

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Robert Manestar

Josip Petanjek

Sabina Pintar

DOKUMENTACIJA ZA PROJEKT
CRYPTO++

PROJEKTNI RAD

Varaždin, 2018.

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
V A R A Ž D I N

Robert Manestar, 45156/16-R

Josip Petanjek, 45037/16-R

Sabina Pintar, 44926/16-R

Studij: Informacijski sustavi

DOKUMENTACIJA ZA PROJEKT
CRYPTO++

PROJEKTNI RAD

Mentor:

Doc.dr.sc. Nikola Ivković

Varaždin, listopad 2018.

Sadržaj

1. Uvod	1
2. Instalacija biblioteke Crypto++ (operacijski sustav Linux)	2
3. AES.....	3
3.1. AES – Opis	3
3.2. Prevođenje i pokretanje programa	4
3.3. AES - program.....	5
4. RSA	6
4.1. RSA – Opis.....	6
4.2. RSA – Program.....	7
5. Hash.....	9
5.1. Hash – Opis	Pogreška! Knjižna oznaka nije definirana.
5.2. Hash – Program	9
6. TCP veza server – klijent sa AES enkripcijom i dekripcijom	12
6.1. Klijent – program	12
6.2. Server - program	14
7. Zaključak	16
8. Literatura	18

1. Uvod



Slika 1 Crypto++ Logo

Crypto++ (također poznata i kao CryptoPP, libcrypto++ i libcryptopp) besplatna je i open source C++ biblioteka kriptografskih funkcija (kriptografskih algoritama i shema) koje je napisao Wei Dai, a danas ju održava zajednica volontera.

Crypto++ većinom ima primjenu na fakultetima, open source i nekomercijalnim projektima, ali također i u tvrtkama. Biblioteka je izdana 1995. godine, a prilikom izrade obraćena je pozornost i na prenosivost na različite računalne platforme. Dakle, biblioteka nije ovisna o operacijskom sustavu, a također su podržani i little-endian i big-endian sustavi, te 32-bitne i 64-bitne arhitekture.

Biblioteka je organizirana u nekoliko dijelova koji pokrivaju različita područja kriptografije, a ovdje će detaljnije biti objašnjeni:

- Simetrični algoritmi za kriptiranje (block, stream ciphers)
- Asimetrični algoritmi za kriptiranje (public-key algorithms)
- Hash funkcije

Crypto++ obično osigurava potpune kriptografske implementacije, a često uključuje i manje popularne i rjeđe korištene sheme. Na primjer, algoritmi koji su uključeni u biblioteku su Camellia, koja je otprilike ekvivalentna algoritmu AES (eng. Advanced Encryption Standard) te Whirlpool, hash funkcija otprilike ekvivalentna SHA algoritmu.

Biblioteka također omogućava ostvarivanje jednostavnih operacija kao što su višestruka preciznost cjelobrojnog tipa podataka (eng. multi-precision integers), generiranje i verifikacija prostih brojeva, aritmetika konačnih polja, eliptične krivulje i operacije s polinomima.

2. Instalacija biblioteke Crypto++ (operacijski sustav Linux)

Najjednostavnija opcija za instalaciju na računalu koje koristi operacijski sustav Windows je instalirati Virtual Box i unutar njega instalirati Ubuntu.

Zatim je potrebno otvoriti terminal i ažurirati popis dostupnih komponenti za ažuriranje, a to ćemo napraviti ako u terminalu upišemo sljedeće:

```
sudo apt-get update
```

- ***sudo*** (*eng. superuser do*) – pišemo kada su nam potrebne administracijske ovlasti
- ***apt-get*** (*eng. application get*) – naredba za instalaciju programa. Na Ubuntu se programi uglavnom nalaze na online bazi (repozitoriju) te ih je moguće skinuti (download) direktno iz terminala
- ***update*** – koristi se za dovođenje programa na najnoviju verziju, ukoliko se ne navede ime paketa (kao u našem slučaju), ova naredba će ažurirati sve programe koji su instalirani

Nakon upisivanja te naredbe bit će potrebno unijeti lozinku, i za sve sljedeće *sudo* naredbe koje upisujemo u tom terminalu u roku od 15 minuta bit će zapamćena lozinka, stoga je neće trebati ponovno upisivati.

Nakon toga je potrebno u terminal upisati:

```
sudo apt-get install libcrypto++-dev libcrypto++-doc libcrypto++-utils
```

Obično se instalacija radi upisivanjem naredbe *apt-get install ime-paketa*, a ime paketa se može pronaći na mjestu na kojem se nalazi program. Također je prilikom korištenja ove naredbe potrebno imati administracijske ovlasti, stoga koristimo *sudo*.

