando como a saída dos dados intermediários, pois não teríamos espaço para colocá-los na tela principal do *software*. A tela do *prompt* de comando pode ser observada na Figura 3.4.

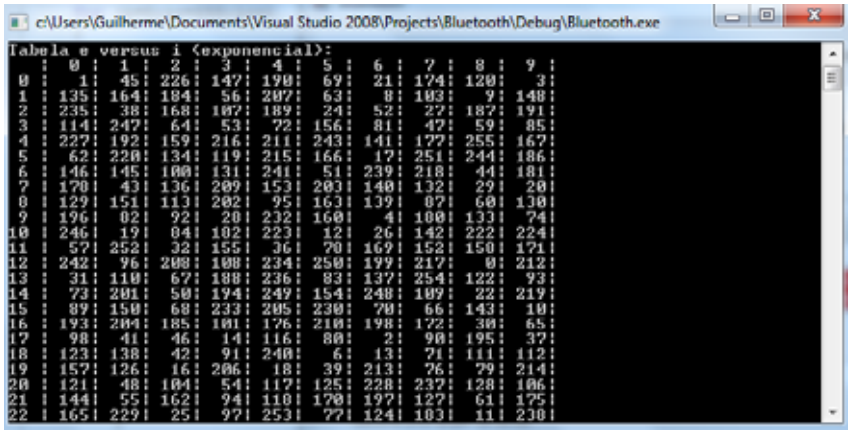


Figura 3.4 – Tela do *prompt* de comando.

Para facilitar a compreensão da execução do *software*, vamos definir a tela de acesso às funcionalidades do sistema como sendo o modo gráfico e a tela do *prompt* de comando como sendo o modo texto.

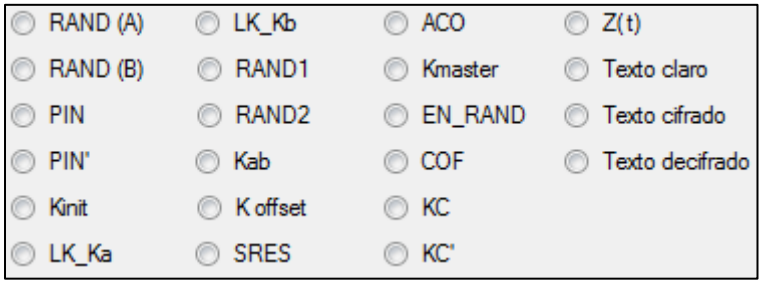


Figura 3.5 – *radio buttons*

Os *radio buttons*, mostrados na Figura 3.5, podem ser escolhidos para mostrar no modo gráfico os dados que estão armazenados no *software*, e só faz sentido escolhê-los após os dados terem sido guardados na memória do *software*. Os dados escolhidos serão mostrados na "Saída de Dados", mostrada na Figura 3.6. Ao longo das explicações das funcionalidades dos botões, informaremos quando os *radio buttons* poderão ter utilidade.

Figura 3.6 – Saída de Dados do modo gráfico

3.3 BOTÕES DE CÁLCULO

O *software* possui nove botões de cálculos, que estão dispostos do lado superior esquerdo da tela de acesso do *software*. A execução dos botões não pode ser aleatória, ou seja, precisamos pressionar os botões de modo sequencial, pois uma rotina de cálculo pode ser dependente de outra. A sequência sugerida é: “Exemplo Padrão”, “Iniciar variáveis”, “Dados Obrigatórios”, “Cálculo Kinit”, “Cálculo LK Ka e LK Kb”, “RAND 1 RAND 2”, “Kab”, “SRES ACO”, “E0” e “Z(t)”.

O botão “Exemplo Padrão” é opcional, porém os “Dados obrigatórios” são realmente obrigatórios, pois são necessários para a execução das rotinas de cálculos. A função do botão “Exemplo Padrão” é preencher os campos: “PIN”, “Comprimento efetivo da chave KC” e “Texto Claro”. Caso o usuário queira, poderá digitar manualmente estes três campos, ou alterá-los após pressionar o botão “Exemplo padrão”.

3.3.1 Botão “Iniciar variáveis”: 

O botão “Iniciar variáveis” tem a função de iniciar os seguintes dados:

- Tabela 3.1;
- Tabela 3.2;
- $RAND_A$;
- $RAND_B$;
- *BD Address* do dispositivo A;
- *BD Address* do dispositivo B;
- *RAND*;

O exemplo 3.5 nos mostra os dados gerados por uma execução do botão “Iniciar variáveis”. Os dados foram copiados do modo texto, devido à dificuldade de impressão referente à cor de fundo, que é preta. Este procedimento de cópia dos dados será realizado em todas as saídas que forem apresentadas no modo texto.

Exemplo 3.5 – Execução do botão “Iniciar variáveis”:

Tabela e versus i (exponencial):

	0	1	2	3	4	5	6	7	8	9
0	1	45	226	147	190	69	21	174	120	3
1	135	164	184	56	207	63	8	103	9	148
2	235	38	168	107	189	24	52	27	187	191
3	114	247	64	53	72	156	81	47	59	85
4	227	192	159	216	211	243	141	177	255	167
5	62	220	134	119	215	166	17	251	244	186
6	146	145	100	131	241	51	239	218	44	181
7	178	43	136	209	153	203	140	132	29	20
8	129	151	113	202	95	163	139	87	60	130
9	196	82	92	28	232	160	4	180	133	74
10	246	19	84	182	223	12	26	142	222	224
11	57	252	32	155	36	78	169	152	158	171
12	242	96	208	108	234	250	199	217	0	212
13	31	110	67	188	236	83	137	254	122	93
14	73	201	50	194	249	154	248	109	22	219
15	89	150	68	233	205	230	70	66	143	10
16	193	204	185	101	176	210	198	172	30	65
17	98	41	46	14	116	80	2	90	195	37
18	123	138	42	91	240	6	13	71	111	112
19	157	126	16	206	18	39	213	76	79	214
20	121	48	104	54	117	125	228	237	128	106
21	144	55	162	94	118	170	197	127	61	175
22	165	229	25	97	253	77	124	183	11	238
23	173	75	34	245	231	115	35	33	200	5
24	225	102	221	179	88	105	99	86	15	161
25	49	149	23	7	58	40				

Tabela j versus l (logaritmo):

	0	1	2	3	4	5	6	7	8	9
0	128	0	176	9	96	239	185	253	16	18
1	159	228	105	186	173	248	192	56	194	101
2	79	6	148	252	25	222	106	27	93	78
3	168	130	112	237	232	236	114	179	21	195
4	255	171	182	71	68	1	172	37	201	250
5	142	65	26	33	203	211	13	110	254	38
6	88	218	50	15	32	169	157	132	152	5
7	156	187	34	140	99	231	197	225	115	198
8	175	36	91	135	102	39	247	87	244	150
9	177	183	92	139	213	84	121	223	170	246
10	62	163	241	17	202	245	209	23	123	147
11	131	188	189	82	30	235	174	204	214	53
12	8	200	138	180	226	205	191	217	208	80
13	89	63	77	98	52	10	72	136	181	86
14	76	46	107	158	210	61	60	3	19	251
15	151	81	117	74	145	113	35	190	118	42
16	95	249	212	85	11	220	55	49	22	116
17	215	119	167	230	7	219	164	47	70	243
18	97	69	103	227	12	162	59	28	133	24
19	4	29	41	160	143	178	90	216	166	126
20	238	141	83	75	161	154	193	14	122	73
21	165	44	129	196	199	54	43	127	67	149
22	51	242	108	104	109	240	2	40	206	221
23	155	234	94	153	124	20	134	207	229	66
24	184	64	120	45	58	233	100	31	146	144
25	125	57	111	224	137	48				

RAND(a)=
89, 55, 28, 33, 100, 88, 51, 174, 131, 225, 34, 131, 102, 84, 180, 73,

RAND(b)=
224, 201, 244, 241, 234, 223, 78, 74, 235, 248, 242, 211, 55, 166, 121, 206,

BD Address(A)=255, 31, 107, 46, 33, 101,

BD Address(B)=255, 224, 125, 244, 48, 8,

RAND=
213, 96, 203, 149, 133, 122, 64, 49, 1, 84, 5, 41, 132, 209, 194, 152,

Após a execução do botão “Iniciar variáveis”, os *radio buttons* RAND (a) e RAND (b) já podem ser escolhidos, mostrando, no modo gráfico a mesma saída do modo texto para estas variáveis. Também faz sentido executar o botão “Mostrar Endereços”, pois ambos já foram iniciados no sistema. A saída do botão “Mostrar Endereços” é colocada no modo gráfico.

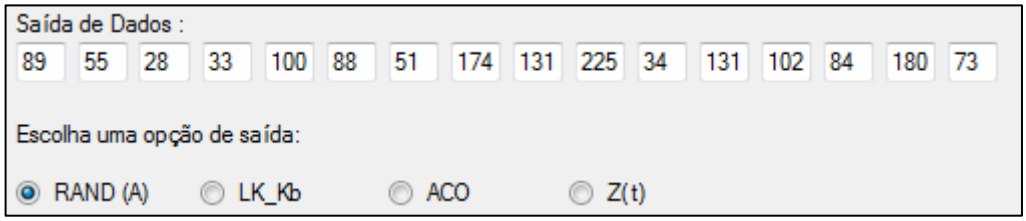


Figura 3.7 – Exemplo da saída de dados para o $RAND_A$

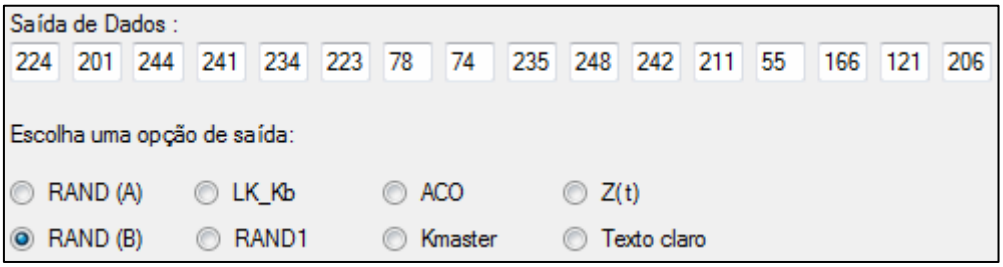


Figura 3.8 – Exemplo da saída de dados para o $RAND_B$

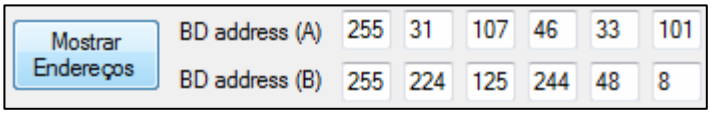
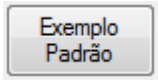


Figura 3.9 – Exemplo dos endereços dos dois dispositivos

3.3.2 Botão “Exemplo Padrão”:



O botão “Exemplo Padrão” tem a função de preencher os campos: “PIN”, “Comprimento efetivo da chave KC” e “Texto Claro”. O exemplo 3.6 mostra a execução do botão “Exemplo Padrão”, que sempre preenche os três campos com os mesmos dados.

Exemplo 3.6:

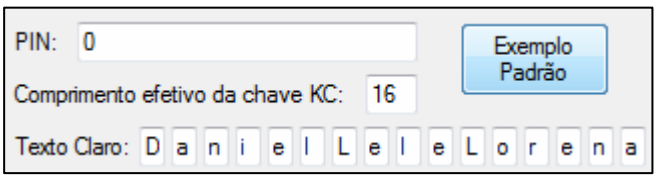


Figura 3.10 – Saída do Exemplo Padrão no modo gráfico

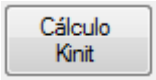
Os dados da execução do botão “Exemplo Padrão” podem ser observados no modo gráfico. O *radio button* “Texto claro” já pode ser usado.

3.3.3 Botão “Dados Obrigatórios”: 

A execução do botão “Dados Obrigatórios” exige que os campos “PIN”, “Comprimento efetivo da chave KC” e “Texto Claro” estejam preenchidos. A saída dos dados é realizada no modo texto, conforme pode ser observado no exemplo 3.7.

Exemplo 3.7:
bits=1, L=1
PIN=0,
PIN'=0, 255, 31, 107, 46, 33, 101,

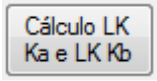
As saídas da execução do botão “Dados Obrigatórios” são o comprimento do PIN em bits e em octetos (L), o valor do PIN convertido para octetos e o valor do PIN' também em octetos. Os *radio buttons* “PIN” e “PIN'” já podem ser usados.

3.3.4 Botão “Cálculo Kinit”: 

A execução do botão “Cálculo Kinit” gera a chave K_{init} e produz uma saída no modo texto e uma saída no modo gráfico, como mostrado no exemplo 3.8.

Exemplo 3.8:
Kinit:
131, 6, 3, 72, 31, 182, 83, 10, 160, 106, 73, 184, 206, 136, 157, 56,

O *radio button* “Kinit” já pode ser utilizado. Devido à praticidade, todas as saídas disponibilizadas no modo gráfico estão em octetos.

3.3.5 Botão “Cálculo LK Ka e LK Kb”: 

Ao executarmos o botão “Cálculo LK Ka e LK Kb”, estamos gerando a chave da unidade A e a chave da unidade B. Podemos verificar a saída da execução deste botão observando o exemplo 3.9.

Exemplo 3.9:

LK_Ka:
Input=
89, 55, 28, 33, 100, 88, 51, 174, 131, 225, 34, 131, 102, 84, 180, 73,

Ka=
94, 64, 10, 32, 206, 141, 24, 56, 5, 193, 145, 254, 248, 161, 108, 215,

LK_Kb:
Input=
224, 201, 244, 241, 234, 223, 78, 74, 235, 248, 242, 211, 55, 166, 121, 206,

Kb=
192, 167, 248, 10, 104, 158, 172, 84, 233, 224, 98, 149, 169, 211, 162, 244,

A saída do modo texto nos mostra a chave da unidade A (LK Ka), representada pela saída Ka e a chave da unidade B (LK Kb), representada pela saída Kb. Ambos os valores já podem ser acessados pelos *radio buttons* “LK Ka” e “LK Kb” no modo gráfico.

3.3.6 Botão “RAND 1 RAND 2”: 

A cada execução do botão “RAND 1 RAND 2” estamos gerando dois números pseudo aleatórios RAND1 e RAND2; vamos observar as saídas geradas por meio do exemplo 3.10.

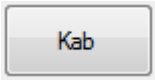
Exemplo 3.10:

RAND1=
210, 190, 176, 232, 135, 128, 4, 93, 18, 105, 245, 203, 72, 130, 182, 165,

RAND2=

165, 234, 57, 36, 76, 204, 83, 222, 74, 142, 27, 154, 27, 75, 147, 99,

No modo texto, podemos observar os dois números RAND1 e RAND2 gerados, no modo gráfico, os *radio buttons* “RAND1” e “RAND2” já podem ser utilizados.

3.3.7 Botão “Kab”: 

Quando pressionado, o botão “Kab” gera a chave de combinação K_{AB}, vamos observar o exemplo 3.11.

Exemplo 3.11:
Kab=
158, 231, 242, 42, 166, 19, 180, 108, 236, 33, 243, 107, 81, 114, 206, 35,

Além de podermos acessar a chave K_{AB} gerada através do modo texto, podemos usar o *radio button* “KAB” para confirmarmos a geração no modo gráfico.

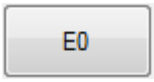
3.3.8 Botão “SRES ACO”: 

O botão “SRES ACO” tem o objetivo de gerar o *SRES* e o *ACO*, como pode ser observado no exemplo 3.12.

Exemplo 3.12:
SRES=
165, 17, 232, 14,

ACO=
75, 193, 216, 39, 167, 167, 139, 8, 68, 249, 129, 10,

Os *radio buttons* “K offset”, “SRES” e “ACO” já podem ser escolhidos no modo gráfico. A saída K_{offset} corresponde à chave interna usada no algoritmo E_0.

3.3.9 Botão “E0”: 

Ao acionarmos o botão “E0” estamos executando a rotina “E0”, na qual são gerados a chave K_{master}, o número *EN_RAND*, o *COF*, a chave K_C, a chave K'_C e os 200 bits da sequência de saída z_t. Vamos observar o exemplo 3.13.

Exemplo 3.13:

Input=
84, 78, 50, 127, 91, 116, 80, 183, 61, 104, 222, 7, 158, 146, 187, 165,

Kmaster:
96, 97, 104, 100, 56, 107, 218, 184, 211, 110, 31, 122, 77, 160, 145, 59,

EN_RAND=
242, 101, 21, 94, 40, 200, 35, 204, 82, 45, 137, 30, 22, 244, 53, 83,

COF=
255, 31, 107, 46, 33, 101, 255, 31, 107, 46, 33, 101,

Input=
242, 101, 21, 94, 40, 200, 35, 204, 82, 45, 137, 30, 22, 244, 53, 83,

K_offset=
73, 132, 71, 165, 235, 204, 111, 59, 58, 83, 192, 59, 254, 71, 4, 190,

Input=
183, 234, 91, 71, 5, 225, 254, 135, 201, 140, 215, 7, 20, 34, 123, 82,

KC=
20, 149, 39, 13, 34, 44, 85, 21, 197, 7, 37, 36, 51, 29, 121, 183,

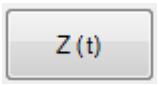
KCl_dec=
20,149,39,13,34,44,85,21,197,7,37,36,51,29,121,183,

200 saídas geradas:
01101001111111000100011101011100010101000110101111
10000100100100000010111011111000001001101011010010
11100111100110111000000101111010101001001000100011
01001110000000110001001000101110001010100010010110

Podemos observar nas saídas que temos vários Inputs, cada um deles representa a entrada de dados para a rotina A_r ou A'_r, o K_offset é a chave K modificada para a rotina A'_r,

KCl_dec é a saíde K'_C e temos as 200 saídas correspondentes à sequência de cifragem. Os *radio buttons* “Kmaster”, “EN_RAND”, “COF”, “KC”, “KC'” já podem ser usados.

3.3.10 Botão “Z(t)”:



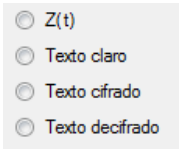
O botão “Z(t)” tem a função de selecionar apenas os últimos 128 bits da sequência de cifragem e convertê-la para decimal para poder mostrá-la no modo gráfico. Observe o exemplo 3.14.

Exemplo 3.14:

128 últimas saídas geradas:
10111110000010011010110100101110011110011011100000010111101010100100010001101001110000000110001001000101110001010100010010110

Z(t) em decimal:
190, 9, 173, 46, 121, 184, 23, 170, 72, 141, 56, 12, 72, 184, 168, 150,

3.3.11 *Radio buttons* Z(t), Texto claro, Texto cifrado e Texto decifrado:



Os *radio buttons* “Z(t)”, “Texto claro”, “Texto cifrado” e “Texto decifrado” já podem ser acionados, concluindo o processo de cifragem/decifragem dos dados.

Exemplo 3.15:

Texto claro do exemplo padrão:

Figura 3.11 – Texto claro.

Ao escolhermos o *radio button* texto claro, teremos a saída de dados preenchida pelo texto claro, como pode ser visto na Figura 3.12.

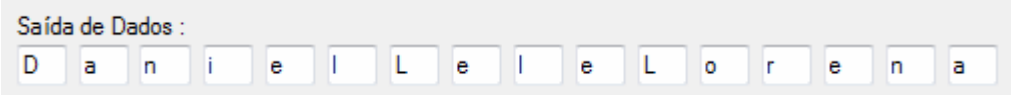


Figura 3.12 – Texto claro na saída de dados.

Na Figura 3.13 podemos ver um exemplo de texto cifrado.

Figura 3.13 – Texto cifrado na saída de dados.

Na Figura 3.14 temos o texto decifrado.

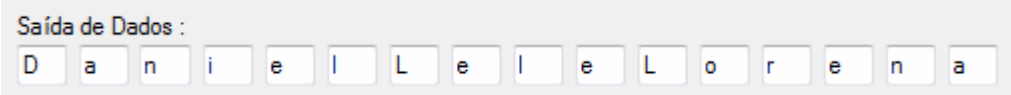
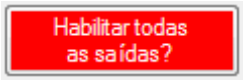


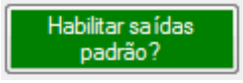
Figura 3.14 – Texto decifrado na saída de dados.

