MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

1 Ten MARINA PESSOA MOTA
1 Ten TAYNÁ LARISSA FISCHER VIEIRA
PEDRO MORAIS DE ARRUDA SIAUDZIONIS

PINPAD POR SOFTWARE USANDO WHITE-BOX CRYPTOGRAPHY

Rio de Janeiro
2018

INSTITUTO MILITAR DE ENGENHARIA

1 Ten MARINA PESSOA MOTA
1 Ten TAYNÁ LARISSA FISCHER VIEIRA
PEDRO MORAIS DE ARRUDA SIAUDZIONIS

PINPAD POR SOFTWARE USANDO WHITE-BOX CRYPTOGRAPHY

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Luiz Carlos Castro Guedes - D.Sc.

Rio de Janeiro
2018

c2018

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80 - Praia Vermelha
Rio de Janeiro - RJ CEP 22290-270

Este exemplar é de propriedade do Instituto Militar de Engenharia, que poderá incluí-lo em base de dados, armazenar em computador, microfilmar ou adotar qualquer forma de arquivamento.

É permitida a menção, reprodução parcial ou integral e a transmissão entre bibliotecas deste trabalho, sem modificação de seu texto, em qualquer meio que esteja ou venha a ser fixado, para pesquisa acadêmica, comentários e citações, desde que sem finalidade comercial e que seja feita a referência bibliográfica completa.

Os conceitos expressos neste trabalho são de responsabilidade do(s) autor(es) e do(s) orientador(es).

005.82 Mota, Marina Pessoa
M917p PINPAD POR SOFTWARE USANDO WHITE-BOX CRYPTOGRAPHY / Marina Pessoa Mota, Tayná Larissa Fischer Vieira, Pedro Morais de Arruda Siaudzionis, orientado por Luiz Carlos Castro Guedes - Rio de Janeiro: Instituto Militar de Engenharia, 2018.

48p.: il.

Projeto de Fim de Curso (PFC) - Instituto Militar de Engenharia, Rio de Janeiro, 2018.

1. Curso de Graduação em Engenharia de Computação - projeto de fim de curso. 2. Criptografia. I. Tayná Larissa Fischer Vieira. II. Pedro Morais de Arruda Siaudzionis. III. Luiz Carlos Castro Guedes. IV. Título. V. Instituto Militar de Engenharia.

INSTITUTO MILITAR DE ENGENHARIA

1 Ten MARINA PESSOA MOTA
1 Ten TAYNÁ LARISSA FISCHER VIEIRA
PEDRO MORAIS DE ARRUDA SIAUDZIONIS

PINPAD POR SOFTWARE USANDO WHITE-BOX CRYPTOGRAPHY

Projeto de Fim de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Instituto Militar de Engenharia, como requisito parcial para a obtenção do título de Engenheiro de Computação.

Orientador: Prof. Luiz Carlos Castro Guedes - D.Sc.

Aprovado em 20 de Setembro de 2018 pela seguinte Banca Examinadora:

Prof. Luiz Carlos Castro Guedes - D.Sc. do IME - Presidente

Prof. Humberto Henriques de Arruda - M.Sc. do IME

Prof. Alex de Vasconcelos Garcia - D.Sc. do IME

Rio de Janeiro
2018

A todos aqueles que apoiaram nossos sonhos, oferecemos hoje nossas conquistas.

AGRADECIMENTOS

Com essas linhas iniciais, gostaríamos de agradecer a todas as pessoas que participaram, de perto ou de longe, de forma fortuita ou não, para o bom andamento deste trabalho. Aos nossos pais, pelo investimento em nossos estudos que nos trouxe até aqui, mas principalmente pela paciência, companheirismo e presença diária em nossas vidas.

Ao nosso professor orientador Dr. Luiz Carlos Castro Guedes, pelo apoio incondicional prestado, pelo interesse com que acompanhou o trabalho, por sua disponibilidade, mesmo em finais de semana e, principalmente, pelo seus ensinamentos que contribuíram para nossa formação como engenheiros.

Aos professores Humberto Arruda e Alex Garcia que gentilmente aceitaram compor a banca examinadora do presente trabalho e contribuíram com avaliações parciais durante o projeto.

À seção de Engenharia de Computação, pela estrutura fornecida durante toda nossa graduação, pelos professores que se empenharam para nossa formação e pelos funcionários que nos ajudaram a todo momento.

Aos nossos amigos, que jamais nos deixaram sozinhos, que são uma família para todos os momentos e com os quais construímos uma sincera amizade.

Por fim, ao Instituto Militar de Engenharia como um todo, representado pelos alunos, professores e servidores, que engrandeceram nossa formação pessoal e acadêmica.

"Win a no win situation by rewriting the rules. "

HARVEY SPECTER

SUMÁRIO

LISTA DE ILUSTRAÇÕES 7

LISTA DE TABELAS 8

LISTA DE SIGLAS 9

1 INTRODUÇÃO 12
1.1 MOTIVAÇÃO 12
1.2 OBJETIVO 13
1.3 JUSTIFICATIVA 13
1.4 METODOLOGIA 13
1.5 ESTRUTURA 14

2 INTRODUÇÃO TEÓRICA 15
2.1 DES 16
2.2 CRIPTOGRAFIA CAIXA-BRANCA 19
2.3 ESTRATÉGIAS CAIXA-BRANCA 20
2.4 WBDES 21
2.5 PINPad 22

3 SWB-3DES 25
3.1 versão inicial: SWBDES 25
3.2 Funcionamento do código do SWB-3DES 27

4 TESTES REALIZADOS 30
4.1 Vetores de teste para o SWBDES 30
4.2 Vetores de teste para o SWB-3DES 34
4.3 SWB-3DES aplicado ao PINPad 38

5 SEGURANÇA DE ALGORITMOS CAIXA-BRANCA 43
5.1 Revisão dos ataques do DES 44
5.2 Segurança SWBDES 45

6 CONCLUSÕES 46

LISTA DE ILUSTRAÇÕES

FIG.2.1 Funcionamento do DES 17
FIG.2.2 Função F 18
FIG.2.3 Implementação de White-Box por Chow et al. (1) 22
FIG.2.4 Bloco de cifras 23
FIG.2.5 Obtenção da *Session Key* 23

FIG.3.1 Função F modificada 26

FIG.4.1 Vetores de teste propostos pela NBS (2) 31
FIG.4.2 Vetor de teste SWBDES 1 32
FIG.4.3 Vetor de teste SWBDES 2 33
FIG.4.4 Vetor de teste SWBDES 3 34
FIG.4.5 Vetores de teste propostos pelo NESSIE (3) 35
FIG.4.6 Vetor de teste SWB-3DES 1 36
FIG.4.7 Vetor de teste SWB-3DES 2 37
FIG.4.8 Vetor de teste SWB-3DES 3 37
FIG.4.9 Exemplo de PINPad na versão DES 38
FIG.4.10 Configuração da *Master Key* 39
FIG.4.11 Menu do PINPad 39
FIG.4.12 Atualização da *Working Key* 40
FIG.4.13 Passar o cartão 41
FIG.4.14 Saída do Simulador 41
FIG.4.15 Opção Ajuda 42
FIG.4.16 Opção Sair 42

LISTA DE TABELAS

TAB.2.1 Permutação inicial IP 16
TAB.2.2 S-box 1 18
TAB.2.3 Permutação P 19
TAB.2.4 Permutação final IP^{-1} 19

TAB.3.1 Método das Relações das Durações 25

LISTA DE SIGLAS

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
DES	Data Encryption Standard
DRM	Digital Right Managements
NBS	National Bureau of Standards
NESSIE	New European Schemes for Signatures, Integrity and Encryption
NIST	National Institute of Standards and Technology
SWB-3DES	Simple White-Box Triple Data Encryption Standard
SWBDES	Simple White-Box Data Encryption Standard
WBC	White-Box Cryptography
WBDES	White-Box Data Encryption Standard

RESUMO

A segurança das chaves é uma das grandes dificuldades dos desenvolvedores atuais. O processo de armazenamento nos dispositivos digitais se tornou extremamente vulnerável, uma vez que, na maioria dos casos, o atacante tem total controle do ambiente de execução, o que caracteriza um ataque caixa-branca. Com o avanço da tecnologia, os bancos se tornaram digitais e o celulares passaram a substituir o PINPad. Mas como armazenar as senhas, se destruir o celular não é uma opção? Numa tentativa de impedir esse tipo de ataque, tenciona-se simular o funcionamento de um PINPad utilizando uma versão caixa-branca do algoritmo 3DES. Para cumprir esse objetivo, foi criado o SWB-3DES, uma simplificação do algoritmo proposto por Chow et al. expandido para o 3DES. Nesse projeto, o algoritmo criado é validado a partir de vetores de teste e por comparação com um exemplo de PINPad publicado. E, a partir de um estudo dos principais tipos de ataques ao DES, é feita uma dedução lógica de quais ataques o nosso algoritmo estaria vulnerável como forma de contextualizar como esse algoritmo pode ser de fato empregado na prática e o que está sendo estudado de melhorias no meio acadêmico.

ABSTRACT

Key security is one of the great difficulties of today's developers. The storage process on digital devices has become extremely vulnerable since, in most cases, the attacker has full control of the execution environment, which characterizes a white box attack. With the advancement of technology, banks have become digital and mobile phones have replaced the PINPad. But how to store passwords, if destroying the cell phone is not an option? In an attempt to prevent type of attack, it is intended to simulate the operation of a PINPad using a white-box version of the 3DES algorithm. To achieve this goal, SWB-3DES was created, a simplification of the algorithm proposed by Chow et al. expanded to 3DES. In this project, the algorithm created is validated by test vectors and by an example of a published PINPad. And, from a study of the main types of attacks of the DES, a logical deduction is made of which attacks our algorithm would be vulnerable in a way to contextualize how this algorithm can actually be used in practice and what is being studied of improvements in the academic environment.

1 INTRODUÇÃO

Manter a segurança das informações é um dos grandes desafios dos desenvolvedores atuais. E a segurança das chaves é uma das grandes dificuldades. O processo de armazenamento nos dispositivos digitais se tornou extremamente vulnerável com o surgimento da rede e com a facilidade de espalhar malwares e spywares.

A criptografia clássica é um tipo de caixa-preta, ela supõe que o atacante sabe o algoritmo, tem acesso à cifra resultante, mas não tem acesso à chave. No entanto, o atacante não tem acesso à execução das operações dinâmicas de encriptação. Um dos possíveis ataques é explorar as vulnerabilidades no armazenamento das chaves.

A criptografia de caixa-branca busca, ao deixar em claro o funcionamento do algoritmo, tenta solucionar o desafio do armazenamento. As chaves estão escondidas no código, o desafio é, então, entender o código e conseguir extrair a chave.

1.1 MOTIVAÇÃO

O PINPad, dispositivo eletrônico utilizado para entradas de senhas nos pagamentos com cartão de crédito e débito, funciona como uma caixa preta. Ele tem um mecanismo de proteção de senha que destrói o chip onde a senha é armazenada caso haja ataque. Com o avanço da tecnologia, os bancos se tornaram digitais e a tendência atual é usar o celular para tudo, inclusive substituindo o PINPad. Mas como armazenar as senhas, se destruir o celular não é uma opção? Utilizando a criptografia de caixa-branca.

A motivação desse trabalho é simular, através da criptografia de caixa branca, o funcionamento do PINPad em dispositivos onde o armazenamento das chaves não é seguro. Escolheu-se o algoritmo do 3DES pela compatibilidade com os host legados, uma vez que este é o algoritmo utilizado nesses dispositivos

1.2 OBJETIVO

No presente trabalho, tenciona-se simular o funcionamento do PINPad em dispositivos onde o armazenamento das chaves não é seguro. Como a maioria dos PINPads utilizam o algoritmo 3DES, será implementado uma versão caixa-branca desse algoritmo.

Para obter esse resultado, foram estabelecidos os seguintes objetivos secundários:

a) Obter um algoritmo caixa-branca para o DES

b) Estender esse algoritmo ao 3DES

c) Simular um PINPad utilizando a versão caixa-branca do 3DES

1.3 JUSTIFICATIVA

Nos últimos anos, houve um crescimento nas vendas online, os bancos e as operadoras de cartão de crédito estão investindo cada vez mais nos meios digitais e se tornou comum realizar transações de crédito e débito online. A tendência é que até as compras em lojas físicas sejam processadas por dispositivos móveis. Contudo, simular um PINPad pode gerar uma dificuldade no armazenamento das senhas.