Nakon upisivanja ove naredbe izvršit će se instalacija Crypto++ biblioteke koja je zatim spremna za korištenje. Za korištenje u drugim razvojnim okruženjima i operacijskim sustavima, cijeli „library“ crypto++ dostupan je na <https://www.cryptopp.com/> ^[1]

3. AES

AES (eng. *Advanced Encryption Standard*), izvorno nazvan Rijndael, je simetrični blok algoritam s javnim izvornim tekstom programa odabran od NIST-a (eng. National Institute of Standards and Technology) kao novi standard za kriptiranje podataka, nakon što je 1997. raspisao natječaj za napredni kriptosustav koji neće imati mane DES-a^[2]. U nastavku ćemo kratko opisati kako je AES izveden te pokazati na primjeru (programu) kako radi.

3.1. AES – Opis

AES je simetrični kriptosustav, a simetrični kriptosustavi zahtijevaju da pošiljatelj i primatelj znaju zajednički tajni ključ.

Ulazni i izlazni blokovi AES algoritma su duljine 128 bitova, dok duljina ključa može biti 128, 192 ili 256 bitova. Broj koraka ovisi o veličini bloka podataka i veličini ključa - za AES, koristi se 10+1 koraka.

AES šifrira blokove od 128 bitova (16 bajtova). Svaki takav blok se može shvatiti kao 16 elemenata polja koji se reprezentiraju pomoću 4×4 matrice s elementima iz $GF(2^8)$. Tu matricu zovemo aes-blok. Za primjer možemo uzeti inačicu ključa koji ima 128 bitova, te 10 rundi, a svaka runda sastoji se od 4 operacije:

- SubBytes – nelinearna supstitucija gdje se svaki element aes-bloka zamjeni inverzom u $GF(2^8)$ pomoću S-kutije,
- ShiftRows – ciklički pomak elemenata i-tog retka aes-bloka za i mjesta ulijevo,
- MixColumn – svaki stupac bloka se tretira kao polinom nad $GF(2^8)$ i množi polinomom modulo $x^4 + 1$,
- AddRoundKey – ključ koraka dodaje se bloku XOR operacijom

Dekripcija se provodi analogno enkripciji, samo što se koraci provode u obrnutom redoslijedu, i koriste se inverzni ključevi.

3.2. Prevođenje i pokretanje programa

Kako bi pokrenuli program napisan u jeziku C++, u kojem smo koristiti biblioteku Crypto++ za kriptiranje pomoću AES algoritma, najprije je potrebno pokrenuti Virtual Box u kojem smo biblioteku instalirali te unutar nje, na po želji odabrano mjesto, spremiti program s nastavkom .cpp.



Slika 2 Spremanje datoteke s nazivom „AESprimjer.cpp” na Desktop

Zatim se u terminalu pozicioniramo u mjesto na koje smo spremili .cpp datoteku pomoću naredbe `cd lokacija/` te upišemo sljedeću naredbu, nakon koje pokrenemo kompajliranje sa

`g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o aevs2 aevs2.cpp -lcryptopp` te pokretanje sa

`./aevs2`

```
osboxes@osboxes:~/Desktop$ g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o aevs2 aevs2.cpp -lcryptopp
aevs2.cpp: In function 'int main()':
aevs2.cpp:35:47: warning: comparison between signed and unsigned integer expressions [-Wsign-compare]
osboxes@osboxes:~/Desktop$ ./aevs2
Unesi tekst za AES enkripciju i dekripciju: 12345 1233
Obični tekst (10 bitova): 12345 1233

Kriptirani tekst - hex (16 bitova): 0xf9 0xc6 0x9b 0x94 0x89 0xdb 0x62 0x9f 0x5f 0x3b 0x9c 0xe5 0x34 0x72 0xca 0x27
Dekriptirani tekst: 12345 1233
```

Slika 3 Prevođenje i pokretanje programa (kriptiranje i dekriptiranje teksta)

Sam program je vrlo jednostavan, traži se unos teksta nad kojim se provodi enkripcija i dekripcija – korištenjem funkcija, klasa i objekata iz **crypto++**. Uneseni tekst se kriptira (korištenjem *CBC* (engl. *cipher block chaining*) mod operacije^[3]), ispiše mu se „hex“ vrijednost, dekriptira se te isti ispiše.