3.3.12 Botão “Habilitar todas as saídas?”:



Caso queiramos habilitar as saídas intermediárias dos processos de cifragem utilizados na tecnologia Bluetooth, devemos acionar este botão e observar as saídas no modo texto. Automaticamente este botão irá mudar para a cor verde, sinalizando a ativação do botão. O botão “Habilitar saídas padrão?” também trocará de cor, para vermelho, indicando que a saída padrão foi desabilitada.

3.3.13 Botão “Habilitar saídas padrão?”:



Este botão habilita as saídas padrão, caso tenha sido desabilitada pelo usuário. Por definição do *software*, a saída padrão já é iniciada habilitada. Em caso de ser desabilitada, a cor do botão irá trocar para vermelho. O resultado desta modificação nas saídas deve ser observado no modo texto.

4 CONCLUSÕES E PERSPECTIVAS DE PESQUISAS FUTURAS PARA A TECNOLOGIA BLUETOOTH

Iniciamos os estudos com a intenção de aprofundar nossos conhecimentos na área de segurança para a tecnologia Bluetooth. Logo após as pesquisas iniciais, resolvemos desenvolver um *software* que nos permitisse testar cada um dos procedimentos internos de segurança desta tecnologia.

Esta idéia acabou por se revelar de grande valia, principalmente do ponto de vista didático, pois possibilitou que muitas das teorias vistas em sala de aula pudessem ter seu funcionamento visualizado na prática, empregando uma tecnologia real, com grandes possibilidades de expandir os seus limites de atuação. Tais teorias incluem, por exemplo, divisão e multiplicação de polinômios binários, registradores de deslocamento com realimentação linear, operações modulares, cifras simétricas, cifragem e decifragem de textos, entre outras,

Também foram investigados por nós alguns pontos de vulnerabilidade da tecnologia Bluetooth, porém, infelizmente não foi possível analisar estes aspectos de segurança com grande profundidade. Mesmo assim, pudemos constatar que é possível nos proteger de ataques por busca exaustiva ao PIN, desde que o seu comprimento em dígitos decimais seja adequado. Com um PIN de comprimento igual a 14 dígitos decimais, ou seja, cerca de 44 dígitos binários, teremos segurança na informação transmitida por mais de 500 anos, usando-se os parâmetros de tempo entre tentativas falhas de cerca de um mili segundo.

Podemos dizer que um dos pontos que poderá ser abordado como trabalho futuro é o estudo mais detalhado na área de vulnerabilidades de segurança da tecnologia Bluetooth. Também seria possível expandir a área de aplicação do nosso *software* para poder simular a comunicação real entre dois dispositivos Bluetooth, de modo que pudéssemos transferir uma sequência de dados mais longa, como um texto, ou simular transferência de arquivos de

imagem ou arquivos de som, já que na sua grande maioria, estes são os arquivos mais transferidos pelos usuários da tecnologia Bluetooth.

O código fonte do *software* utilizado, o qual é descrito posteriormente, encontra-se disponibilizado no CD em anexo. Este CD inclui também a versão executável deste código, permitindo assim o acesso gratuito a estas informações, para aqueles que desejarem desenvolver estudos nesta área.

ANEXOS

ANEXO 1 – Bluetooth.cpp

```
// Bluetooth Criptography
//

#include "stdafx.h"
#include "Bluetooth.h"

using namespace Bluetooth;

int main(array<System::String ^> ^args)
{
	entrada_texto=true;
	print=false;

	Application::EnableVisualStyles();
	Application::SetCompatibleTextRenderingDefault(false);

	Application::Run(gcnew Form1());
	return 0;
}
```