Algoritmos de caixa-branca têm se mostrado uma alternativa para ambientes onde o armazenamento das chaves não é seguro. Trazer o algoritmo do PINPad, normalmente o 3DES, para um ambiente de caixa-branca pode ser a solução para a maioria dos ataques criptográficos e evitar a clonagem do cartão.

1.4 METODOLOGIA

Como primeiro passo na realização do projeto foi feito um estudo acerca do que são algoritmos de caixa-branca focando nas versões caixa-branca do DES. Dentre os algoritmos encontrados, escolhemos trabalhar com a implementação de Wong et al. (4) do algoritmo proposto pelo Chow et al. (1).

Apesar de publicada, a implementação desse algoritmo continha muitos erros e numa tentativa de consertá-los, foi criada uma versão simplificada do algoritmo de Chow et al., que chamamos de SWBDES.

A partir dessa versão simplificada, foi criada uma versão expandida ao 3DES, o SWB-3DES, uma versão caixa-branca do 3DES e um dos objetivos principais de nosso trabalho.

Foi criado, então, um simulador do PINPad, utilizando tanto a versão caixa-branca do 3DES, como a versão caixa-preta.

Foram validados tanto os algoritmos caixa-branca do DES e 3DES através de vetores de teste quanto o PINPad comparando os resultados com o exemplo publicado pela ANSI X9.8 (5).

Por fim, foi feito um outro estudo, dessa vez acerca da Segurança de algoritmos caixa-branca, definindo conceitos como implementação caixa-branca fraca e forte e fazendo uma revisão dos principais ataques ao WBDES e verificado se houve alguma perda de segurança relacionada à simplificação realizada no SWBDES.

1.5 ESTRUTURA

No capítulo 2, será apresentada uma breve introdução à criptografia e a fundamentação teórica acerca do DES, criptografia de caixa-branca e PINPad. Além de serem apresentados conceitos iniciais de criptografia caixa-branca e será introduzida o WBDES, uma versão caixa-baixa do DES proposta por Chow et al.

No capítulo 3, mostraremos o funcionamento do SWB-3DES, uma versão simplificada do algoritmo de caixa-branca do DES proposto anteriormente por Chow et al. e expandido pro 3DES. Será apresentado também o SWBDES, a versão simplificada inicial.

No capítulo 4, serão mostrados os resultados para vetores de testes para o SWBDES e para o SWB-3DES, será mostrada também uma simulação de um PINPad que utiliza o SWB-3DES para ofuscar a *Master Key*.

No capítulo 5, discorreremos acerca da segurança de algoritmos de caixa-branca, introduzindo conceitos e revisando os principais ataques do WBDES.

No capítulo 6 mostraremos as conclusões e propostas de trabalhos futuros.

2 INTRODUÇÃO TEÓRICA

Onde há conflitos, há necessidade da proteção das informações. A ameaça de interceptação de dados levou ao surgimento de códigos, de modo que apenas quem escrevia e o destinatário legítimo poderiam entender o conteúdo transmitido. A criptografia, surgiu assim, como um conjunto de técnicas que quando aplicadas, conferiam segurança ao transporte das mensagens.

Com o passar dos anos surgiram diversos algoritmos, a maioria deles, em sua projeção, considera que serão executados em um ambiente seguro, onde o atacante só teria acesso, no máximo, às entradas e saídas do algoritmo, que funcionaria como uma caixa-preta.

Hoje em dia, o problema está na premissa do ambiente seguro. A criptografia é usada em dispositivos abertos, como computadores, celulares ou tablets. Nesses dispositivos, além do atacante obter a implementação do software, o binário é visível e alterável, ou seja, ele também tem controle da plataforma de execução. Nesse caso, o desafio é implementar um algoritmo de uma forma que, mesmo nesse tipo de ataque, não seja possível extrair as informações (6).

O termo criptografia caixa-branca foi introduzido por Chow et al. em um artigo onde além de descrever o contexto de aplicação, fez uma implementação para o algoritmo do DES que será o foco desse trabalho (7).

O principal desafio desse tipo de criptografia é implementar um algoritmo que mantenha a informação segura mesmo quando o atacante tenha pleno conhecimento e controle do ambiente onde o algoritmo é executado. Esse objetivo vai em linha com o princípio de Kerckhoff que foi formulado ainda no século XIX e diz que um sistema criptográfico deve ser seguro mesmo que toda informação sobre o sistema seja conhecida, menos a chave.

Nesse contexto, algoritmos criptográficos caixa-branca são uma opção interessante a ser estudada, pois, até agora, todas as implementações acadêmicas de algoritmos caixa-branca conhecidas já foram quebradas (8).

2.1 DES

O DES e o 3DES estão especificados nas Federal Information Processing Standards Publications (FIPS PUBS 16-3), que são emitidas pelo NIST (9). O 3DES é a aplicação do DES três vezes, criptografando com uma chave, decriptografando com uma segunda chave e criptografando com uma terceira (que pode ser, ou não, igual à primeira). Então basta entender o funcionamento do DES para entender o do 3DES.

O DES é uma cifra de blocos de Feistel cuja chave é composta por uma sequência binária de 64 bits, dos quais 56 são usados pelo algoritmo. Os outros 8 bits servem de teste de paridade (verificando a integridade da chave). A partir dessa chave de 64 bits, são geradas 16 subchaves diversificadas de 48 bits, cada uma usada em uma rodada pela função.

O DES opera em blocos de 64 bits, primeiro ele realiza uma permutação inicial IP no bloco do texto em claro como pode ser vista na TAB 2.1 abaixo.

TAB. 2.1: Permutação inicial IP

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

Fonte: FIPS PUB 46-3 (9)

Divide-se, então, o bloco resultante em duas partes de mesmo tamanho, L_0 e R_0. Em seguida, são efetuadas 16 rodadas, cada uma com uma das subchaves. Para cada rodada computa-se:

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus \mathrm{F}(\mathrm{R_i}, \mathrm{K_i}).$$

Então a cifra é dada por (R_{n+1}, L_{n+1}).

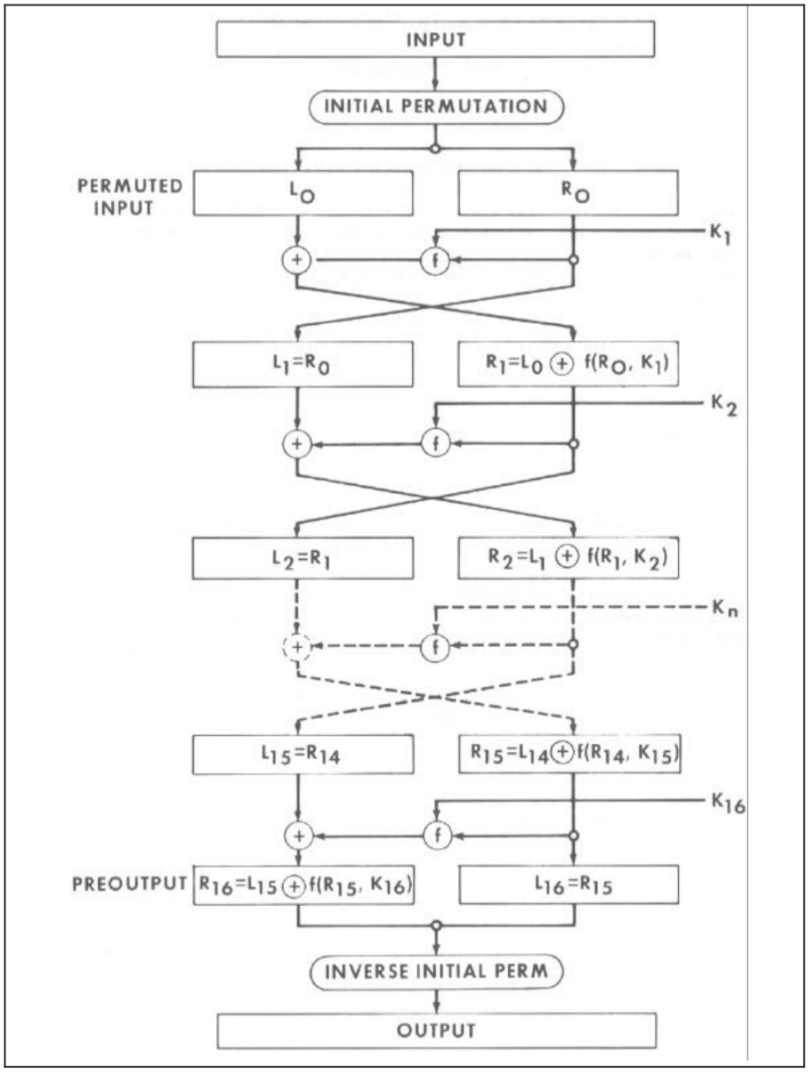


FIG. 2.1: Funcionamento do DES

Fonte: FIPS PUB 46-3 (9)

A função F consiste em 4 estágios: Expansão, Mistura de Chaves, Substituição e Permutação P.

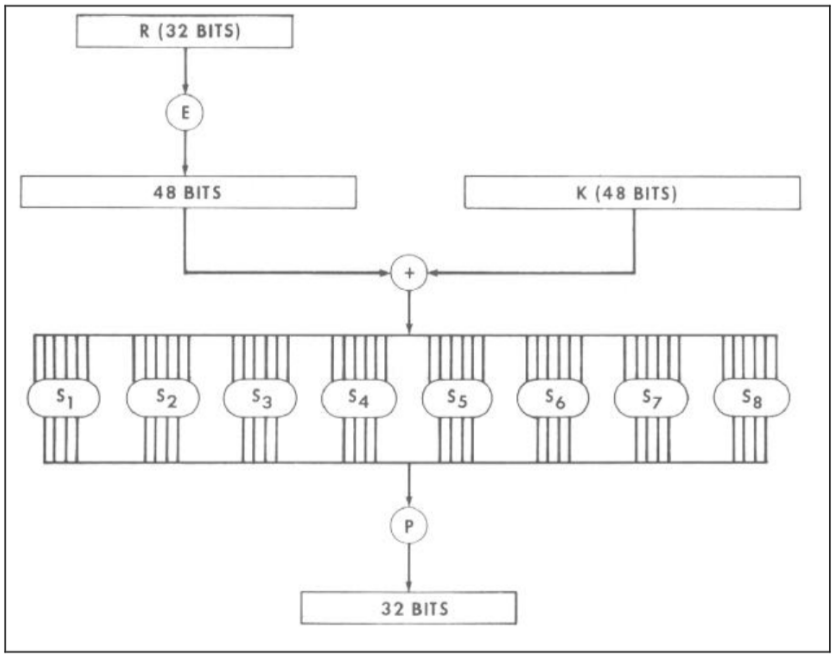


FIG. 2.2: Função F

Fonte: FIPS PUB 46-3 (9)

Na expansão, o bloco de 32 bits, metade do texto em claro (R_i) é expandido para 48 bits. Na etapa Mistura de Chaves, esses 48 bits são somados, pela operação XOR, à subchave correspondente.

A próxima etapa é a Substituição, nela cada 6 bits do bloco serve como entrada de uma tabela chamada S-box (diferente a cada rodada), onde a saída é de 4 bits. Assim, os 48 bits são substituídos por 32 bits.

TAB. 2.2: S-box 1

S-box 1

x	x0x	x1x	x2x	x3x	x4x	x5x	x6x	x7x	x8x	x9x	x10x	x11x	x12x	x13x	x14x	x15x
0y0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0y1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
1y0	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
1y1	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0

Fonte: FIPS PUB 46-3 (9)

Por último, há uma permutação fixa, permutação P (TAB 2.3), que rearranja os 32 bits.

TAB. 2.3: Permutação P

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Fonte: FIPS PUB 46-3 (9)

Ao final da última rodada, ocorre a permutação final, IP^{-1}, que é a inversa da permutação inicial. Ela pode ser observada na TAB 2.4 abaixo.