3.3. AES - program

```
1. #include <iostream>
2. #include <iomanip>
3.
4. #include "cryptopp/modes.h"
5. #include "cryptopp/aes.h"
6. #include "cryptopp/filters.h"
7.
8. using CryptoPP::AES;
9. using namespace std;
10.
11. int main() {
12.
13.     byte kljuc[ AES::DEFAULT_KEYLENGTH ], iv[ AES::BLOCKSIZE ];
14.     memset( kljuc, 0x00, AES::DEFAULT_KEYLENGTH ); //ključ
15.     memset( iv, 0x00, AES::BLOCKSIZE ); //inicijalni vektor
16.
17.     // Obicni tekst
18.     string obicnitekst="";
19.     cout<<"Unesi tekst za AES enkripciju i dekripciju: ";
20.     getline(cin,obicnitekst);
21.     cout << "Obicni tekst ( " << obicnitekst.size() << " bytes): ";
22.     cout << obicnitekst;
23.     cout << endl << endl;
24.
25.     // Kriptirani tekst
26.     string kriptirantekst;
27.     AES::Encryption aesEncryption(kljuc, AES::DEFAULT_KEYLENGTH); //konkretna enkripcija
28.     CryptoPP::CBC_Mode_ExternalCipher::Encryption cbcEncryption( aesEncryption, iv ); //cipher
        block chaining - mod operacije
29.
30.     CryptoPP::StreamTransformationFilter stfEncryptor(cbcEncryption, new CryptoPP::StringSink(
        kriptirantekst ) ); //postavljanje enkriptora
31.     stfEncryptor.Put( reinterpret_cast<const unsigned char*>( obicnitekst.c_str() ), obicnitek
        st.length() + 1 );
32.     stfEncryptor.MessageEnd();
33.
34.     cout << "Kriptirani tekst - hex ( " << kriptirantekst.size() << " bytes): ";
35.     for( int i = 0; i < kriptirantekst.size(); i++ ) {
36.         cout << "0x" << hex << (0xFF & static_cast<byte>(kriptirantekst[i])) << " ";
37.     }
38.     cout << endl << endl;
39.
40.     // Dekriptirani tekst
41.     string dekriptirantekst;
42.     AES::Decryption aesDecryption(kljuc, AES::DEFAULT_KEYLENGTH);
43.     CryptoPP::CBC_Mode_ExternalCipher::Decryption cbcDecryption( aesDecryption, iv );
44.
45.     CryptoPP::StreamTransformationFilter stfDecryptor(cbcDecryption, new CryptoPP::StringSink(
        dekriptirantekst ) );
46.     stfDecryptor.Put( reinterpret_cast<const unsigned char*>( kriptirantekst.c_str() ), kript
        irantekst.size() );
47.     stfDecryptor.MessageEnd();
48.
49.     cout << "Dekriptirani tekst: ";
50.     cout << dekriptirantekst;
51.     cout << endl << endl;
52.
53.     return 0;
54. }
55. // g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o aesv2 aesv2.cpp -lcryptopp
56. // ./aesv2
```


4. RSA

RSA (*Rivest–Shamir–Adleman*) je djelo Ron Rivesta, Adi Shamira i Leonard Adlemana. Temelji se na problemu cjelovitih faktora. Sustav je razvijen 1977. i patentiran je od strane Massachusetts Institute of Technology. ^[4] U nastavku ćemo kratko opisati kako je RSA izveden te pokazati na primjeru (programu) kako radi.

4.1. RSA – Opis

RSA spada pod asimetrične krypto sustave, pošiljatelj i primatelj ne dijele isti tajni ključ. Javni ključ se koristi za kriptiranje i dostupan je svima. Privatni ključ se koristi za dekriptiranje i poznat je samo primatelju. ^[6]

Ukratko, RSA koristi dva ključa - javni ključ i privatni ključ. Pod RSA, javni ključ je $\{n, e\}$, a privatni ključ je $\{n, e, d\}$. e je javni eksponent, a d je privatni eksponent. Javni je ključ za masovnu potrošnju i obično je objavljen ili dostupan. Privatni ključ treba ostati tajna.

U RSA, enkripcija je jednostavno $c = m^e$. Dakle, naša je zadaća kodirati niz kao „integer“ u pripremi za enkripciju m – *tajna riječ*, c – *kriptirana riječ*.

Dekripcija se izvodi podizanjem c do privatnog eksponenta ($m = c^d$). Budući da je privatni eksponent d dio privatnog ključa, privatni ključ mora se koristiti za poništavanje operacije šifriranja. ^[5]

U našem programu koristimo generirane ključeve pomoću pseduoslučajnih brojeva, ali je prikazano i drugo rješenje s prije spomenutim $\{n, e, d\}$ parametrima. Nadalje generirani modul, privatni te javni eksponent ispisujemo i tražimo unos „tajne“. Unos pretvaramo u „integer“ te ga sa :

```
pubKey.ApplyFunction(tajna_rijec);
```

enkriptiramo (uz pomoć javnog ključa), suprotno tome, dekriptiramo (uz privatni ključ) ga pomoću :

```
dekriptirana_rijec = privKey.CalculateInverse  
(pseudo_slucajan_generiran_broj, enkriptirana_rijec);
```

te ispisujemo rezultat, dekriptiranu tajnu riječ.