ANEXO 2 – Bluetooth.h

```
#pragma once
#using <mscorlib.dll>
#include<math.h>
#include<stdio.h>
#include<time.h>
#include<conio.h>
#include<iostream>
#include<complex>
#include<vector>
using namespace std;
using namespace System;

int RAND_a[16]={0},RAND_b[16]={0},BD_ADDRESS_A_dec[6]={0},BD_ADDRESS_B_dec[6]={0};
int PIN_dec[16]={0},PINl_dec[16]={0},LK_KA[16]={0},LK_KB[16]={0},Kinit[16]={0};
int RAND1[16]={0},RAND2[16]={0},Kmaster[16]={0},Kab[16]={0},KC[16]={0};
int K_offset[16]={0},SRES[16]={0},ACO[16]={0},COF[12]={0},EN_RAND[16]={0};
int KCl_dec[16]={0},L,Ll,plain_text[17]={0},cipher_text[17]={0};
int plain_text_dec[17]={0},L_effective,decipher_text[16]={0},Octet[16][8]={0};
int binary[128]={0},nbits,number,i,p,j,x,y,z,Bpi[18][16]={0},AuxBpi[18][16]={0};
int BpiBinary[18][128]={0},Octetaux[17][8]={0},k[18][129]={0},Fase[10][10][16]={0};
int round,str1d[30]={0},kdec[18][16]={0},str1b[16][8]={0},random[16],X[16]={0};
int X_E21[16]={0},Y_E21[16]={0},PIN_bin[64]={0},Faseaux[9][10][16]={0};
int X_E22[16]={0},Y_E22[16]={0},Xtable[256]={0},Ytable[256][2]={0};
int grau_POL1,grau_POL2,grau_POL3,POL1[128]={0},POL2[128]={0},POL3[128]={0};
int RAND[16]={0},ENTRADA[16]={0},ENTRADA_ARL[16]={0},LINK_KEY[16]={0};
int LFSR1[25]={0},ent_LFSR1[49]={0},LFSR1_24,LFSR2[31]={0},ent_LFSR2[55]={0};
int LFSR2_30,LFSR3[33]={0},ent_LFSR3[49]={0},LFSR3_32,LFSR4[39]={0};
int ent_LFSR4[55]={0},LFSR4_38,SAIDA_AR[16]={0},zt_dec[17]={0};
int t1=0,t2=0,t3=0,t4=0,t=0,ct[240]={0},zt[240]={0},yt[240]={0},st[240]={0},T1,T2;
int Pmul_aux[256]={0},Pmul_res[256]={0},Pmul1[129]={0},Pmul2[129]={0},KCl[129]={0};
unsigned __int64 decimal,PIN;
bool LK_KA_bool,LK_KB_bool,Kinit_bool,Kmaster_bool,x_bool,acha_grau,teste,KC_bool;
bool E1_key,E1_ARL_key,E3_key,teste1_bool,print,printbin;
bool RA,RB,RB1,RB2,RB3,RB4,RB5,RB6,RB7,entrada_texto;
char str1[30]={0},plain_text_string[16]={0},chKinit[1];

void Converts(unsigned __int64 localnumber)
{
        unsigned __int64 localresto;
        int n,saida;
        localresto=localnumber;
        for (n=0;n<=63;n++){ binary[n]=0;} //zerar o vetor
        while (localnumber>=2)
                {
                        n--;
                        localresto=localnumber%2;
                        localnumber=(localnumber-localresto)/2;
                        binary[n]=localresto;
                }
        n--;
        binary[n]=localnumber;
        if (printbin)
                {
                        if (localnumber==0) printf("%d",0);
                                else saida=0;
```

```
                        if (localnumber!=0)
                                for (n=0;n<=63;n++)
                                        {
                                                if (binary[n]!=0) saida=1;
                                                if (saida==1) printf("%d",binary[n]);
                                        }
                        printf("\n");
                }
}
void binary_to_decimal(int local_binary[8],int startbit,int endbit)
{
        int ilocal;
        number=0;
        for (ilocal=startbit;ilocal<=endbit;ilocal++)
                number=number+(local_binary[ilocal]*Math::Pow(2,(endbit-ilocal)));
}
void xor(int n1,int n2)
{
        int n1b[64]={0},n2b[64]={0};
        Converts(n1);
        for (y=0;y<=7;y++)
                {
                        n1b[7-y]=binary[63-y];
                }
        Converts(n2);
        for (y=0;y<=7;y++)
                {
                        n2b[7-y]=binary[63-y];
                }
        for (y=0;y<=7;y++)
                {
                        n1b[y]=n1b[y]+n2b[y];
                        if (n1b[y]==2) n1b[y]=0;
                }
        binary_to_decimal(n1b,0,7);
}
void Calculate_AuxBpi(int plocal,int ilocal,int mlocal)
{
        int aux,ylocal,xlocal;
        xlocal=45;
        aux=xlocal;
        ylocal=17*plocal+ilocal+1;
        xlocal=xlocal*xlocal;
        while (ylocal>=3)
        {
                xlocal=(xlocal*aux)%mlocal;
                ylocal--;
        }
        x=xlocal;
}
void Calculate_Bpi(int plocal,int mlocal)
{
        int aux,ylocal,xlocal;
        xlocal=45;
        aux=xlocal;
        ylocal=plocal;
        while (ylocal>=2)
        {
                xlocal=(xlocal*aux)%mlocal;
                ylocal--;
```

```
		}
		x=xlocal;
		if (ylocal==0) x=1;
}
void rotate3bitsleft(void)
{
		int auxp0,auxp1,auxp2;
		for (i=0;i<=16;i++)
		{
			auxp0=Octet[i][0];
			auxp1=Octet[i][1];
			auxp2=Octet[i][2];
			for (p=0;p<=4;p++) Octet[i][p]=Octet[i][p+3];
			Octet[i][5]=auxp0;
			Octet[i][6]=auxp1;
			Octet[i][7]=auxp2;
		}
}
void Octet16(void)
{
		int aux=0;
		for (p=0;p<=7;p++)
		{
			for (i=0;i<=15;i++)
				aux=aux+Octet[i][p];
			Octet[16][p]=aux%2;
		}
}
void Print_K(int klocal)
{
		j=0;
		for (i=0;i<=15;i++)
		{
			for (p=0;p<=7;p++)
			{
				k[klocal][j]=Octet[i][p];//converte 16 Octetos para chave de 128 bits
				j++;
			}
			binary_to_decimal(k[klocal],8*i,8*i+7);
			kdec[klocal][i]=(Bpi[klocal][i]+number)%256;
			Converts(kdec[klocal][i]);
			for (y=0;y<=7;y++) k[klocal][8*i+7-y]=binary[63-y];
		}
		if (print)
		{
			printf("K%d\n",klocal);
			for (i=0;i<=15;i++)
				printf("%d,",kdec[klocal][i]);
			printf("\n");
			j=0;
			for (i=0;i<=127;i++)
			{
				printf("%d",k[klocal][i]);
				j++;
				if (j==8)
				{
					printf(" ");
					j=0;
				}
			}
```

```
                printf("\n");
        }
}
void rotate1Octetleft(void)
{
        for (p=0;p<=7;p++) Octetaux[17][p]=Octet[0][p];
        for (i=0;i<=16;i++)
                for (p=0;p<=7;p++) Octetaux[i-1][p]=Octet[i][p];
        for (p=0;p<=7;p++) Octetaux[16][p]=Octetaux[17][p];
}
void CopyOctetauxtoOctet(void)
{
        for (i=0;i<=16;i++)
                for (p=0;p<=7;p++) Octet[i][p]=Octetaux[i][p];
}
void CalculateBpiBinary(void)
{
        for (p=0;p<=18;p++)
                for (i=0;i<=128;i++) BpiBinary[p][i]=0;
        for (p=2;p<=17;p++)
        {
                j=127;
                for (i=0;i<=15;i++)
                {
                        Converts(Bpi[p][i]);
                        for (y=0;y<=7;y++)
                        {
                                BpiBinary[p][127-j]=binary[63-y];
                                j--;
                        }
                }
        }
}
void PHT(int round,int step,int startoctet,int xlocal,int ylocal)
{
        Fase[round][step+1][startoctet]=(2*xlocal+ylocal)%256;
        Fase[round][step+1][startoctet+1]=(xlocal+ylocal)%256;
}
void Calculate_Fase_x_y(int round,int xlocal,int ylocal)
{
        Converts(Fase[round-1][10][xlocal]);
        for (y=0;y<=7;y++)
        {
                binary[63-y]=binary[63-y]+k[(2*round)-1][(8*xlocal+7)-y];
                if (binary[63-y]==2) binary[63-y]=0;
        }
        binary_to_decimal(binary,56,63);
        Fase[round][1][xlocal]=number;
        binary_to_decimal(k[2*round-1],8*ylocal,(8*ylocal+7));
        number=(number+Fase[round-1][10][ylocal])%256;
        Fase[round][1][ylocal]=number;
        Fase[round][2][xlocal]=Xtable[Fase[round][1][xlocal]];
        Fase[round][2][ylocal]=Ytable[Fase[round][1][ylocal]][1];
        binary_to_decimal(k[2*round],8*xlocal,(8*xlocal+7));
        Fase[round][3][xlocal]=(number+Fase[round][2][xlocal])%256;
        Converts(Fase[round][2][ylocal]);
        for (y=0;y<=7;y++)
        {
                binary[63-y]=binary[63-y]+k[2*round][(8*ylocal+7)-y];
                if (binary[63-y]==2) binary[63-y]=0;
```

```
		}
		binary_to_decimal(binary,56,63);
		Fase[round][3][ylocal]=number;
		PHT(round,3,xlocal,Fase[round][3][xlocal],Fase[round][3][ylocal]);
}

void Calculate_Fase2_x_y(int round,int xlocal,int ylocal)
{
		Converts(Fase[round-1][10][ylocal]);
		for (y=0;y<=7;y++)
		{
			binary[63-y]=binary[63-y]+k[2*round-1][(8*ylocal+7)-y];
			if (binary[63-y]==2) binary[63-y]=0;
		}
		binary_to_decimal(binary,56,63);
		Fase[round][1][ylocal]=number;
		binary_to_decimal(k[2*round-1],8*xlocal,8*xlocal+7);
		number=(number+Fase[round-1][10][xlocal])%256;
		Fase[round][1][xlocal]=number;
		Fase[round][2][ylocal]=Xtable[Fase[round][1][ylocal]];
		Fase[round][2][xlocal]=Ytable[Fase[round][1][xlocal]][1];
		binary_to_decimal(k[2*round],8*ylocal,8*ylocal+7);
		Fase[round][3][ylocal]=(number+Fase[round][2][ylocal])%256;
		Converts(Fase[round][2][xlocal]);
		for (y=0;y<=7;y++)
		{
			binary[63-y]=binary[63-y]+k[2*round][8*xlocal+7-y];
			if (binary[63-y]==2) binary[63-y]=0;
		}
		binary_to_decimal(binary,56,63);
		Fase[round][3][xlocal]=number;
		PHT(round,3,xlocal,Fase[round][3][xlocal],Fase[round][3][ylocal]);
}

void input(bool auto_input,bool E21_key,bool E22_key)
{
		if ((round==3)&(auto_input))
		{
			str1[0]='D';str1[1]='A';str1[2]='N';str1[3]='I';str1[4]='E';str1[5]='L';
			str1[6]='L';str1[7]='E';str1[8]='L';str1[9]='E';
			str1[10]='L';str1[11]='O';str1[12]='R';str1[13]='E';str1[14]='N';
			str1[15]='A';
			for (i=0;i<=15;i++)
				str1d[i]=str1[i];
			j=2;p=2;
		}
		if ((round==3)&(E1_key))
		{
			for (i=0;i<=15;i++) str1d[i]=RAND[i];
			j=2;p=2;
		}
		if ((round==3)&(E21_key))
		{
			for (i=0;i<=15;i++) str1d[i]=Y_E21[i];
			j=2;p=2;
		}
		if ((round==3)&(E22_key))
		{
			for (i=0;i<=15;i++) str1d[i]=Y_E22[i];
			j=2;p=2;
```

```
	}
	if ((round==3)&(E3_key))
	{
		for (i=0;i<=15;i++) str1d[i]=EN_RAND[i];
		j=2;p=2;
	}
	if (round==1)
	{
		for (i=0;i<=15;i++) str1d[i]=ENTRADA[i];
		j=0;p=0;
		printf("Input=\n");
		for (i=0;i<=15;i++) printf("%d, ",str1d[i]);
		printf("\n\n");
	}
	if (round==8)
	{
		for (i=0;i<=15;i++) str1d[i]=kdec[17][i];
		j=9;p=8;
	}
	if (print)
	{
		printf("INPUT =\n");
		for(i=0;i<=15;i++)
		{
			if (str1d[i]<10) printf(" ");
			if (str1d[i]<100) printf(" ");
			printf("%d, ",str1d[i]);
		}
		printf("\n");
	}
	Fase[j][10][1]=(str1d[1]+Fase[p][10][1])%256;
	Fase[j][10][2]=(str1d[2]+Fase[p][10][2])%256;
	Fase[j][10][5]=(str1d[5]+Fase[p][10][5])%256;
	Fase[j][10][6]=(str1d[6]+Fase[p][10][6])%256;
	Fase[j][10][9]=(str1d[9]+Fase[p][10][9])%256;
	Fase[j][10][10]=(str1d[10]+Fase[p][10][10])%256;
	Fase[j][10][13]=(str1d[13]+Fase[p][10][13])%256;
	Fase[j][10][14]=(str1d[14]+Fase[p][10][14])%256;
	xor(str1d[0],Fase[p][10][0]); Fase[j][10][0]=number;
	xor(str1d[3],Fase[p][10][3]); Fase[j][10][3]=number;
	xor(str1d[4],Fase[p][10][4]); Fase[j][10][4]=number;
	xor(str1d[7],Fase[p][10][7]); Fase[j][10][7]=number;
	xor(str1d[8],Fase[p][10][8]); Fase[j][10][8]=number;
	xor(str1d[11],Fase[p][10][11]); Fase[j][10][11]=number;
	xor(str1d[12],Fase[p][10][12]); Fase[j][10][12]=number;
	xor(str1d[15],Fase[p][10][15]); Fase[j][10][15]=number;
	if (print)
	{
		printf("Fase[%d][10]:\n",j);
		for(i=0;i<=15;i++)
		{
			if (Fase[j][10][i]<10) printf(" ");
			if (Fase[j][10][i]<100) printf(" ");
			printf("%d, ",Fase[j][10][i]);
		}
		printf("\n");
	}
}

void permutation(int round,int newfase)
```

```
{
	Fase[round][newfase][0]=Fase[round][newfase-1][8];
	Fase[round][newfase][1]=Fase[round][newfase-1][11];
	Fase[round][newfase][2]=Fase[round][newfase-1][12];
	Fase[round][newfase][3]=Fase[round][newfase-1][15];
	Fase[round][newfase][4]=Fase[round][newfase-1][2];
	Fase[round][newfase][5]=Fase[round][newfase-1][1];
	Fase[round][newfase][6]=Fase[round][newfase-1][6];
	Fase[round][newfase][7]=Fase[round][newfase-1][5];
	Fase[round][newfase][8]=Fase[round][newfase-1][10];
	Fase[round][newfase][9]=Fase[round][newfase-1][9];
	Fase[round][newfase][10]=Fase[round][newfase-1][14];
	Fase[round][newfase][11]=Fase[round][newfase-1][13];
	Fase[round][newfase][12]=Fase[round][newfase-1][0];
	Fase[round][newfase][13]=Fase[round][newfase-1][7];
	Fase[round][newfase][14]=Fase[round][newfase-1][4];
	Fase[round][newfase][15]=Fase[round][newfase-1][3];
}

void kbin_to_kdec()
{
	int jl,il;
	for (jl=1;jl<=17;jl++)
		for (il=0;il<=15;il++)
		{
			binary_to_decimal(k[jl],8*il,8*il+7);
			kdec[jl][il]=number;
			if (print)
			{
				if (il==0) printf("K[%d]=\n",jl);
				if (number<100) printf(" ");
				if (number<10) printf(" ");
				printf("%d, ",kdec[jl][il]);
				if (il==15) printf("\n");
			}
		}
}

void saferplus(bool ARL,bool auto_key,bool E21_key,bool E22_key)
{
	for (i=0;i<=10;i++)
		for (j=0;j<=10;j++)
			for (p=0;p<=16;p++) Fase[i][j][p]=0;
	if (print)
	{
		if (ARL) printf("AR':\n");
		else printf("AR:\n");
	}
	for (p=2;p<=17;p++)
		for (i=0;i<=15;i++)
		{
			Calculate_AuxBpi(p,i,257);
			AuxBpi[p][i]=x;
		}
	for (p=2;p<=17;p++)
		for (i=0;i<=15;i++)
		{
			Calculate_Bpi(AuxBpi[p][i],257);
			Bpi[p][i]=x%256;
		}
```

```
if (auto_key)
        for (i=0;i<=15;i++)
        {
                Converts(i);
                for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];//chave automática
        }
if (E1_ARL_key)
        for (i=0;i<=15;i++)
        {
                Converts(ENTRADA_ARL[i]);
                for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
        }
if (E21_key)
        for (i=0;i<=15;i++)
        {
                if ((RA)|(RB)) Converts(Kinit[i]);// se RA ou RB então chave = Kinit
                for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
        }
if (E22_key)
        for (i=0;i<=15;i++)
        {
                Converts(Kab[i]);// se E22 então chave = Kab
                for (p=0;p<=7;p++)
                        Octet[i][7-p]=binary[63-p];
        }
if (E3_key)
        for (i=0;i<=15;i++)
        {
                Converts(LINK_KEY[i]);// se E3 então chave =
                for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
        }
Octet16();
CalculateBpiBinary();
for (p=0;p<=17;p++)
        for (i=0;i<=127;i++) k[p][i]=0; //zerar k[][]
Print_K(1); //Encontra K1 e mostra na tela
for (z=2;z<=17;z++)
{
        rotate3bitsleft();
        rotate1Octetleft();
        CopyOctetauxtoOctet();
        Print_K(z); //Encontra Kz e mostra na tela
}
kbin_to_kdec();//transforma as chaves intermediárias de binário para decimal
for (round=1;round<=8;round++)
{
        if (print)_getch();
        if (round==1) input(false,false,false);
        if ((round==3)&(ARL)) input(false,E21_key,E22_key);
        Calculate_Fase_x_y(round,0,1); Calculate_Fase_x_y(round,4,5);
        Calculate_Fase_x_y(round,8,9); Calculate_Fase_x_y(round,12,13);
        Calculate_Fase2_x_y(round,2,3); Calculate_Fase2_x_y(round,6,7);
        Calculate_Fase2_x_y(round,10,11); Calculate_Fase2_x_y(round,14,15);
        permutation(round,5);
        for (i=0;i<=8;i++)
                PHT(round,5,2*i,Fase[round][5][2*i],Fase[round][5][2*i+1]);
        permutation(round,7);
        for (i=0;i<=8;i++)
                PHT(round,7,2*i,Fase[round][7][2*i],Fase[round][7][2*i+1]);
        permutation(round,9);
```

```
for (i=0;i<=8;i++)
	PHT(round,9,2*i,Fase[round][9][2*i],Fase[round][9][2*i+1]);
if (print)
{
	for (j=0;j<=10;j++)
	{
		printf("Fase[%d][%d]:\n",round,j);
		for (i=0;i<=15;i++)
		{
			if (Fase[round][j][i]<100) printf(" ");
			if (Fase[round][j][i]<10) printf(" ");
			printf("%d, ",Fase[round][j][i]);
		}
		printf("\n");
	}
}
if (round==8) input(false,E21_key,E22_key);
if ((round==8)&(Kinit_bool))
{
	printf("Kinit:\n");
	for (i=0;i<=15;i++)
	{
		if (Fase[round+1][10][i]<100) printf(" ");
		if (Fase[round+1][10][i]<10) printf(" ");
		printf("%d, ",Fase[round+1][10][i]);
	}
	printf("\n");
}
if ((round==8)&(Kmaster_bool))
{
	printf("Kmaster:\n");
	for (i=0;i<=15;i++)
	{
		if (Fase[round+1][10][i]<100) printf(" ");
		if (Fase[round+1][10][i]<10) printf(" ");
		printf("%d, ",Fase[round+1][10][i]);
	}
	printf("\n");
}
if ((round==8)&(E1_key))
{
	printf("Saida AR=\n");
	for (i=0;i<=15;i++)
	{
		if (Fase[round+1][10][i]<100) printf(" ");
		if (Fase[round+1][10][i]<10) printf(" ");
		printf("%d, ",Fase[round+1][10][i]);
	}
	printf("\n");
}
if ((round==8)&(E21_key))
{
	if (LK_KA_bool) printf("Ka=\n");
	if (LK_KB_bool) printf("Kb=\n");
	for (i=0;i<=15;i++)
	{
		if (Fase[round+1][10][i]<100) printf(" ");
		if (Fase[round+1][10][i]<10) printf(" ");
		printf("%d, ",Fase[round+1][10][i]);
	}
```

```
                printf("\n");
            }
            if ((round==8)&(teste1_bool))
            {
                printf("teste1:\n");
                for (i=0;i<=15;i++)
                {
                    if (Fase[round+1][10][i]<100) printf(" ");
                    if (Fase[round+1][10][i]<10) printf(" ");
                    printf("%d, ",Fase[round+1][10][i]);
                }
                printf("\n");
            }
        }
}

void E_box(int BD_ADDRESS_dec[16],int L)// (E BLOCK)
{
        for (i=0;i<=15;i++)
        {
            X[i]=BD_ADDRESS_dec[i%L];
            if (print) printf("%d, ",X[i]);
        }
        if (print) printf("\n");
}

void Calculate_K_offset()
{
        if (print)
        {
            printf("Kdec[1]= \n");
            for (i=0;i<=15;i++)
            {
                if (kdec[1][i]<10) printf(" ");
                if (kdec[1][i]<100) printf(" ");
                printf("%d, ",kdec[1][i]);
            }
            printf("\n");
        }
        K_offset[0]=(kdec[1][0]+233)%256; K_offset[2]=(kdec[1][2]+223)%256;
        K_offset[4]=(kdec[1][4]+179)%256; K_offset[6]=(kdec[1][6]+149)%256;
        K_offset[9]=(kdec[1][9]+229)%256; K_offset[11]=(kdec[1][11]+193)%256;
        K_offset[13]=(kdec[1][13]+167)%256; K_offset[15]=(kdec[1][15]+131)%256;
        xor(kdec[1][1],229); K_offset[1]=number;
        xor(kdec[1][3],193); K_offset[3]=number;
        xor(kdec[1][5],167); K_offset[5]=number;
        xor(kdec[1][7],131); K_offset[7]=number;
        xor(kdec[1][8],233); K_offset[8]=number;
        xor(kdec[1][10],223); K_offset[10]=number;
        xor(kdec[1][12],179); K_offset[12]=number;
        xor(kdec[1][14],149); K_offset[14]=number;
        if (print)
        {
            printf("233, 229, 223, 193, 179, 167, 149, 131, ");
            printf("233, 229, 223, 193, 179, 167, 149, 131,\n\n");
        }
        printf("K_offset= \n");
        for (i=0;i<=15;i++)
        {
            if (K_offset[i]<10) printf(" ");
```

```
                if (K_offset[i]<100) printf(" ");
                printf("%d, ",K_offset[i]);
        }
        printf("\n");
        for (i=0;i<=15;i++) kdec[1][i]=K_offset[i];
}

void E21(bool autokey)
{
        if (autokey)
        {
                for (i=0;i<=15;i++) X_E21[i]=rand()%256;
                xor(X_E21[15],6); X_E21[15]=number;
                for (i=0;i<=15;i++) Y_E21[i]=BD_ADDRESS_A_dec[i%6];
        }
        if (RA)//RAND(a)
        {
                for (i=0;i<=15;i++)
                {
                        X_E21[i]=RAND_a[i];
                        ENTRADA[i]=RAND_a[i];
                }
                for (i=0;i<=15;i++) Y_E21[i]=BD_ADDRESS_A_dec[i%6];
        }
        if (RB)//RAND(b)
        {
                for (i=0;i<=15;i++)
                {
                        X_E21[i]=RAND_b[i];
                        ENTRADA[i]=RAND_b[i];
                }
                for (i=0;i<=15;i++) Y_E21[i]=BD_ADDRESS_B_dec[i%6];
        }
        if (print)
        {
                printf("X_E21=\n");
                for (i=0;i<=15;i++) printf("%d, ",X_E21[i]);
                printf("\n");
                printf("Y_E21=\n");
                for (i=0;i<=15;i++) printf("%d, ",Y_E21[i]);
                printf("\n");
        }
        saferplus(true,false,true,false);
        if (LK_KA_bool) for (i=0;i<=15;i++) LK_KA[i]=Fase[9][10][i];
        if (LK_KB_bool) for (i=0;i<=15;i++) LK_KB[i]=Fase[9][10][i];
}

void E22()
{
        if (Kinit_bool)
        {
                for (i=0;i<=15;i++) Y_E22[i]=rand()%256;
                xor(Y_E22[15],Ll); Y_E22[15]=number;
                for (i=0;i<=15;i++)
                {
                        X_E22[i]=PINl_dec[i%Ll];
                        ENTRADA[i]=X_E22[i];
                }
                if (print)
                {
```

```
                        printf("L'= %d\n",Ll);
                        printf("X_E22=\n");
                        for (i=0;i<=15;i++) printf("%d, ",X_E22[i]);
                        printf("\n");
                        printf("Y_E22=\n");
                        for (i=0;i<=15;i++) printf("%d, ",Y_E22[i]);
                        printf("\n");
                }
                saferplus(true,false,false,true);
                for (i=0;i<=15;i++) Kinit[i]=Fase[9][10][i];
        }
        if (Kmaster_bool)
        {
                for (i=0;i<=15;i++)
                {
                        X_E22[i]=RAND1[i];
                        Y_E22[i]=RAND2[i];
                        ENTRADA[i]=X_E22[i];
                }
                Ll=16;
                saferplus(true,false,false,true);
                for (i=0;i<=15;i++) Kmaster[i]=Fase[9][10][i];
        }
}

void E3(void)
{
//gerar EN_RAND:
        printf("EN_RAND= \n");
        for (i=0;i<=15;i++) EN_RAND[i]=rand()%256;
        for (i=0;i<=15;i++) printf("%d, ",EN_RAND[i]);
//gerar COF:
        printf("\n\nCOF= \n");
        for (i=0;i<=11;i++)
        {
                COF[i]=BD_ADDRESS_A_dec[i%6];
                printf("%d, ",COF[i]);
        }
        printf("\n\n");
//link_key=Kmaster
        for (i=0;i<=15;i++)
        {
                Converts(Kmaster[i]);// chave=Kmaster
                for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
        }
//AR
        for (i=0;i<=15;i++) ENTRADA[i]=EN_RAND[i];
        saferplus(false,false,false,false);
//xor 16
        for (i=0;i<=15;i++) SAIDA_AR[i]=Fase[9][10][i];
        for (i=0;i<=15;i++)
        {
                xor(SAIDA_AR[i],EN_RAND[i]);
                ENTRADA_ARL[i]=number;
        }
//E => (COF=96 e L=12)
        E_box(COF,12);
//add+16
        for (i=0;i<=15;i++)         ENTRADA_ARL[i]=(ENTRADA_ARL[i]+X[i])%256;
//offset=K~
```

```
		for (i=0;i<=15;i++) kdec[1][i]=Kmaster[i];
		Calculate_K_offset();
		for (i=0;i<=15;i++)
		{
			Converts(kdec[1][i]);// chave=Koffset
			for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
		}
//A'r
		for (i=0;i<=15;i++) ENTRADA[i]=ENTRADA_ARL[i];
		saferplus(true,false,false,false);
//KC:
		printf("KC= \n");
		for (i=0;i<=15;i++)
		{
			KC[i]=Fase[9][10][i];
			printf("%d, ",KC[i]);
		}
		printf("\n");
}

void E1(void)
{
//RAND
		//Calculado no início do software
//K=link key = Kab
		for (i=0;i<=15;i++)
		{
			Converts(Kab[i]);// se E1 e AR então chave=kab
			for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
		}
//AR
		for (i=0;i<=15;i++) ENTRADA[i]=RAND[i];
		saferplus(false,false,false,false);
//xor 16
		for (i=0;i<=15;i++) SAIDA_AR[i]=Fase[9][10][i];
		for (i=0;i<=15;i++)
		{
			xor(SAIDA_AR[i],RAND[i]);
			ENTRADA_ARL[i]=number;
		}
//E (address=48 e L=6)
		E_box(BD_ADDRESS_A_dec,6);
//add+16
		for (i=0;i<=15;i++)		ENTRADA_ARL[i]=(ENTRADA_ARL[i]+X[i])%256;
//Koffset
		for (i=0;i<=15;i++) kdec[1][i]=Kab[i];
		Calculate_K_offset();
		for (i=0;i<=15;i++)
		{
			Converts(kdec[1][i]);// chave=Koffset
			for (p=0;p<=7;p++) Octet[i][7-p]=binary[63-p];
		}
//A'r
		for (i=0;i<=15;i++) ENTRADA[i]=ENTRADA_ARL[i];
		saferplus(true,false,false,false);
//SRES
		printf("SRES= \n");
		for (i=0;i<=3;i++)
		{
			SRES[i]=Fase[9][10][i];
```

```
                printf("%d, ",SRES[i]);
        }
//ACO
        printf("\n\nACO= \n");
        for (i=4;i<=15;i++)
        {
                ACO[i]=Fase[9][10][i];
                printf("%d, ",ACO[i]);
        }
        printf("\n");
}

void Divides_polynomials(void) //acha o módulo da divisão entre dois polinômios
{
        for (i=0;i<=grau_POL1;i++) POL3[i]=POL1[i];
        grau_POL3=grau_POL1;
        j=0;
        while (grau_POL3>=grau_POL2)
        {
                acha_grau=true;
                for (i=0;i<=grau_POL1;i++)
                {
                        xor(POL3[i+j],POL2[i]); POL3[i+j]=number;
                }
                grau_POL3=0;
                for (i=0;i<=grau_POL1;i++)
                if ((POL3[i]!=0)&(acha_grau))
                {
                        grau_POL3=grau_POL1-i;
                        acha_grau=false;
                        j=i;
                }
        }
        if (print)
        {
                printf("\nGrau POL1=%d\nPOL1=\n",grau_POL1);
                j=0;
                for (i=0;i<=grau_POL1;i++)
                {
                        j=j+1;
                        printf("%d,",POL1[i]);
                        if (j%32==0) printf("\n");
                }
                printf("\nGrau POL2=%d\nPOL2=\n",grau_POL2);
                j=0;
                for (i=0;i<=grau_POL2;i++)
                {
                        j=j+1;
                        printf("%d,",POL2[i]);
