Criptografia na Segunda Guerra Mundial: Uma Abordagem Computacional em Linguagem C

Autores: Taylor Rodrigues, Vitor Almeida, Vitor Sousa

Resumo

Durante a Segunda Guerra Mundial, a criptografia foi um elemento crucial para a comunicação segura entre as nações. Este artigo apresenta uma abordagem computacional inspirada nas técnicas criptográficas utilizadas no período, com ênfase na construção de um simulador em linguagem C. O programa é baseado na conversão de pares hexadecimais e em uma função polinomial de grau sete para decodificação condicional de mensagens. São apresentados os métodos utilizados na implementação, os resultados obtidos com testes práticos e a relevância histórica e técnica do projeto. A proposta alia conceitos históricos e fundamentos matemáticos, permitindo compreender melhor a aplicação da lógica de programação em contextos de segurança da informação.

1. Introdução

A criptografia tem sido utilizada há séculos como meio de proteger informações sensíveis, sendo particularmente importante em contextos militares. Durante a Segunda Guerra Mundial, dispositivos como a máquina Enigma evidenciaram a sofisticação dos métodos criptográficos utilizados. Inspirado nesse contexto, este trabalho visa desenvolver um programa que simule um processo de decodificação simples, utilizando pares hexadecimais e lógica computacional. O objetivo é aproximar os estudantes dos conceitos fundamentais da criptografia, por meio de uma aplicação prática em linguagem C.

2. Metodologia

2.1 Estrutura do Programa

O programa foi desenvolvido na linguagem C e está estruturado em três principais partes:

1. Inclusão de bibliotecas padrão.
2. Definição da função polinomial de decodificação.
3. Estrutura principal com entrada de dados, decodificação e repetição.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

As bibliotecas utilizadas permitem realizar entrada e saída de dados (stdio.h), conversão (stdlib.h), manipulação de strings (string.h) e cálculos matemáticos (math.h).

2.2 Função Polinomial

A função Funcao_decodificadora aplica um polinômio de grau 7 para avaliar cada posição de caractere:

```
double Funcao_decodificadora(int x, float b) {
    double a0 = 348.11, a1 = -25.91, a2 = 13.52, a3 = -4.8;
    double a4 = 1.2, a5 = -0.2, a6 = 0.03, a7 = -0.001;
    return a0 + (a1 + b) * x + a2 * pow(x,2) + a3 * pow(x,3)
            + a4 * pow(x,4) + a5 * pow(x,5) + a6 * pow(x,6) + a7 * pow(x,7);
}
```

O parâmetro x representa a posição do caractere na mensagem, enquanto b é uma variável ajustável entre -5 e 5, que simula uma chave adicional de decodificação.

2.3 Lógica Principal

O programa solicita ao usuário a quantidade de mensagens, o valor de b, e a mensagem codificada em formato hexadecimal. Em seguida, divide a string em pares de caracteres, converte para decimal e aplica a função decodificadora.

```
for(i = 0; i < strlen(codigo); i += 2){
    par[0] = codigo[i];
    par[1] = codigo[i + 1];
    par[2] = '\0';

    int valor = strtol(par, NULL, 16);
    resultado = Funcao_decodificadora(posicao, b);

    if (resultado >= 0) {
        printf("%c", valor);
    }
}
```

A repetição continua conforme a escolha do usuário.

2.4 Análise dos Limites de Tipos de Dados

Como complemento à função de decodificação, foi desenvolvido um trecho de código para ilustrar os limites dos tipos de dados primários na linguagem C. Esse recurso tem como objetivo garantir a segurança na manipulação de dados, prevenindo problemas como estouros de valor (overflow) durante a execução do programa.

O código utiliza a biblioteca <limits.h>, que fornece constantes definidas para os valores mínimo e máximo permitidos pelos principais tipos de dados, como char, int, short, long e suas versões não assinadas. O programa exibe esses limites de maneira clara, facilitando a análise do programador sobre os intervalos de valores que cada tipo pode suportar.

Adicionalmente, o código demonstra o comportamento do overflow, um fenômeno que ocorre quando um valor excede a capacidade de armazenamento de uma variável. Por exemplo, ao atribuir o valor 255 a uma variável do tipo unsigned char (que tem 255 como valor máximo), e em seguida somar 1 a esse valor, o resultado retorna a 0, ilustrando o ciclo de valores (overflow) típico em variáveis sem sinal. Esse conceito é de grande relevância no campo da criptografia e da segurança da informação, uma vez que falhas na manipulação dos dados podem resultar em vulnerabilidades críticas.

Esse exemplo enfatiza a importância de compreender os limites dos tipos de dados, especialmente em sistemas que envolvem codificação e decodificação de informações sensíveis, onde a precisão e o controle rigoroso dos dados são essenciais.

3. Resultados

Durante os testes, foram utilizadas diversas mensagens com diferentes valores de b no intervalo de -5 a 5. As mensagens foram parcialmente decodificadas, dependendo do resultado retornado pela função. A variação do parâmetro b afetou diretamente a quantidade de caracteres revelados, simulando a presença de uma chave criptográfica variável.

4. Discussão

Os resultados demonstram como alterações nos parâmetros de entrada influenciam a decodificação. A estrutura do programa permite simular características importantes da criptografia real: controle de acesso por chave (parâmetro b), uso de pares hexadecimais (codificação de mensagens) e filtragem por função matemática (validação condicional). Apesar de didático, o programa fornece base conceitual para introduzir conceitos de segurança e criptografia digital.

5. Conclusão

Este trabalho proporcionou uma integração entre fundamentos históricos e computacionais da criptografia. A implementação do simulador em C permitiu compreender a lógica por trás da codificação de mensagens, utilizando matemática e programação. A experiência reforça a importância de práticas interdisciplinares no ensino de lógica e algoritmos.

Referências

SINGH, S. *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography*. Anchor, 1999.

STALLINGS, W. *Cryptography and Network Security: Principles and Practice*. Pearson, 2017.

KERNIGHAN, B.; RITCHIE, D. *The C Programming Language*. Prentice Hall, 1988.

Apêndice

Repositório Git com o código completo do projeto:

https://github.com/TAYLOR-RD/Trabalho_De_Logica