4.2. RSA – Program

```
1. #include <iostream>
2. #include <iomanip>
3. using namespace std;
4.
5. #include <iomanip>
6. using std::hex;
7.
8. #include <string>
9. using std::string;
10.
11. #include "cryptopp/rsa.h"
12. using CryptoPP::RSA;
13.
14. #include "cryptopp/integer.h"
15. using CryptoPP::Integer;
16.
17. #include "cryptopp/osrng.h"
18. using CryptoPP::AutoSeededRandomPool;
19.
20. int main(int argc, char** argv)
21. {
22.     AutoSeededRandomPool psgb; //pseudo slucajan generiran broj
23.
24.     // generiranje
25.     RSA::PrivateKey privKey;
26.     privKey.GenerateRandomWithKeySize(psgb, 256); //generiraj privatni kljuc
27.     RSA::PublicKey pubKey(privKey); //generiraj javni kljuc
28.
29.     cout << "modul: " << hex << privKey.GetModulus() << endl;
30.     cout << "privatni eksponent (d): " << hex << privKey.GetPrivateExponent() << endl;
31.     cout << "javni eksponent (e): " << hex << privKey.GetPublicExponent() << endl;
32.     cout << endl;
33.
34.     ///
35.     //n, e i d zadani
36.     //Integer n("0xbeadb3d839f3b5f"), e("0x11"), d("0x21a5ae37b9959db9");
37.
38.     //RSA::PrivateKey privKey;
39.     //privKey.Initialize(n, e, d);
40.
41.     //RSA::PublicKey pubKey;
42.     //pubKey.Initialize(n, e);
43.     ///
44.
45.     string obicnitekst, dekrptirantekst;
46.     Integer tr, er, dr;
47.
48.     // Obicni tekst
49.     obicnitekst="";
50.     cout<<"Unesi tekst za RSA enkripciju i dekrpciju: ";
51.     getline(cin,obicnitekst);
52.     cout << "Obicni tekst (" << obicnitekst.size() << " bytes): ";
53.     cout << obicnitekst;
54.     cout << endl << endl;
55.
56.     // obicnitekst -> big endian
57.     tr = Integer((const byte *)obicnitekst.data(), obicnitekst.size());
58.     cout << "tajna rijec (int): " << hex << tr << endl;
59.
60.     // Enkripcija
61.     er = pubKey.ApplyFunction(tr);
62.     cout << "enkriptirana rijec (int): " << hex << er << endl;
63.
```

```

64. // Dekripcija
65. dr = privKey.CalculateInverse(psgb, er);
66. cout << "dekriptirana rijec (int): " << hex << dr << endl;
67.
68. // dekriptiranitekst - enkodiranje za ispis
69. size_t req = dr.MinEncodedSize();
70. dekriptiranitekst.resize(req);
71. dr.Encode((byte *)dekriptiranitekst.data(), dekriptiranitekst.size());
72.
73. cout << "Dekriptirani tekst: " << dekriptiranitekst << endl;
74.
75. return 0;
76. }
77. // g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o rsav2 rsav2.cpp -lcryptopp -lpthread
78. // ./rsav2

```

5. Hash

SHA (eng. Secure Hash Algorithm) je klasa kriptografskih funkcija za sažimanje, odnosno algoritama koji blok podataka varijabilne dužine preslikavaju u niz bitova fiksne dužine. Ulazni podaci poruke obično se u kontekstu kriptografije nazivaju poruka (eng. message), a izlazni podaci se nazivaju sažetak (eng. digest). Dakle, korištenjem hash algoritama dolazi do sažimanja unesene poruke prilikom čega nastaje digest koji je uvijek jednake duljine, neovisno o veličini unesene poruke. Jedna od glavnih značajki algoritama za sažimanje jest da se iz digesta ne može dobiti izvorni set podataka, kako se primjerice može kod enkripcije gdje se uz posjedovanje odgovarajućeg ključa dekripcijom može vratiti u izvorni oblik. Primjena funkcija sažimanja je kod stvaranja digitalnog potpisa, sažimanja i pohranjivanja lozinki, izvođenja ključeva za ostvarenje sigurne komunikacije, integriteta podataka, itd.

