



# **Projektová dokumentace**

**ISA**

Přenos souboru skrz skrytý kanál

# Obsah

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Úvod do problematiky</b>                                       | <b>1</b>  |
| 1.1      | ICMP . . . . .  | 1         |
| 1.2      | AES . . . . .   | 1         |
| <b>2</b> | <b>Popis programu</b>   | <b>1</b>  |
| <b>3</b> | <b>Implementace</b>   | <b>1</b>  |
| 3.1      | Parsování argumentů . . . . .                                     | 1         |
| 3.2      | Šifrování . . . . .   | 2         |
| 3.3      | Struktura pro vlastní protokol . . . . .                          | 2         |
| 3.4      | Klient . . . . .  | 3         |
| 3.5      | Server . . . . .  | 3         |
| <b>4</b> | <b>Návod na použití</b>   | <b>4</b>  |
| <b>5</b> | <b>Testování</b>  | <b>4</b>  |
| 5.1      | 1. fáze - Arch Linux . . . . .                                    | 4         |
| 5.1.1    | Poslání textového souboru . . . . .                               | 4         |
| 5.1.2    | Poslání již existujícího souboru . . . . .                        | 5         |
| 5.1.3    | Poslání textového souboru, kde soubor má v názvu mezeru . . . . . | 6         |
| 5.1.4    | Poslání obrázku . . . . .   | 7         |
| 5.1.5    | Poslání obrázku většího než 10MB . . . . .                        | 8         |
| 5.1.6    | Poslání videa . . . . .   | 9         |
| 5.2      | 2. fáze - Ubuntu na Arch Linux . . . . .                          | 9         |
| 5.2.1    | Poslání textového souboru . . . . .                               | 9         |
| 5.2.2    | Poslání obrázku většího než 10MB . . . . .                        | 10        |
| 5.2.3    | Poslání videa . . . . .   | 11        |
| <b>6</b> | <b>Závěr</b>  | <b>11</b> |
| <b>7</b> | <b>Použité zdroje</b>   | <b>12</b> |

# 1 Úvod do problematiky

Cílem tohoto projektu je přenos souboru přes skrytý kanál.

Pro přenos dat přes síť se používají pakety. Paket je blok dat maximální velikosti standardně 1500 bytů. Každý paket má v sobě určité hlavičky, které znázorňují, které data paket přenáší, a k čemu slouží. Druhy paketů se rozlišují podle protokolů, které paket přenáší. Jeden z protokolů je například ICMP.

Dalšími druhy jsou: TCP, ARP, UDP atd.. Jedno ze základních rozdělení paketů je rozdělení na IPv4 nebo IPv6.

V tomto projektu budeme jako skrytý kanál používat ICMP paket. Aby byl přenos bezpečný, posílané data budeme šifrovat. Na šifrování použijeme šifru *AES*.

## 1.1 ICMP

ICMP neboli Internet Control Message Protocol je protokol, který primárně slouží pro oznámení chyb a diagnostiku sítě. Většinou se používá ke kontrole, jestli se data dostanou na požadovanou adresu v nějakém rozumném čase. Je to jeden z nejdůležitějších protokolů pro hlášení chyb a testování.

Nejjednodušší způsob, jak generovat ICMP pakety, je například pomocí příkazu **ping**. ICMP pakety se dají zneužít. Například ICMP flood attack nebo ping of death. Pro naše účely ICMP paket zneužijeme k posílání dat skrze něj.

## 1.2 AES

Advanced Encryption Standard je bloková šifra, která šifruje 128 bitů dat pomocí šifrovacího klíče. Šifrovací klíč může mít velikost 128, 192, a nebo 256 bitů.

Algoritmus provede několik transformací dat v iteracích. Čím větší velikost klíče, tím více iterací jednotlivých transformací šifra potřebuje.

První transformací je substituce dat pomocí substituční tabulky. Druhá a třetí transformace míchají data. Poslední transformace využívá části šifrovacího klíče na šifrování jednotlivých sloupců dat.

# 2 Popis programu

Program slouží k posílání šifrovaných souborů přes skrytý kanál. Je k tomu použit ICMP paket, do kterého jsou vložena potřebná data k rozšifrování souboru. Program slouží zároveň jako klient, tak i jako server, záleží pouze na tom, jakým způsobem je spuštěn. Ke spuštění programu je potřeba mít root práva. To ve většině případů znamená pouštět program přes příkaz **sudo**.

