Proba practica
CECOI PAVEL

Biletul 8.

1. Enumeraţi condiţiile funcţionării criptosistemului DES.

   Algoritmul DES este o combinaţie complexă, folosind două blocuri fundamentale în criptografie: substituţia şi permutarea (transpoziţia). Acest cifru bloc acceptă un bloc de 64 de biţi la intrare şi generează un bloc cifrat de 64 de biţi. DES este un algoritm simetric. Acelaşi algoritm şi aceeaşi cheie sunt folosiţi atât la criptare cât şi la decriptare. Algoritmul este constituit din 16 cicluri repetate ale blocurilor fundamentale. Textul iniţial este descompus în blocuri de 64 de biţi. Cheia este de 64 biţi din care doar 56 sunt efectivi, ceilalţi fiind biţi de paritate. Folosirea substituţiei provoacă confuzie prin sistematica substituire a unor biţi cu alţii. Transpoziţiile provoacă difuzie prin re-ordonarea biţilor. Algoritmul foloseşte numai operaţii aritmetice şi logice clasice cu număr de până la 64 de biţi, ceea ce face relativ uşor de implementat atât software cât mai ales hardware: unul din scopurile declarate ale algoritmului fiind uşoara lui implementare hardware într-un cip specializat. Parcurgerea celor 16 cicluri.

2. Modificaţi meniul specific aplicaţiei criptografice astfel încât să interacţioneze cu shell-ul în modul următor: în cazul introducerii doar a numelui aplicaţiei sau a numelui să afişeze „nameofapp: no option and message”, în cazul introducerii doar a opţiunilor să afişeze „nameofapp: no option”, în cazul introducerii doar a mesajului să afişeze „nameofapp: no message”, în cazul introducerii şi a opţiunilor şi a mesajului necriptat sau al celui criptat aplicaţia să pornească funcţia respectivă ce criptează sau decriptează mesajul iar în cazul unor opţiuni inexistente să afişeze „nameofapp: no such argument: nameofargument”. La fel trebuie de implementat şi o metodă de standardizare a erorilor. Opţiunile existente vor fi „-cript” pentru criptare şi „-decript” pentru decriptare. Meniul trebuie plasat în fişierul „main.c”.

Programul:

```
#inlcude <iostream>
#include <string.h>
#include "custom_fluid.h"


int main(int argc, char "argc[ ])
```

```
{
if (arfc == 1) {
std::cout<<argv[0]) << “: No option and message\n”;
Return 0;


}
Std::string option = argv[1];
 If (argc == 2 && option.else() ==2) {
std::cout<<argv[0] << “: no message\n”;
Return 0;


}


Std::string word=argv[2];


If (option == “-c”)


{
Std::cout << “criptat: “ << fluid_encrypt(word) << \n”;
}
else  if (option == “-d”)
{
std::cout<< “Decriptat: “ <<fluid_encrypt(word) <<\n” ;
}
else
{
Std::cout <<argv[0] << “: no such argument” \ “” << option << “\| \n”;
}


Return 0;
}
```

3. Creaţi o metodă prin combinarea criptosistemului CAESAR cu VIGENERE, astfel încât mai întâi să cripteze cu primul apoi cu al doilea. Modalitatea de implementare să fie următoarea: funcţia „vigenere_encrypt” să funcţioneze după algoritmii menţionaţi mai

jos, iar funcţia „vigenere_decrypt” trebuie creată din cea de criptare. encrypt=(MSGk+seed+KEYk+seed)%62; | unde k=(i+cezar_key)%62; | unde seed=timestamp; Dacă mesajul este „NuPotVeniAzi”, cheia vigenere „Hello”, cheia cezar „7” şi seed-ul „1576089612”, atunci mesajul criptat va fi „ICoDLQwC7cu0”. Metoda trebuie să fie valabilă pentru toate cazurile posibile sau cel puţin φ=0,0012, şi evident fără erori de procesare. Funcţiile implementaţi-le în fişierul „new2_vigenere.c” şi intercalează-le cu fişierul „main.c” din exerciţiul precedent.

```
#include "text_analyser.h"
	#include "buffer.h"
	#include "math.h"
	#include
	#include

	void GetFrequencies(text_stats_t* text_stats, buffer_t* text, long int step, long int shift)
	{

		unsigned long int i;

		if (text->size == 0)
		{
			printf("Analyse des frequences d'un texte vide impossible !\n");
			exit(-1);
		}

		for (i = 0; i < 26; i++)
		{
			text_stats->cnt[i] = 0;
		}
		text_stats->size = 0;

		for (i=shift; i < text->size; i += step)
		{
			text_stats->cnt[(unsigned char)text->content[i]]++;
			text_stats->size ++;
		}

		for (i=0; i < 26; i++)
		{
			text_stats->freq[i] = (float) text_stats->cnt[i] / (float) text_stats->size;
		}
	}
```

```
	void GetCoincidence(text_stats_t* stats)
	{
		int i;

		stats->coincidence = 0;
		for (i = 0; i < 26; i++)
		{
			stats->coincidence += ((float) stats->cnt[i]/stats->size) * ((float)
(stats->cnt[i]-1) / (stats->size - 1));
		}
	}

	// Affiche un comparatif de fréquences
	void DrawFrequencies(float freq[26], float ref_freq[26], int height)
	{
		float max = 0;
		int heights[26], heights_ref[26];
		unsigned char c;
		int x, y;
		div_t div_result;

		// On cherche la fréquence maximum
		for (c = 0; c < 26; c++)
		{
			if (freq[c] > max) max = freq[c];
			if (ref_freq[c] > max) max = ref_freq[c];
		}

		// On calcule la hauteur de chaque 'barre'
		for (c = 0; c < 26; c++)
		{
			heights[c] = round_p(freq[c] / max * (float) height);
			heights_ref[c] = round_p(ref_freq[c] / max * (float) height);
		}

		// Dessin du graphe
		for (y = 0; y < height; y++)
		{
			for (x = 0; x <= 3*26-1; x++)
			{
				div_result = div(x,3);
				if (div_result.rem == 0)
				{
					if (heights[div_result.quot] >= height - y)
```

```
                                {
                                        printf("#");
                                } else {
                                        printf(" ");
                                }
                        }
                        else if (div_result.rem == 1 )
                        {
                                if (heights_ref[div_result.quot] >= height - y)
                                {
                                        printf("|");
                                } else {
                                        printf(" ");
                                }
                        }
                        else printf(" ");
                }
                printf("\n");
        }

        // Affichage des lettres
        for (x = 0; x <= 3*26-1; x++)
        {
                div_result = div(x,3);
                if (div_result.rem < 2)
                {
                        printf("%c", 65 + div_result.quot);
                } else printf(" ");
        }
        printf("\n");
}
```