Najpoznatije funkcije sažimanja su MD4, MD5, SHA-0, SHA-1, SHA-2 te SHA-3, no MD4 i MD5 se danas smatraju prilično nesigurnim jer su probijene kolizijskim napadom, a najčešće se koristi SHA-2, većinom u 256-bitnoj varijanti koji ćemo kratko opisati na primjeru te pokazati kako algoritam izvesti uz pomoć Crypto++ biblioteke.

5.1. Hash – Opis

Riječ je o jednostavnom programu u kojem se koristeći biblioteku Crypto++ izvodi sažimanje poruke koju sami unesemo. Za unesenu istu poruku, uvijek ćemo dobiti isti sažetak, no ukoliko napravimo i malu promjenu u unosu ulaznih podataka, npr. promijenimo li samo jedan znak, dolazi do velike promjene u izlaznom sažetku. Zbog toga se može reći da je rezultat hash funkcije digitalni „otisak prsta“. Za sažimanje se u programu koristi funkcija

*CalculateDigest(byte *digest, const byte *input, size_t length)*

koja je ugrađena u Crypto++ biblioteku, gdje argument *digest* predstavlja pokazivač na buffer koji će primiti hash, *input* predstavlja naš unos, a *length* veličinu unosa, u bajtovima.

5.2. Hash - Program

5.2.1. Primjer 1

```
1. #include <cryptopp/hex.h>
2. #include <cryptopp/sha.h>
3. #include <cryptopp/base64.h>
4. #include <iostream>
5. #include <string>
6.
7. int main()
8. {
9.     CryptoPP::SHA256 hash;
10.    byte digest[CryptoPP::SHA256::DIGESTSIZE];
11.
12.    //unos poruke
13.    std::string message;
14.    std::cout << "Upisite poruku: ";
15.    std::getline(std::cin, message);
16.
17.    //virtual void HashTransformation::CalculateDigest(byte *digest, const byte *input,
18.    //size_t length)
19.    //Ažurira/izracunava funkciju sazimanja od određenog input-a
20.    hash.CalculateDigest(digest, (const byte *)message.c_str(), message.size());
21.
22.    //transformacije buffer-a
23.    CryptoPP::HexEncoder encoder;
24.    std::string output;
25.    encoder.Attach(new CryptoPP::StringSink(output));
26.    encoder.Put(digest, sizeof(digest));
27.    encoder.MessageEnd();
28.
29.    //ispis sazetka
30.    std::cout << "Rezultat: " << output << std::endl;
31.    return 0;
32. }
```

5.2.2. Primjer 2

```
1. #include <cryptopp/hex.h>
2. #include <cryptopp/sha.h>
3. #include <cryptopp/base64.h>
4. #include <iostream>
5. #include <string>
6.
7. int main()
8. {
9.     CryptoPP::SHA256 hash;
10.    byte digest[CryptoPP::SHA256::DIGESTSIZE];
11.    std::string username, password, salt, output;
12.
13.    ///unos
14.    std::cout << "Unesite korisnicko ime: ";
15.    std::getline(std::cin, username);
16.    std::cout << std::endl << "Unesite lozinku: ";
17.    std::getline(std::cin, password);
18.    salt = username + password;
19.
20.    //Azurira/izracunava funkciju sazimanja od određenog input-a
21.    hash.CalculateDigest(digest, (const byte *)salt.c_str(), salt.size());
```

```
22.  
23. //transformacije buffer-a  
24.     CryptoPP::HexEncoder encoder;  
25.     CryptoPP::StringSink *SS = new CryptoPP::StringSink(output);  
26.     encoder.Attach(SS);  
27.     encoder.Put(digest, sizeof(digest));  
28.     encoder.MessageEnd();  
29.  
30. //ispis  
31.     std::cout << "Rezultat: " << output << std::endl;  
32.     return 0;  
33. }
```

6. TCP veza server – klijent sa AES enkripcijom i dekripcijom poruka

Nakon izvođenja više, nešto jednostavnijih programa htjeli smo primijeniti koncepte enkripcije i dekripcije na nešto što bi se moglo koristiti u pravome svijetu, u području mrežnih komunikacija.

Odlučili smo da ćemo primijeniti kriptiranje na poruke poslane između klijenta i servera. Na klijentskoj strani se prvo upisuje ključ za kriptiranje, pomoću AES kriptira se poruka te se šalje serveru, uz pomoć TCP protokola. Server traži odgovarajući ključ, prima kriptiranu poruku te ispisuje njezinu „hex“ vrijednost (kriptiranu), te ju dekriptira i takvu ju ispisuje, ako je unesen točan ključ dekriptirana poruka će biti identična unesenoj sa klijentske strane.