# 3 Implementace

Projekt je implementován v jazyce C++. Celá implementace je uložena v souboru **secret.cpp**. Nejdříve je nutné přeložit pomocí příkazu **make** a poté je možné program **secret** použít.

Program se skládá ze dvou hlavních částí: klient a server. Při spuštění programu lze specifikovat, která část se má spustit pomocí argumentu **-l**. Pokud je tento argument zadán, program funguje jako server, pokud ne, tak jako klient.

## 3.1 Parsování argumentů

Program začne parsováním argumentů. Pro tento účel existuje třída `ArgumentParser`, která má atributy pro každý možný argument programu. Pro parsování se používá funkce `getopt` z knihovny *getopt.h*.

Mezi možné argumenty patří:

- -r file : Pokud je spuštěn klient, argument je povinný a specifikuje, který soubor se má poslat. Tento argument u serveru nemá vliv.
- -s IP : Pokud je spuštěn klient, argument je povinný a specifikuje, na kterou adresu se má soubor poslat. Tento argument u serveru také nemá žádný vliv.
- -l : Pokud je zadán argument, spustí se program jako server, pokud ne, spustí se jako klient.

### 3.2 Šifrování

Program používá AES šifrování. Jako klíč je použit *xnorek01*. Pro nastavení klíčů jsou použity funkce **AES\_set\_encrypt\_key** a **AES\_set\_decrypt\_key**. Na zašifrování se používá funkce **AES\_encrypt**, na rozšifrování obdobně **AES\_decrypt**. Tyto funkce šifrují 16 bytů dat. Pokud není velikost dat 16 bytů, tak funkce sama zbytek bytů doplní.

```
char *encrypt(char *data, int length) {
    unsigned char *output = (unsigned char *)calloc(length +
                                                    COMPLEMENT(length), sizeof(char));

    if (output == nullptr) {
        std::cerr << "Allocation failed" << std::endl;
        return nullptr;
    }
    for (int shift = 0; shift < length; shift += AES_BLOCK_SIZE) {
        AES_encrypt((unsigned char*)(data + shift), (output +
                                                    shift), &key_e);
    }
    return (char *)output;
}
```

Uvedený kód výše je funkce pro zašifrování textu. Nejdříve si musíme alokovat dostatek místa pro výsledek šifrování. Na to je použita funkce `calloc`. Velikost je určena parametrem funkce `length`, ke kterému je přičten doplněk, aby velikost byla dělitelná 16. Na doplněk je použito makro *COMPLEMENT*.

```
#define COMPLEMENT(x) (AES_BLOCK_SIZE - (x % AES_BLOCK_SIZE))
```

Poté funkce iteruje skrze pole zadané parametrem `data` a každých 16 bytů zašifruje a uloží do pole `output`. Po zašifrování funkce vrací pole `output` s plně zašifrovanými daty.

Funkce na rozšifrování dat `decrypt` funguje analogicky, pouze se volá funkce **AES\_decrypt** místo funkce **AES\_encrypt**.

### 3.3 Struktura pro vlastní protokol

V posílaných paketech je uložena vlastní struktura s názvem `secrethdr`, díky které je možné jednoduše identifikovat jestli se jedná o paket poslaný klientem a také umožňuje snadný přístup k datům.

```
struct secrethdr {
    const char id[AES_BLOCK_SIZE];
```

```

int type;
int length = 0;
char data[MAX_DATA_LEN];
};

```

Proměnná `id` je identifikátor paketu. V programu je tento identifikátor zvolen jako slovo *Secretga*. Pomocí tohoto identifikátoru server zjistí, jestli to je paket od klienta, a podle toho pokračuje dál.

Proměnná `type` je typ paketu. Existují 3 typy. Typ 1 je úvodní paket, který místo dat obsahuje název souboru. Typ 2 je typ, díky kterému server ví, že má přenesená data do souboru. Poslední typ 3 se používá na kontrolu, jestli byly odchyceny všechny odeslané pakety.

Proměnná `length` udává jakou velikost mají poslaná data. Tato proměnná je nutná, abychom po rozšifrování dat věděli, jakou velikost zapsat do souboru a nevypisovali nechtěná data přidělená funkcí `AES_decrypt`. Nechtěná data vzniknou tehdy, když velikost dat není dělitelná 16.