                        if (j%32==0) printf("\n");
                }
                printf("\nGrau POL3=%d\nPOL3=\n",grau_POL3);
                teste=false;
                for (i=0;i<=grau_POL1;i++)
                {
                        if ((POL3[i]==1)&(teste==false))
                        {
                                teste=true;
                                j=i;
                        }
```

```
                        if (teste) printf("%d,",POL3[i]);
                }
                printf("\n");
        }
        for (i=0;i<=127;i++)
        {
                if (i+j<=127) POL3[i]=POL3[i+j];
                else POL3[i]=0;
        }
}

void Multiply_polynomials(void)
{
        for (i=0;i<=127;i++)
        {
                if (Pmul2[128-i]==1)
                        for (j=128;j>=1;j--)
                        {
                                Pmul_aux[128+j-i]=Pmul1[j];
                                Pmul_res[128+j]=Pmul_res[128+j]+Pmul_aux[128+j];
                                if (Pmul_res[128+j]==2)
                                {
                                        Pmul_res[128+j]=0;
                                        Pmul_res[127+j]=Pmul_res[127+j]+1;
                                }
                                if (Pmul_res[128+j]==3)
                                {
                                        Pmul_res[128+j]=1;
                                        Pmul_res[127+j]=Pmul_res[127+j]+1;
                                }
                        }
                for (j=0;j<=256;j++) Pmul_aux[j]=0;
        }
        if (print)
        {
                printf("\nPmul_res=\n");
                for (i=128;i<=256;i++) printf("%d",Pmul_res[i]);
                printf("\n");
        }
}

void LFSRs(void)
{
        int aux_ct;
        Converts(BD_ADDRESS_A_dec[2]); for (j=0;j<=7;j++) ent_LFSR1[j]=binary[56+j];
        Converts(170); for (j=0;j<=7;j++) ent_LFSR1[8+j]=binary[56+j];
        Converts(KCl_dec[12]); for (j=0;j<=7;j++) ent_LFSR1[16+j]=binary[56+j];
        Converts(KCl_dec[8]); for (j=0;j<=7;j++) ent_LFSR1[24+j]=binary[56+j];
        Converts(KCl_dec[4]); for (j=0;j<=7;j++) ent_LFSR1[32+j]=binary[56+j];
        Converts(KCl_dec[0]); for (j=0;j<=7;j++) ent_LFSR1[40+j]=binary[56+j];
        ent_LFSR1[48]=0;//clock24=0
        Converts(BD_ADDRESS_A_dec[3]); for (j=0;j<=7;j++) ent_LFSR2[j]=binary[56+j];
        Converts(BD_ADDRESS_A_dec[0]); for (j=0;j<=7;j++) ent_LFSR2[8+j]=binary[56+j];
        Converts(KCl_dec[13]); for (j=0;j<=7;j++) ent_LFSR2[16+j]=binary[56+j];
        Converts(KCl_dec[9]); for (j=0;j<=7;j++) ent_LFSR2[24+j]=binary[56+j];
        Converts(KCl_dec[5]); for (j=0;j<=7;j++) ent_LFSR2[32+j]=binary[56+j];
        Converts(KCl_dec[1]); for (j=0;j<=7;j++) ent_LFSR2[40+j]=binary[56+j];
        ent_LFSR2[48]=0;ent_LFSR2[49]=1;ent_LFSR2[50]=0;ent_LFSR2[51]=1;
        ent_LFSR2[52]=0;ent_LFSR2[53]=0;ent_LFSR2[54]=1;
        Converts(BD_ADDRESS_A_dec[4]); for (j=0;j<=7;j++) ent_LFSR3[j]=binary[56+j];
```

```
Converts(170);//clock2=10101010
for (j=0;j<=7;j++) ent_LFSR3[8+j]=binary[56+j];
Converts(KCl_dec[14]); for (j=0;j<=7;j++) ent_LFSR3[16+j]=binary[56+j];
Converts(KCl_dec[10]); for (j=0;j<=7;j++) ent_LFSR3[24+j]=binary[56+j];
Converts(KCl_dec[6]); for (j=0;j<=7;j++) ent_LFSR3[32+j]=binary[56+j];
Converts(KCl_dec[2]); for (j=0;j<=7;j++) ent_LFSR3[40+j]=binary[56+j];
ent_LFSR3[48]=1;
Converts(BD_ADDRESS_A_dec[5]); for (j=0;j<=7;j++) ent_LFSR4[j]=binary[56+j];
Converts(BD_ADDRESS_A_dec[1]); for (j=0;j<=7;j++) ent_LFSR4[8+j]=binary[56+j];
Converts(KCl_dec[15]); for (j=0;j<=7;j++) ent_LFSR4[16+j]=binary[56+j];
Converts(KCl_dec[11]); for (j=0;j<=7;j++) ent_LFSR4[24+j]=binary[56+j];
Converts(KCl_dec[7]); for (j=0;j<=7;j++) ent_LFSR4[32+j]=binary[56+j];
Converts(KCl_dec[3]); for (j=0;j<=7;j++) ent_LFSR4[40+j]=binary[56+j];
ent_LFSR4[48]=0;ent_LFSR4[49]=1;ent_LFSR4[50]=0;ent_LFSR4[51]=1;
ent_LFSR4[52]=1;ent_LFSR4[53]=1;ent_LFSR4[54]=1;
if (print)
{
        printf("\nENTRADA LFSR1=\n");
        for (i=0;i<=48;i++)         printf("%d",ent_LFSR1[i]);
        printf("\n\nENTRADA LFSR2=\n");
        for (i=0;i<=54;i++)         printf("%d",ent_LFSR2[i]);
        printf("\n\nENTRADA LFSR3=\n");
        for (i=0;i<=48;i++) printf("%d",ent_LFSR3[i]);
        printf("\n\nENTRADA LFSR4=\n");
        for (i=0;i<=54;i++) printf("%d",ent_LFSR4[i]);
        printf("\n");
}
t=0;t1=0;t2=0;t3=0;t4=0;
for (p=1;p<=239;p++)
{
        for (i=38;i>=0;i--)
        {
                if ((p>=26)&(i<=24))//chave fechada quando p>=26, 24=comprimento LFSR1-1
                {
                        if (i==24)
                        {
                                LFSR1_24=LFSR1[24];
                                t1=t1+1;
                        }
                        if ((i!=0)&(i!=8)&(i!=12)&(i!=20)) LFSR1[i]=LFSR1[i-1];
                        if ((i==20)|(i==12)|(i==8))
                        {
                                LFSR1[i]=LFSR1[i-1]+LFSR1_24;
                                if (LFSR1[i]==2) LFSR1[i]=0;
                        }
                        if (i==0)
                        {
                                LFSR1[i]=ent_LFSR1[0]+LFSR1_24;
                                if (LFSR1[i]==2) LFSR1[i]=0;
                        }
                }
                if ((p<26)&(i<=24)) //chave aberta até p<26, 24=comprimento LFSR1-1
                {
                        LFSR1[i]=LFSR1[i-1];
                        if (i==0) LFSR1[0]=ent_LFSR1[0];
                }
                if ((p>=32)&(i<=30)) //chave fechada quando p>=32, 30=comprimento LFSR2-1
                {
                        if (i==30)
                        {
```

```
                LFSR2_30=LFSR2[30];
                t2=t2+1;
        }
        if ((i!=0)&(i!=12)&(i!=16)&(i!=24)) LFSR2[i]=LFSR2[i-1];
        if ((i==24)|(i==16)|(i==12))
        {
                LFSR2[i]=LFSR2[i-1]+LFSR2_30;
                if (LFSR2[i]==2) LFSR2[i]=0;
        }
        if (i==0)
        {
                LFSR2[i]=ent_LFSR2[0]+LFSR2_30;
                if (LFSR2[i]==2) LFSR2[i]=0;
        }
}
if ((p<32)&(i<=30))//chave aberta até p<32, 30=comprimento LFSR2-1
{
        LFSR2[i]=LFSR2[i-1];
        if (i==0) LFSR2[0]=ent_LFSR2[0];
}
if ((p>=34)&(i<=32))//chave fechada quando p>=34, 32=comprimento LFSR3-1
{
        if (i==32)
        {
                LFSR3_32=LFSR3[32];
                t3=t3+1;
        }
        if ((i!=0)&(i!=4)&(i!=24)&(i!=28)) LFSR3[i]=LFSR3[i-1];
        if ((i==28)|(i==24)|(i==4))
        {
                LFSR3[i]=LFSR3[i-1]+LFSR3_32;
                if (LFSR3[i]==2) LFSR3[i]=0;
        }
        if (i==0)
        {
                LFSR3[i]=ent_LFSR3[0]+LFSR3_32;
                if (LFSR3[i]==2) LFSR3[i]=0;
        }
}
if ((p<34)&(i<=32))//chave aberta até p<34, 32=comprimento LFSR3-1
{
        LFSR3[i]=LFSR3[i-1];
        if (i==0) LFSR3[0]=ent_LFSR3[0];
}
if (p>=40)//chave fechada quando p>=40
{
        if (i==38)
        {
                LFSR4_38=LFSR4[38];
                t4=t4+1;
        }
        if ((i!=0)&(i!=4)&(i!=28)&(i!=36)) LFSR4[i]=LFSR4[i-1];
        if ((i==36)|(i==28)|(i==4))
        {
                LFSR4[i]=LFSR4[i-1]+LFSR4_38;
                if (LFSR4[i]==2) LFSR4[i]=0;
        }
        if (i==0)
        {
                LFSR4[i]=ent_LFSR4[0]+LFSR4_38;
```

```
                                        if (LFSR4[i]==2) LFSR4[i]=0;
                                }
                        }
                        if (p<40) //chave aberta até p<40
                        {
                                LFSR4[i]=LFSR4[i-1];
                                if (i==0) LFSR4[0]=ent_LFSR4[0];
                        }
                }
                t=t+1;
                if ((t>=39)&(t<=239))
                {
                        if (t==39) ct[39]=0;ct[38]=0;
                        //cálculo de y(t)
                        yt[t]=LFSR1[24]+LFSR2[24]+LFSR3[32]+LFSR4[32];
                        //cálculo de s(t+1)
                        st[t+1]=(yt[t]+ct[t]);
                        if ((st[t+1]==0)|(st[t+1]==1)) st[t+1]=0;
                        if ((st[t+1]==2)|(st[t+1]==3)) st[t+1]=1;
                        if ((st[t+1]==4)|(st[t+1]==5)) st[t+1]=2;
                        if ((st[t+1]==6)|(st[t+1]==7)) st[t+1]=3;
                        //cálculo de c(t+1)
                        if (ct[t]==0) T1=0; if (ct[t]==1) T1=1;
                        if (ct[t]==2) T1=2;             if (ct[t]==3) T1=3;
                        if (ct[t-1]==0) T2=0; if (ct[t-1]==1) T2=3;
                        if (ct[t-1]==2) T2=1; if (ct[t-1]==3) T2=2;
                        xor(T1,T2); aux_ct=number;
                        xor(aux_ct,st[t+1]); ct[t+1]=number;
                        //cálculo de z(t)
                        zt[t]=(yt[t]+(ct[t]%2))%2;
                        //saidas:
                        if (print)
                        {
                                if ((t%50)==0) _getch();
                                printf("\nx%d_1=%d, x%d_2=%d, ",t,LFSR1[24],t,LFSR2[24]);
                                printf("x%d_3=%d, x%d_4=%d\n",t,LFSR3[32],t,LFSR4[32]);
                                printf("yt[%d]=%d, st[%d]=%d",t,yt[t],t+1,st[t+1]);
                                printf(", ct[%d]=%d, zt[%d]=%d\n",t+1,ct[t+1],t,zt[t]);
                        }
                }
                if (p<=54)
                {
                for (j=0;j<=48;j++) ent_LFSR1[j]=ent_LFSR1[j+1];//desloca entrada do LFSR1
                for (j=0;j<=54;j++) ent_LFSR2[j]=ent_LFSR2[j+1];//desloca entrada do LFSR2
                for (j=0;j<=48;j++) ent_LFSR3[j]=ent_LFSR3[j+1];//desloca entrada do LFSR3
                for (j=0;j<=54;j++) ent_LFSR4[j]=ent_LFSR4[j+1];//desloca entrada do LFSR4
                }
}
if (print)
{
        printf("\nSaidas dos LFSRs:\n");
        printf("\nLFSR1=       ");
        for (i=0;i<=24;i++) printf("%d",LFSR1[i]);
        printf("\nLFSR2=       ");
        for (i=0;i<=30;i++)  printf("%d",LFSR2[i]);
        printf("\nLFSR3=");
        for (i=0;i<=32;i++)  printf("%d",LFSR3[i]);
        printf("\nLFSR4=");
        for (i=0;i<=38;i++) printf("%d",LFSR4[i]);
        printf("\nt1=%d\nt2=%d\nt3=%d\nt4=%d\n",t1,t2,t3,t4);
```

```
                printf("%d simbolos gerados\n",t4);
        }
        j=0;
        printf("\n200 saidas geradas:\n");
        for (i=40;i<=239;i++)
        {
                j=j+1;
                printf("%d",zt[i]);
                if (j%50==0) printf("\n");
        }
}

void find_KCl(void)
{
        int g1_bin[16][128]={0},g2_bin[16][128]={0},g1[16][16]={0},g2[16][16]={0};
        //definição dos polinômios g1(x)
        g1[1][14]=1;g1[1][15]=29; g1[2][13]=1;g1[2][15]=63;
        g1[3][12]=1;g1[3][13]=1;g1[3][15]=219; g1[4][11]=1;g1[4][15]=175;
        g1[5][10]=1;g1[5][15]=57; g1[6][9]=1;g1[6][14]=2;g1[5][15]=145;
        g1[7][8]=1;g1[7][15]=149; g1[8][7]=1;g1[8][15]=27;
        g1[9][6]=1;g1[9][14]=6;g1[9][15]=9; g1[10][5]=1;g1[10][14]=2;g1[10][15]=21;
        g1[11][4]=1;g1[11][14]=1;g1[11][15]=59; g1[12][3]=1;g1[12][15]=221;
        g1[13][2]=1;g1[13][14]=4;g1[13][15]=157;
        g1[14][1]=1;g1[14][14]=1;g1[14][15]=79; g1[15][0]=1;g1[15][15]=231;
        //definição dos polinômios g2(x)
        g2[1][1]=226;g2[1][2]=117;g2[1][3]=160;g2[1][4]=171;g2[1][5]=210;g2[1][6]=24;
        g2[1][7]=212;g2[1][8]=207;g2[1][9]=146;g2[1][10]=139;g2[1][11]=155;
        g2[1][12]=191;g2[1][13]=108;g2[1][14]=176;g2[1][15]=143;
        g2[2][1]=1;g2[2][2]=227;g2[2][3]=246;g2[2][4]=61;g2[2][5]=118;g2[2][6]=89;
        g2[2][7]=179;g2[2][8]=127;g2[2][9]=24;g2[2][10]=194;g2[2][11]=88;g2[2][12]=207;
        g2[2][13]=246;g2[2][14]=239;g2[2][15]=239;
        g2[3][2]=1;g2[3][3]=190;g2[3][4]=246;g2[3][5]=108;g2[3][6]=108;g2[3][7]=58;
        g2[3][8]=177;g2[3][9]=3;g2[3][10]=10;g2[3][11]=90;g2[3][12]=25;g2[3][13]=25;
        g2[3][14]=128;g2[3][15]=139;
        g2[4][3]=1;g2[4][4]=106;g2[4][5]=184;g2[4][6]=153;g2[4][7]=105;g2[4][8]=222;
        g2[4][9]=23;g2[4][10]=70;g2[4][11]=127;g2[4][12]=211;g2[4][13]=115;
        g2[4][14]=106;g2[4][15]=217;
        g2[5][4]=1;g2[5][5]=99;g2[5][6]=6;g2[5][7]=50;g2[5][8]=145;g2[5][9]=218;
        g2[5][10]=80;g2[5][11]=236;g2[5][12]=85;g2[5][13]=113;g2[5][14]=82;g2[5][15]=71;
        g2[6][6]=44;g2[6][7]=147;g2[6][8]=82;g2[6][9]=170;g2[6][10]=108;g2[6][11]=192;
        g2[6][12]=84;g2[6][13]=70;g2[6][14]=131;g2[6][15]=17;
        g2[7][7]=179;g2[7][8]=247;g2[7][9]=255;g2[7][10]=252;g2[7][11]=226;
        g2[7][12]=121;g2[7][13]=243;g2[7][14]=160;g2[7][15]=115;
        g2[8][8]=161;g2[8][9]=171;g2[8][10]=129;g2[8][11]=91;
        g2[8][12]=199;g2[8][13]=236;g2[8][14]=128;g2[8][15]=37;
        g2[9][9]=2;g2[9][10]=201;g2[9][11]=128;g2[9][12]=17;g2[9][13]=216;
        g2[9][14]=176;g2[9][15]=77;
        g2[10][10]=5;g2[10][11]=142;g2[10][12]=36;g2[10][13]=249;g2[10][14]=164;
        g2[10][15]=187;
        g2[11][11]=12;g2[11][12]=167;g2[11][13]=96;g2[11][14]=36;g2[11][15]=215;
        g2[12][12]=28;g2[12][13]=156;g2[12][14]=38;g2[12][15]=185;
        g2[13][13]=38;g2[13][14]=217;g2[13][15]=227;
        g2[14][14]=67;g2[14][15]=119; g2[15][15]=137;
        if ((L_effective>0)&(L_effective<16))
        {
                for (i=0;i<=15;i++)
                {
                        Converts(KC[i]);
                        for (j=0;j<=7;j++) POL1[8*i+7-j]=binary[63-j];
                }
```

```
                for (i=0;i<=128;i++) if (POL1[i]==1)
                        {
                                grau_POL1=127-i;
                                i=129;
                        }
                for (i=0;i<=128;i++) POL1[i]=POL1[i+127-grau_POL1];
                for (i=0;i<=15;i++)
                {
                        Converts(g1[L_effective][i]);
                        for (j=0;j<=7;j++) POL2[8*i+7-j]=binary[63-j];
                }
                for (i=0;i<=128;i++) if (POL2[i]==1)
                        {
                                grau_POL2=127-i;
                                i=129;
                        }
                for (i=0;i<=128;i++) POL2[i]=POL2[i+127-grau_POL2];
                Divides_polynomials();
                for (i=0;i<=15;i++)
                {
                        Converts(g2[L_effective][i]);
                        for (j=0;j<=7;j++) Pmul1[8*i+8-j]=binary[63-j];
                }
                Pmul2[0]=0;
                for (i=0;i<=grau_POL3;i++) Pmul2[128+i-grau_POL3]=POL3[i];
                if (print)
                {
                        printf("\nPmul1=\n");
                        for (i=0;i<=128;i++) printf("%d",Pmul1[i]);
                        printf("\nPmul2=\n");
                        for (i=0;i<=128;i++) printf("%d",Pmul2[i]);
                }
                Multiply_polynomials();
                for (i=1;i<=128;i++) KCl[i-1]=Pmul_res[i+128];
                if (print)
                {
                        printf("\nKCl=\n");
                        for (i=0;i<=127;i++) printf("%d",KCl[i]);
                }
                p=0;
                for (i=0;i<=15;i++)
                {
                        binary_to_decimal(KCl,8*i,8*i+7);
                        KCl_dec[p]=number;
                        p=p+1;
                }
        }
        else for (i=0;i<=15;i++) KCl_dec[i]=KC[i];
        printf("\nKCl_dec=\n");
        for (i=0;i<=15;i++) printf("%d, ",KCl_dec[i]);
        printf("\n");
}

void E0(void)
{
        Kmaster_bool=true;
        E22();
        Kmaster_bool=false;
        KC_bool=true;
        E3();
```

```
	KC_bool=false;
	find_KCl();
	LFSRs();
}

void Start_variables(void)
{
	for (i=0;i<=255;i++)
	{
		Calculate_Bpi(i,257);
		Xtable[i]=x%256;
	}
	printf("Tabela e versus i (exponencial):\n");
	printf("   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
	j=0;
	for (i=0;i<=255;i++)
	{
		if (i%10==0)
		{
			printf("\n");
			if (j<10) printf(" ");
			printf("%d | ",j);
			j=j+1;
		}
		if (Xtable[i]<10) printf(" ");
		if (Xtable[i]<100) printf(" ");
		printf("%d| ",Xtable[i]);
	}
	printf("\n\nTabela j versus l (logaritmo):\n");
	printf("   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |");
	for (i=0;i<=255;i++)
		{
			Ytable[i][0]=Xtable[i];
			Ytable[i][1]=i;
		}
	x_bool=true;
	while (x_bool)
	{
		x_bool=false;
		for (i=0;i<=254;i++)
		if (Ytable[i][0]>Ytable[i+1][0])
		{
			x=Ytable[i][0]; y=Ytable[i][1];
			Ytable[i][0]=Ytable[i+1][0]; Ytable[i][1]=Ytable[i+1][1];
			Ytable[i+1][0]=x; Ytable[i+1][1]=y;
			x_bool=true;
		}
	}
	j=0;
	for (i=0;i<=255;i++)
	{
		if (i%10==0)
		{
			printf("\n");
			if (j<10) printf(" ");
			printf("%d | ",j);
			j=j+1;
		}
		if (Ytable[i][1]<10) printf(" ");
		if (Ytable[i][1]<100) printf(" ");
```

```
            printf("%d| ",Ytable[i][1]);
        }
        BD_ADDRESS_A_dec[0]=255;    BD_ADDRESS_B_dec[0]=255;
        BD_ADDRESS_A_dec[1]=31;     BD_ADDRESS_B_dec[1]=224;
        BD_ADDRESS_A_dec[2]=107;    BD_ADDRESS_B_dec[2]=125;
        BD_ADDRESS_A_dec[3]=46;     BD_ADDRESS_B_dec[3]=244;
        BD_ADDRESS_A_dec[4]=33;     BD_ADDRESS_B_dec[4]=48;
        BD_ADDRESS_A_dec[5]=101;    BD_ADDRESS_B_dec[5]=8;
        E_box(BD_ADDRESS_A_dec,6);
        srand(time(0));
        printf("\n\nRAND(a)= \n");
        for (i=0;i<=15;i++) RAND_a[i]=rand()%256;
        for (i=0;i<=15;i++) printf("%d, ",RAND_a[i]);
        printf("\n\nRAND(b)= \n");
        for (i=0;i<=15;i++) RAND_b[i]=rand()%256;
        for (i=0;i<=15;i++) printf("%d, ",RAND_b[i]);
        printf("\n\nBD Address(A)=");
        for (i=0;i<=5;i++) printf("%d, ",BD_ADDRESS_A_dec[i]);
        printf("\n\nBD Address(B)=");
        for (i=0;i<=5;i++) printf("%d, ",BD_ADDRESS_B_dec[i]);
        printf("\n\nRAND= \n");
        for (i=0;i<=15;i++) RAND[i]=rand()%256;
        for (i=0;i<=15;i++) printf("%d, ",RAND[i]);
        printf("\n");}

namespace Bluetooth {
        using namespace System;
        using namespace System::ComponentModel;
        using namespace System::Collections;
        using namespace System::Windows::Forms;
        using namespace System::Data;
        using namespace System::Drawing;

        public ref class Form1 : public System::Windows::Forms::Form
        {
        public:
            Form1(void) {InitializeComponent();}
        protected:
            ~Form1()
            {
                if (components)
                {
                    delete components;
                }
            }
        protected:
        private: System::Windows::Forms::TextBox^  textBox1;
        private: System::Windows::Forms::Button^  button2;
        private: System::Windows::Forms::RadioButton^  radioButton1;
        private: System::Windows::Forms::RadioButton^  radioButton2;
        private: System::Windows::Forms::TextBox^  textBox2;
        private: System::Windows::Forms::TextBox^  textBox3;
        private: System::Windows::Forms::TextBox^  textBox4;
        private: System::Windows::Forms::TextBox^  textBox5;
        private: System::Windows::Forms::TextBox^  textBox6;
        private: System::Windows::Forms::TextBox^  textBox7;
        private: System::Windows::Forms::TextBox^  textBox8;
        private: System::Windows::Forms::TextBox^  textBox9;
        private: System::Windows::Forms::TextBox^  textBox10;
        private: System::Windows::Forms::TextBox^  textBox11;
```

```
private: System::Windows::Forms::TextBox^  textBox12;
private: System::Windows::Forms::TextBox^  textBox13;
private: System::Windows::Forms::TextBox^  textBox14;
private: System::Windows::Forms::TextBox^  textBox15;
private: System::Windows::Forms::TextBox^  textBox16;
private: System::Windows::Forms::TextBox^  textBox17;
private: System::Windows::Forms::TextBox^  textBox18;
private: System::Windows::Forms::TextBox^  textBox19;
private: System::Windows::Forms::TextBox^  textBox20;
private: System::Windows::Forms::TextBox^  textBox21;
private: System::Windows::Forms::TextBox^  textBox22;
private: System::Windows::Forms::TextBox^  textBox23;
private: System::Windows::Forms::TextBox^  textBox24;
private: System::Windows::Forms::TextBox^  textBox25;
private: System::Windows::Forms::TextBox^  textBox26;
private: System::Windows::Forms::TextBox^  textBox27;
private: System::Windows::Forms::TextBox^  textBox28;
private: System::Windows::Forms::Label^  label1;
private: System::Windows::Forms::Label^  label2;
private: System::Windows::Forms::RadioButton^  radioButton3;
private: System::Windows::Forms::RadioButton^  radioButton4;
private: System::Windows::Forms::RadioButton^  radioButton5;
private: System::Windows::Forms::RadioButton^  radioButton6;
private: System::Windows::Forms::RadioButton^  radioButton7;
private: System::Windows::Forms::RadioButton^  radioButton8;
private: System::Windows::Forms::RadioButton^  radioButton9;
private: System::Windows::Forms::RadioButton^  radioButton10;
private: System::Windows::Forms::RadioButton^  radioButton11;
private: System::Windows::Forms::RadioButton^  radioButton12;
private: System::Windows::Forms::Label^  label3;
private: System::Windows::Forms::TextBox^  textBox29;
private: System::Windows::Forms::TextBox^  textBox30;
private: System::Windows::Forms::TextBox^  textBox31;
private: System::Windows::Forms::TextBox^  textBox32;
private: System::Windows::Forms::TextBox^  textBox33;
private: System::Windows::Forms::TextBox^  textBox34;
private: System::Windows::Forms::TextBox^  textBox35;
private: System::Windows::Forms::TextBox^  textBox36;
private: System::Windows::Forms::TextBox^  textBox37;
private: System::Windows::Forms::TextBox^  textBox38;
private: System::Windows::Forms::TextBox^  textBox39;
private: System::Windows::Forms::TextBox^  textBox40;
private: System::Windows::Forms::TextBox^  textBox41;
private: System::Windows::Forms::TextBox^  textBox42;
private: System::Windows::Forms::TextBox^  textBox43;
private: System::Windows::Forms::TextBox^  textBox44;
private: System::Windows::Forms::RadioButton^  radioButton13;
private: System::Windows::Forms::RadioButton^  radioButton14;
private: System::Windows::Forms::RadioButton^  radioButton15;
private: System::Windows::Forms::RadioButton^  radioButton16;
private: System::Windows::Forms::RadioButton^  radioButton17;
private: System::Windows::Forms::RadioButton^  radioButton18;
private: System::Windows::Forms::Label^  label5;
private: System::Windows::Forms::RadioButton^  radioButton19;
private: System::Windows::Forms::RadioButton^  radioButton20;
private: System::Windows::Forms::RadioButton^  radioButton21;
private: System::Windows::Forms::Button^  button4;
private: System::Windows::Forms::Button^  button5;
private: System::Windows::Forms::Button^  button6;
private: System::Windows::Forms::Button^  button7;
```

```
        private: System::Windows::Forms::Button^  button8;
        private: System::Windows::Forms::Button^  button9;
        private: System::Windows::Forms::Button^  button10;
        private: System::Windows::Forms::Label^  label6;
        private: System::Windows::Forms::Button^  button15;
        private: System::Windows::Forms::TextBox^  textBox45;
        private: System::Windows::Forms::TextBox^  textBox46;
        private: System::Windows::Forms::Label^  label7;
        private: System::Windows::Forms::Label^  label8;
        private: System::Windows::Forms::Label^  label9;
        private: System::Windows::Forms::Button^  button3;
        private: System::Windows::Forms::Label^  label10;
        private: System::Windows::Forms::RadioButton^  radioButton22;
        private: System::Windows::Forms::Button^  button1;
        private: System::Windows::Forms::Button^  button11;
        private: System::Windows::Forms::Button^  button12;
        private: System::ComponentModel::IContainer^  components;
        private:

#pragma region Windows Form Designer generated code
                void InitializeComponent(void){
                        this->textBox1 = (gcnew System::Windows::Forms::TextBox());
                        this->button2 = (gcnew System::Windows::Forms::Button());
                        this->radioButton1 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton2 = (gcnew System::Windows::Forms::RadioButton());