6.1. Klijent – program

```
1. #include <sys/types.h>
2. #include <sys/socket.h>
3. #include <netdb.h>
4. #include <string.h>
5. #include <stdio.h>
6. #include <unistd.h>
7. #include <iostream>
8. #include <string>
9. #include <cstdlib>
10. #include "cryptopp/modes.h"
11. #include "cryptopp/aes.h"
12. #include "cryptopp/filters.h"
13.
14. /* IPV4 makro konstanta navodi IPV4 adresu na koju se klijent povezuje */
15. #define IPV4 "127.0.0.1"
16. /* PORT makro navodi broj TCP porta na koji se klijent povezuje */
17. #define PORT "7252"
18. /* maksimalna velicina poruke koju klijent moze prihvatiti od servera */
19. #define VELICINA 200
20.
21. int main(int argc, char **argv){
22.
23.     byte key[ CryptoPP::AES::DEFAULT_KEYLENGTH ], iv[ CryptoPP::AES::BLOCKSIZE ];
24.
25.     int kljuc, n;
26.     std::string plaintext;
27.     std::string ciphertext;
28.
29.     int opisnik; /* socket descriptor klijenta */
30.     int procitano; /* broj procitanih okteta koje nam je server poslao */
31.     char medjusprennik[VELICINA] = {'\0'}; /* medjusprennik za pohranu poruke sa ser
vera*/
32.     struct addrinfo upute; /* struktura za parametriziranje getaddrinfo poziva (u en
gl.*/
33.     struct addrinfo *rezultat; /* struktura koja ce sadrzavati popunjene informacije
o */
34.
35.
36.     std::cout<<"Unesite kljuc: ";std::cin>>kljuc;
37.     std::cout<<"Unesite poruku: ";std::cin>>plaintext;
38.
```

```

39.     memset( key, kljuc, CryptoPP::AES::DEFAULT_KEYLENGTH ); //kljuc
40.     memset( iv, kljuc, CryptoPP::AES::BLOCKSIZE ); //inicijalni vektor
41.
42.     CryptoPP::AES::Encryption aesEncryption(key, CryptoPP::AES::DEFAULT_KEYLENGTH);
    //priprema objekta za enkripciju
43.     CryptoPP::CBC_Mode_ExternalCipher::Encryption cbcEncryption( aesEncryption, iv )
    ; //postavljanje mod-a operacije CBC
44.     CryptoPP::StreamTransformationFilter stfEncryptor(cbcEncryption, new CryptoPP::S
tringSink( ciphertext ) );
45.
46.     stfEncryptor.Put( reinterpret_cast<const unsigned char*>( plaintext.c_str() ), p
laintext.length() + 1 );
47.     stfEncryptor.MessageEnd();
48.
49.     n=ciphertext.length();
50.
51.     std::cout<<"Velicina enkriptirane poruke u Bajtovima: "<<n;
52.
53.     char polje[n+1];
54.     strcpy(polje,ciphertext.c_str());
55.
56.     /* dohvatanje adrese servera */
57.     memset(&upute, 0, sizeof(struct addrinfo));
58.     upute.ai_family = AF_INET;
59.     upute.ai_socktype = SOCK_STREAM;
60.     getaddrinfo(IPV4, PORT, &upute, &rezultat);
61.
62.     /* kreiranje prikljucnice (socket-a) */
63.     opisnik = socket(rezultat->ai_family, rezultat->ai_socktype, rezultat-
>ai_protocol);
64.
65.     /* povezivanje na server */
66.     connect(opisnik, rezultat->ai_addr, rezultat->ai_addrlen);
67.     if( send(opisnik , polje , strlen(polje) , 0) < 0) {
68.         puts("Slanje fail");
69.         return 1;
70.     }
71.
72.     /* učitavanje poruke sa servera u lokalni medjusprennik */
73.     bzero(medjusprennik,200);
74.     procitano = recv(opisnik, medjusprennik, VELICINA, 0);
75.
76.     if(procitano > 0 && procitano < VELICINA) {
77.         medjusprennik[procitano] = '\0';
78.     }
79.
80.     printf("%s\n", medjusprennik);
81.     close(opisnik);
82.
83.     return 0;
84. }
85. // g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o klijentAES klijentAES.cpp -
lcryptopp
86. // ./klijentAES