TAB. 2.4: Permutação final IP^{-1}

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

Fonte: FIPS PUB 46-3 (9)

2.2 CRIPTOGRAFIA CAIXA-BRANCA

O conceito de criptografia caixa-branca foi citado pela primeira vez por Chow et al. (1). Para definir o que seria um algoritmo caixa-branca, ele definiu, inicialmente, o que seria um ataque em ambiente caixa-preta e um ataque em ambiente caixa-branca.

O ataque em ambiente caixa-preta é aquele em que se assume que o atacante não tem conhecimento da chave e nem pode observar o algoritmo sendo executado, apesar de saber qual é o algoritmo que está sendo executado e ter acesso ao texto cifrado.

Em um ataque em ambiente caixa-branca, assume-se que o atacante tem total controle do ambiente de execução, o que lhe permite uma série de vantagens comparativas.

Com esses dois conceitos esclarecidos, ele definiu criptografia caixa-branca da seguinte forma:

Definição 1 (Chow et al. (1)): A criptografia caixa-branca é uma técnica de ofuscação que tem como objetivo implementar uma cifra que, mesmo quando o adversário tem acesso a implementação e ao ambiente de execucação, não é possível recuperar informação sobre a chave.

Entretanto, em sua tese de doutorado, Wyseur (7) citou alguns problemas com essa definição. Um dos problemas citados é que em certo sentido, a criptografia caixa-branca pode ser vista com uma técnica para esconder a chave em uma chave ainda maior. Dessa forma, a noção de recuperar informações sobre a chave torna-se sem sentido.

Então, Wyseur propôs uma definição que fizesse mais sentido ao contexto de ambiente caixa-branca.

Definição 2 (Wyseur (7)): O objetivo de um algoritmo caixa-branca é implementar uma cifra de tal forma que, dentro do contexto da aplicação pretendida, ter total acesso a implementação criptográfica não represente qualquer vantagem em comparação com um adversário lidando com a implementação com uma caixa-preta.

Percebe-se que essa definição de Wyseur apesar de vaga, corrige o problema citado na definição de Chow e alinha o conceito de cifra caixa-branca com os ambientes de ataque definidos pelo próprio Chow.

2.3 ESTRATÉGIAS CAIXA-BRANCA

A fim de implementar as cifras de bloco convencionais em ambiente caixa-branca, Chow et al. definiu duas estratégias principais que servem de bases para a construção. São elas:

1. Avaliação parcial: Consiste em avaliar previamente todas as operações envolvendo

a chave. Para o caso do algoritmo do DES, foco de estudo desse trabalho, consiste em substituir as S-boxes por tabelas específicas dependentes da chave e da rodada:

$$T_i(x) = S_i(x \oplus k_i)$$

2. Tabularização: É o processo de mapear as transformações lineares de um algoritmo na sua *lookup tables* correspondente.

Além dessas duas estratégias, Chow et al. também citou sobre a técnica de **randomização**. Essa técnica consiste em aplicar funções e suas respectivas inversas no conjunto de *lookup tables*, de forma que a composição delas continue gerando a mesma saída. O objetivo dessa técnica é aumentar o grau de aleatorização das tabelas e dificultar um ataque que observe a execução do algoritmo.

2.4 WBDES

O algoritmo caixa-branca do DES foi proposto inicialmente por Chow et al. (1). Quando publicado, o próprio autor comentou que o nível de segurança do algoritmo era desconhecido e que era um algoritmo mais lento que a implementação caixa-preta do mesmo.

Posteriormente, em 2005, Link e Neumann propuseram algumas alterações para aumentar a robustez e eficiência do algoritmo (10). Quando proposta, a variante era resistente a todos os ataques conhecidos até então. Entretanto, já em 2007, Wyseur et al. (6) propuseram um ataque diferencial que permitia a recuperação da chave nessa variante. No mesmo ano, Goubin et al (11). apresentaram um ataque similar.

No presente trabalho, apresentaremos a ideia geral do algoritmo para que posteriormente sejam estudadas eventuais alterações para a implementação no PINPad.

A criptografia caixa-branca tem como premissa utilizar a própria implementação como defesa. Para isso, utilizam-se as técnicas de ofuscação citadas anteriormente: avaliação parcial, tabularização e randomização.

A randomização é necessária pelo pressuposto de que o atacante conhece seu sistema. Dessa forma, há uma prevenção contra a introdução de pontos de interrupção ou de sobrescrita dos componentes. (6)

A transformação de um algoritmo em sua versão caixa-branca segue a seguinte estratégia: difundir a chave em toda a implementação e randomizar os blocos de construção.

A difusão impede o adversário de analisar as partes individualmente e o força a analisar toda a implementação.

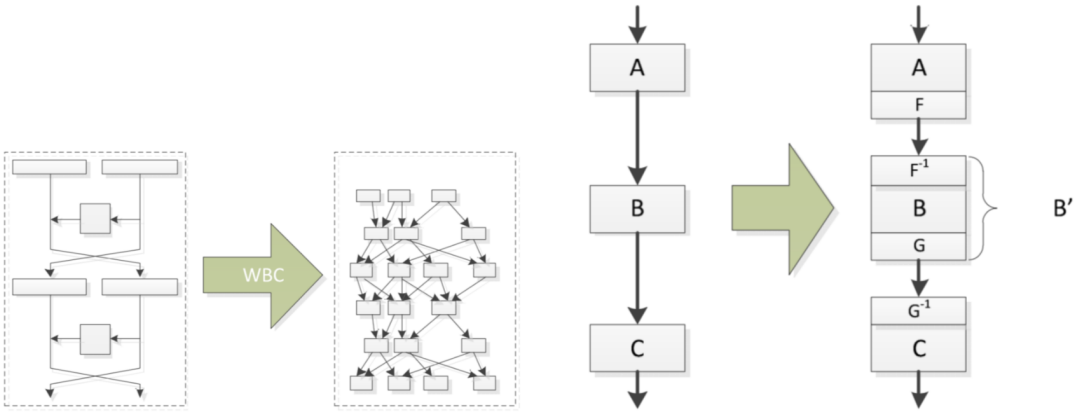
Essa transformação pode ser resumida em quatro passos.

No primeiro passo, aplica-se a técnica de avaliação parcial. Reorganiza-se a cifra de forma a tornar adjacente as operações de substituição (S-box) e a operação que inclui a chave da rodada, então codifica-se a chave na própria S-box, resumindo $T_k(x) = S(x) + k$.

O segundo passo é o que o artigo do Chow et al. (1) chama de *mixing bijections*. Operações afins anuladoras são injetadas de forma a tornar as permutações em matrizes menos esparsas.

O próximo passo é decompor todas as operações em tabelas de consulta, até mesmo as operações XOR. É a técnica de tabularização.

E por último, aplica-se a técnica de randomização. São injetadas codificações aleatórias nas tabelas de consulta.



(a) Transformação nas tabelas de consulta (b) Tabelas de consulta codificadas

FIG. 2.3: Implementação de White-Box por Chow et al. (1)

Fonte: White-Box Cryptography: Hiding Keys in Software (6)

Na FIG. 2.3(b), temos F e G que são funções de randomização.

2.5 PINPAD

Uma das grandes preocupações relacionada a segurança computacional é proporcionar um ambiente seguro para as transações financeiras eletrônicas. Nessas operações, é

necessário um canal para checar a validade da identificação do usuário, por isso tem-se o *Personal Identification Number* (PIN).

O PIN só é conhecido pelo o cliente e pelo banco emissor do cartão de crédito. Esse número tem que ser criptografado de forma que não seja decodificado se interceptado por terceiros. Em muitas operações financeiras, o dispositivo usado é o PINPad, que entrega uma solução segura, rápida e confiável para tais transações financeiras. O PIN pode ser inserido no dispositivo pelo teclado e então criptogrado de forma segura.

Quando o cliente digita o PIN no teclado, é determinado o *PIN-block* com as informações do cartão de crédito e do PIN. Esse dado é criptografado usando o DES e uma *Session Key*, como mostra o diagrama abaixo (FIG. 2.4). (5)

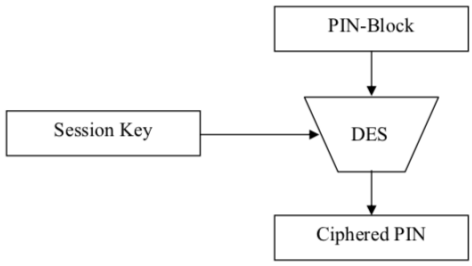


FIG. 2.4: Bloco de cifras

Fonte: ANSI X9.8

A *Session Key* é determinada decriptografando com o DES a *Working Key*, utilizando como chave a *Master Key*, como pode ser observado na FIG. 2.5.

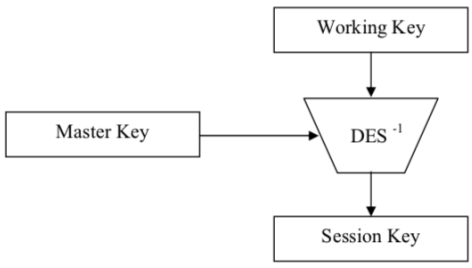


FIG. 2.5: Obtenção da *Session Key*

Fonte: ANSI X9.8

A *Master Key* é a chave que está contida dentro do PINPad. Já a *Working Key* é determinada pelo sistema e muda com o tempo. Dessa forma, a chave que é usada para

gerar os blocos cifrados, a *Session Key*, varia. Esse algoritmo deixa a comunicação com a operadora de cartão de crédito mais segura, visto que a amostra de dados criptografados com a mesma chave é pequena. Então, por mais que a troca de mensagens seja interceptada, não é tão simples obter as informações cifradas do cartão.

Esse algoritmo não é confiável em um dispositivo aberto, onde o armazenamento das chaves não é seguro. Por isso, a necessidade de uma versão caixa-branca. Dessa forma, a *Master Key* estaria ofuscada no algoritmo.

3 SWB-3DES

Na versão caixa-branca de Chow et al.(1) todo o DES é tabelado, incluindo a Permutação Inicial, Expansão, Permutação Final.

A tabularização dessas operações serve apenas para tentar aumentar a ofuscação, mas não carrega informação referente a chave, não possuindo assim, caráter criptográfico.

Neste trabalho, será proposto, então, uma versão simplificada da implementação de Wong et al.(4) do algoritmo caixa-branca proposto por Chow et al.(1) para o DES. Em seguida, essa versão será expandida para o 3DES, como alvo de aplicação pelo PINPad.

3.1 VERSÃO INICIAL: SWBDES

A implementação de Wong et al. (4) apesar de publicada continha muitos erros e, numa tentativa de consertá-los e de viabilizar o nosso trabalho, criamos uma versão simplificada.

A nossa versão do algoritmo de (1) não apresenta todas as tabularizações. A TAB. 3.1 a seguir explicita essas diferenças.

TAB. 3.1: Método das Relações das Durações

Operação	WBDES	SWBDES
Permutação Inicial	sim	não
Expansão	sim	não
Mistura de chaves	sim	sim
Substituição	sim	sim
Permutação P	sim	não
Permutação Final IP^{-1}	sim	não

O algoritmo do SWBDES tem a estrutura semelhante com a do DES explicitado na FIG 2.1 , com a alteração apenas na cifra de Feistel. Na função F, a chave não pode estar explícita, por isso utiliza-se as *lookup tables*.

A função F do WBDES consiste em três etapas: Expansão, Substituição e Permutação. A parte da Expansão e da Permutação são iguais as do DES da FIG. 2.2. O que muda é Substituição, que na versão caixa-branca já apresenta as chaves misturadas nas *lookup tables*, conforme pode ser observado na FIG. 3.1 abaixo.