Pole `data` uschovává posílané data. V úvodním paketu je zde obsažen název souboru, a poté jsou zde uložena zašifrovaná data souboru.

### 3.4 Klient

Klient začíná ve funkci `client`. Zde za pomoci struktury `addrinfo`, a s ní související funkce `getaddrinfo`, dostaneme základní informace, abychom mohli vytvořit soket. Jedna z takových informací je například typ protokolu (IPv4 nebo IPv6). Poté se pomocí funkce vytvoří soket. Jakmile je soket vytvořen, zavolá se funkce `send_file`, která zařídí posílání souboru.

Ve funkci `send_file` se nejdříve zašifruje název souboru a identifikátor paketu. Poté se vytvoří potřebné hlavičky paketů. V tomto případě to je ICMP hlavička a již zmiňovaná struktura `secrethdr`. Jako ICMP kód se používá `ICMP_ECHO`. Nejprve se vytvoří první paket, kde se místo dat přenesou název souboru, aby server věděl, jaký soubor vytvořit, popřípadě přepsat. Pro paket se vypočítá checksum z funkce poskytnuté v souborech k předmětu ISA. Jakmile je paket připraven, pošle se přes příkaz `sendto` na danou IP.

Následuje čtení souboru. Čte se po částech velkých 1024 bytů, aby se data mohla vlézt do standardní velikosti paketu. Načtená data se zašifrují a uloží do atributu `data` struktury `secrethdr`. Znovu se vypočítá checksum a paket je poslán. Tento proces se v iteracích opakuje dokud se nenačtou všechna data ze souboru. Během čtení se také počítá celková délka. Tato délka je využita v konečném paketu pro kontrolu na straně serveru, jestli se neztratily pakety. Po poslání souboru se tedy pošle poslední paket bez data, sloužící pouze pro kontrolu. Tímto klient končí.

Během posílání dat je použita funkce `poll` z knihovny `poll.h`, aby se pakety odesílaly pouze tehdy, jakmile jsou připravené.

### 3.5 Server

Pokud je program spuštěn jako server, zavolá se funkce `server`. Zde se chytají pakety. Pro chytání paketů je využita knihovna `pcap/pcap.h`. Jako filter je použit `icmp or icmpv6`, který chytá pouze ICMP pakety. Rozhraní, na kterém se pakety zachycují, se nazývá *any*.

Následuje samotné chytání paketů. Na to se zde používá upravený kód z [2]. Změnou je použití `pcap_loop` místo `pcap_next`. Pokud nenaskytne chyba během otevírání rozhraní, případně nastavování filtru, volá se funkce `pcap_loop`, která když zachytí packet, volá callback funkci `gotPacket`.

Ve funkci `got_packet` se nejdříve pomocí struktury `sll_header` (Linux cooked capture) zjistí typ protokolu a podle zjištěného protokolu se přetypuje paket na strukturu `secrethdr`.

Nyní se rozšifruje položka na místě, kde by paket poslaný klientem měl identifikátor. Pokud se identifikátor shoduje s globální proměnnou udávající zvolený identifikátor, pokračuje se dál. V opačném případě se nejedná o paket poslaný klientem a funkce končí.

Pokud se jedná o požadovaný paket, následuje rozdělení podle typu paketu (atribut *type* struktury *secrethdr*). Jedná-li se o typ *START*, rozšifruje se název souboru a vytvoří se, případně přepíše, pokud již soubor existoval. Je-li to typ *TRANSFER*, data se rozšifrují a ukládají se do vektoru *buffer*. Ten, pokud dosáhne velikosti 5MiB, je vypsán do souboru a vynulován. Tento buffer je použit z důvodu, aby nedocházelo k zahlcení programu díky I/O operacím. Jedná-li se o poslední typ *END*, vypíše se zbytek bufferu a provede se kontrola, jestli nebyl ztracen nějaký paket. Poté klient vypíše hlášku "File transfered", aby uživatel dostal zpětnou vazbu.

Jakmile funkce **got\_paket** skončí, server se vrací do **pcap\_loop**, kde čeká na další paket.