```


6.2. Server - program

```
1. #include <sys/types.h>
2. #include <sys/socket.h>
3. #include <netdb.h>
4. #include <errno.h>
5. #include <string>
6. #include <string.h>
7. #include <stdio.h>
8. #include <unistd.h>
9. #include <iostream>
10. #include "cryptopp/modes.h"
11. #include "cryptopp/aes.h"
12. #include "cryptopp/filters.h"
13.
14. /* IPV4 makro konstanta navodi IPV4 adresu na koju se klijent povezuje */
15. #define IPV4 "127.0.0.1"
16. /* PORT makro navodi broj TCP porta na koji se klijent povezuje */
17. #define PORT "7252"
18. #define ERR 1;
19. #define OK 0;
20. #define VELICINA_REDA_CEKANJA 5
21.
22. int main(int argc, char **argv) {
23.
24.     byte key[ CryptoPP::AES::DEFAULT_KEYLENGTH ], iv[ CryptoPP::AES::BLOCKSIZE ];
25.
26.     int izbor=0;
27.     std::string ciphertext;
28.     std::string decryptedtext;
29.     int kljuc;
30.     char client_message[5000];
31.
32.     int povratna; /* privremena varijabla za pohranu povratnih vrijednosti funkcijskih poziva */
33.     int opisnik, opisnik_klijent; /* socket descriptor: jedan od servera i jedan koji predstavlja trenutno obradivanog klijenta */
34.     struct addrinfo upute; /* struktura za parametriziranje getaddrinfo poziva (u engl. literaturi obicno 'hints') */
35.     struct addrinfo *rezultat; /* pokazivac na strukturu koja ce sadrzavati popunjene informacije o loopback adresi servera */
36.     struct sockaddr_storage adresa_klijent; /* struktura koja ce sadrzavati informacije o povezanom klijentu */
37.     socklen_t adresa_klijent_velicina; /* velicina strukture sockaddr_storage koja se popunjava pozivom accept() */
38.
39.     std::cout<<"Unesite kljuc: ";std::cin>>kljuc;
40.     memset( key, kljuc, CryptoPP::AES::DEFAULT_KEYLENGTH ); //ključ
41.     memset( iv, kljuc, CryptoPP::AES::BLOCKSIZE ); //inicijalni vektor
42.
43.     /* dohvatanje strukture lokalne adrese */
44.     memset(&upute, 0, sizeof(struct addrinfo));
45.     upute.ai_family = AF_INET; /* koristi se IPv4 */
46.     upute.ai_socktype = SOCK_STREAM;
47.     povratna = getaddrinfo(IPV4, PORT, &upute, &rezultat);
48.     if(povratna != 0 ) {
49.         printf("getaddrinfo(): %s (%d)\n", gai_strerror(povratna), povratna);
50.         return ERR;
51.     }
52.
53.     /* kreiranje opisnika prikljucnice (socket) */
54.     opisnik = socket(rezultat->ai_family, rezultat->ai_socktype, rezultat->ai_protocol);
55.     povratna = bind(opisnik, rezultat->ai_addr, rezultat->ai_addrlen);
56.     if(povratna == -1) {
57.         int brojgreske = errno;
```

```

58.         printf("bind(): %s (%d)\n", strerror(brojgreske), brojgreske);
59.         freeaddrinfo(rezultat);
60.         return ERR;
61.     }
62.     freeaddrinfo(rezultat);
63.     povratna = listen(opisnik, VELICINA_REDA_CEKANJA);
64.     if(povratna == -1) {
65.         int brojgreske = errno;
66.         printf("listen(): %s (%d)\n", strerror(brojgreske), brojgreske);
67.         return ERR;
68.     }
69.
70.     puts("Krenula petlja");
71.     while(1) {
72.         adresa_klijent_velicina = sizeof adresa_klijent;
73.         opisnik_klijent = accept(opisnik, (struct sockaddr *)&adresa_klijent, &adresa_klijent_
velicina);
74.         if(opisnik_klijent == -1) {
75.             int brojgreske = errno;
76.             printf("accept(): %s (%d)\n", strerror(brojgreske), brojgreske);
77.             return ERR;
78.         }
79.
80.         printf("Povezao se klijent.\n");
81.
82.         bzero(client_message, 5000);
83.         read(opisnik_klijent, client_message, 5000);
84.
85.         std::cout << std::endl << "Enkriptirana poruka:" << std::endl;
86.
87.         for( int i = 0; i < strlen(client_message); i++ )
88.             std::cout << "0x" << std::hex << (0xFF & static_cast<byte>(client_message[i])) <
< " ";
89.
90.         ciphertext = client_message;
91.         std::cout << std::endl << std::endl;
92.
93.         CryptoPP::AES::Decryption aesDecryption(key, CryptoPP::AES::DEFAULT_KEYLENGTH); //dekr
ipcija
94.         CryptoPP::CBC_Mode_ExternalCipher::Decryption cbcDecryption( aesDecryption, iv ); // p
ostavljanje moda operacije CBC
95.         CryptoPP::StreamTransformationFilter stfDecryptor(cbcDecryption, new CryptoPP::StringS
ink( decryptedtext ) ); //postavljanje dekriptora
96.
97.         stfDecryptor.Put( reinterpret_cast<const unsigned char*>( ciphertext.c_str() ), ciphe
rtext.size() );
98.         stfDecryptor.MessageEnd();
99.
100.        std::cout<<"Dekriptirana poruka: "<<decryptedtext<<std::endl;
101.        decryptedtext.clear();
102.        ciphertext.clear();
103.
104.        if(povratna == -1) {
105.            int brojgreske = errno;
106.            printf("send(): %s (%d)\n", strerror(brojgreske), brojgreske);
107.            return ERR;
108.        }
109.
110.        close(opisnik_klijent);
111.        puts("Kraj veze...");
112.    }
113.    return OK;
114.}
115.// g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o serverAES serverAES.cpp -lcryptopp
116.    // ./serverAES