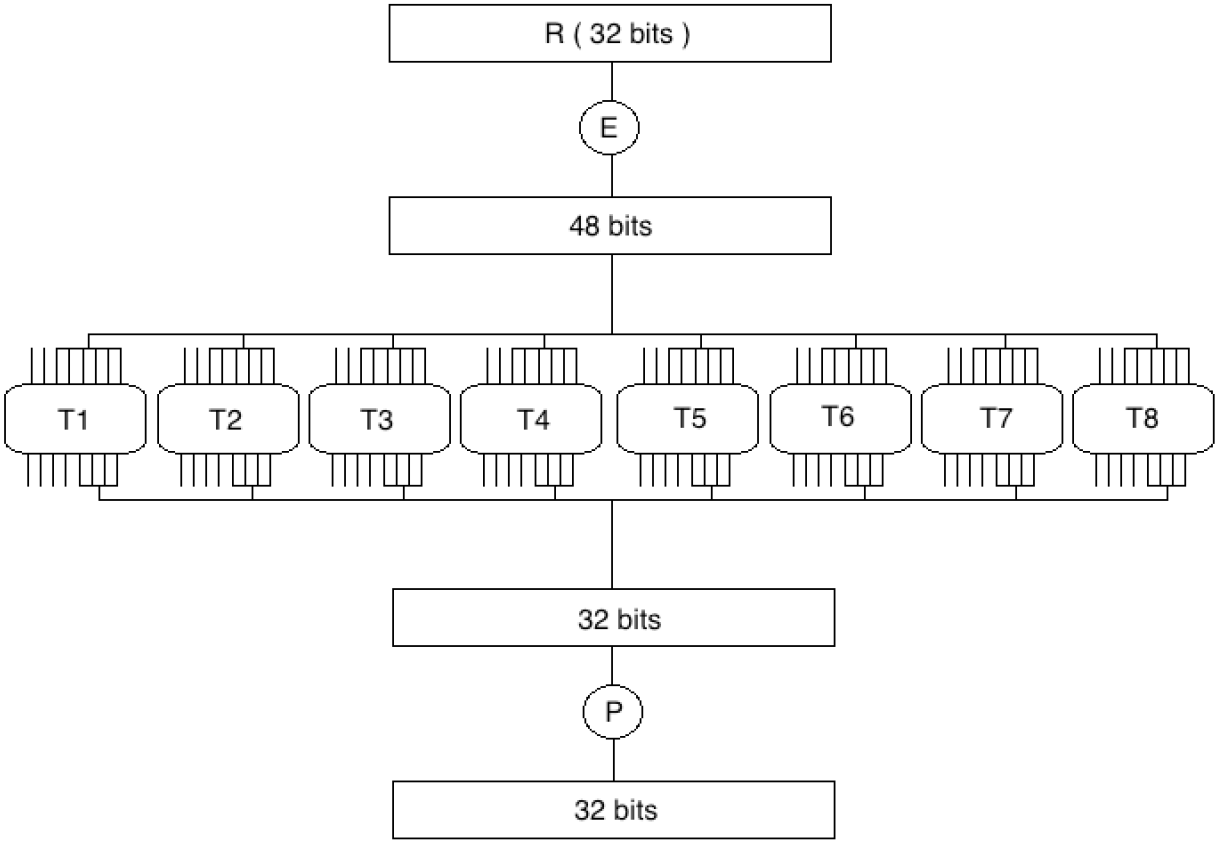


FIG. 3.1: Função F modificada

Na etapa da Substituição utilizam-se as T-boxes, as quais possuem entradas de 8 bits e saídas de 8 bits. Sendo que nessa simplificação do WBDES utiliza-se apenas os quatro bits menos significativos de saída de cada T-box. Assim obtêm-se 32 bits na saída.

As T-boxes armazenam a informação da chave e das S-box originais do DES. Visto que para cada rodada do DES usa-se uma chave obtida a partir da inicial, tem-se que para cada uma das dezesseis rodadas do algoritmo é necessário ter oito T-boxes diferentes.

Para construir as T-boxes são realizados os seguintes passos. Com os seis bits menos significativos da entrada é feito um XOR com os 6 bits relativos da chave da rodada, e então, utiliza-se esse valor para substituir o uso da S-box. Os outros quatro bits mais significativos da saída da T-box são oriundos do primeiro, segundo, terceiro e oitavo bit da entrada, respectivamente. Dessa forma, são construídas 128 T-Boxes, com 256 entradas cada.

3.2 FUNCIONAMENTO DO CÓDIGO DO SWB-3DES

Essa versão é uma aplicação do SWBDES três vezes, criptografando com uma chave, decriptografando com a segunda e criptografando novamente com a primeira. No algoritmo é dado uma chave de 16 bytes, na qual os 8 primeiros bytes formam a primeira chave, e os oito últimos formam a segunda chave.

Inicialmente é necessário inserir o texto inicial que será utilizado em hexadecimal. Depois disso, esse texto é divido em blocos de 8 bytes que, em seguida, são transformados para binário. A outra informação que precisa ser inserida é se deseja-se criptografar ou decriptografar. Essa parte será realizada em blocos como pode-se verificar no pseudocódigo a seguir.

```
if encrypt == True then
    for cada bloco i do texto original do
        bloco[i]=encryptSWB3DES(bloco[i],NonLinearTBoxes1,NonLinearTBoxes2);
    end
else
    for cada bloco i do texto original do
        bloco[i]=decryptSWB3DES(bloco[i],NonLinearTBoxes1,NonLinearTBoxes2);
    end
end
```

Algorithm 1: SWB3DES main

A função encryptSWB3DES() recebe um texto para ser criptogrado, usando SWB-3DES, e as duas T-boxes necessárias para isso. Como pode-se visualizar no Algoritmo 2, é aplicado o SWBDES criptografando com o primeiro conjunto de T-boxes, depois decriptografando com o segundo conjunto de T-boxes e em seguida repetindo o primeiro passo. A função decryptSWB3DES() é semelhante à encryptSWB3DES(), só que as etapadas são invertidas, primeiro decriptofra, depois criptografa e então decriptografa novamente.

input : data, non_linear_tboxes1[16][8][256], non_linear_tboxes2[16][8][256]
output: bloco

```
Permutation(&data, true);
data = encryptSWBDES(data,non_linear_tboxes1);
data = decryptSWBDES(data,non_linear_tboxes2);
data = encryptSWBDES(data,no _linear_tboxes1);
Permutation(&data, false);
```

Algorithm 2: Função encryptSWB3DES

Como pode-se visualizar na FIG 2.1 as funções encryptSWBDES() e decryptSWBDES() realizam cada uma das 16 rodadas do algoritmo. Com isso, a estrutura dessas funções é mostrada a seguir, em que srounds() representa uma rodada.

input : data,non_linear_tboxes[16][8][256]
output: data

for *cada rodada i das 16* **do**
    srounds(&data, i, non_linear_tboxes);
    inverte os 32bits do lado direito e com os 32 bits do lado esquerdo de data;
end

Algorithm 3: Função encryptSWBDES

Para entender o código de srounds() é melhor antes entender o funcionamento de outra função, a addbit(). Essa função copia um bit da posição x de uma variável, na posição y de outra variável. Tal função é importante em operações de expansão e comprenssão.

Em cada execução da srounds() são realizadas as operações descritas na rodadas da FIG 2.1, sendo a função de Feistel a da FIG 3.1. O primeiro passo é expandir o bloco da direita de 32 bits para 48bits. Para isso, utiliza-se um vetor DesExpansion[48] que possui os bits que devem ser copiados na ordem, assim como no algoritmo do DES.

O passo seguinte é a substituição. Tendo obtido os 48bits, eles são separados em 8 blocos em in[], obtendo-se, assim, as entradas para as T-boxes. Em seguida, é feito a substituição dado a rodada, a posição da T-box e o valor de entrada.

Depois, tem-se a permutação. Para realizá-la, usa-se um vetor Pbox[32] que possui

a ordem que deve ser permutada. Com o auxílio de uma variável temporária e usando a função addbit(), essa etapa é concluída.

Por último, é realizada a operação em que o bloco da direita da próxima rodada é o XOR do resultado obtido até agora com o bloco da esquerda do início. Já o bloco da esquerda da próxima rodada é o da direita do início.

input : *data,round,non_linear_tboxes[16][8][256]
output: é void, apenas altera o valor em *data

right_block = 0;
right_block_temp = 0;
for *cada bit i dos 48* **do**
| addbit(&right_block, *data, (DesExpansion[i] + 31), i);
end

for *cada i de 0 até 7* **do**
| in[i] = *seis bits relativos de right_block, de acordo com o i*;
end
for *cada i de 0 até 7* **do**
| substitued = non_linear_tboxes[round][i][in[i]]; right_block = *substitued na posição correta, de acordo com o i*;
end

right_block_temp = 0;
for *cada i de 0 até 31* **do**
| addbit(&right_block_temp, right_block, Pbox[i] - 1, i);
end
right_block = right_block_temp;

right_block = right_block XOR *bloco esquerdo do início*;
*data = *bloco direito do início concatenado com o right_block*;

Algorithm 4: Função srounds

4 TESTES REALIZADOS

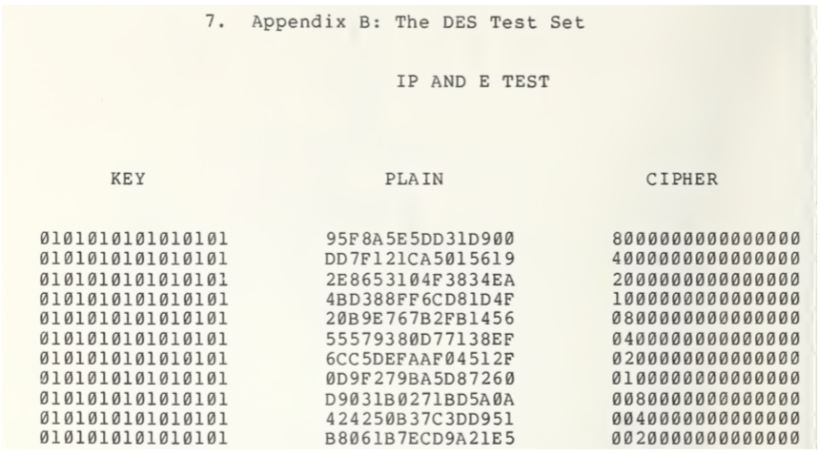
Foram realizados testes com as versões caixa-branca do DES e do 3DES de forma a confirmar que os resultados correspodiam às versões originais dos algoritmos.

Nestes testes, compara-se a tripla chave, texto em claro e cifra obtida a partir dos algoritmos caixa-branca com triplas publicadas como vetores de teste.

4.1 VETORES DE TESTE PARA O SWBDES

Na execução de nossos testes, utilizaram-se alguns dos vetores testes propostos pela NBS, National Bureau of Standards, para validação do DES (2).

Abaixo, FIG. 4.1, encontra-se uma amostra dos vetores de teste propostos pela NBS. Os testes foram construídos de forma a validar cada um dos componentes do DES: permutação inicial e final, matriz de expansão, permutação dos dados, permutação das chaves e as tabelas de substituição, as S-boxes.