                        this->textBox2 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox3 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox4 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox5 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox6 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox7 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox8 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox9 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox10 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox11 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox12 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox13 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox14 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox15 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox16 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox17 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox18 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox19 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox20 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox21 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox22 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox23 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox24 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox25 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox26 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox27 = (gcnew System::Windows::Forms::TextBox());
                        this->textBox28 = (gcnew System::Windows::Forms::TextBox());
                        this->label1 = (gcnew System::Windows::Forms::Label());
                        this->label2 = (gcnew System::Windows::Forms::Label());
                        this->radioButton3 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton4 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton5 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton6 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton7 = (gcnew System::Windows::Forms::RadioButton());
                        this->radioButton8 = (gcnew System::Windows::Forms::RadioButton());
```

```
this->radioButton9 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton10 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton11 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton12 = (gcnew System::Windows::Forms::RadioButton());
this->label3 = (gcnew System::Windows::Forms::Label());
this->textBox29 = (gcnew System::Windows::Forms::TextBox());
this->textBox30 = (gcnew System::Windows::Forms::TextBox());
this->textBox31 = (gcnew System::Windows::Forms::TextBox());
this->textBox32 = (gcnew System::Windows::Forms::TextBox());
this->textBox33 = (gcnew System::Windows::Forms::TextBox());
this->textBox34 = (gcnew System::Windows::Forms::TextBox());
this->textBox35 = (gcnew System::Windows::Forms::TextBox());
this->textBox36 = (gcnew System::Windows::Forms::TextBox());
this->textBox37 = (gcnew System::Windows::Forms::TextBox());
this->textBox38 = (gcnew System::Windows::Forms::TextBox());
this->textBox39 = (gcnew System::Windows::Forms::TextBox());
this->textBox40 = (gcnew System::Windows::Forms::TextBox());
this->textBox41 = (gcnew System::Windows::Forms::TextBox());
this->textBox42 = (gcnew System::Windows::Forms::TextBox());
this->textBox43 = (gcnew System::Windows::Forms::TextBox());
this->textBox44 = (gcnew System::Windows::Forms::TextBox());
this->radioButton13 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton14 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton15 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton16 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton17 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton18 = (gcnew System::Windows::Forms::RadioButton());
this->label5 = (gcnew System::Windows::Forms::Label());
this->radioButton19 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton20 = (gcnew System::Windows::Forms::RadioButton());
this->radioButton21 = (gcnew System::Windows::Forms::RadioButton());
this->button4 = (gcnew System::Windows::Forms::Button());
this->button5 = (gcnew System::Windows::Forms::Button());
this->button6 = (gcnew System::Windows::Forms::Button());
this->button7 = (gcnew System::Windows::Forms::Button());
this->button8 = (gcnew System::Windows::Forms::Button());
this->button9 = (gcnew System::Windows::Forms::Button());
this->button10 = (gcnew System::Windows::Forms::Button());
this->label6 = (gcnew System::Windows::Forms::Label());
this->button15 = (gcnew System::Windows::Forms::Button());
this->textBox45 = (gcnew System::Windows::Forms::TextBox());
this->textBox46 = (gcnew System::Windows::Forms::TextBox());
this->label7 = (gcnew System::Windows::Forms::Label());
this->label8 = (gcnew System::Windows::Forms::Label());
this->label9 = (gcnew System::Windows::Forms::Label());
this->button3 = (gcnew System::Windows::Forms::Button());
this->label10 = (gcnew System::Windows::Forms::Label());
this->radioButton22 = (gcnew System::Windows::Forms::RadioButton());
this->button1 = (gcnew System::Windows::Forms::Button());
this->button11 = (gcnew System::Windows::Forms::Button());
this->button12 = (gcnew System::Windows::Forms::Button());
this->SuspendLayout();
this->textBox1->Location = System::Drawing::Point(33, 268);
this->textBox1->Name = L"textBox1";
this->textBox1->Size = System::Drawing::Size(25, 20);
this->textBox1->TabIndex = 1;
this->textBox1->TextChanged +=
gcnew System::EventHandler(this, &Form1::textBox1_TextChanged);
this->button2->Location = System::Drawing::Point(353, 138);
this->button2->Name = L"button2";
```

```
this->button2->Size = System::Drawing::Size(75, 37);
this->button2->TabIndex = 2;
this->button2->Text = L" Mostrar\r\nEndereços";
this->button2->UseVisualStyleBackColor = true;
this->button2->Click +=
gcnew System::EventHandler(this, &Form1::button2_Click);
this->radioButton1->AutoSize = true;
this->radioButton1->Location = System::Drawing::Point(33, 334);
this->radioButton1->Name = L"radioButton1";
this->radioButton1->Size = System::Drawing::Size(72, 17);
this->radioButton1->TabIndex = 3;
this->radioButton1->TabStop = true;
this->radioButton1->Text = L"RAND (A)";
this->radioButton1->UseVisualStyleBackColor = true;
this->radioButton1->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton1_CheckedChanged);
this->radioButton2->AutoSize = true;
this->radioButton2->Location = System::Drawing::Point(33, 357);
this->radioButton2->Name = L"radioButton2";
this->radioButton2->Size = System::Drawing::Size(72, 17);
this->radioButton2->TabIndex = 4;
this->radioButton2->TabStop = true;
this->radioButton2->Text = L"RAND (B)";
this->radioButton2->UseVisualStyleBackColor = true;
this->radioButton2->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton2_CheckedChanged);
this->textBox2->Location = System::Drawing::Point(64, 268);
this->textBox2->Name = L"textBox2";
this->textBox2->Size = System::Drawing::Size(25, 20);
this->textBox2->TabIndex = 5;
this->textBox2->TextChanged +=
gcnew System::EventHandler(this, &Form1::textBox2_TextChanged);
this->textBox3->Location = System::Drawing::Point(94, 268);
this->textBox3->Name = L"textBox3";
this->textBox3->Size = System::Drawing::Size(26, 20);
this->textBox3->TabIndex = 6;
this->textBox4->Location = System::Drawing::Point(126, 268);
this->textBox4->Name = L"textBox4";
this->textBox4->Size = System::Drawing::Size(25, 20);
this->textBox4->TabIndex = 7;
this->textBox5->Location = System::Drawing::Point(157, 268);
this->textBox5->Name = L"textBox5";
this->textBox5->Size = System::Drawing::Size(25, 20);
this->textBox5->TabIndex = 8;
this->textBox6->Location = System::Drawing::Point(188, 268);
this->textBox6->Name = L"textBox6";
this->textBox6->Size = System::Drawing::Size(25, 20);
this->textBox6->TabIndex = 9;
this->textBox7->Location = System::Drawing::Point(219, 268);
this->textBox7->Name = L"textBox7";
this->textBox7->Size = System::Drawing::Size(25, 20);
this->textBox7->TabIndex = 10;
this->textBox8->Location = System::Drawing::Point(250, 268);
this->textBox8->Name = L"textBox8";
this->textBox8->Size = System::Drawing::Size(25, 20);
this->textBox8->TabIndex = 11;
this->textBox9->Location = System::Drawing::Point(281, 268);
this->textBox9->Name = L"textBox9";
this->textBox9->Size = System::Drawing::Size(25, 20);
this->textBox9->TabIndex = 17;
```

```
this->textBox10->Location = System::Drawing::Point(312, 268);
this->textBox10->Name = L"textBox10";
this->textBox10->Size = System::Drawing::Size(25, 20);
this->textBox10->TabIndex = 16;
this->textBox11->Location = System::Drawing::Point(343, 268);
this->textBox11->Name = L"textBox11";
this->textBox11->Size = System::Drawing::Size(25, 20);
this->textBox11->TabIndex = 15;
this->textBox12->Location = System::Drawing::Point(374, 268);
this->textBox12->Name = L"textBox12";
this->textBox12->Size = System::Drawing::Size(25, 20);
this->textBox12->TabIndex = 14;
this->textBox13->Location = System::Drawing::Point(405, 268);
this->textBox13->Name = L"textBox13";
this->textBox13->Size = System::Drawing::Size(25, 20);
this->textBox13->TabIndex = 13;
this->textBox14->Location = System::Drawing::Point(436, 268);
this->textBox14->Name = L"textBox14";
this->textBox14->Size = System::Drawing::Size(25, 20);
this->textBox14->TabIndex = 12;
this->textBox15->Location = System::Drawing::Point(467, 268);
this->textBox15->Name = L"textBox15";
this->textBox15->Size = System::Drawing::Size(25, 20);
this->textBox15->TabIndex = 21;
this->textBox16->Location = System::Drawing::Point(498, 268);
this->textBox16->Name = L"textBox16";
this->textBox16->Size = System::Drawing::Size(25, 20);
this->textBox16->TabIndex = 20;
this->textBox17->Location = System::Drawing::Point(518, 134);
this->textBox17->Name = L"textBox17";
this->textBox17->Size = System::Drawing::Size(25, 20);
this->textBox17->TabIndex = 27;
this->textBox18->Location = System::Drawing::Point(549, 134);
this->textBox18->Name = L"textBox18";
this->textBox18->Size = System::Drawing::Size(25, 20);
this->textBox18->TabIndex = 26;
this->textBox19->Location = System::Drawing::Point(580, 134);
this->textBox19->Name = L"textBox19";
this->textBox19->Size = System::Drawing::Size(25, 20);
this->textBox19->TabIndex = 25;
this->textBox20->Location = System::Drawing::Point(611, 134);
this->textBox20->Name = L"textBox20";
this->textBox20->Size = System::Drawing::Size(25, 20);
this->textBox20->TabIndex = 24;
this->textBox21->Location = System::Drawing::Point(642, 134);
this->textBox21->Name = L"textBox21";
this->textBox21->Size = System::Drawing::Size(25, 20);
this->textBox21->TabIndex = 23;
this->textBox22->Location = System::Drawing::Point(673, 134);
this->textBox22->Name = L"textBox22";
this->textBox22->Size = System::Drawing::Size(25, 20);
this->textBox22->TabIndex = 22;
this->textBox23->Location = System::Drawing::Point(518, 160);
this->textBox23->Name = L"textBox23";
this->textBox23->Size = System::Drawing::Size(25, 20);
this->textBox23->TabIndex = 33;
this->textBox24->Location = System::Drawing::Point(549, 160);
this->textBox24->Name = L"textBox24";
this->textBox24->Size = System::Drawing::Size(25, 20);
this->textBox24->TabIndex = 32;
```

```
this->textBox25->Location = System::Drawing::Point(580, 160);
this->textBox25->Name = L"textBox25";
this->textBox25->Size = System::Drawing::Size(25, 20);
this->textBox25->TabIndex = 31;
this->textBox26->Location = System::Drawing::Point(611, 160);
this->textBox26->Name = L"textBox26";
this->textBox26->Size = System::Drawing::Size(25, 20);
this->textBox26->TabIndex = 30;
this->textBox27->Location = System::Drawing::Point(642, 160);
this->textBox27->Name = L"textBox27";
this->textBox27->Size = System::Drawing::Size(25, 20);
this->textBox27->TabIndex = 29;
this->textBox28->Location = System::Drawing::Point(673, 160);
this->textBox28->Name = L"textBox28";
this->textBox28->Size = System::Drawing::Size(25, 20);
this->textBox28->TabIndex = 28;
this->label1->AutoSize = true;
this->label1->Location = System::Drawing::Point(434, 138);
this->label1->Name = L"label1";
this->label1->Size = System::Drawing::Size(78, 13);
this->label1->TabIndex = 34;
this->label1->Text = L"BD address (A)";
this->label1->Click +=
gcnew System::EventHandler(this, &Form1::label1_Click);
this->label2->AutoSize = true;
this->label2->Location = System::Drawing::Point(434, 162);
this->label2->Name = L"label2";
this->label2->Size = System::Drawing::Size(78, 13);
this->label2->TabIndex = 35;
this->label2->Text = L"BD address (B)";
this->radioButton3->AutoSize = true;
this->radioButton3->Location = System::Drawing::Point(33, 380);
this->radioButton3->Name = L"radioButton3";
this->radioButton3->Size = System::Drawing::Size(43, 17);
this->radioButton3->TabIndex = 36;
this->radioButton3->TabStop = true;
this->radioButton3->Text = L"PIN";
this->radioButton3->UseVisualStyleBackColor = true;
this->radioButton3->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton3_CheckedChanged);
this->radioButton4->AutoSize = true;
this->radioButton4->Location = System::Drawing::Point(33, 403);
this->radioButton4->Name = L"radioButton4";
this->radioButton4->Size = System::Drawing::Size(45, 17);
this->radioButton4->TabIndex = 37;
this->radioButton4->TabStop = true;
this->radioButton4->Text = L"PIN\"";
this->radioButton4->UseVisualStyleBackColor = true;
this->radioButton4->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton4_CheckedChanged);
this->radioButton5->AutoSize = true;
this->radioButton5->Location = System::Drawing::Point(33, 426);
this->radioButton5->Name = L"radioButton5";
this->radioButton5->Size = System::Drawing::Size(45, 17);
this->radioButton5->TabIndex = 38;
this->radioButton5->TabStop = true;
this->radioButton5->Text = L"Kinit";
this->radioButton5->UseVisualStyleBackColor = true;
this->radioButton5->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton5_CheckedChanged);
```

```
this->radioButton6->AutoSize = true;
this->radioButton6->Location = System::Drawing::Point(33, 449);
this->radioButton6->Name = L"radioButton6";
this->radioButton6->Size = System::Drawing::Size(57, 17);
this->radioButton6->TabIndex = 39;
this->radioButton6->TabStop = true;
this->radioButton6->Text = L"LK_Ka";
this->radioButton6->UseVisualStyleBackColor = true;
this->radioButton6->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton6_CheckedChanged);
this->radioButton7->AutoSize = true;
this->radioButton7->Location = System::Drawing::Point(125, 334);
this->radioButton7->Name = L"radioButton7";
this->radioButton7->Size = System::Drawing::Size(57, 17);
this->radioButton7->TabIndex = 40;
this->radioButton7->TabStop = true;
this->radioButton7->Text = L"LK_Kb";
this->radioButton7->UseVisualStyleBackColor = true;
this->radioButton7->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton7_CheckedChanged);
this->radioButton8->AutoSize = true;
this->radioButton8->Location = System::Drawing::Point(126, 357);
this->radioButton8->Name = L"radioButton8";
this->radioButton8->Size = System::Drawing::Size(62, 17);
this->radioButton8->TabIndex = 41;
this->radioButton8->TabStop = true;
this->radioButton8->Text = L"RAND1";
this->radioButton8->UseVisualStyleBackColor = true;
this->radioButton8->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton8_CheckedChanged);
this->radioButton9->AutoSize = true;
this->radioButton9->Location = System::Drawing::Point(126, 380);
this->radioButton9->Name = L"radioButton9";
this->radioButton9->Size = System::Drawing::Size(62, 17);
this->radioButton9->TabIndex = 42;
this->radioButton9->TabStop = true;
this->radioButton9->Text = L"RAND2";
this->radioButton9->UseVisualStyleBackColor = true;
this->radioButton9->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton9_CheckedChanged);
this->radioButton10->AutoSize = true;
this->radioButton10->Location = System::Drawing::Point(219, 357);
this->radioButton10->Name = L"radioButton10";
this->radioButton10->Size = System::Drawing::Size(63, 17);
this->radioButton10->TabIndex = 43;
this->radioButton10->TabStop = true;
this->radioButton10->Text = L"Kmaster";
this->radioButton10->UseVisualStyleBackColor = true;
this->radioButton10->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton10_CheckedChanged);
this->radioButton11->AutoSize = true;
this->radioButton11->Location = System::Drawing::Point(125, 403);
this->radioButton11->Name = L"radioButton11";
this->radioButton11->Size = System::Drawing::Size(44, 17);
this->radioButton11->TabIndex = 44;
this->radioButton11->TabStop = true;
this->radioButton11->Text = L"Kab";
this->radioButton11->UseVisualStyleBackColor = true;
this->radioButton11->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton11_CheckedChanged);
```

```
this->radioButton12->AutoSize = true;
this->radioButton12->Location = System::Drawing::Point(219, 426);
this->radioButton12->Name = L"radioButton12";
this->radioButton12->Size = System::Drawing::Size(39, 17);
this->radioButton12->TabIndex = 45;
this->radioButton12->TabStop = true;
this->radioButton12->Text = L"KC";
this->radioButton12->UseVisualStyleBackColor = true;
this->radioButton12->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton12_CheckedChanged);
this->label3->AutoSize = true;
this->label3->Location = System::Drawing::Point(30, 252);
this->label3->Name = L"label3";
this->label3->Size = System::Drawing::Size(91, 13);
this->label3->TabIndex = 46;
this->label3->Text = L"Saída de Dados :";
this->label3->Click +=
gcnew System::EventHandler(this, &Form1::label3_Click);
this->textBox29->Location = System::Drawing::Point(376, 100);
this->textBox29->Name = L"textBox29";
this->textBox29->Size = System::Drawing::Size(13, 20);
this->textBox29->TabIndex = 62;
this->textBox30->Location = System::Drawing::Point(392, 100);
this->textBox30->Name = L"textBox30";
this->textBox30->Size = System::Drawing::Size(13, 20);
this->textBox30->TabIndex = 61;
this->textBox31->Location = System::Drawing::Point(408, 100);
this->textBox31->Name = L"textBox31";
this->textBox31->Size = System::Drawing::Size(13, 20);
this->textBox31->TabIndex = 60;
this->textBox32->Location = System::Drawing::Point(424, 100);
this->textBox32->Name = L"textBox32";
this->textBox32->Size = System::Drawing::Size(13, 20);
this->textBox32->TabIndex = 59;
this->textBox33->Location = System::Drawing::Point(440, 100);
this->textBox33->Name = L"textBox33";
this->textBox33->Size = System::Drawing::Size(13, 20);
this->textBox33->TabIndex = 58;
this->textBox34->Location = System::Drawing::Point(456, 100);
this->textBox34->Name = L"textBox34";
this->textBox34->Size = System::Drawing::Size(13, 20);
this->textBox34->TabIndex = 57;
this->textBox35->Location = System::Drawing::Point(472, 100);
this->textBox35->Name = L"textBox35";
this->textBox35->Size = System::Drawing::Size(13, 20);
this->textBox35->TabIndex = 56;
this->textBox36->Location = System::Drawing::Point(488, 100);
this->textBox36->Name = L"textBox36";
this->textBox36->Size = System::Drawing::Size(13, 20);
this->textBox36->TabIndex = 55;
this->textBox37->Location = System::Drawing::Point(504, 100);
this->textBox37->Name = L"textBox37";
this->textBox37->Size = System::Drawing::Size(13, 20);
this->textBox37->TabIndex = 54;
this->textBox38->Location = System::Drawing::Point(520, 100);
this->textBox38->Name = L"textBox38";
this->textBox38->Size = System::Drawing::Size(13, 20);
this->textBox38->TabIndex = 53;
this->textBox39->Location = System::Drawing::Point(536, 100);
this->textBox39->Name = L"textBox39";
```

```
this->textBox39->Size = System::Drawing::Size(13, 20);
this->textBox39->TabIndex = 52;
this->textBox40->Location = System::Drawing::Point(552, 100);
this->textBox40->Name = L"textBox40";
this->textBox40->Size = System::Drawing::Size(13, 20);
this->textBox40->TabIndex = 51;
this->textBox41->Location = System::Drawing::Point(568, 100);
this->textBox41->Name = L"textBox41";
this->textBox41->Size = System::Drawing::Size(13, 20);
this->textBox41->TabIndex = 50;
this->textBox42->Location = System::Drawing::Point(584, 100);
this->textBox42->Name = L"textBox42";
this->textBox42->Size = System::Drawing::Size(13, 20);
this->textBox42->TabIndex = 49;
this->textBox43->Location = System::Drawing::Point(600, 100);
this->textBox43->Name = L"textBox43";
this->textBox43->Size = System::Drawing::Size(13, 20);
this->textBox43->TabIndex = 48;
this->textBox44->Location = System::Drawing::Point(616, 100);
this->textBox44->Name = L"textBox44";
this->textBox44->Size = System::Drawing::Size(13, 20);
this->textBox44->TabIndex = 47;
this->radioButton13->AutoSize = true;
this->radioButton13->Location = System::Drawing::Point(219, 449);
this->radioButton13->Name = L"radioButton13";
this->radioButton13->Size = System::Drawing::Size(41, 17);
this->radioButton13->TabIndex = 64;
this->radioButton13->TabStop = true;
this->radioButton13->Text = L"KC\'";
this->radioButton13->UseVisualStyleBackColor = true;
this->radioButton13->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton13_CheckedChanged);
this->radioButton14->AutoSize = true;
this->radioButton14->Location = System::Drawing::Point(125, 449);
this->radioButton14->Name = L"radioButton14";
this->radioButton14->Size = System::Drawing::Size(54, 17);
this->radioButton14->TabIndex = 65;
this->radioButton14->TabStop = true;
this->radioButton14->Text = L"SRES";
this->radioButton14->UseVisualStyleBackColor = true;
this->radioButton14->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton14_CheckedChanged);
this->radioButton15->AutoSize = true;
this->radioButton15->Location = System::Drawing::Point(219, 334);
this->radioButton15->Name = L"radioButton15";
this->radioButton15->Size = System::Drawing::Size(47, 17);
this->radioButton15->TabIndex = 66;
this->radioButton15->TabStop = true;
this->radioButton15->Text = L"ACO";
this->radioButton15->UseVisualStyleBackColor = true;
this->radioButton15->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton15_CheckedChanged);
this->radioButton16->AutoSize = true;
this->radioButton16->Location = System::Drawing::Point(219, 380);
this->radioButton16->Name = L"radioButton16";
this->radioButton16->Size = System::Drawing::Size(77, 17);
this->radioButton16->TabIndex = 67;
this->radioButton16->TabStop = true;
this->radioButton16->Text = L"EN_RAND";
this->radioButton16->UseVisualStyleBackColor = true;
```

```
this->radioButton16->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton16_CheckedChanged);
this->radioButton17->AutoSize = true;
this->radioButton17->Location = System::Drawing::Point(219, 403);
this->radioButton17->Name = L"radioButton17";
this->radioButton17->Size = System::Drawing::Size(46, 17);
this->radioButton17->TabIndex = 68;
this->radioButton17->TabStop = true;
this->radioButton17->Text = L"COF";
this->radioButton17->UseVisualStyleBackColor = true;
this->radioButton17->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton17_CheckedChanged);
this->radioButton18->AutoSize = true;
this->radioButton18->Location = System::Drawing::Point(126, 426);
this->radioButton18->Name = L"radioButton18";
this->radioButton18->Size = System::Drawing::Size(61, 17);
this->radioButton18->TabIndex = 69;
this->radioButton18->TabStop = true;
this->radioButton18->Text = L"K offset";
this->radioButton18->UseVisualStyleBackColor = true;
this->radioButton18->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton18_CheckedChanged);
this->label5->AutoSize = true;
this->label5->Location = System::Drawing::Point(30, 307);
this->label5->Name = L"label5";
this->label5->Size = System::Drawing::Size(149, 13);
this->label5->TabIndex = 70;
this->label5->Text = L"Escolha uma opção de saída:";
this->radioButton19->AutoSize = true;
this->radioButton19->Location = System::Drawing::Point(309, 357);
this->radioButton19->Name = L"radioButton19";
this->radioButton19->Size = System::Drawing::Size(78, 17);
this->radioButton19->TabIndex = 71;
this->radioButton19->TabStop = true;
this->radioButton19->Text = L"Texto claro";
this->radioButton19->UseVisualStyleBackColor = true;
this->radioButton19->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton19_CheckedChanged);
this->radioButton20->AutoSize = true;
this->radioButton20->Location = System::Drawing::Point(309, 380);
this->radioButton20->Name = L"radioButton20";
this->radioButton20->Size = System::Drawing::Size(87, 17);
this->radioButton20->TabIndex = 72;
this->radioButton20->TabStop = true;
this->radioButton20->Text = L"Texto cifrado";
this->radioButton20->UseVisualStyleBackColor = true;
this->radioButton20->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton20_CheckedChanged);
this->radioButton21->AutoSize = true;
this->radioButton21->Location = System::Drawing::Point(309, 403);