```

7. Zaključak

Kriptografija predstavlja jedan od najčešće korištenih vrsta računalne sigurnosti. Temelji se na primjeni algoritama (tzv. CIPHERA i DECIPHERA) koji stvarni sadržaj poruke pretvaraju u šifrirani tekst (eng. cyphertext), te uz pomoć odgovarajućih privatnih i/ili javnih ključeva vraćaju stvarni sadržaj datoteke. U ovome projektnom radu uz pomoć Crypto++ biblioteka implementirali smo neke od kriptografskih algoritama koji su se kroz povijest iskazali prilično značajnima.

Primjena Crypto++ biblioteke klasa kriptografskih algoritama i shema predstavlja vrlo kvalitetno rješenje kod problema koji se često javljaju u aplikacijskim izvedbama. Vrlo velika prednost Crypto++ biblioteke je njezina raspoloživost, veliki raspon raznolikih algoritama za kriptiranje te činjenica da je često ažurirana zbog svojeg open-source svojstva (izvorni kod je dostupan javnosti i ažuriran od strane Crypto++ zajednice). Biblioteka je besplatna na korištenje i dostupna na raznim operacijskim sustavima (Windows, iOS, Linux...) te je kompatibilna s širokim brojem C++ prevodioca (*engl. compilers*). S obzirom da je biblioteka namijenjena programskom jeziku koji je poprilično „blizu sklopovlju“ implementacije kriptografskih rješenja će u velikom broju slučajeva biti brža u odnosu na performanse u usporedbi s rješenjima izvedenim u nekim drugim programskim jezicima. Primijenili smo biblioteku na AES, koji je simetrični kriptosustav, a simetrični kriptosustavi zahtijevaju da pošiljalac i primalac znaju zajednički tajni ključ. Nadalje primijenili smo ga na RSA koji spada pod asimetrične krypto sustave, pošiljalac i primalac ne dijele isti tajni ključ. Javni ključ se koristi za kriptiranje i dostupan je svima. Privatni ključ se koristi za dekriptiranje i poznat je samo primalcu.

Naposljetku, pri izradi našeg primjera TCP veza server – klijent sa AES enkripcijom i dekripcijom poruka, naučili smo puno o podjeli zadataka, korištenju različitih funkcija, objekata i klasa iz neke biblioteke a najviše smo naučili proučavanjem same dokumentacije i objašnjena na Crypto++ Wiki. Objektno orijentirano programiranje predstavljalo je najveći problem u izvođenju projekta, pošto nismo toliko iskusni u tom načinu razmišljanja, a samo korištenje biblioteke je otežano problemima kompatibilnosti.

8. Raspodjela poslova

Sabina Pinter	<ul style="list-style-type: none">• Uvod• Instalacija biblioteke Crypto++• Hash – opis i program
Robert Manestar	<ul style="list-style-type: none">• TCP veza server – klijent sa AES enkripcijom i dekripcijom poruka• Zaključak
Josip Petanjek	<ul style="list-style-type: none">• AES – opis i program• RSA – opis i program

9. Literatura

- [1] <https://www.cryptopp.com/>
- [2] https://www.cryptopp.com/wiki/Advanced_Encryption_Standard
- [3] https://www.cryptopp.com/wiki/CBC_Mode
- [4] https://www.cryptopp.com/wiki/RSA_Cryptography
- [5] https://www.cryptopp.com/wiki/Raw_RSA
- [6] [https://elf.foi.hr/pluginfile.php/11828/mod_resource/content/8/8.%20Sigurnost do TLS-a.pdf](https://elf.foi.hr/pluginfile.php/11828/mod_resource/content/8/8.%20Sigurnost%20do%20TLS-a.pdf) – slajd 32