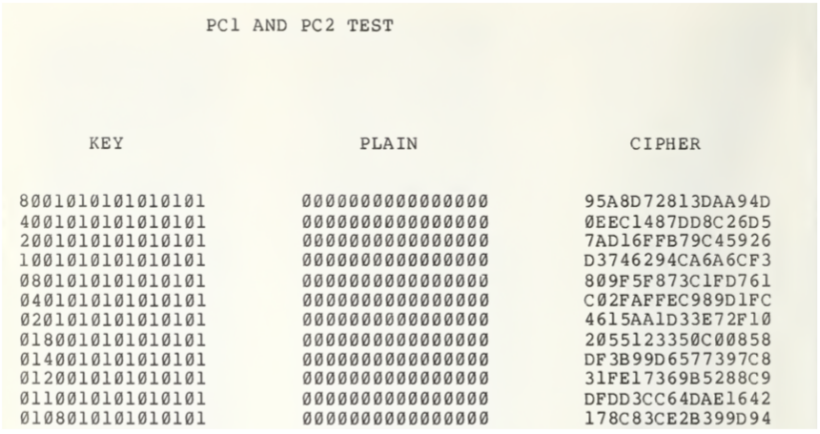


7. Appendix B: The DES Test Set

IP AND E TEST

KEY	PLAIN	CIPHER
0101010101010101	95F8A5E5DD31D900	8000000000000000
0101010101010101	DD7F121CA5015619	4000000000000000
0101010101010101	2E8653104F3834EA	2000000000000000
0101010101010101	4BD388FF6CD81D4F	1000000000000000
0101010101010101	20B9E767B2FB1456	0800000000000000
0101010101010101	55579380D77138EF	0400000000000000
0101010101010101	6CC5DEFAAF04512F	0200000000000000
0101010101010101	0D9F279BA5D87260	0100000000000000
0101010101010101	D9031B0271BD5A0A	0080000000000000
0101010101010101	424250B37C3DD951	0040000000000000
0101010101010101	B8061B7ECD9A21E5	0020000000000000

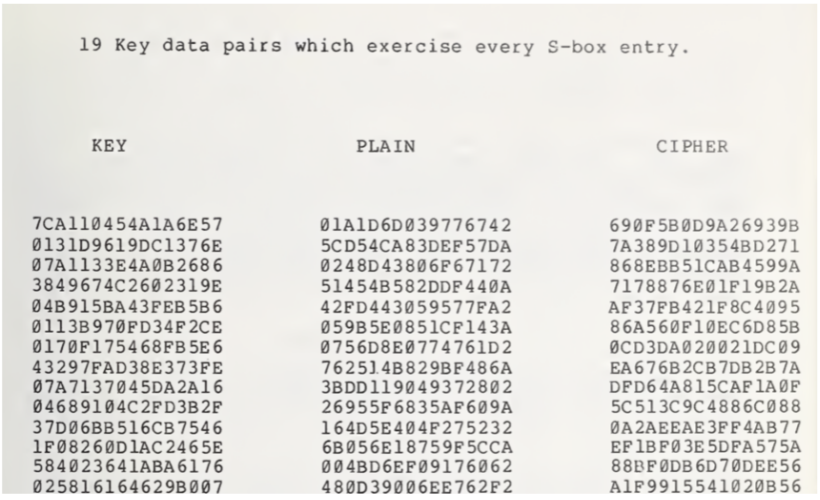
(a) Vetores de teste 1



PC1 AND PC2 TEST

KEY	PLAIN	CIPHER
8001010101010101	0000000000000000	95A8D72813DAA94D
4001010101010101	0000000000000000	0EEC1487DD8C26D5
2001010101010101	0000000000000000	7AD16FFB79C45926
1001010101010101	0000000000000000	D3746294CA6A6CF3
0801010101010101	0000000000000000	809F5F873C1FD761
0401010101010101	0000000000000000	C02FAFFEC989D1FC
0201010101010101	0000000000000000	4615AA1D33E72F10
0180010101010101	0000000000000000	2055123350C00858
0140010101010101	0000000000000000	DF3B99D6577397C8
0120010101010101	0000000000000000	31FE17369B5288C9
0110010101010101	0000000000000000	DFDD3CC64DAE1642
0108010101010101	0000000000000000	178C83CE2B399D94

(b) Vetores de teste 2



19 Key data pairs which exercise every S-box entry.

KEY	PLAIN	CIPHER
7CA110454A1A6E57	01A1D6D039776742	690F5B0D9A26939B
0131D9619DC1376E	5CD54CA83DEF57DA	7A389D10354BD271
07A1133E4A0B2686	0248D43806F67172	868EBB51CAB4599A
3849674C2602319E	51454B582DDF440A	7178876E01F19B2A
04B915BA43FEB5B6	42FD443059577FA2	AF37FB421F8C4095
0113B970FD34F2CE	059B5E0851CF143A	86A560F10EC6D85B
0170F175468FB5E6	0756D8E0774761D2	0CD3DA020021DC09
43297FAD38E373FE	762514B829BF486A	EA676B2CB7DB2B7A
07A7137045DA2A16	3BDD119049372802	DFD64A815CAF1A0F
04689104C2FD3B2F	26955F6835AF609A	5C513C9C4886C088
37D06BB516CB7546	164D5E404F275232	0A2AEEAE3FF4AB77
1F08260D1AC2465E	6B056E18759F5CCA	EF1BF03E5DFA575A
584023641ABA6176	004BD6EF09176062	88BF0DB6D70DEE56
025816164629B007	480D39006EE762F2	A1F9915541020B56

(c) Vetores de teste 3

FIG. 4.1: Vetores de teste propostos pela NBS (2)

Fonte: NBS

Para testar o SWBDES é necessário, primeiro, gerar as tabelas de consulta, T-boxes. Para isso, a chave é passada para o gerador das tabelas, LTablesGen. Recompila-se o main, já que ele utiliza as T-boxes, e depois o executamos.

Como pode ser observado na figura abaixo, FIG. 4.2, pede-se o texto em hexadecimal e o modo, ou seja decriptografar ou criptografar. O resultado do algoritmo é, então, impresso.

Ilustraremos aqui três dos vetores de testes propostos pela NBS.

No primeiro teste foi usado como vetor de teste a chave 0x0101010101010101, o texto em claro 0x95F8A5E5DD31D900 e a cifra resultante 0x8000000000000000 no modo criptografar. Executamos, também, a operação inversa, decriptografar e o resultado correspondeu ao texto em claro.

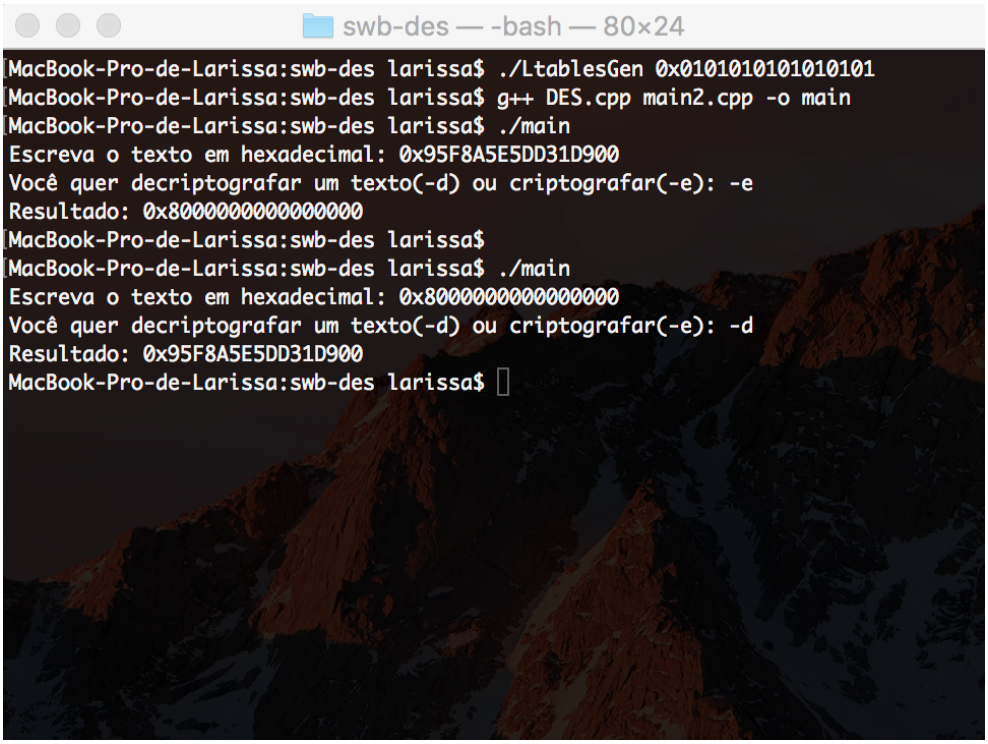


FIG. 4.2: Vetor de teste SWBDES 1

No segundo teste, FIG. 4.3, foi usado como vetor de teste a chave 0x8001010101010101, o texto em claro 0x0000000000000000 e a cifra resultante 0x95A8D72813DAA94D no modo criptografar. Como no exemplo anterior, executamos também no modo decriptografar.

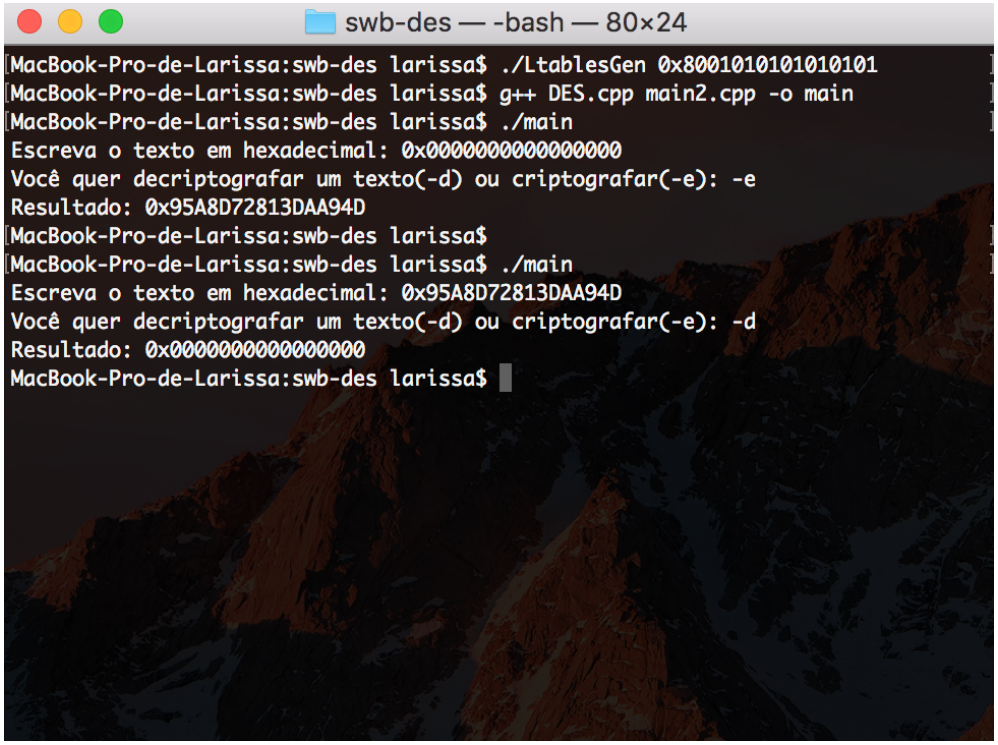


FIG. 4.3: Vetor de teste SWBDES 2

Por último, FIG. 4.4, foi usado como vetor de teste a chave 0x7CA110454A1A6E57, o texto em claro 0x01A1D6D039776742 e a cifra resultante 690F5B0D9A26939B no modo criptografar e o inverso no modo decriptografar.

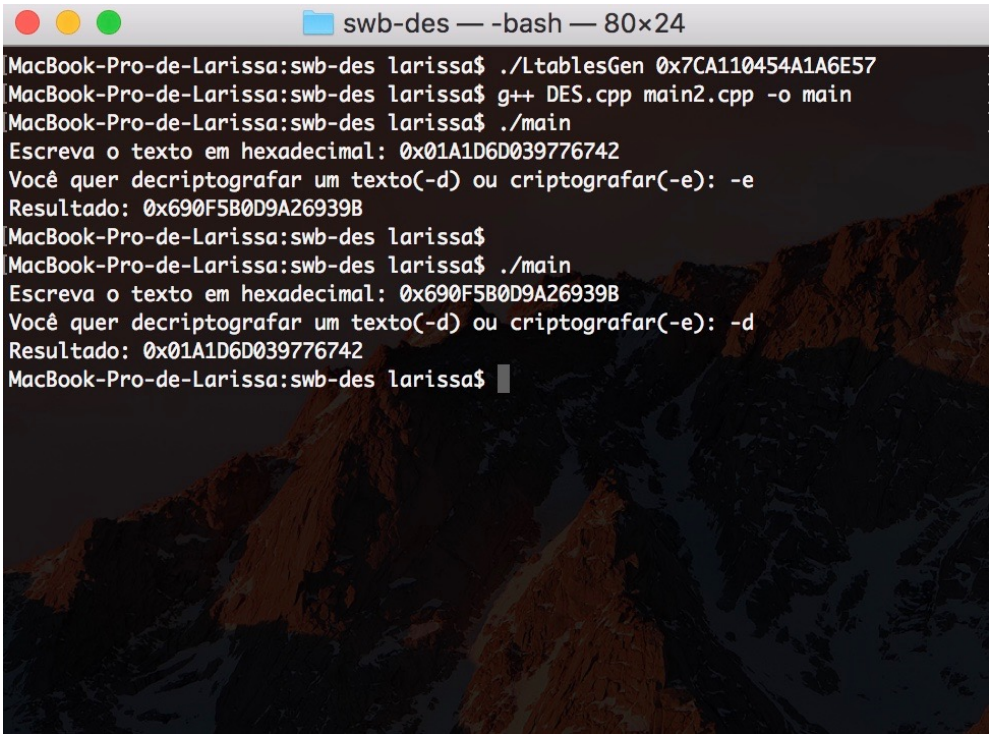


FIG. 4.4: Vetor de teste SWBDES 3

Em todos os testes acima, os resultados corresponderam com os vetores testes propostos pela NBS, FIG. 4.1.