this->radioButton21->Name = L"radioButton21";
this->radioButton21->Size = System::Drawing::Size(99, 17);
this->radioButton21->TabIndex = 73;
this->radioButton21->TabStop = true;
this->radioButton21->Text = L"Texto decifrado";
this->radioButton21->UseVisualStyleBackColor = true;
this->radioButton21->CheckedChanged +=
gcnew System::EventHandler(this, &Form1::radioButton21_CheckedChanged);
this->button4->Location = System::Drawing::Point(33, 91);
this->button4->Name = L"button4";
```

```
this->button4->Size = System::Drawing::Size(75, 37);
this->button4->TabIndex = 75;
this->button4->Text = L"Dados\r\nObrigatórios";
this->button4->UseVisualStyleBackColor = true;
this->button4->Click +=
gcnew System::EventHandler(this, &Form1::button4_Click);
this->button5->Location = System::Drawing::Point(33, 134);
this->button5->Name = L"button5";
this->button5->Size = System::Drawing::Size(75, 37);
this->button5->TabIndex = 76;
this->button5->Text = L"Cálculo\r\nKinit";
this->button5->UseVisualStyleBackColor = true;
this->button5->Click +=
gcnew System::EventHandler(this, &Form1::button5_Click);
this->button6->Location = System::Drawing::Point(33, 177);
this->button6->Name = L"button6";
this->button6->Size = System::Drawing::Size(75, 37);
this->button6->TabIndex = 77;
this->button6->Text = L"Cálculo LK\r\nKa e LK Kb";
this->button6->UseVisualStyleBackColor = true;
this->button6->Click +=
gcnew System::EventHandler(this, &Form1::button6_Click);
this->button7->Location = System::Drawing::Point(114, 48);
this->button7->Name = L"button7";
this->button7->Size = System::Drawing::Size(75, 37);
this->button7->TabIndex = 78;
this->button7->Text = L"RAND 1\r\nRAND 2";
this->button7->UseVisualStyleBackColor = true;
this->button7->Click +=
gcnew System::EventHandler(this, &Form1::button7_Click);
this->button8->Location = System::Drawing::Point(114, 91);
this->button8->Name = L"button8";
this->button8->Size = System::Drawing::Size(75, 37);
this->button8->TabIndex = 79;
this->button8->Text = L"Kab";
this->button8->UseVisualStyleBackColor = true;
this->button8->Click +=
gcnew System::EventHandler(this, &Form1::button8_Click);
this->button9->Location = System::Drawing::Point(114, 134);
this->button9->Name = L"button9";
this->button9->Size = System::Drawing::Size(75, 37);
this->button9->TabIndex = 80;
this->button9->Text = L"SRES\r\nACO";
this->button9->UseVisualStyleBackColor = true;
this->button9->Click +=
gcnew System::EventHandler(this, &Form1::button9_Click);
this->button10->Location = System::Drawing::Point(114, 177);
this->button10->Name = L"button10";
this->button10->Size = System::Drawing::Size(75, 37);
this->button10->TabIndex = 81;
this->button10->Text = L"E0";
this->button10->UseVisualStyleBackColor = true;
this->button10->Click +=
gcnew System::EventHandler(this, &Form1::button10_Click);
this->label6->AutoSize = true;
this->label6->Location = System::Drawing::Point(59, 20);
this->label6->Name = L"label6";
this->label6->Size = System::Drawing::Size(98, 13);
this->label6->TabIndex = 86;
this->label6->Text = L"Botões de Cálculos";
```

```
this->label6->Click +=
gcnew System::EventHandler(this, &Form1::label6_Click);
this->button15->Location = System::Drawing::Point(33, 48);
this->button15->Name = L"button15";
this->button15->Size = System::Drawing::Size(75, 37);
this->button15->TabIndex = 87;
this->button15->Text = L"Iniciar\r\nvariáveis";
this->button15->UseVisualStyleBackColor = true;
this->button15->Click +=
gcnew System::EventHandler(this, &Form1::button15_Click);
this->textBox45->Location = System::Drawing::Point(345, 48);
this->textBox45->Name = L"textBox45";
this->textBox45->Size = System::Drawing::Size(170, 20);
this->textBox45->TabIndex = 88;
this->textBox45->TextChanged +=
gcnew System::EventHandler(this, &Form1::textBox45_TextChanged);
this->textBox46->Location = System::Drawing::Point(489, 74);
this->textBox46->Name = L"textBox46";
this->textBox46->Size = System::Drawing::Size(26, 20);
this->textBox46->TabIndex = 89;
this->label7->AutoSize = true;
this->label7->Location = System::Drawing::Point(311, 51);
this->label7->Name = L"label7";
this->label7->Size = System::Drawing::Size(28, 13);
this->label7->TabIndex = 90;
this->label7->Text = L"PIN:";
this->label8->AutoSize = true;
this->label8->Location = System::Drawing::Point(311, 77);
this->label8->Name = L"label8";
this->label8->Size = System::Drawing::Size(171, 13);
this->label8->TabIndex = 91;
this->label8->Text = L"Comprimento efetivo da chave KC:";
this->label9->AutoSize = true;
this->label9->Location = System::Drawing::Point(373, 20);
this->label9->Name = L"label9";
this->label9->Size = System::Drawing::Size(98, 13);
this->label9->TabIndex = 92;
this->label9->Text = L"Dados obrigatórios:";
this->label9->Click +=
gcnew System::EventHandler(this, &Form1::label9_Click);
this->button3->Location = System::Drawing::Point(195, 48);
this->button3->Name = L"button3";
this->button3->Size = System::Drawing::Size(75, 37);
this->button3->TabIndex = 93;
this->button3->Text = L"Z ( t)";
this->button3->UseVisualStyleBackColor = true;
this->button3->Click +=
gcnew System::EventHandler(this, &Form1::button3_Click);
this->label10->AutoSize = true;
this->label10->Location = System::Drawing::Point(311, 103);
this->label10->Name = L"label10";
this->label10->Size = System::Drawing::Size(64, 13);
this->label10->TabIndex = 94;
this->label10->Text = L"Texto Claro:";
this->radioButton22->AutoSize = true;
this->radioButton22->Location = System::Drawing::Point(309, 334);
this->radioButton22->Name = L"radioButton22";
this->radioButton22->Size = System::Drawing::Size(44, 17);
this->radioButton22->TabIndex = 96;
this->radioButton22->TabStop = true;
```

```
            this->radioButton22->Text = L"Z( t)";
            this->radioButton22->UseVisualStyleBackColor = true;
            this->radioButton22->CheckedChanged +=
            gcnew System::EventHandler(this, &Form1::radioButton22_CheckedChanged);
            this->button1->Location = System::Drawing::Point(536, 48);
            this->button1->Name = L"button1";
            this->button1->Size = System::Drawing::Size(75, 37);
            this->button1->TabIndex = 97;
            this->button1->Text = L"Exemplo\r\nPadrão";
            this->button1->UseVisualStyleBackColor = true;
            this->button1->Click +=
            gcnew System::EventHandler(this, &Form1::button1_Click_1);
            this->button11->BackColor = System::Drawing::Color::Red;
            this->button11->ForeColor = System::Drawing::Color::White;
            this->button11->Location = System::Drawing::Point(353, 193);
            this->button11->Name = L"button11";
            this->button11->Size = System::Drawing::Size(118, 37);
            this->button11->TabIndex = 98;
            this->button11->Text = L"Habilitar todas\r\nas saídas\?";
            this->button11->UseVisualStyleBackColor = false;
            this->button11->Click +=
            gcnew System::EventHandler(this, &Form1::button11_Click);
            this->button12->BackColor = System::Drawing::Color::Green;
            this->button12->ForeColor = System::Drawing::Color::White;
            this->button12->Location = System::Drawing::Point(477, 193);
            this->button12->Name = L"button12";
            this->button12->Size = System::Drawing::Size(118, 37);
            this->button12->TabIndex = 99;
            this->button12->Text = L"Habilitar saídas\r\npadrão\?";
            this->button12->UseVisualStyleBackColor = false;
            this->button12->Click +=
            gcnew System::EventHandler(this, &Form1::button12_Click);
            this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
            this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
            this->ClientSize = System::Drawing::Size(728, 515);
this->Controls->Add(this->button12);this->Controls->Add(this->button11);
this->Controls->Add(this->button1);this->Controls->Add(this->radioButton22);
this->Controls->Add(this->label10);this->Controls->Add(this->button3);
this->Controls->Add(this->label9);this->Controls->Add(this->label8);
this->Controls->Add(this->label7);this->Controls->Add(this->textBox46);
this->Controls->Add(this->textBox45);this->Controls->Add(this->button15);
this->Controls->Add(this->label6);this->Controls->Add(this->button10);
this->Controls->Add(this->button9);this->Controls->Add(this->button8);
this->Controls->Add(this->button7);this->Controls->Add(this->button6);
this->Controls->Add(this->button5);this->Controls->Add(this->button4);
this->Controls->Add(this->radioButton21);
this->Controls->Add(this->radioButton20);
this->Controls->Add(this->radioButton19);this->Controls->Add(this->label5);
this->Controls->Add(this->radioButton18);
this->Controls->Add(this->radioButton17);
this->Controls->Add(this->radioButton16);
this->Controls->Add(this->radioButton15);
this->Controls->Add(this->radioButton14);
this->Controls->Add(this->radioButton13);
this->Controls->Add(this->textBox29);this->Controls->Add(this->textBox30);
this->Controls->Add(this->textBox31);this->Controls->Add(this->textBox32);
this->Controls->Add(this->textBox33);this->Controls->Add(this->textBox34);
this->Controls->Add(this->textBox35);this->Controls->Add(this->textBox36);
this->Controls->Add(this->textBox37);this->Controls->Add(this->textBox38);
this->Controls->Add(this->textBox39);this->Controls->Add(this->textBox40);
```

```
this->Controls->Add(this->textBox41);this->Controls->Add(this->textBox42);
this->Controls->Add(this->textBox43);this->Controls->Add(this->textBox44);
this->Controls->Add(this->label3);this->Controls->Add(this->radioButton12);
this->Controls->Add(this->radioButton11);
this->Controls->Add(this->radioButton10);
this->Controls->Add(this->radioButton9);
this->Controls->Add(this->radioButton8);
this->Controls->Add(this->radioButton7);
this->Controls->Add(this->radioButton6);
this->Controls->Add(this->radioButton5);
this->Controls->Add(this->radioButton4);
this->Controls->Add(this->radioButton3);
this->Controls->Add(this->label2);this->Controls->Add(this->label1);
this->Controls->Add(this->textBox23);this->Controls->Add(this->textBox24);
this->Controls->Add(this->textBox25);this->Controls->Add(this->textBox26);
this->Controls->Add(this->textBox27);this->Controls->Add(this->textBox28);
this->Controls->Add(this->textBox17);this->Controls->Add(this->textBox18);
this->Controls->Add(this->textBox19);this->Controls->Add(this->textBox20);
this->Controls->Add(this->textBox21);this->Controls->Add(this->textBox22);
this->Controls->Add(this->textBox15);this->Controls->Add(this->textBox16);
this->Controls->Add(this->textBox9);this->Controls->Add(this->textBox10);
this->Controls->Add(this->textBox11);this->Controls->Add(this->textBox12);
this->Controls->Add(this->textBox13);this->Controls->Add(this->textBox14);
this->Controls->Add(this->textBox8);this->Controls->Add(this->textBox7);
this->Controls->Add(this->textBox6);this->Controls->Add(this->textBox5);
this->Controls->Add(this->textBox4);this->Controls->Add(this->textBox3);
this->Controls->Add(this->textBox2);this->Controls->Add(this->radioButton2);
this->Controls->Add(this->radioButton1);this->Controls->Add(this->button2);
this->Controls->Add(this->textBox1);
                    this->Name = L"Form1";
                    this->Text = L"Bluetooth";
                    this->Load += gcnew System::EventHandler(this, &Form1::Form1_Load);
                    this->ResumeLayout(false);
                    this->PerformLayout();}
#pragma endregion
private: System::Void radioButton1_CheckedChanged
                     (System::Object^  sender, System::EventArgs^  e) {
          textBox1->Text = System::Convert::ToString(RAND_a[0]);
          textBox2->Text = System::Convert::ToString(RAND_a[1]);
          textBox3->Text = System::Convert::ToString(RAND_a[2]);
          textBox4->Text = System::Convert::ToString(RAND_a[3]);
          textBox5->Text = System::Convert::ToString(RAND_a[4]);
          textBox6->Text = System::Convert::ToString(RAND_a[5]);
          textBox7->Text = System::Convert::ToString(RAND_a[6]);
          textBox8->Text = System::Convert::ToString(RAND_a[7]);
          textBox9->Text = System::Convert::ToString(RAND_a[8]);
          textBox10->Text = System::Convert::ToString(RAND_a[9]);
          textBox11->Text = System::Convert::ToString(RAND_a[10]);
          textBox12->Text = System::Convert::ToString(RAND_a[11]);
          textBox13->Text = System::Convert::ToString(RAND_a[12]);
          textBox14->Text = System::Convert::ToString(RAND_a[13]);
          textBox15->Text = System::Convert::ToString(RAND_a[14]);
          textBox16->Text = System::Convert::ToString(RAND_a[15]);}
private: System::Void radioButton2_CheckedChanged
                     (System::Object^  sender, System::EventArgs^  e) {
          textBox1->Text = System::Convert::ToString(RAND_b[0]);
          textBox2->Text = System::Convert::ToString(RAND_b[1]);
          textBox3->Text = System::Convert::ToString(RAND_b[2]);
          textBox4->Text = System::Convert::ToString(RAND_b[3]);
          textBox5->Text = System::Convert::ToString(RAND_b[4]);
```

```
            textBox6->Text = System::Convert::ToString(RAND_b[5]);
            textBox7->Text = System::Convert::ToString(RAND_b[6]);
            textBox8->Text = System::Convert::ToString(RAND_b[7]);
            textBox9->Text = System::Convert::ToString(RAND_b[8]);
            textBox10->Text = System::Convert::ToString(RAND_b[9]);
            textBox11->Text = System::Convert::ToString(RAND_b[10]);
            textBox12->Text = System::Convert::ToString(RAND_b[11]);
            textBox13->Text = System::Convert::ToString(RAND_b[12]);
            textBox14->Text = System::Convert::ToString(RAND_b[13]);
            textBox15->Text = System::Convert::ToString(RAND_b[14]);
            textBox16->Text = System::Convert::ToString(RAND_b[15]);}
private: System::Void radioButton3_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            if (L>=1) textBox1->Text = System::Convert::ToString(PIN_dec[0]);
            else textBox1->Text = "";
            if (L>=2) textBox2->Text = System::Convert::ToString(PIN_dec[1]);
            else textBox2->Text = "";
            if (L>=3) textBox3->Text = System::Convert::ToString(PIN_dec[2]);
            else textBox3->Text = "";
            if (L>=4) textBox4->Text = System::Convert::ToString(PIN_dec[3]);
            else textBox4->Text = "";
            if (L>=5) textBox5->Text = System::Convert::ToString(PIN_dec[4]);
            else textBox5->Text = "";
            if (L>=6) textBox6->Text = System::Convert::ToString(PIN_dec[5]);
            else textBox6->Text = "";
            if (L>=7) textBox7->Text = System::Convert::ToString(PIN_dec[6]);
            else textBox7->Text = "";
            if (L>=8) textBox8->Text = System::Convert::ToString(PIN_dec[7]);
            else textBox8->Text = "";
            if (L>=9) textBox9->Text = System::Convert::ToString(PIN_dec[8]);
            else textBox9->Text = "";
            if (L>=10) textBox10->Text = System::Convert::ToString(PIN_dec[9]);
            else textBox10->Text = "";
            if (L>=11) textBox11->Text = System::Convert::ToString(PIN_dec[10]);
            else textBox11->Text = "";
            if (L>=12) textBox12->Text = System::Convert::ToString(PIN_dec[11]);
            else textBox12->Text = "";
            if (L>=13) textBox13->Text = System::Convert::ToString(PIN_dec[12]);
            else textBox13->Text = "";
            if (L>=14) textBox14->Text = System::Convert::ToString(PIN_dec[13]);
            else textBox14->Text = "";
            if (L>=15) textBox15->Text = System::Convert::ToString(PIN_dec[14]);
            else textBox15->Text = "";
            if (L>=16) textBox16->Text = System::Convert::ToString(PIN_dec[15]);
            else textBox16->Text = "";}
private: System::Void radioButton4_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            if (L+6>=1) textBox1->Text = System::Convert::ToString(PINl_dec[0]);
            else textBox1->Text = "";
            if (L+6>=2) textBox2->Text = System::Convert::ToString(PINl_dec[1]);
            else textBox2->Text = "";
            if (L+6>=3) textBox3->Text = System::Convert::ToString(PINl_dec[2]);
            else textBox3->Text = "";
            if (L+6>=4) textBox4->Text = System::Convert::ToString(PINl_dec[3]);
            else textBox4->Text = "";
            if (L+6>=5) textBox5->Text = System::Convert::ToString(PINl_dec[4]);
            else textBox5->Text = "";
            if (L+6>=6) textBox6->Text = System::Convert::ToString(PINl_dec[5]);
            else textBox6->Text = "";
            if (L+6>=7) textBox7->Text = System::Convert::ToString(PINl_dec[6]);
```

```
            else textBox7->Text = "";
            if (L+6>=8) textBox8->Text = System::Convert::ToString(PINl_dec[7]);
            else textBox8->Text = "";
            if (L+6>=9) textBox9->Text = System::Convert::ToString(PINl_dec[8]);
            else textBox9->Text = "";
            if (L+6>=10) textBox10->Text = System::Convert::ToString(PINl_dec[9]);
            else textBox10->Text = "";
            if (L+6>=11) textBox11->Text = System::Convert::ToString(PINl_dec[10]);
            else textBox11->Text = "";
            if (L+6>=12) textBox12->Text = System::Convert::ToString(PINl_dec[11]);
            else textBox12->Text = "";
            if (L+6>=13) textBox13->Text = System::Convert::ToString(PINl_dec[12]);
            else textBox13->Text = "";
            if (L+6>=14) textBox14->Text = System::Convert::ToString(PINl_dec[13]);
            else textBox14->Text = "";
            if (L+6>=15) textBox15->Text = System::Convert::ToString(PINl_dec[14]);
            else textBox15->Text = "";
            if (L+6>=16) textBox16->Text = System::Convert::ToString(PINl_dec[15]);
            else textBox16->Text = "";}
private: System::Void radioButton5_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e){
            textBox1->Text = System::Convert::ToString(Kinit[0]);
            textBox2->Text = System::Convert::ToString(Kinit[1]);
            textBox3->Text = System::Convert::ToString(Kinit[2]);
            textBox4->Text = System::Convert::ToString(Kinit[3]);
            textBox5->Text = System::Convert::ToString(Kinit[4]);
            textBox6->Text = System::Convert::ToString(Kinit[5]);
            textBox7->Text = System::Convert::ToString(Kinit[6]);
            textBox8->Text = System::Convert::ToString(Kinit[7]);
            textBox9->Text = System::Convert::ToString(Kinit[8]);
            textBox10->Text = System::Convert::ToString(Kinit[9]);
            textBox11->Text = System::Convert::ToString(Kinit[10]);
            textBox12->Text = System::Convert::ToString(Kinit[11]);
            textBox13->Text = System::Convert::ToString(Kinit[12]);
            textBox14->Text = System::Convert::ToString(Kinit[13]);
            textBox15->Text = System::Convert::ToString(Kinit[14]);
            textBox16->Text = System::Convert::ToString(Kinit[15]);}
private: System::Void radioButton6_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e){
            textBox1->Text = System::Convert::ToString(LK_KA[0]);
            textBox2->Text = System::Convert::ToString(LK_KA[1]);
            textBox3->Text = System::Convert::ToString(LK_KA[2]);
            textBox4->Text = System::Convert::ToString(LK_KA[3]);
            textBox5->Text = System::Convert::ToString(LK_KA[4]);
            textBox6->Text = System::Convert::ToString(LK_KA[5]);
            textBox7->Text = System::Convert::ToString(LK_KA[6]);
            textBox8->Text = System::Convert::ToString(LK_KA[7]);
            textBox9->Text = System::Convert::ToString(LK_KA[8]);
            textBox10->Text = System::Convert::ToString(LK_KA[9]);
            textBox11->Text = System::Convert::ToString(LK_KA[10]);
            textBox12->Text = System::Convert::ToString(LK_KA[11]);
            textBox13->Text = System::Convert::ToString(LK_KA[12]);
            textBox14->Text = System::Convert::ToString(LK_KA[13]);
            textBox15->Text = System::Convert::ToString(LK_KA[14]);
            textBox16->Text = System::Convert::ToString(LK_KA[15]);}
private: System::Void radioButton7_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e){
            textBox1->Text = System::Convert::ToString(LK_KB[0]);
            textBox2->Text = System::Convert::ToString(LK_KB[1]);
            textBox3->Text = System::Convert::ToString(LK_KB[2]);
```

```
            textBox4->Text = System::Convert::ToString(LK_KB[3]);
            textBox5->Text = System::Convert::ToString(LK_KB[4]);
            textBox6->Text = System::Convert::ToString(LK_KB[5]);
            textBox7->Text = System::Convert::ToString(LK_KB[6]);
            textBox8->Text = System::Convert::ToString(LK_KB[7]);
            textBox9->Text = System::Convert::ToString(LK_KB[8]);
            textBox10->Text = System::Convert::ToString(LK_KB[9]);
            textBox11->Text = System::Convert::ToString(LK_KB[10]);
            textBox12->Text = System::Convert::ToString(LK_KB[11]);
            textBox13->Text = System::Convert::ToString(LK_KB[12]);
            textBox14->Text = System::Convert::ToString(LK_KB[13]);
            textBox15->Text = System::Convert::ToString(LK_KB[14]);
            textBox16->Text = System::Convert::ToString(LK_KB[15]);}
private: System::Void radioButton8_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e){
            textBox1->Text = System::Convert::ToString(RAND1[0]);
            textBox2->Text = System::Convert::ToString(RAND1[1]);
            textBox3->Text = System::Convert::ToString(RAND1[2]);
            textBox4->Text = System::Convert::ToString(RAND1[3]);
            textBox5->Text = System::Convert::ToString(RAND1[4]);
            textBox6->Text = System::Convert::ToString(RAND1[5]);
            textBox7->Text = System::Convert::ToString(RAND1[6]);
            textBox8->Text = System::Convert::ToString(RAND1[7]);
            textBox9->Text = System::Convert::ToString(RAND1[8]);
            textBox10->Text = System::Convert::ToString(RAND1[9]);
            textBox11->Text = System::Convert::ToString(RAND1[10]);
            textBox12->Text = System::Convert::ToString(RAND1[11]);
            textBox13->Text = System::Convert::ToString(RAND1[12]);
            textBox14->Text = System::Convert::ToString(RAND1[13]);
            textBox15->Text = System::Convert::ToString(RAND1[14]);
            textBox16->Text = System::Convert::ToString(RAND1[15]);}
private: System::Void radioButton9_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e) {
            textBox1->Text = System::Convert::ToString(RAND2[0]);
            textBox2->Text = System::Convert::ToString(RAND2[1]);
            textBox3->Text = System::Convert::ToString(RAND2[2]);
            textBox4->Text = System::Convert::ToString(RAND2[3]);
            textBox5->Text = System::Convert::ToString(RAND2[4]);
            textBox6->Text = System::Convert::ToString(RAND2[5]);
            textBox7->Text = System::Convert::ToString(RAND2[6]);
            textBox8->Text = System::Convert::ToString(RAND2[7]);
            textBox9->Text = System::Convert::ToString(RAND2[8]);
            textBox10->Text = System::Convert::ToString(RAND2[9]);
            textBox11->Text = System::Convert::ToString(RAND2[10]);
            textBox12->Text = System::Convert::ToString(RAND2[11]);
            textBox13->Text = System::Convert::ToString(RAND2[12]);
            textBox14->Text = System::Convert::ToString(RAND2[13]);
            textBox15->Text = System::Convert::ToString(RAND2[14]);
            textBox16->Text = System::Convert::ToString(RAND2[15]);}
private: System::Void radioButton10_CheckedChanged
                        (System::Object^ sender, System::EventArgs^ e) {
            textBox1->Text = System::Convert::ToString(Kmaster[0]);
            textBox2->Text = System::Convert::ToString(Kmaster[1]);
            textBox3->Text = System::Convert::ToString(Kmaster[2]);
            textBox4->Text = System::Convert::ToString(Kmaster[3]);
            textBox5->Text = System::Convert::ToString(Kmaster[4]);
            textBox6->Text = System::Convert::ToString(Kmaster[5]);
            textBox7->Text = System::Convert::ToString(Kmaster[6]);
            textBox8->Text = System::Convert::ToString(Kmaster[7]);
            textBox9->Text = System::Convert::ToString(Kmaster[8]);