## 4 Návod na použití

Pro detailnější návod lze využít soubor **secret.1**, který slouží jako manuálová stránka. Manuálová stránka lze zobrazit pomocí následujícího příkazu:

```
$ man -l secret.1
```

Spuštění klienta a poslání souboru s názvem file na lokální počítač:

```
$ sudo ./secret -r test -s 127.0.0.1
$ sudo ./secret -r test -s localhost
```

Spuštění serveru:

```
$ sudo ./secret -l
```

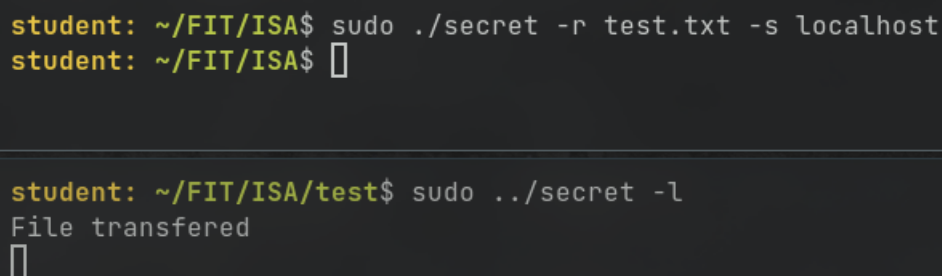
## 5 Testování

Testování bylo prováděno ve 2 fázích. První fází bylo testování na lokálním počítači, kde se posílaly soubory pouze lokálně. Druhá fáze testování spočívala v posílání souborů z virtuálního stroje zpátky na lokální počítač. Textové soubory byly porovnávány pomocí příkazu **diff** (v obrázcích z testování je proveden výpis pomocí příkazu **cat** pro porovnání souborů). Obrázky a videa pomocí příkazu **md5sum**.

Testované prostředí: Ubuntu 20.04.2 - virtuální stroj, Arch Linux 5.14.15 - lokální počítač

### 5.1 1. fáze - Arch Linux

#### 5.1.1 Poslání textového souboru



```
student: ~/FIT/ISA$ sudo ./secret -r test.txt -s localhost
student: ~/FIT/ISA$ █

student: ~/FIT/ISA/test$ sudo ../secret -l
File transfered
█
```

```
student: ~/FIT/ISA$ cat test.txt
The standard Lorem Ipsum passage, used since the 1500s

"Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum."

Section 1.10.32 of "de Finibus Bonorum et Malorum", written by Cicero in 45 BC
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ cat test.txt
The standard Lorem Ipsum passage, used since the 1500s

"Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum."

Section 1.10.32 of "de Finibus Bonorum et Malorum", written by Cicero in 45 BC
student: ~/FIT/ISA/test$
```

### 5.1.2 Poslání již existujícího souboru

```
student: ~/FIT/ISA$ sudo ./secret -r test.txt -s localhost
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ sudo ../secret -l
File already exists. File will be overwritten
File transfered

```

```
student: ~/FIT/ISA$ cat test.txt
New file that will overwrite already existing file test.txt
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ cat test.txt
New file that will overwrite already existing file test.txt
student: ~/FIT/ISA/test$
```

### 5.1.3 Poslání textového souboru, kde soubor má v názvu mezeru

```
student: ~/FIT/ISA$ sudo ./secret -r 'file with space' -s localhost
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ sudo ../secret -l
File transfered

```



```
student: ~/FIT/ISA$ cat file\ with\ space
We're no strangers to love
You know the rules and so do I
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ cat file\ with\ space
We're no strangers to love
You know the rules and so do I
A full commitment's what I'm thinking of
You wouldn't get this from any other guy
I just wanna tell you how I'm feeling
Gotta make you understand
Never gonna give you up
Never gonna let you down
Never gonna run around and desert you
Never gonna make you cry
Never gonna say goodbye
student: ~/FIT/ISA/test$
```

#### 5.1.4 Poslání obrázku

```
student: ~/FIT/ISA$ sudo ./secret -r image.jpg -s localhost
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ sudo ../secret -l
File transfered

```

```
student: ~/FIT/ISA$ md5sum -b image.jpg
69e384b55deb3014a3f39d7503eb076f *image.jpg
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ md5sum -b image.jpg
69e384b55deb3014a3f39d7503eb076f *image.jpg
student: ~/FIT/ISA/test$
```