4.2 VETORES DE TESTE PARA O SWB-3DES

O funcionamento do SWB-3DES é bastante semelhante ao do SWBDES. A principal diferença é que no lugar de uma chave de 8 bytes, utilizaremos uma chave de 16 bytes, onde os 8 primeiros correspondem à primeira chave e os 8 bytes finais à segunda chave.

O 3DES consiste em encriptar, decriptar e encriptar de novo (criptografar) ou decriptar, encriptar e decriptar (decriptografar). No modo criptografar, a primeira chave será usada nas encriptações e a segunda na decriptação. No modo decriptografar o inverso ocorre, a primeira chave será usada nas decriptações e a segunda na encriptação.

Como vetores de teste, utilizamos os propostos pelo NESSIE (3), um projeto de criptografia Europeu comparado ao NIST AES PROCESS.

```
**********************************************************************************
*Project NESSIE - New European Schemes for Signature, Integrity, and Encryption*
**********************************************************************************

Primitive Name: Triple-Des (2-Key)
==================================
Key size: 128 bits
Block size: 64 bits

Test vectors -- set 1
=====================

Set 1, vector#  0:
                         key=80000000000000000000000000000000
                       plain=0000000000000000
                      cipher=FAFD5084374FCE34
                   decrypted=0000000000000000
           Iterated 100 times=01EA5B8A6C2FB1C0
          Iterated 1000 times=ACEE9B10C53F4754

Set 1, vector#  1:
                         key=40000000000000000000000000000000
                       plain=0000000000000000
                      cipher=60CC37B7B537A1DC
                   decrypted=0000000000000000
           Iterated 100 times=2F152081D6E98061
          Iterated 1000 times=129F660736F1DA96

Set 1, vector#  2:
                         key=20000000000000000000000000000000
                       plain=0000000000000000
                      cipher=BE3E7304FE92C2BC
                   decrypted=0000000000000000
           Iterated 100 times=9BD7D49B9B986043
          Iterated 1000 times=98C611CDCB072DBB
```

(a) Vetores de teste 1

```
Set 3, vector# 50:
                         key=3232323232323232323232323232323232
                       plain=3232323232323232
                      cipher=AC978C247863388F
                   decrypted=3232323232323232
           Iterated 100 times=353E49075AC19356
          Iterated 1000 times=74986B6B48A70B9B

Set 3, vector# 51:
                         key=3333333333333333333333333333333333
                       plain=3333333333333333
                      cipher=0432ED386F2DE328
                   decrypted=3333333333333333
           Iterated 100 times=8D6B18F726B5BD30
          Iterated 1000 times=FDB76D63EF051ADD

Set 3, vector# 52:
                         key=3434343434343434343434343434343434
                       plain=3434343434343434
                      cipher=D254014CB986B3C2
                   decrypted=3434343434343434
           Iterated 100 times=085509B51C375B80
          Iterated 1000 times=31E9566BB30E081E
```

(b) Vetores de teste 2

```
Test vectors -- set 8
=====================

Set 8, vector#  0:
                         key=000102030405060708090A0B0C0D0E0F
                      cipher=0011223344556677
                       plain=E4FC19D69463B783
                   encrypted=0011223344556677

Set 8, vector#  1:
                         key=2BD6459F82C5B300952C49104881FF48
                      cipher=EA024714AD5C4D84
                       plain=8598538A8ECF117D
                   encrypted=EA024714AD5C4D84
```

(c) Vetores de teste 3

FIG. 4.5: Vetores de teste propostos pelo NESSIE (3)

Fonte: NESSIE

Como no SWBDES, ilustraremos com três vetores de teste.

No primeiro exemplo, foi utilizada a chave 0x80000000000000000000000000000000, o texto em claro 0x0000000000000000 e a cifra resultante 0xFAFD5084374FCE34 como vetor de teste no modo criptografar e o inverso (trocando chave e texto em claro) no modo decriptografar. Comparando a imagem abaixo, FIG. 4.6 com a da FIG. 4.5, observa-se que os vetores são correspondentes.

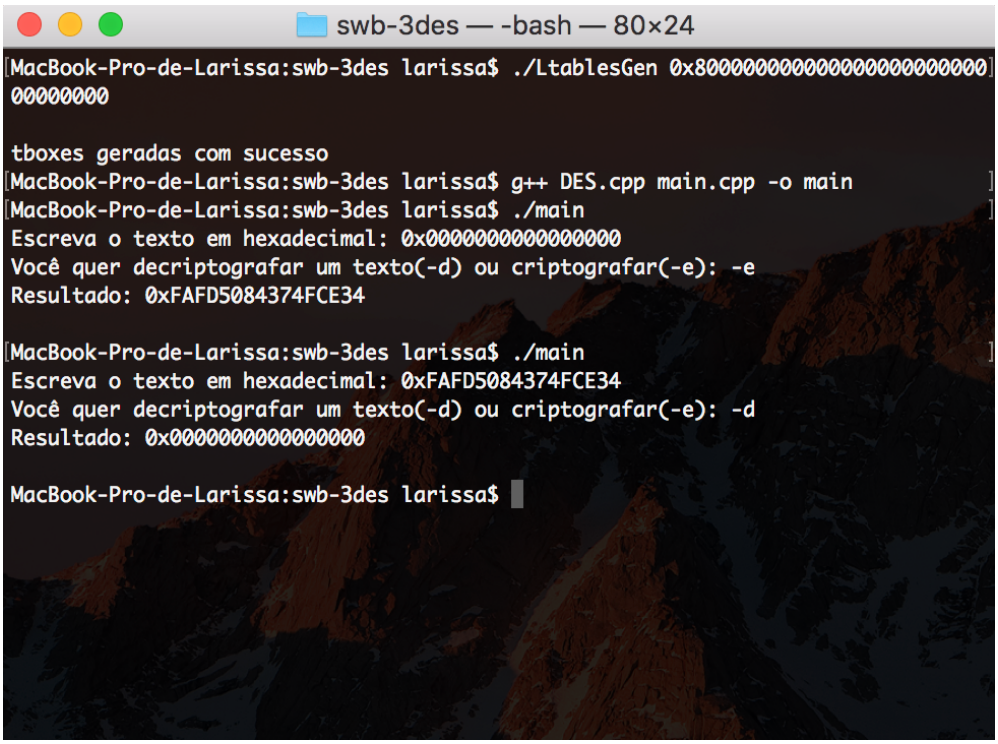


FIG. 4.6: Vetor de teste SWB-3DES 1

O segundo tem como vetor de teste a chave 0x32323232323232323232323232323232, o texto em claro 0x3232323232323232 e a cifra resultante 0xAC978C247863388F no modo encriptar e o inverso no modo decriptar. Comparando a imagem abaixo, FIG. 4.7, com a da FIG. 4.5, observa-se que os vetores são correspondentes.

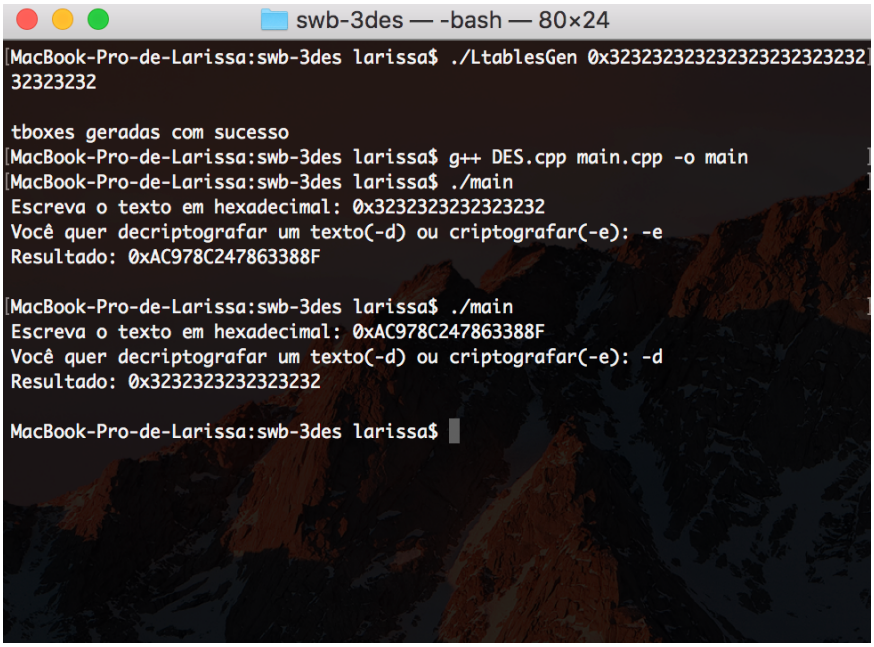


FIG. 4.7: Vetor de teste SWB-3DES 2

E por último, foi utilizado a chave 0x000102030405060708090A0B0C0D0E0F, o texto em claro 0xE4FC19D69463B783 e a cifra resultante 0xFAFD5084374FCE34 no modo criptografar e o inverso no modo decriptografar. Comparando a imagem abaixo, FIG. 4.8, com a da FIG. 4.5, observa-se que os vetores são correspondentes.

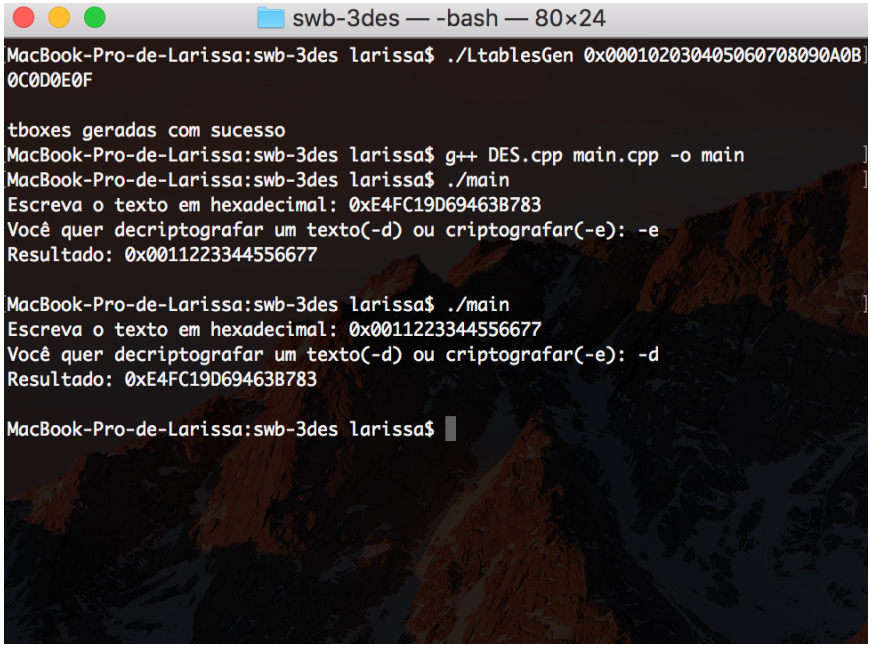


FIG. 4.8: Vetor de teste SWB-3DES 3

4.3 SWB-3DES APLICADO AO PINPAD

Após a expansão para o SWB-3DES, foi feita uma simulação de um PINPad.

Como foi explicado no capítulo 2, seção 2.5, primeiro obtém-se a *Session Key*. No nosso PINPad, para que isso ocorra, decriptografa-se a *Working Key* utilizando a *Master Key* na forma ofuscada através do algoritmo SWB-3DES.

Essa *Session Key* é utilizada como chave no 3DES, criptografando o *PIN-Block* e obtendo-se o *Ciphered PIN*. Neste caso não há uma necessidade de utilizar a versão caixa-branca, já que a *Session Key* muda periodicamente (ao mudar a *Working Key*) e ela é obtida a partir de um algoritmo caixa-branca.