```

```
            textBox10->Text = System::Convert::ToString(Kmaster[9]);
            textBox11->Text = System::Convert::ToString(Kmaster[10]);
            textBox12->Text = System::Convert::ToString(Kmaster[11]);
            textBox13->Text = System::Convert::ToString(Kmaster[12]);
            textBox14->Text = System::Convert::ToString(Kmaster[13]);
            textBox15->Text = System::Convert::ToString(Kmaster[14]);
            textBox16->Text = System::Convert::ToString(Kmaster[15]);}
private: System::Void radioButton11_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            textBox1->Text = System::Convert::ToString(Kab[0]);
            textBox2->Text = System::Convert::ToString(Kab[1]);
            textBox3->Text = System::Convert::ToString(Kab[2]);
            textBox4->Text = System::Convert::ToString(Kab[3]);
            textBox5->Text = System::Convert::ToString(Kab[4]);
            textBox6->Text = System::Convert::ToString(Kab[5]);
            textBox7->Text = System::Convert::ToString(Kab[6]);
            textBox8->Text = System::Convert::ToString(Kab[7]);
            textBox9->Text = System::Convert::ToString(Kab[8]);
            textBox10->Text = System::Convert::ToString(Kab[9]);
            textBox11->Text = System::Convert::ToString(Kab[10]);
            textBox12->Text = System::Convert::ToString(Kab[11]);
            textBox13->Text = System::Convert::ToString(Kab[12]);
            textBox14->Text = System::Convert::ToString(Kab[13]);
            textBox15->Text = System::Convert::ToString(Kab[14]);
            textBox16->Text = System::Convert::ToString(Kab[15]);}
private: System::Void radioButton12_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            textBox1->Text = System::Convert::ToString(KC[0]);
            textBox2->Text = System::Convert::ToString(KC[1]);
            textBox3->Text = System::Convert::ToString(KC[2]);
            textBox4->Text = System::Convert::ToString(KC[3]);
            textBox5->Text = System::Convert::ToString(KC[4]);
            textBox6->Text = System::Convert::ToString(KC[5]);
            textBox7->Text = System::Convert::ToString(KC[6]);
            textBox8->Text = System::Convert::ToString(KC[7]);
            textBox9->Text = System::Convert::ToString(KC[8]);
            textBox10->Text = System::Convert::ToString(KC[9]);
            textBox11->Text = System::Convert::ToString(KC[10]);
            textBox12->Text = System::Convert::ToString(KC[11]);
            textBox13->Text = System::Convert::ToString(KC[12]);
            textBox14->Text = System::Convert::ToString(KC[13]);
            textBox15->Text = System::Convert::ToString(KC[14]);
            textBox16->Text = System::Convert::ToString(KC[15]);}
private: System::Void textBox1_TextChanged
                                (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void button1_Click
                                (System::Object^  sender, System::EventArgs^  e) {
            entrada_texto=false;
            plain_text[1]=System::Convert::ToChar(textBox29->Text);
            plain_text[2]=System::Convert::ToChar(textBox30->Text);
            plain_text[3]=System::Convert::ToChar(textBox31->Text);
            plain_text[4]=System::Convert::ToChar(textBox32->Text);
            plain_text[5]=System::Convert::ToChar(textBox33->Text);
            plain_text[6]=System::Convert::ToChar(textBox34->Text);
            plain_text[7]=System::Convert::ToChar(textBox35->Text);
            plain_text[8]=System::Convert::ToChar(textBox36->Text);
            plain_text[9]=System::Convert::ToChar(textBox37->Text);
            plain_text[10]=System::Convert::ToChar(textBox38->Text);
            plain_text[11]=System::Convert::ToChar(textBox39->Text);
            plain_text[12]=System::Convert::ToChar(textBox40->Text);
```

```
            plain_text[13]=System::Convert::ToChar(textBox41->Text);
            plain_text[14]=System::Convert::ToChar(textBox42->Text);
            plain_text[15]=System::Convert::ToChar(textBox43->Text);
            plain_text[16]=System::Convert::ToChar(textBox44->Text);}
private: System::Void button2_Click
                                    (System::Object^ sender, System::EventArgs^ e) {
          this->textBox17->Text = System::Convert::ToString(BD_ADDRESS_A_dec[0]);
   this->textBox18->Text = System::Convert::ToString(BD_ADDRESS_A_dec[1]);
   this->textBox19->Text = System::Convert::ToString(BD_ADDRESS_A_dec[2]);
   this->textBox20->Text = System::Convert::ToString(BD_ADDRESS_A_dec[3]);
   this->textBox21->Text = System::Convert::ToString(BD_ADDRESS_A_dec[4]);
   this->textBox22->Text = System::Convert::ToString(BD_ADDRESS_A_dec[5]);
   this->textBox23->Text = System::Convert::ToString(BD_ADDRESS_B_dec[0]);
   this->textBox24->Text = System::Convert::ToString(BD_ADDRESS_B_dec[1]);
   this->textBox25->Text = System::Convert::ToString(BD_ADDRESS_B_dec[2]);
   this->textBox26->Text = System::Convert::ToString(BD_ADDRESS_B_dec[3]);
   this->textBox27->Text = System::Convert::ToString(BD_ADDRESS_B_dec[4]);
   this->textBox28->Text = System::Convert::ToString(BD_ADDRESS_B_dec[5]);}
private: System::Void radioButton13_CheckedChanged
                            (System::Object^ sender, System::EventArgs^ e) {
            textBox1->Text = System::Convert::ToString(KCl_dec[0]);
            textBox2->Text = System::Convert::ToString(KCl_dec[1]);
            textBox3->Text = System::Convert::ToString(KCl_dec[2]);
            textBox4->Text = System::Convert::ToString(KCl_dec[3]);
            textBox5->Text = System::Convert::ToString(KCl_dec[4]);
            textBox6->Text = System::Convert::ToString(KCl_dec[5]);
            textBox7->Text = System::Convert::ToString(KCl_dec[6]);
            textBox8->Text = System::Convert::ToString(KCl_dec[7]);
            textBox9->Text = System::Convert::ToString(KCl_dec[8]);
            textBox10->Text = System::Convert::ToString(KCl_dec[9]);
            textBox11->Text = System::Convert::ToString(KCl_dec[10]);
            textBox12->Text = System::Convert::ToString(KCl_dec[11]);
            textBox13->Text = System::Convert::ToString(KCl_dec[12]);
            textBox14->Text = System::Convert::ToString(KCl_dec[13]);
            textBox15->Text = System::Convert::ToString(KCl_dec[14]);
            textBox16->Text = System::Convert::ToString(KCl_dec[15]);}
private: System::Void radioButton18_CheckedChanged
                            (System::Object^ sender, System::EventArgs^ e) {
            textBox1->Text = System::Convert::ToString(K_offset[0]);
            textBox2->Text = System::Convert::ToString(K_offset[1]);
            textBox3->Text = System::Convert::ToString(K_offset[2]);
            textBox4->Text = System::Convert::ToString(K_offset[3]);
            textBox5->Text = System::Convert::ToString(K_offset[4]);
            textBox6->Text = System::Convert::ToString(K_offset[5]);
            textBox7->Text = System::Convert::ToString(K_offset[6]);
            textBox8->Text = System::Convert::ToString(K_offset[7]);
            textBox9->Text = System::Convert::ToString(K_offset[8]);
            textBox10->Text = System::Convert::ToString(K_offset[9]);
            textBox11->Text = System::Convert::ToString(K_offset[10]);
            textBox12->Text = System::Convert::ToString(K_offset[11]);
            textBox13->Text = System::Convert::ToString(K_offset[12]);
            textBox14->Text = System::Convert::ToString(K_offset[13]);
            textBox15->Text = System::Convert::ToString(K_offset[14]);
            textBox16->Text = System::Convert::ToString(K_offset[15]);}
private: System::Void radioButton14_CheckedChanged
                            (System::Object^ sender, System::EventArgs^ e) {
            textBox1->Text = System::Convert::ToString(SRES[0]);
            textBox2->Text = System::Convert::ToString(SRES[1]);
            textBox3->Text = System::Convert::ToString(SRES[2]);
            textBox4->Text = System::Convert::ToString(SRES[3]);
```

```
            textBox5->Text = ""; textBox6->Text = ""; textBox7->Text = "";
            textBox8->Text = ""; textBox9->Text = ""; textBox10->Text = "";
            textBox11->Text = ""; textBox12->Text = ""; textBox13->Text = "";
            textBox14->Text = ""; textBox15->Text = ""; textBox16->Text = "";}
private: System::Void radioButton15_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            textBox1->Text = ""; textBox2->Text = "";
            textBox3->Text = ""; textBox4->Text = "";
            textBox5->Text = System::Convert::ToString(ACO[4]);
            textBox6->Text = System::Convert::ToString(ACO[5]);
            textBox7->Text = System::Convert::ToString(ACO[6]);
            textBox8->Text = System::Convert::ToString(ACO[7]);
            textBox9->Text = System::Convert::ToString(ACO[8]);
            textBox10->Text = System::Convert::ToString(ACO[9]);
            textBox11->Text = System::Convert::ToString(ACO[10]);
            textBox12->Text = System::Convert::ToString(ACO[11]);
            textBox13->Text = System::Convert::ToString(ACO[12]);
            textBox14->Text = System::Convert::ToString(ACO[13]);
            textBox15->Text = System::Convert::ToString(ACO[14]);
            textBox16->Text = System::Convert::ToString(ACO[15]);}
private: System::Void radioButton17_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            textBox1->Text = System::Convert::ToString(COF[0]);
            textBox2->Text = System::Convert::ToString(COF[1]);
            textBox3->Text = System::Convert::ToString(COF[2]);
            textBox4->Text = System::Convert::ToString(COF[3]);
            textBox5->Text = System::Convert::ToString(COF[4]);
            textBox6->Text = System::Convert::ToString(COF[5]);
            textBox7->Text = System::Convert::ToString(COF[6]);
            textBox8->Text = System::Convert::ToString(COF[7]);
            textBox9->Text = System::Convert::ToString(COF[8]);
            textBox10->Text = System::Convert::ToString(COF[9]);
            textBox11->Text = System::Convert::ToString(COF[10]);
            textBox12->Text = System::Convert::ToString(COF[11]);
            textBox13->Text = ""; textBox14->Text = "";
            textBox15->Text = ""; textBox16->Text = "";}
private: System::Void radioButton16_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
            textBox1->Text = System::Convert::ToString(EN_RAND[0]);
            textBox2->Text = System::Convert::ToString(EN_RAND[1]);
            textBox3->Text = System::Convert::ToString(EN_RAND[2]);
            textBox4->Text = System::Convert::ToString(EN_RAND[3]);
            textBox5->Text = System::Convert::ToString(EN_RAND[4]);
            textBox6->Text = System::Convert::ToString(EN_RAND[5]);
            textBox7->Text = System::Convert::ToString(EN_RAND[6]);
            textBox8->Text = System::Convert::ToString(EN_RAND[7]);
            textBox9->Text = System::Convert::ToString(EN_RAND[8]);
            textBox10->Text = System::Convert::ToString(EN_RAND[9]);
            textBox11->Text = System::Convert::ToString(EN_RAND[10]);
            textBox12->Text = System::Convert::ToString(EN_RAND[11]);
            textBox13->Text = System::Convert::ToString(EN_RAND[12]);
            textBox14->Text = System::Convert::ToString(EN_RAND[13]);
            textBox15->Text = System::Convert::ToString(EN_RAND[14]);
            textBox16->Text = System::Convert::ToString(EN_RAND[15]);}
private: System::Void radioButton19_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
                    Console::Clear();
                    printf("\nTexto claro:\n");
                    for (i=1;i<=16;i++) printf("%d,",plain_text[i]);
                    printf("\nTexto claro (decimal):\n");
```

```
                for (i=1;i<=16;i++)  printf("%c",plain_text[i]);
            textBox1->Text = System::Char::ToString(plain_text[1]);
            textBox2->Text = System::Char::ToString(plain_text[2]);
            textBox3->Text = System::Char::ToString(plain_text[3]);
            textBox4->Text = System::Char::ToString(plain_text[4]);
            textBox5->Text = System::Char::ToString(plain_text[5]);
            textBox6->Text = System::Char::ToString(plain_text[6]);
            textBox7->Text = System::Char::ToString(plain_text[7]);
            textBox8->Text = System::Char::ToString(plain_text[8]);
            textBox9->Text = System::Char::ToString(plain_text[9]);
            textBox10->Text = System::Char::ToString(plain_text[10]);
            textBox11->Text = System::Char::ToString(plain_text[11]);
            textBox12->Text = System::Char::ToString(plain_text[12]);
            textBox13->Text = System::Char::ToString(plain_text[13]);
            textBox14->Text = System::Char::ToString(plain_text[14]);
            textBox15->Text = System::Char::ToString(plain_text[15]);
            textBox16->Text = System::Char::ToString(plain_text[16]);}
private: System::Void radioButton20_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                printf("\nTexto cifrado (decimal):\n");
                for (i=1;i<=16;i++)
                {
                        xor(plain_text[i],zt_dec[i-1]); cipher_text[i]=number;
                        printf("%d,",cipher_text[i]);
                }
                printf("\nTexto cifrado:\n");
                for (i=1;i<=16;i++) printf("%c",cipher_text[i]);
            textBox1->Text = System::Char::ToString(cipher_text[1]);
            textBox2->Text = System::Char::ToString(cipher_text[2]);
            textBox3->Text = System::Char::ToString(cipher_text[3]);
            textBox4->Text = System::Char::ToString(cipher_text[4]);
            textBox5->Text = System::Char::ToString(cipher_text[5]);
            textBox6->Text = System::Char::ToString(cipher_text[6]);
            textBox7->Text = System::Char::ToString(cipher_text[7]);
            textBox8->Text = System::Char::ToString(cipher_text[8]);
            textBox9->Text = System::Char::ToString(cipher_text[9]);
            textBox10->Text = System::Char::ToString(cipher_text[10]);
            textBox11->Text = System::Char::ToString(cipher_text[11]);
            textBox12->Text = System::Char::ToString(cipher_text[12]);
            textBox13->Text = System::Char::ToString(cipher_text[13]);
            textBox14->Text = System::Char::ToString(cipher_text[14]);
            textBox15->Text = System::Char::ToString(cipher_text[15]);
            textBox16->Text = System::Char::ToString(cipher_text[16]);}
private: System::Void radioButton21_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                printf("\nTexto decifrado (decimal):\n");
                for (i=1;i<=16;i++)
                {
                        xor(cipher_text[i],zt_dec[i-1]); decipher_text[i]=number;
                        printf("%d,",decipher_text[i]);
                }
                printf("\nTexto decifrado:\n");
                for (i=1;i<=16;i++) printf("%c",decipher_text[i]);
            textBox1->Text = System::Char::ToString(decipher_text[1]);
            textBox2->Text = System::Char::ToString(decipher_text[2]);
            textBox3->Text = System::Char::ToString(decipher_text[3]);
            textBox4->Text = System::Char::ToString(decipher_text[4]);
            textBox5->Text = System::Char::ToString(decipher_text[5]);
```

```
            textBox6->Text = System::Char::ToString(decipher_text[6]);
            textBox7->Text = System::Char::ToString(decipher_text[7]);
            textBox8->Text = System::Char::ToString(decipher_text[8]);
            textBox9->Text = System::Char::ToString(decipher_text[9]);
            textBox10->Text = System::Char::ToString(decipher_text[10]);
            textBox11->Text = System::Char::ToString(decipher_text[11]);
            textBox12->Text = System::Char::ToString(decipher_text[12]);
            textBox13->Text = System::Char::ToString(decipher_text[13]);
            textBox14->Text = System::Char::ToString(decipher_text[14]);
            textBox15->Text = System::Char::ToString(decipher_text[15]);
            textBox16->Text = System::Char::ToString(decipher_text[16]);}
private: System::Void Form1_Load
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void checarTabelasToolStripMenuItem_Click
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void button15_Click
                        (System::Object^  sender, System::EventArgs^  e) {
         Start_variables();}
private: System::Void button4_Click
                        (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                int x,y;
           entrada_texto=false;
           plain_text[1]=System::Convert::ToChar(textBox29->Text);
           plain_text[2]=System::Convert::ToChar(textBox30->Text);
           plain_text[3]=System::Convert::ToChar(textBox31->Text);
           plain_text[4]=System::Convert::ToChar(textBox32->Text);
           plain_text[5]=System::Convert::ToChar(textBox33->Text);
           plain_text[6]=System::Convert::ToChar(textBox34->Text);
           plain_text[7]=System::Convert::ToChar(textBox35->Text);
           plain_text[8]=System::Convert::ToChar(textBox36->Text);
           plain_text[9]=System::Convert::ToChar(textBox37->Text);
           plain_text[10]=System::Convert::ToChar(textBox38->Text);
           plain_text[11]=System::Convert::ToChar(textBox39->Text);
           plain_text[12]=System::Convert::ToChar(textBox40->Text);
           plain_text[13]=System::Convert::ToChar(textBox41->Text);
           plain_text[14]=System::Convert::ToChar(textBox42->Text);
           plain_text[15]=System::Convert::ToChar(textBox43->Text);
           plain_text[16]=System::Convert::ToChar(textBox44->Text);
                PIN=System::Convert::ToDouble(textBox45->Text);
                L_effective=System::Convert::ToInt64(textBox46->Text);
                j=0;
                Converts(PIN);
                for (i=0;i<=63;i++)
                {
                        PIN_bin[i]=binary[i];
                        if ((PIN_bin[i]==1)&(j==0)) j=64-i;
                }
                L=1; y=j; p=0;
                while (y>8) {y=y-8; L=L+1;}
                for (i=8-L;i<=7;i++)
                {
                        binary_to_decimal(PIN_bin,8*i,8*i+7);
                        PIN_dec[p]=number; p=p+1;
                }
                if (j==0) j=1;
                printf("bits=%d, L=%d\n",j,L);
                if (15-L>=5) x=5; if (15-L<5) x=15-L;
                if (L<16)
                {
```

```
                    for (i=0;i<=L-1;i++) PINl_dec[i]=PIN_dec[i];
                    for (i=0;i<=x;i++) PINl_dec[L+i]=BD_ADDRESS_A_dec[i];
                }
                if (L==16) for (i=0;i<=L-1;i++) PINl_dec[i]=PIN_dec[i];
                printf("\nPIN=");
                for (i=0;i<=L-1;i++) printf("%d, ",PIN_dec[i]);
                if (L+6>=16) Ll=16; if (L+6<16) Ll=L+6;
                printf("\n\nPIN'=");
                for (i=0;i<=Ll-1;i++) printf("%d, ",PINl_dec[i]);
                printf("\n");}
private: System::Void button5_Click
                    (System::Object^  sender, System::EventArgs^  e) {
        Console::Clear();
        Kinit_bool=true;
        E22();
        Kinit_bool=false;}
private: System::Void button6_Click
                    (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                printf("LK_Ka:\n");
                LK_KA_bool=true;RA=true;E21(false);RA=false;LK_KA_bool=false;
                printf("LK_Kb:\n");
                LK_KB_bool=true;RB=true;E21(false);RB=false;LK_KB_bool=false;}
private: System::Void button7_Click
                    (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                printf("RAND1= \n");
                for (i=0;i<=15;i++) RAND1[i]=rand()%256;
                for (i=0;i<=15;i++) printf("%d, ",RAND1[i]);
                printf("\n\nRAND2= \n");
                for (i=0;i<=15;i++) RAND2[i]=rand()%256;
                for (i=0;i<=15;i++) printf("%d, ",RAND2[i]);
                printf("\n");}
private: System::Void button8_Click
                    (System::Object^  sender, System::EventArgs^  e) {
                Console::Clear();
                printf("Kab=\n");
                for (i=0;i<=15;i++)
                {
                    xor(LK_KA[i],LK_KB[i]);          Kab[i]=number;
                    printf("%d, ",Kab[i]);
                }
                printf("\n");}
private: System::Void button9_Click
                    (System::Object^  sender, System::EventArgs^  e) {
        Console::Clear();
        E1();}
private: System::Void button10_Click
                    (System::Object^  sender, System::EventArgs^  e) {
        Console::Clear();
        E0();}
private: System::Void textBox45_TextChanged
                    (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void button3_Click
                    (System::Object^  sender, System::EventArgs^  e) {
        Console::Clear();
        printf("128 ultimas saidas geradas:\n");
        j=0;
        for (i=112;i<=239;i++)
        {
```

```
                j=j+1; printf("%d",zt[i]);
                if (j%64==0) printf("\n");
        }
        printf("\nZ(t) em decimal:\n");
        for (i=0;i<=15;i++)
        {
                binary_to_decimal(zt,112+8*i,112+8*i+7); zt_dec[i]=number;
                printf("%d, ",zt_dec[i]);
        }}
private: System::Void radioButton22_CheckedChanged
                        (System::Object^  sender, System::EventArgs^  e) {
          textBox1->Text = System::Convert::ToString(zt_dec[0]);
          textBox2->Text = System::Convert::ToString(zt_dec[1]);
          textBox3->Text = System::Convert::ToString(zt_dec[2]);
          textBox4->Text = System::Convert::ToString(zt_dec[3]);
          textBox5->Text = System::Convert::ToString(zt_dec[4]);
          textBox6->Text = System::Convert::ToString(zt_dec[5]);
          textBox7->Text = System::Convert::ToString(zt_dec[6]);
          textBox8->Text = System::Convert::ToString(zt_dec[7]);
          textBox9->Text = System::Convert::ToString(zt_dec[8]);
          textBox10->Text = System::Convert::ToString(zt_dec[9]);
          textBox11->Text = System::Convert::ToString(zt_dec[10]);
          textBox12->Text = System::Convert::ToString(zt_dec[11]);
          textBox13->Text = System::Convert::ToString(zt_dec[12]);
          textBox14->Text = System::Convert::ToString(zt_dec[13]);
          textBox15->Text = System::Convert::ToString(zt_dec[14]);
          textBox16->Text = System::Convert::ToString(zt_dec[15]);}
private: System::Void button1_Click_1
                        (System::Object^  sender, System::EventArgs^  e) {
          textBox29->Text = "D"; textBox30->Text = "a"; textBox31->Text = "n";
          textBox32->Text = "i"; textBox33->Text = "e"; textBox34->Text = "l";
          textBox35->Text = "L"; textBox36->Text = "e"; textBox37->Text = "l";
          textBox38->Text = "e"; textBox39->Text = "L"; textBox40->Text = "o";
          textBox41->Text = "r"; textBox42->Text = "e"; textBox43->Text = "n";
          textBox44->Text = "a"; textBox45->Text = "0"; textBox46->Text = "16";}
private: System::Void button11_Click
                        (System::Object^  sender, System::EventArgs^  e) {
        print=true;
        this->button11->BackColor=System::Drawing::Color::Green;
        this->button11->ForeColor = System::Drawing::Color::White;
        this->button12->BackColor=System::Drawing::Color::Red;
        this->button12->ForeColor = System::Drawing::Color::White;}
private: System::Void button12_Click
                        (System::Object^  sender, System::EventArgs^  e) {
        print=false;
        this->button12->BackColor=System::Drawing::Color::Green;
        this->button12->ForeColor = System::Drawing::Color::White;
        this->button11->BackColor=System::Drawing::Color::Red;
        this->button11->ForeColor = System::Drawing::Color::White;}
private: System::Void label1_Click
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void label6_Click
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void label9_Click
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void textBox2_TextChanged
                        (System::Object^  sender, System::EventArgs^  e) {}
private: System::Void label3_Click
                        (System::Object^  sender, System::EventArgs^  e) {}
};}
```