### 5.1.5 Poslání obrázku většího než 10MB

```
student: ~/FIT/ISA$ sudo ./secret -r img10MB.jpg -s localhost
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ sudo ../secret -l
File transfered

```

```
student: ~/FIT/ISA$ md5sum -b img10MB.jpg
a53ec97be1343c08b573ddea8ff2d854 *img10MB.jpg
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ md5sum -b img10MB.jpg
a53ec97be1343c08b573ddea8ff2d854 *img10MB.jpg
student: ~/FIT/ISA/test$
```

### 5.1.6 Poslání videa

```
student: ~/FIT/ISA$ sudo ./secret -r video.mp4 -s localhost
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ sudo ../secret -l
File transfered

```

```
student: ~/FIT/ISA$ md5sum -b video.mp4
dd4e86bb0aa056bc91a67f370abbb7d3 *video.mp4
student: ~/FIT/ISA$
```

```
student: ~/FIT/ISA/test$ md5sum -b video.mp4
dd4e86bb0aa056bc91a67f370abbb7d3 *video.mp4
student: ~/FIT/ISA/test$
```

## 5.2 2. fáze - Ubuntu na Arch Linux

### 5.2.1 Poslání textového souboru

```
student@student-vm:~/ISA$ sudo ./secret -r test.txt -s 192.168.0.172
[sudo] password for student:
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ sudo ./secret -l
File transfered

```

```
student@student-vm:~/ISA$ cat test.txt
The standard Lorem Ipsum passage, used since the 1500s

"Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum."

Section 1.10.32 of "de Finibus Bonorum et Malorum", written by Cicero in 45 BC
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ cat test.txt
The standard Lorem Ipsum passage, used since the 1500s

"Lorem ipsum dolor sit amet, consectetur adipiscing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
Duis aute irure dolor in reprehenderit in voluptate velit
esse cillum dolore eu fugiat nulla pariatur. Excepteur
sint occaecat cupidatat non proident, sunt in culpa qui
officia deserunt mollit anim id est laborum."

Section 1.10.32 of "de Finibus Bonorum et Malorum", written by Cicero in 45 BC
student: ~/FIT/ISA$
```

### 5.2.2 Poslání obrázku většího než 10MB

```
student@student-vm:~/ISA$ sudo ./secret -r img10MB.jpg -s 192.168.0.172
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ sudo ./secret -l
File transfered

```

```
student@student-vm:~/ISA$ md5sum -b img10MB.jpg
a53ec97be1343c08b573ddea8ff2d854 *img10MB.jpg
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ md5sum -b img10MB.jpg
a53ec97be1343c08b573ddea8ff2d854 *img10MB.jpg
student: ~/FIT/ISA$
```

### 5.2.3 Poslání videa

```
student@student-vm:~/ISA$ sudo ./secret -r video.mp4 -s 192.168.0.172
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ sudo ./secret -l
File transfered

```

```
student@student-vm:~/ISA$ md5sum -b video.mp4
dd4e86bb0aa056bc91a67f370abbb7d3 *video.mp4
student@student-vm:~/ISA$
```

```
student: ~/FIT/ISA$ md5sum -b video.mp4
dd4e86bb0aa056bc91a67f370abbb7d3 *video.mp4
student: ~/FIT/ISA$
```

## 6 Závěr

Při správném postupu klient posílá ICMP pakety, které obsahují zašifrovaný soubor. Správně je i posílá na zadanou IP adresu. Server zvládne ochytávat příchozí pakety, kontrolovat, jsou-li to pakety zaslané klientem, a pokud ano, dešifrovat jejich obsah a zapsat přenášený soubor do lokální složky, kde je server spuštěn.

## 7 Použité zdroje

- [1] Bernstein, C.; Cobb, M.: AES encryption. [online], 2021.  
URL <https://searchsecurity.techtarget.com/definition/Advanced-Encryption-Standard>
- [2] CARSTENS, T.: Programming with pcap. [online], 2012.  
URL <https://www.tcpdump.org/pcap.html>
- [3] Cloudflare: What is ICMP? [online].  
URL <https://www.cloudflare.com/learning/ddos/glossary/internet-control-message-protocol-icmp/>
- [4] TORVALDS, L.: [online], 2013.  
URL <https://sites.uclouvain.be/SysInfo/usr/include/linux/>