Na ANSI X9.8 (5), temos um exemplo de resultados do PINPad na versão DES (FIG. 4.9 abaixo). Como o nosso algoritmo utiliza uma chave de 16 bytes, duplicamos a *Master Key* e a *Working Key*. Os resultados devem se manter, já que o 3DES, e consequentemente o SWB-3DES, com chaves igual equivale ao DES.

A *Master Key* utilizada no teste foi então 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 31 e a *Working Key* utilizada foi 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32 32. Tanto o PIN como o número do cartão (*Account*) foram mantidos.

Example:

```
Master Key      = 31 31 31 31 31 31 31 31
Working Key     = 32 32 32 32 32 32 32 32
PIN             = 123456
Account         = 1234567890123456

Intermediate PIN-Block   = 06 12 34 56 FF FF FF FF
PAN-Block                = 00 00 45 67 89 01 23 45
PIN-Block                = 06 12 71 31 76 FE DC BA

Ciphered PIN             = 02 AC 8C 05 CB 4C 86 D4
```

FIG. 4.9: Exemplo de PINPad na versão DES

Primeiro, são geradas as *lookup tables* com a *Master Key* ofuscada (FIG. 4.10), em seguida o PINPad é recompilado para que a *Master Key* seja atualizada. Em um PINPad real, essa chave é configurada na máquina, de forma que cada máquina possua uma *Master Key* própria. No nosso simulador a configuração equivale a gerar essas tabelas, e caso seja sempre utilizada a mesma *Master Key* esse passo não precisa ser executado novamente.

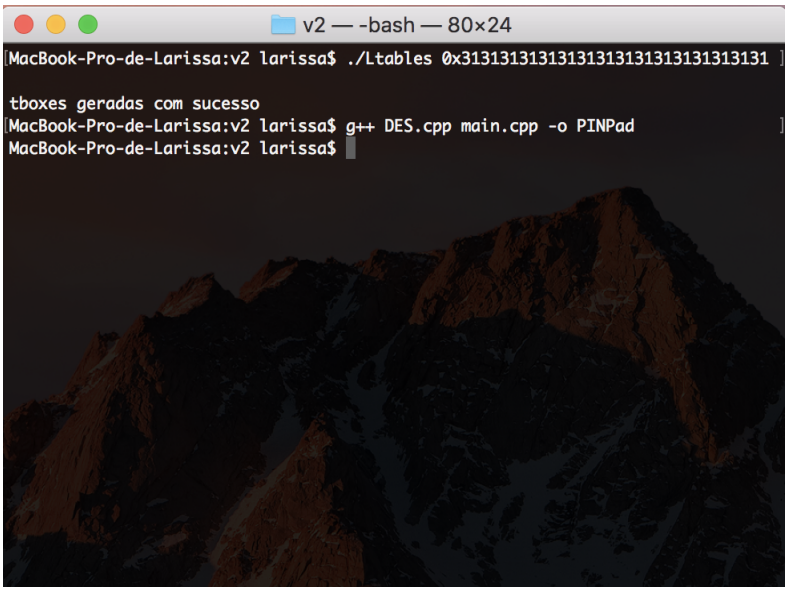


FIG. 4.10: Configuração da *Master Key*

Dando início à simulação propriamente dita, executa-se o PINPad. Um Menu é mostrado (FIG. 4.11), nele encontram-se as opções de: 1 - Passar o cartão, 2 - Atualizar a *Working Key*, 3 - Ajuda ou 4 - Sair.

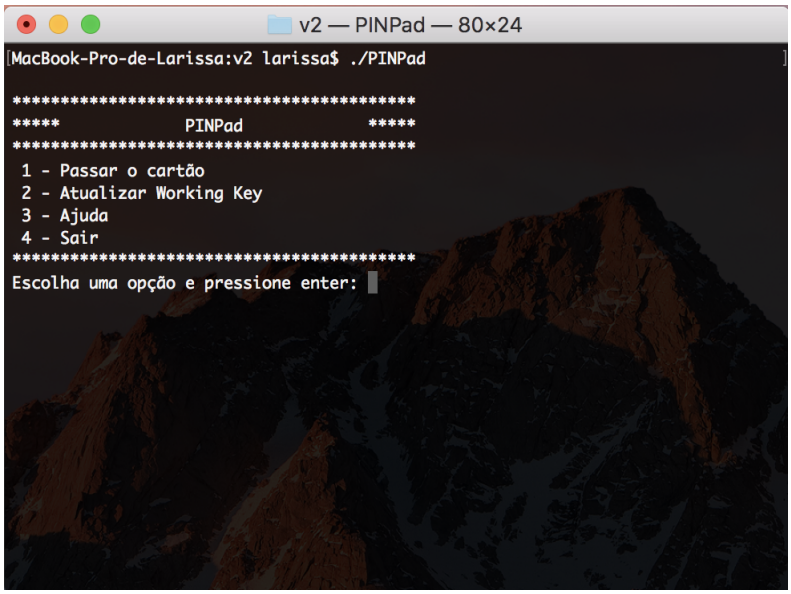


FIG. 4.11: Menu do PINPad

Continuando ao nosso teste, vamos atualizar a *Working Key* para a correspondente de 16 bytes da chave proposta na FIG. 4.9. A atualização ocorre como na FIG. 4.12 abaixo. Novamente, não é necessária a atualização toda vez que "Passar o cartão", apenas quando deseja-se trocar a *Working Key*.

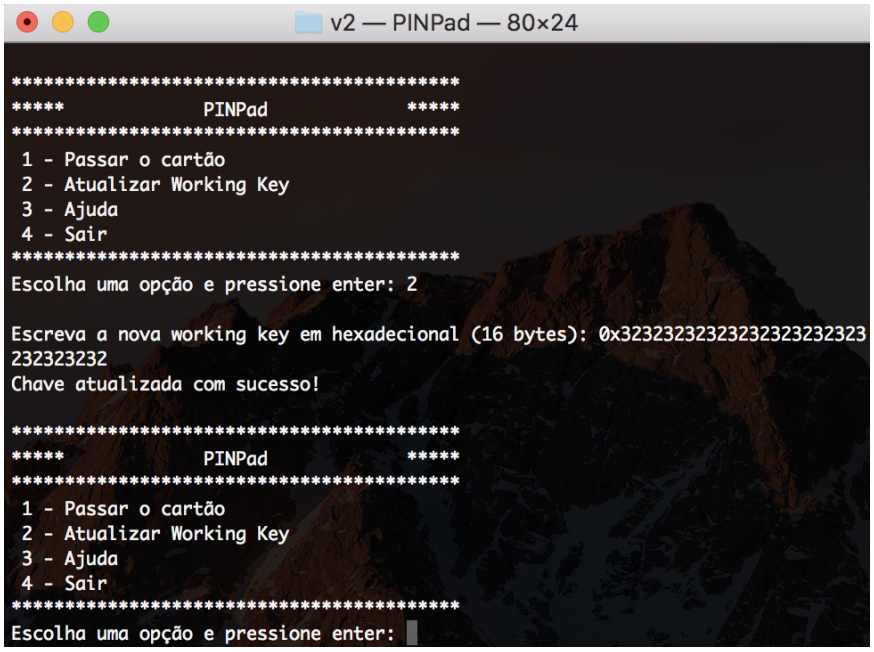


FIG. 4.12: Atualização da *Working Key*

Agora vamos simular o "Passar o cartão". Ao pressionar essa opção, são mostradas para fins didáticos a *Working Key* e a *Session Key* resultante. Em um PINPad real essa informações não são mostradas na tela, mas a *Working Key* é enviada junto com o *Ciphered PIN* ao servidor.

É pedido para que se digite o número do cartão (que seria obtido ao se inserir ou passar o cartão) e o PIN. Na FIG. 4.13 abaixo, pode-se observar que o simulador retorna o *Ciphered PIN*. Novamente, é apenas para fins didáticos de modo a facilitar a visualização, a saída real, que será enviada ao servidor, é um arquivo contendo tanto a *Working Key* como o *Ciphered PIN* (FIG. 4.14).

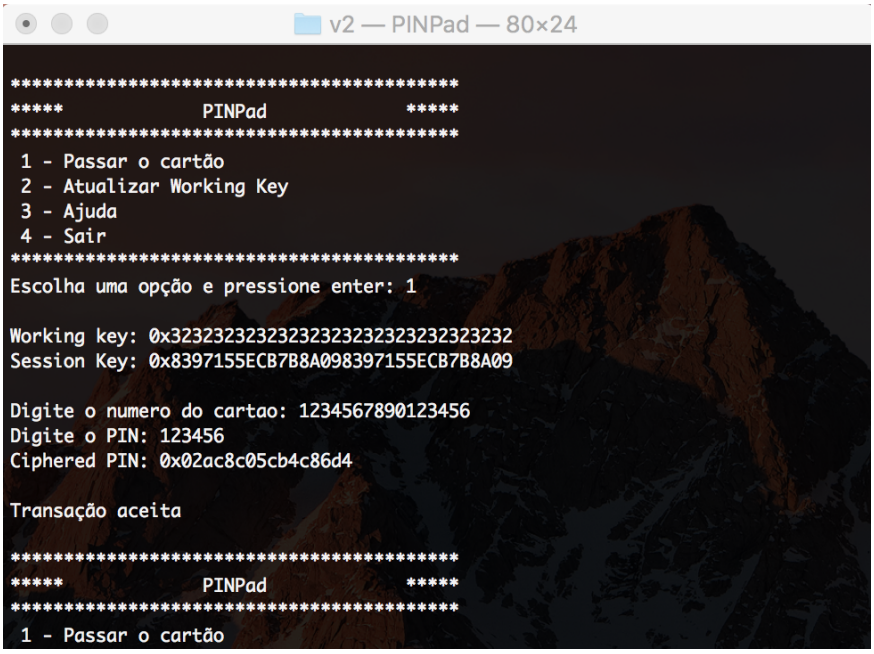


FIG. 4.13: Passar o cartão

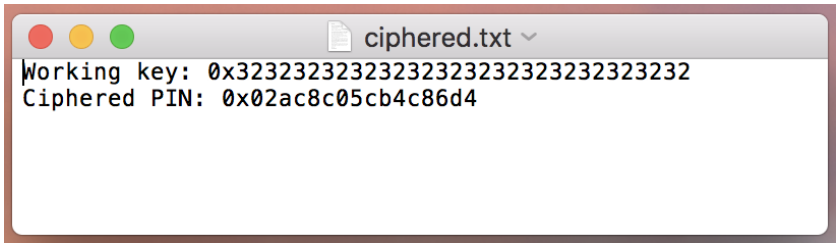


FIG. 4.14: Saída do Simulador

Abaixo, encontra-se o que obtem-se ao pressionar Ajuda (FIG. 4.15) ou Sair (FIG. 4.16). Na opção Ajuda, temos os passos para atualizar a *Master Key* ou a *Working Key* e a informação de que *Session Key* é gerada automaticamente.

A opção sair apenas encerra a simulação.