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BLUETOOTH SIG. *Bluetooth Basics*. disponível em: http://www.bluetooth.com/English/Technology/Pages/Basics.aspx, acesso em junho de 2010.

[2] BLUETOOTH SIG. *Bluetooth Core System Architecture*. disponível em: http://www.bluetooth.com/English/Technology/Works/Pages/Core_System_Architecture.aspx, acesso em junho de 2010.

[3] 802.15.1-2005 ™ IEEE Standard for Information Technology. *Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements*. New York, USA 2005.

[4] LAGARIAS, J. C. Pseudorandom Numbers. *Statistical Science.* 8: 31--39, 1993.

[5] MASSEY, J. L. and RUEPPEL, R. *Method of, and apparatus for, transforming a digital data sequence into an encoded form*. U.S. Patent No. 4,797,922, 1989.

[6] RUEPPEL, R. *Analysis and Design of Stream Ciphers*. Springer-Verlag, New York, 1986.

[7] KLAPPER A. and GORESKY, M. Feedback Shift Registers, 2-Adic Span, and Combiners with Memory. *Journal of Cryptology*. 1997, 10: 111-147.

[8] HAMMING, R. W. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, Dover paperback reprint, 1986.

[9] MASSEY, J. L.; KHACHATRIAN, G. and KUREGIAN, M. Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard. *1st Advanced Encryption Standard Canditate Conference*. CA, Aug: 20-22, 1998, pp:- 1-14.

[10] MASSEY, J. L. SAFER K-64: A Byte-Oriented Block-Ciphering Algorithm. Fast Software Encryption. *Cambridge Security Workshop Proceedings*. Springer, 1994, pp:-: 1-17.

[11] BAKHTIARI, S.; SAFAVI-NAINI R. and PIEPRZYK J. *Cryptographic Hash Functions: A Survey*. Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Australia, 95-09, July 1995.

[12] LIPMAA, H. On Differential Properties of Pseudo-Hadamard Transform and Related Mappings (Extended Abstract). In Alfred Menezes and Palash Sarkar, editors, *INDOCRYPT 2002*, volume 2551 of *Lecture Notes in Computer Science*, pages 48--61, Hyderabad, India, December 15--18, 2002. Springer-Verlag.

[13] BANAHAN, M.; BRADY, D. and DORAN, M. *The C Book*. second edition, Addison Wesley, 1991.

[14] STEPHEN, R. G. F. *Pro Visual C++/CLI and .NET 3.5 Platform*. Apress, 2009.

[15] SANCHEZ, J. and CANTON, M. P. (2007). *Microcontroller programming : the microchip PIC*. Boca Raton, FL: CRC Press, p. 37.

[16] HORTON, I. *Beginning Visual C++ 2008*. Wiley Publishing, Inc - 2008.

[17] THOMAS H. C.; CHARLES E. L.; RONALD L. R. and CLIFFORD S. *Introduction to Algorithms*. Second Edition. MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. Section 31.3: Modular arithmetic, pp. 862–868.

[18] BEARD, J. K. Binary Polynomial Division. *EE521 – Analog and Digital Communications*. April 20, 2006, pp. 1-5.

[19] GATHEN, J. V. Z. and GERHARD, J. *Modern Computer Algebra*. Second edition, Cambridge University Press, 2003.

[20] BACH, E. Efficient prediction of Marsaglia–Zaman random number generators. *IEEE Transactions On Information Theory*, 44(3), 1253-7, 1998.

[21] JANSEN, C. J. A. Information theory of shift registers. In: *Proceedings of the Tenth Symposium on InformationTheory in the Benelux* (A. M. Barbe, ed.), Werkgemeenschap voor Inf.- & Communicatietheorie, Enschede, 1989, pp. 153–160.

[22] HARDY, G. and WRIGHT, E. *An Introduction to the Theory of Numbers*. Oxford University Press, Oxford, 1979.

[23] KUMAR, A.; MANJUNATH D. and KURI J. *Wireless Networking*. Elsevier, Burlington – USA, 2008.

[24] GEHRMANN, C.; PERSSON, J. and SMEETS, B. *Bluetooth Security*. Artech House, 2004.

[25] BLUETOMORROW.COM. *Play Station 3 and Bluetooth*. Disponível em: http://www.bluetomorrow.com/bluetooth-products/gaming-products/playstation3-bluetooth.html, acesso em junho de 2010.

[26] BLUETOMORROW.COM. *Nintendo Wii and Bluetooth*. Disponível em: http://www.bluetomorrow.com/bluetooth-products/gaming-products/nintendo-wii-bluetooth.html, acesso em junho de 2010.