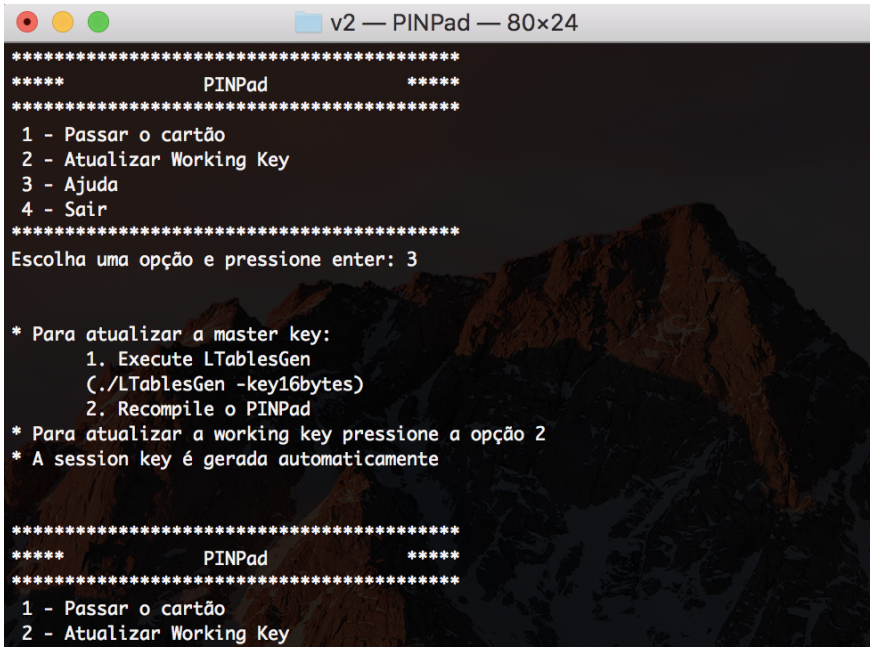


FIG. 4.15: Opção Ajuda

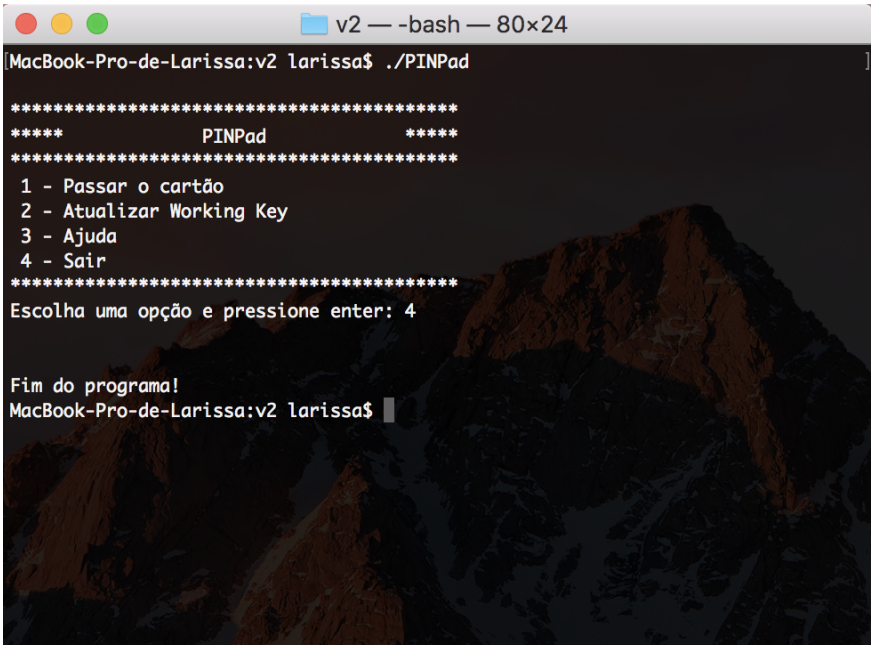


FIG. 4.16: Opção Sair

5 SEGURANÇA DE ALGORITMOS CAIXA-BRANCA

Como citado ao longo do trabalho, os algoritmos caixa-branca surgiram a partir de um esforço para se construir cifras resistentes a um ataque com acesso ao ambiente de operação.

As cifras propostas por Chow et al.(1) tinham o propósito de ser utilizadas em aplicações DRM. Os algoritmos propostos eram alterações no AES e DES e seguiram basicamente a mesma ideia: implementar os algoritmos como uma rede de *lookup tables*.

Não demorou para que as fragilidades desses algoritmos começassem a aparecer. E, como citado no trabalho de Biryukov et al.(12), essas fragilidades se deviam a forma como a chave era injetada e à alta difusão, o que facilitava os ataques.

Antes de entrar na revisão dos ataques conhecidos durante os mais de 15 anos de pesquisa, é necessário formalizar o problema, isto é, qual era o objetivo dos ataques. Muitos deles tinham como objetivo recuperar a chave (a usada em aplicações caixa-preta do DES e AES), ou seja, trabalham com a definição proposta inicialmente por Chow.

Com o surgimento da definição de Wyseur, novos tipos de ataques surgiram. Os ataques que tinham como objetivo simplesmente recuperar o texto em claro, dado que temos o texto cifrado. A esse tipo de ataque a versão caixa-branca tanto do AES quanto do DES não são seguras, pois ambos são facilmente inversíveis com as *lookup tables*.

Dessa forma, duas noções de algoritmos caixa-branca surgiram e estão sintetizadas no trabalho de Biryukov et al. (12)

Definição 3 (Implementação caixa-branca fraca): Uma implementação caixa-branca fraca tem como finalidade proteger a chave e seus derivados de ataques que têm como objetivo recuperar a chave.

Definição 4 (Implementação caixa-branca forte): Uma implementação caixa-branca forte tem como finalidade impedir a recuperação do texto em claro, dado que o atacante tem a rotina de encriptação com a chave ofuscada nas *lookup tables*.

Com essas definições, pode-se observar que a versão caixa-branca tanto do DES quanto do AES são implementações fracas, inclusive a versão simplificada implementada nesse

trabalho. As implementações fortes ainda estão sendo estudadas e podem vir a substituir sistemas de chave pública e privada.

A próxima seção será dedicada a entender as fragilidades que já foram encontradas no algoritmo proposto inicialmente por Chow e como as alterações feitas em nossa implementação afetam de fato a segurança.

Não é escopo desse trabalho fazer uma criptanálise do algoritmo, seja no de Chow et al. seja na nossa implementação. O objetivo é contextualizar como esse algoritmo pode ser de fato empregado na prática e o que está sendo estudado de melhorias no meio acadêmico.

5.1 REVISÃO DOS ATAQUES DO DES

Pouco tempo da publicação do algoritmo por Chow et al.(1), as primeiras fraquezas já apareceram. Jacob et al.(13) apresentou um ataque de inserção de erros na execução que permitia recuperar a chave.

Nesse ataque, duas fraquezas foram exploradas: a segurança dos resultados intermediários (verificou-se que não era segura o suficiente), pois estavam sujeitas a alterações. Além disso, a própria possibilidade de poder identificar as rodadas sendo executadas era por si só uma fraqueza.

A principal conclusão do trabalho de Jacob et al. foi que a ofuscação da chave no código, de fato, fornecia alguma segurança contra ataques que visavam à recuperação da chave, mas não era suficiente como única forma de proteção da chave.

A primeira melhoria no algoritmo veio em 2005, proposta por Link et al. (10). Em seu trabalho, afirmava-se que as alterações que eles propuseram protegia contra os ataques propostos por Jacob et al.(13).

Todavia, já em 2007, um ataque contra esse algoritmo surgiu. Brecht Wyseur propôs em sua tese de doutorado(7) um ataque de criptoanálise diferencial que permitia recuperar a chave. Na verdade, a grande contribuição de Wyseur foi quebrar todas as implementações caixa-branca conhecidas até então.

Esse fato é corroborado por Lin et al.(14). Em um dos mais recentes estudos, ele comenta que quase todas as implementações caixa-branca conhecidas já foram quebradas.

Em seu estudo, eles proprõem uma nova cifra caixa-branca que não é derivada de algoritmos caixa-preta. O novo algoritmo é baseada em uma rede de Feistel não balanceada e na estrutura ASASASA.

5.2 SEGURANÇA SWBDES

Como citado anteriormente, nossa implementação é uma simplificação do algoritmo de Chow et al. Por isso, está sujeita aos mesmos ataques que quebraram essa cifra. Em particular, as alterações feitas diminuiram o nível de ofuscação da chave, dessa forma, é de se esperar que o próprio ataque de Jacob et al. permita sua recuperação.

O próprio fato de não ter sido feito a tabularização das operações afins do DES é um comprometedor da segurança, pois as *lookup tables* dessas operações serviriam para confundir o atacante sobre onde buscar informações da chave. Contudo, há um detalhe importante a ser observado, as *lookup tables* não carregam informação da chave como mostrado anteriormente.

Dessa forma, inevitavelmente, as tabelas atacadas seriam as que carregam informação da chave e poderiam ser identificadas na memória por serem menos esparsas e mais aleatórias que as outras. Como o objetivo principal do trabalho era ofuscar a chave, isso foi feito somente através da tabularização das operaçõs envolvendo a chave.

6 CONCLUSÕES

Este trabalho objetivou simular o funcionamento do PINPad utilizando uma versão caixa-branca do 3DES. Para isso, foi tomado como base o algoritmo do DES proposto para ser usado nesse ambiente, que foi publicado por Chow et al. (1).

Para, de fato, implementar o DES em ambiente caixa-branca, algumas alterações foram feitas no algoritmo original, gerando uma versão simplificada do que foi originalmente proposto. Essa variante foi chamada de SWBDES. Esse algoritmo segue as principais estratégias de implementação de cifras de bloco em ambientes caixa-branca. São elas: a avaliação parcial e a tabularização.

A simplificação feita foi a não tabularização das operações afins do DES. Essa decisão foi tomada, pois as *lookup tables* dessas operações não carregam informação sobre a chave, apenas aumentam a ofuscação; e o esforço necessário fugiria ao escopo desse trabalho.

Essa simplificação diminuiu um pouco a ofuscação da chave, mas como já foi amplamente estudado: a segurança desse algoritmo, mesmo com a tabularização das operações afins, não é muito alta, pois o mesmo foi desenvolvido para ser utilizado em ambiente caixa-preta.

O trabalho atual serve de base para utilização de dispositivos abertos, como um celular, na realização de operações financeiras. O fato de estar sendo utilizado o 3DES garante a compatibilidade das transações com os hosts legados. A segurança não foi a principal preocupação, pois esta já é garantida pelo TLS, que é utilizado na rede TCP/IP para o envio do PIN cifrado.

Como sugestão de trabalho futuro, sugere - se uma biblioteca com tabularização de todas as operações do DES, além disso pode se fazer a criação de um aplicativo para dispositivos iOS e Android que simule o processo de cifra da senha, tomando como base a biblioteca implementada nesse trabalho e que simule, também, o envio do PIN na rede TCP/IP.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] S. Chow, P. Eisen, H. Johnson, and P. C. V. Oorschot, "A white-box des implementation for drm applications," *Lecture Notes in Computer Science Digital Rights Management*, p. 1–15, 2003.

[2] J. Gait, "Validating the correctness of hardware implementations of the nbs data encryption standard," 1977.

[3] I. Project NESSIE New European Schemes for Signature and Encryption, "The nessie book: Final report of european project," 2004.

[4] D. Wong, J. Monin, and H. Bonnin, "Implémentation et analyse d'une white-box du des," 2014.

[5] "Ansi x9.8, master-session pin pad cryptography," 2003. American National Standards Institute, X9-Financial Services.

[6] B. Wyseur, "White-box cryptography: Hiding keys in software," *NAGRA Kudelski Group, Switzerland*, 2012.

[7] B. Wyseur, "White-box cryptography," *Katholieke Universiteit Leuven – Faculteit Ingenieurswetenschappen Arenbergkasteel, B-3001 Heverlee (Belgium)*, 2009.

[8] W. Michiels, P. Gorissen, and H. D. L. Hollmann, "Cryptanalysis of a generic class of white-box implementations," *Selected Areas in Cryptography Lecture Notes in Computer Science*, p. 414–428, 2009.

[9] Des, "Data encryption standard," in *In FIPS PUB 46, Federal Information Processing Standards Publication*, pp. 46–2, 1977.

[10] H. Link and W. Neumann, "Clarifying obfuscation: improving the security of white-box des," *International Conference on Information Technology: Coding and Computing (ITCC05) - Volume II*, 2005.

[11] L. Goubin, J.-M. Masereel, and M. Quisquater, "Cryptanalysis of white box des implementations," *Selected Areas in Cryptography Lecture Notes in Computer Science*, p. 278–295.

[12] A. Biryukov, C. Bouillaguet, and D. Khovratovich, "Cryptographic schemes based on the asasa structure: Black-box, white-box, and public-key (extended abstract)," *Lecture Notes in Computer Science Advances in Cryptology – ASIACRYPT 2014*, p. 63–84, 2014.

[13] M. Jacob, D. Boneh, and E. Felten, "Attacking an obfuscated cipher by injecting faults," *Lecture Notes in Computer Science Digital Rights Management*, p. 16–31, 2003.

[14] T.-T. Lin, X.-J. Lai, W.-J. Xue, and Y. Jia, "A new feistel-type white-box encryption scheme," *Journal of Computer Science and Technology*, vol. 32, no. 2, p. 386–395, 2017.